МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДЕНО
Решением кафедры и допущено
«11» октября 2021 г., протокол №3

Заведующий кафедрой
«Вычислительная техника»
 К.В.Святов

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ
ПО ОСНОВНОЙ ПРОФЕССИОНАЛЬНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
ВЫСШЕГО ОБРАЗОВАНИЯ – ПРОГРАММЫ МАГИСТРАТУРЫ

Направление подготовки
09.04.01 – Информатика и вычислительная техника

Программа подготовки
Искусственный интеллект в автоматизации проектирования

Квалификация выпускника
Магистр

Формы обучения
Очная

Ульяновск 2021г.

Методическое обеспечение по программе магистратуры 09.04.01 «Информатика и вычислительная техника» профиль «Искусственный интеллект в автоматизации проектирования»

1. Соснин П.И., Валюх В.В. Моделирование рассуждений в человеко-компьютерной деятельности: учебное пособие. – Ульяновск: УлГТУ, 2018. – 145 с. URL: http://venec.ulstu.ru/lib/disk/2017/458.pdf

2. Соснин П.И., Маклаев В.А., Перцев А.А. Управление знаниями и опытом в проектной организации: учебное пособие – Ульяновск: УлГТУ, 2018. – 2018. – 213 с. URL: http://venec.ulstu.ru/lib/disk/2017/464.pdf

3. Барский, А. Б. Искусственный интеллект и логические нейронные сети : учебное пособие / А. Б. Барский. — Санкт-Петербург : Интермедия, 2019. — 360 c. — ISBN 978-5-4383-0155-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/95270.html (дата обращения: 14.01.2021). — Режим доступа: для авторизир. пользователей

4. Глубокое обучение на Python / Франсуа Шолле ; [пер. с англ. А. Киселев]. - Санкт-Петербург [и др.] : Питер, 2019. - 397 с. – ISBN 978-5-4461-0770-4

5. Рекомендательные системы на практике / Ким Фальк ; пер. с англ. Д. М. Павлова. - Москва : ДМК Пресс, 2020. - 447 с. – ISBN 978-5-97060-774-9

6. Введение в информационный поиск / Кристофер Д. Маннинг, Прабхакар Рагхаван, Хайнрих Шютце ; [пер. с англ. Д. А. Клюшина]. - Москва: Вильямс, 2011. - 520 с. - ISBN 978-5-8459- 1623-5

7. Орельен, Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем / Ж. Орельен. – Москва: Вильямс, 2018. – 688 с. – ISBN 978-5-9500296-2-2, 978-1-491-96229-9.

8. Гудфеллоу Я., Бенджио И., Курвилль А. - Глубокое обучение - Издательство "ДМК Пресс" - 2018 - 652с. - ISBN: 978-5-97060-618-6 - Текст электронный // ЭБС ЛАНЬ - URL: https://e.lanbook.com/book/107901

9. http://learn.ulstu.ru – Курс «Архитектурное моделирование в проектировании интеллектуальных систем»

10. Соснин П.И., Валюх В.В. Человеко-компьютерное взаимодействие. – Ульяновск: УлГТУ, 2020. – 119 с. URL: http://venec.ulstu.ru/lib/disk/2021/23.pdf

11. http://learn.ulstu.ru – курс «Экспериментальные исследования в проектировании интеллектуальных систем»

12. Волк В.К. Базы данных : учебное пособие. Ч.1. Проектирование и программирование / В.К. Волк ; Министерство науки и высшего образования Российской Федерации, Курганский государственный университет ; [науч. ред. В.А. Симахин]. - Курган : Издательство Курганского государственного университета, 2018.

13. Волк В.К. Базы данных : учебное пособие. Ч.2. Администрирование / В.К. Волк ; Министерство образования и науки Российской Федерации, Курганский государственный университет ; [науч. ред. В.А. Симахин]. - Курган : Издательство Курганского государственного университета, 2018. - 127, [1] с. - Библиогр.: с. 127. - ISBN 978-5-4217- 0440-9.

14. Python в веб - приложениях, фреймворк Django / Виталий Грибачев. - Санкт-Петербург : АЙСИНГ, 2015. - 247 с. – ISBN 978-5-91753-106-9

15. Python. Создание приложений / Уэсли Дж. Чан ; [пер. с англ. О. Л. Пелявского, К. А. Птицына]. - 3-е изд. - Москва [и др.] : Вильямс, 2015. - 808 с. – ISBN 978-5-8459-1793-5

16. Разработка приложений на языке программирования Python с использованием Фреймворка Django / Д. А. Ахметшин. - Казань : Школа, 2019. - 115 с. – ISBN 978-5-00162-058-7

17. https://www.intuit.ru/studies/courses/1178/330/info - Курс «Проектирование информационных систем»

a. Бондарева И.О. Методические рекомендации к практическим работам по дисциплине

«Методы управления знаниями и принятием решений» студентов направления 09.04.01

«Информатика и вычислительная техника», программа магистратуры «Искусственный интеллект в автоматизации проектирования», – Ульяновск, 2021. 87 стр.

b. Бондарева И.О. Методические рекомендации к самостоятельной работе студентов по дисциплине «Методы управления знаниями и принятием решений» студентов направления

09.04.01 «Информатика и вычислительная техника», программа магистратуры

«Искусственный интеллект в автоматизации проектирования», – Ульяновск, 2021. 11 стр.

18. .http://learn.ulstu.ru – курс «Интеллектуальное управление мобильными роботами»

19. https://github.com/ulstu/robotics

20. Рассказова Ж.В. Рабочая тетрадь к курсу «Методология и методы научного исследования» / Ж.В. Рассказова. — Владикавказ : Северо-Осетинский государственный педагогический институт, 2020. — 78 с. — ISBN 978-5-98935-226-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/101487.html.

— Режим доступа: для авторизир. пользователей

21. Методология научного творчества : учебное пособие. — Тюмень : Тюменский индустриальный университет, 2020. — 80 с. — ISBN 978-5-9961-2391-9. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/115077.html. — Режим доступа: для авторизир. Пользователей

22. Киценко Т.П. Методология, планирование и обработка результатов эксперимента в научных исследованиях : учебно-методическое пособие / Киценко Т.П., Лахтарина С.В., Егорова Е.В..

— Макеевка : Донбасская национальная академия строительства и архитектуры, ЭБС АСВ, 2020. — 70 с. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/93862.html. — Режим доступа: для авторизир.

Пользователей

23. Методы одномерного и многомерного безусловного поиска: методические указания для выполнения практических работ по дисциплине «Методы оптимизации» для обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» [Электронный ресурс] / Составитель: Т.В. Хоменко. – Ульяновск: УлГТУ, 2021. – 34с. – Режим доступа – http://learn.fist.ulstu.ru

24. Методы оптимизации дискретных задач: методические указания для выполнения практических работ по дисциплине «Методы оптимизации» для обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» [Электронный ресурс] / Составитель: Т.В. Хоменко. – Ульяновск: УлГТУ, 2021. – 24с. – Режим доступа – http://learn.fist.ulstu.ru

25. Методы многокритериальной оптимизации: методические указания для выполнения практических работ по дисциплине «Методы оптимизации» для обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» [Электронный ресурс] / Составитель: Т.В. Хоменко. – Ульяновск: УлГТУ, 2021. – 26с. – Режим доступа: http://learn.fist.ulstu.ru

26. Методы решения задач линейного программирования: методические указания для выполнения расчётно-графической работы по дисциплине «Методы оптимизации» для обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» [Электронный ресурс] / Составитель: Т.В. Хоменко. – Ульяновск: УлГТУ, 2021. – 45с. – Режим доступа – http://learn.fist.ulstu.ru

27. Соснин П.И., Валюх В.В. Человеко-компьютерное взаимодействие. – Ульяновск: УлГТУ, 2020. – 119 с. URL: http://venec.ulstu.ru/lib/disk/2021/23.pdf

28. http://learn.ulstu.ru – курс «Аналитическое моделирование в проектировании автоматизированных систем»

29. Буслов В.А. Компьютерные технологии в науке и образовании, Воронеж : ГОУВПО "Воронежский государственный технический университет"

30. Назаркин, О.А. Современные технологии разработки распределенных вычислительных систем : учеб. пособие / В.А. Алексеев; О.А. Назаркин .— Липецк : Изд-во ЛГТУ, 2017 .— 68с. — ISBN 978-5-88247-840-6 .— URL: https://rucont.ru/efd/652002 (дата обращения: 25.09.2021)

31. Алексеева, М. Б. Анализ инновационной деятельности : учебник и практикум для бакалавриата и магистратуры [Текст] / М. Б. Алексеева, П. П. Ветренко. — М. : Издательство Юрайт, 2017. – 303 с.

32. Гончаренко, Л. П. Инновационный менеджмент : учебник для академического бакалавриата [Текст] / Л. П. Гончаренко, Б. Т. Кузнецов, Т. С. Булышева, В. М. Захарова ; под общ. ред. Л. П. Гончаренко. — 2-е изд., перераб. и доп. – М. : Юрайт, 2016. – 487 с.

33. Друкер, П.Ф. Менеджмент. Вызовы XXI века [Текст] / П.Ф. Друкер ; пер. с англ. Н. Макарова. – М.: Манн, Иванов и Фербер, 2012. – 256 с.

34. Кремер, Н. Ш. Математическая статистика : учебник и практикум для академического бакалавриата [Текст] / Н. Ш. Кремер. – М. : Юрайт, 2017. – 259 с.

35. Тарасенко, Ф.П. Прикладной системный анализ. Учебное пособие [Текст] / Ф.П. Тарасенко. – М.: КноРус, 2010. – 224 с.

36. Рыков, С. П. Основы научных исследований : учебное пособие для вузов / С. П. Рыков. — Санкт-Петербург : Лань, 2021. — 132 с. — ISBN 978-5-8114-5902-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/159496 (дата обращения: 03.10.2021). — Режим доступа: для авториз. пользователей.

37. Курс «Практика и дипломное проектирование студентов кафедры ВТ». Режим доступа: http://learn.ulstu.ru/

38. Ульяновский гос. технический ун-т. Кафедра "Вычислительная техника". Общие требования к выполнению выпускной квалификационной работы: методические указания / сост. В. Н. Арефьев. - Ульяновск: УлГТУ, 2012. - 82 с.: ил

39. Барвинский А. А. Курс лекций по психологии и педагогике. Раздел «Психология и педагогика высшей школы» : учебное пособие / А. А. Барвинский. – Сумы : Сумский государственный университет, 2015 – 110с.

40. Блинов, В. И. Методика преподавания в высшей школе : учебно-практическое пособие / В. И. Блинов, В. Г. Виненко, И. С. Сергеев. — Москва : Издательство Юрайт, 2019. — 315 с. — (Образовательный процесс). — ISBN 978-5-534-02190-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/432114.

41. Воронина А.В. Управление персоналом: учеб. пособие / А.В. Воронина, О.Г Сорокина, Л.Ю. Сербинович, А.В. Охотников; под ред. А.В. Ворониной; ФГБОУ ВО РГУПС. - Ростов н/Д. - 2017. - 186 с.

42. Воронина, А.В. Профессиональное самоопределение и управление коллективом [Электронный ресурс] : учеб.-метод. пособие для практ. занятий и самостоят. работы(направление подгот. "Сервис") / А. В. Воронина ; ФГБОУ ВО РГУПС. - Ростов н/Д : [б. и.], 2017. - 86 с.

43. Дудина, М. Н. Дидактика высшей школы: от традиций к инновациям : учебное пособие для вузов / М. Н. Дудина. — Москва : Издательство Юрайт, 2020. — 151 с. — (Высшее образование). — ISBN 978-5-534-00830-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/453318.

44. Игнатова, В. В. Педагогика и психология высшей школы : учебное пособие / В. В. Игнатова, Н. А. Красноперова, С. А. Сапрыгина. — Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2018. — 98 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/147445.

45. Кашапов, М. М. Профессиональное становление педагога. Психолого-акмеологические

основы : учебное пособие для вузов / М. М. Кашапов, Т. В. Огородова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 183 с. — (Высшее образование). — ISBN 978-5- 534-08306-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/454223.

46. Коргова, М. А. Менеджмент организации : учебное пособие для вузов / М. А. Коргова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2021. — 197 с. — (Высшее образование).— ISBN 978-5-534-10829-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/474145.

47. Коржуев, А. В. Теория обучения : учебное пособие / А. В. Коржуев, В. А. Попков. — Москва: Академический Проект, 2020. — 269 с. — ISBN 978-5-8291-2737-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/132379.

48. Куклина, Е. Н. Организация самостоятельной работы студента : учебное пособие для вузов / Е. Н. Куклина, М. А. Мазниченко, И. А. Мушкина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 235 с. — (Университеты России). — ISBN 978-5-534-06270-0.— Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/437654.

49. Марасанов, Г. И. Психология инновационной активности руководителя : сборник научных трудов / Г. И. Марасанов. — Москва : Когито-центр, 2018. — 236 с. — ISBN 978-5-89353-529-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/109401.

50. Образцов, П. И. Основы профессиональной дидактики : учебное пособие для вузов / П. И. Образцов. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 230 с. — (Высшее образование). — ISBN 978-5-534-07767-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/449587.

51. Овсянникова, О. А. Психология и педагогика высшей школы : учебное пособие для вузов / О. А. Овсянникова. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 236 с. — ISBN 978-5-8114-7369-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/159491

52. Педагогика и психология высшей школы, Учебно-методическое пособие, Клименко В.А., Островский С.Н., Шершнёва Т.В., 2020

53. Плаксина, И. В. Интерактивные образовательные технологии : учебное пособие для академического бакалавриата / И. В. Плаксина. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 151 с. — (Бакалавр. Академический курс). — ISBN 978-5-534- 07623-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/434374.

54. Попков, В. А. Теория и практика высшего образования : учебник для вузов / В. А. Попков, А. В. Коржуев. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2017. — 342 с. — (Образовательный процесс). — ISBN 978-5-534-01224-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/399654.

55. Самойлова, И. В. Психология и педагогика высшей школы : учебное пособие / И. В. Самойлова. — Пенза : ПГАУ, 2018. — 267 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131187.

56. Спиридонова, Е. А. Управление инновациями : учебник и практикум для вузов / Е. А. Спиридонова. — Москва : Издательство Юрайт, 2020. — 298 с. — (Высшее образование). — ISBN 978-5-534-06608-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/455349.

57. Технология профессионально-ориентированного обучения в высшей школе : учебное пособие / П. И. Образцов, А. И. Уман, М. Я. Виленский ; под редакцией В. А. Сластенина. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 258 с. — (Образовательный процесс). — ISBN 978-5-534-07122-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/438216.

58. Шарипов, Ф. В. Педагогика и психология высшей школы : учебное пособие / Ф. В.

Шарипов.— Москва : Логос, 2020. — 448 c. — ISBN 978-5-98704-587-9. — Текст : электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/163116.

59. Защита компьютерной информации : учебное пособие / Е. С. Бондарев, В. М. Васюков, П. Р. Грушевский, О. В. Скулябина. — Санкт-Петербург : БГТУ "Военмех" им. Д.Ф. Устинова, 2019. — 146 с. — ISBN 978-5-907054-82-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/157086 (дата обращения: 30.09.2021).

60. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2021. — 312 с. — (Высшее образование). — ISBN 978-5-9916-9043-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/471159 (дата обращения: 30.09.2021).

61. Трайнев, В. А. Системный подход к обеспечению информационной безопасности предприятия (фирмы) : монография / В. А. Трайнев. — Москва : Дашков и К, 2018. — 332 с.— ISBN 978-5-394-03016-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/103788 (дата обращения: 30.09.2021).

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.01 Прикладные интеллектуальные системы и экспертные системы
Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Методическое пособие по дисциплине Б1.В.01 «Прикладные интеллектуальные системы и экспертные системы» доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/161064

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.02 машинное обучение

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В. В. Воронина, А. В. Михеев,

Н. Г. Ярушкина, К. В. Святов

ТЕОРИЯ И ПРАКТИКА МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Ульяновск
УлГТУ
2017

УДК 004.8 (075)
ББК 32.973-018.1я7
В12

Рецензенты:

Главный научный сотрудник ФНЦП АО «НПО «Марс», д-р техн. наук Токмаков Г. П.

Начальник расчетно-теоретического отдела ОАО «Ульяновское конструкторское бюро приборостроения», канд. техн. наук Сорокин М. Ю.

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Воронина, Валерия Вадимовна

В12 Теория и практика машинного обучения : учебное пособие / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. – Ульяновск : УлГТУ, 2017. – 290 с.

ISBN 978-5-9795-1712-4

Учебное пособие рассматривает вопросы, связанные с анализом данных: модели, алгоритмы, методы и их реализацию на языке Python. Особое внимание уделено анализу временных рядов. Книга предназначена для студентов группы направлений 09, а также для студентов других групп направлений, изучающих дисциплины, связанные с разработкой приложений в области анализа данных, в том числе TimeSeriesDataMinig и DataMining.

УДК 004.8 (075)
ББК 32.973.2-018.2я7

ISBN 978-5-9795-1712-4

Содержание

Введение

На протяжении всего периода своего развития человечество постоянно ищет способы автоматизации тех или иных видов деятельности, в итоге перекладывая на «плечи» механизмов все более обширные и сложные обязанности. Изначально мечты о роботах, выполняющих всю работу, находили воплощение лишь в литературных произведениях, однако с развитием технологий им открылся путь и в реальность. Сейчас уже не столь фантастичными звучат слова песни «До чего дошел прогресс» (к/ф «Приключения Электроника»): «До чего дошел прогресс – труд физический исчез, да и умственный заменит механический процесс». Ведь эти строки достаточно полно отражают суть такой дисциплины, как «машинное обучение», благодаря которой их воплощение в жизнь становится возможным. Однако стоит заметить, что до полной реализации описанного в песне еще очень далеко: пока машинное обучение не способно заменить умственный труд, и в ближайшее время вряд ли сможет. Но шаги в этом направлении активно совершаются.

С теоретической стороны машинное обучение – дисциплина, находящаяся на пересечении математической статистики, численных методов оптимизации, теории вероятностей, а также дискретного анализа. С помощью ее методов происходит решение задачи извлечения знаний из данных, которой занимается еще только формирующаяся область «Интеллектуальный анализ данных» (DataMining). Согласно Википедии [1]: «В теории искусственного интеллекта и экспертных систем, знание – это совокупность утверждений о мире, свойствах объектов, закономерностях процессов и явлений, а также правил логического вывода одних утверждений из других и правил использования их для принятия решений. Главное отличие знаний от данных состоит в их структурности и активности: появление в базе знаний новых фактов или установление новых связей между ними может стать источником изменений в принятии решений». Если,

опираясь на это определение знания, рассмотреть DataMining как процесс, то его сутью будет нахождение в «сырых» данных знаний, обладающих свойствами нетривиальности, интерпретируемости и практической полезности для принятия решений в различных сферах деятельности человека. Причем, эти знания ранее неизвестны.

С практической же стороны машинное обучение нацелено на создание систем, способных адаптироваться к решению различных задач без явного кодирования алгоритма, то есть систем, способных обучаться.

В данной книге рассмотрены основные модели и алгоритмы машинного обучения, в том числе для анализа временных рядов. Наряду с этим представлены и их практические реализации на языке Python. В последних разделах книги обучающемуся предлагаются контрольные вопросы по пройденным темам, а также задачи для выполнения, с помощью которых он сможет проверить себя и закрепить полученные навыки.

Книга подготовлена при реализации проекта №2.4760.2017/8.9 «Исследование и разработка моделей, методов и алгоритмов гибридизации нечетких предметных онтологий, логического вывода и интеллектуального анализа временных рядов» в рамках государственного задания Минобрнауки Российской Федерации.

О ЗАДАЧАХ МАШИННОГО ОБУЧЕНИЯ

Как было сказано выше, машинное обучение нацелено на создание систем, способных получать знания из данных, а также способных с помощью обучения улучшать показатели своей работы. Таким образом, можно утверждать, что машинное обучение является одной из областей науки о данных (DataScience). На рисунке 1 представлены ее составляющие и показано место машинного обучения среди них.

Рисунок 1. Области науки о данных

Сложность работы в сфере DataMining обуславливается неточностью, противоречивостью, разнородностью, неполнотой данных, которые при этом могут иметь гигантские объемы. Для их обработки требуются специальные программные средства: инструменты преобразования «сырых» данных в информацию, а информацию в знания. Кроме того, алгоритмы обработки данных должны иметь возможность обучаться по прецедентам. Машинное обучение (МО) как раз и занимается разрешением подобных сложностей. Причем в настоящее время усилия сконцентрированы на повышении

не столько качества решения, сколько скорости и оптимальности его получения.

Как видно из рисунка 1, машинное обучение пересекается с анализом данных, то есть имеет общие черты, но также и свою специфику. Анализ данных – это огромная и уже относительно не молодая область математики и информатики. Под нее попадает всяческая обработка данных. Например, обработка данных физического эксперимента статистическими методами для получения обобщенной информации (усредненных значений, точных значений, доверительных интервалов, оценок и тому подобное), обработка сигнала методом Фурье, даже подсчет минимального и максимального значений или отношения каких-то величин может быть анализом данных, если эти операции влекут за собой важные выводы, знания о данных и позволяют решить поставленную задачу. Так получение дисперсии некой величины является анализом данных, но не является в чистом виде машинным обучением. Хотя важно отметить, что методы статистики имеют большое значение и в машинном обучении, о чем мы еще будем говорить. Отличительной особенностью методов машинного обучения является то, что они способны решать широкий спектр задач без явного кодирования алгоритма решения. Например, метод Регрессии входит в машинное обучение. Регрессия давно применяется в экономике для прогнозирования и оценки разнообразных параметров, потому что данная математическая модель в некотором роде универсальна. Однако стоит заметить, что регрессия появилась в математике гораздо раньше, чем ее стали использовать в экономике, а вот машинным обучением это стало только тогда, когда все эти методы начали рассматриваться не как обособленные механизмы решения задач из специальных областей, а как методы преобразования информации вообще.

На рисунке 2 представлена диаграмма распределения задач, которые приходится решать в сфере машинного обучения. Необходимо отметить, что соотношение областей на данной диаграмме

отражает лишь субъективное видение отрасли авторами на момент написания книги.

Рисунок 2. Типы задач машинного обучения

Понятно, что разработка программ в области машинного обучения весьма дорога. Поэтому применять его следует лишь в случаях, когда это действительно необходимо. В таблице 1 представлены некоторые признаки, по которым можно принять решении о необходимости применения к задаче методов машинного обучения.

Таблица 1. Ключевые характеристики решаемых задач

Целесообразно применение классических методов	**Целесообразно применение методов машинного обучения**
Наличие четко формализованного алгоритма (например, поиск клиентов, которые тратят больше всего денежных средств в месяц или чаще всего покупают. Так называемая RFM сегментация).	Отсутствие четко формализованного алгоритма решения ввиду высокой вариативности решения (например, распознавание рукописного текста).
Не допускается неопределенность поведения модели (например, расчет местоположения самолета по строго заданным формулам).	Допускается неопределенность поведения модели (например, предварительный контроль качества детали, где допускается определенная доля брака).

Окончание таблицы 1

Целесообразно применение классических методов	Целесообразно применение методов машинного обучения
Экономическая целесообразность (например, методы МО дают 98% качества, классическое решение дает 90% качества, что является приемлемым, но решение на основе МО в 2 раза дороже и реализуется дольше). Поэтому здесь выбирается приоритетный критерий.	

Кроме того, для разработки решений на основе МО необходимы оцифрованные данные. Они являются ключевым аспектом, от которого зависит качество и полнота решения. Таким образом, очень важно понимать, что для применения методов машинного обучения требуются собранные и специально подготовленные данные.

Как правило, процесс машинного обучения проходит несколько этапов. На рисунке 3 представлен обобщенный алгоритм, выполняющийся при решении задач методами рассматриваемой области.

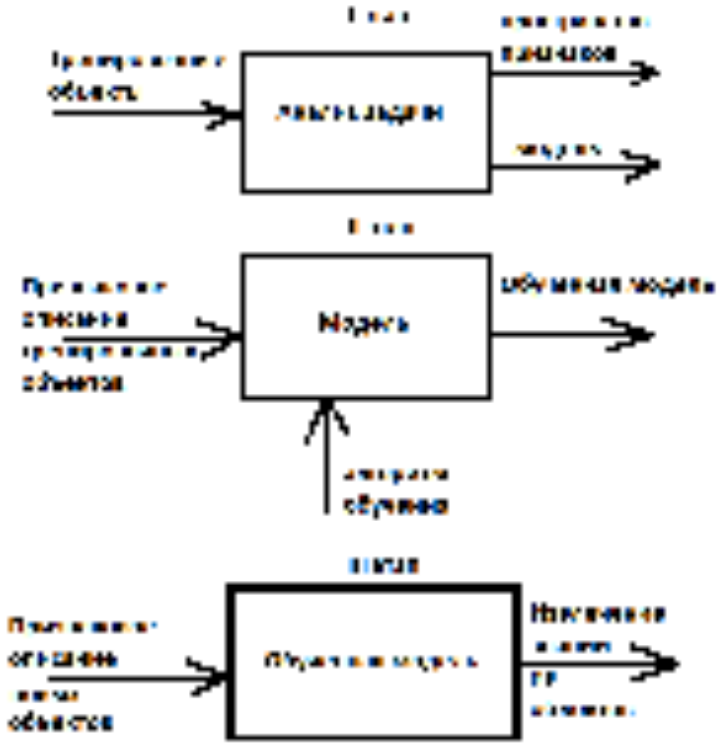


Рисунок 3. Этапы машинного обучения

Необходимо отметить, что данная иллюстрация отражает идеальный случай, когда после обучения мы сразу получили работающую модель. Однако в реальности, как правило, между вторым и третьим этапом есть промежуточное, но очень важное действие – оценка качества модели. И именно по его результатам принимается решение о переходе на следующий этап или о возврате к одному из предыдущих. Этот вопрос мы рассмотрим подробнее далее, здесь же введем определение пространства признаков, для лучшего понимания схемы с рисунка 3.

Пространство признаков

Пространство признаков – это N-мерное пространство, где N – число измеряемых характеристик объектов, выделенное для конкретной задачи. При этом объекты в пространстве признаков задаются N-мерными векторами, каждая компонента которых представляет собой значение определенной характеристики.

Рассмотрим пример. Пусть необходимо разработать функцию для сайта салона красоты, которая будет предлагать клиентам определенные косметические процедуры. Понятно, что они будут разными для разных возрастных и гендерных групп. То есть в этой задаче ключевыми будут характеристики пола и возраста людей. Графическое отображение пространства признаков, а также заданные в нем объекты A и B показаны на рисунке 4. В векторном виде объект A с характеристиками «пол» = «женский» и «возраст» = 40 будет задан как: $A= \{40, ж\}$. Объект B с возрастом 50 и мужским полом – $B= \{50, м\}$.

Как видно из рассмотренного примера, признаки объектов могут быть разными по своей природе. Существует следующая типизация:

1. Бинарные: $F \epsilon \{0,1\}$.
2. Номинальные: $|F| < \infty$.
3. Номинальные, упорядоченные: $|F| < \infty$, $F_i < F_{i+1} \ldots < F_{i+n}$.
4. Количественные: $F \epsilon R$.

Каждый тип признаков обрабатывается по-разному, о чем скажем далее. Пока же продолжим разговор о других аспектах рассматриваемой дисциплины. Как неоднократно было сказано, работа в сфере машинного обучения нацелена на создание систем, способных обучаться. Рассмотрим формальное определение понятия «обучение».

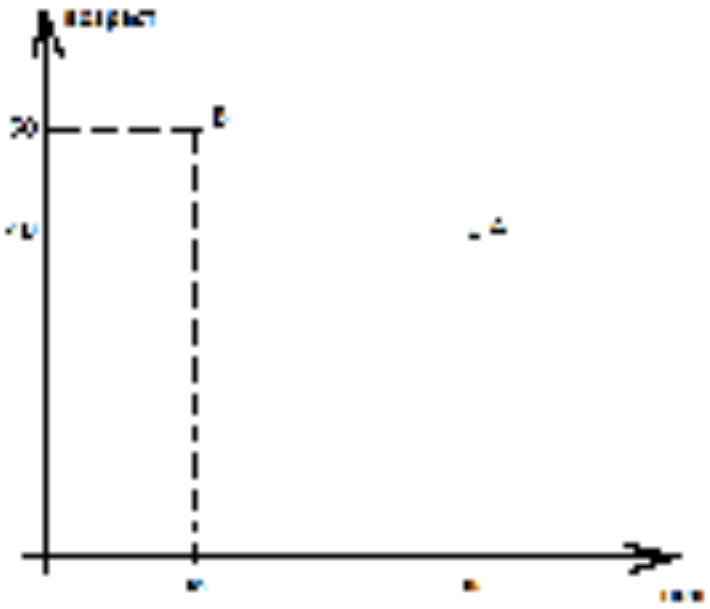


Рисунок 4. Объекты в пространстве признаков

Формальное определение понятия «обучение»

Формально понятие «обучение» специалисты определяют так: «Пусть есть некое множество задач класса *C,* для которого задано некоторое множество опыта *EX* и определена мера качества *L*. Тогда о наличии обучения на опыте *EX* относительно класса задач *C* в смысле меры качества *L* можно говорить в том случае, если при предъявлении нового опыта *EX'* возрастает качество решения задачи класса *C,* измеряемое мерой *L»*.

Существуют два основных типа обучения: с учителем и без учителя. Более подробно о них будет рассказано далее, здесь же рассмотрим формальную постановку задачи обучения для самого первого (и самого простого) типа – обучения с учителем.

Пусть X – некоторое множество объектов, по которым необходимо получить некоторое множество ответов Y. При этом предполагается, что существует некоторая зависимость $y: X \to Y$. Тогда задача обучения будет формулироваться так: дано $\{x_1 \dots x_n\} \in X$ – обучающая выборка, и $\{y_1 \dots y_n\}$ – известные ответы, причем $y_i = y(x_i)$. Найти: $a: X \to Y$ – решающую функцию, приближающую y на всем множестве X.

Данная задача в свою очередь порождает ряд подзадач:

1. Определение способа задания объектов;
2. Определение способа задания ответов;
3. Определение способа построения функции a;
4. Определение способа приближения для a и y.

Все эти подзадачи решаются последовательно, и результаты решения предыдущей, как правило, влияют на решение текущей. Поэтому самой важной среди них будет определение способа описания данных. Можно выделить следующие типы входных данных при обучении:

1. Координаты объектов в пространстве признаков;
2. Временной ряд;
3. Сигнал;
4. Изображение;
5. Видеоряд;
6. Описание взаимоотношений между объектами.

На основе описанного выше введем формальное описание приведенных на рисунке 3 второго и третьего этапов машинного обучения.

Пусть $X = \{x_1 \dots x_k\}$ – исследуемые объекты. $F = \{f_1 \dots f_N\}$ – пространство признаков, в котором эти объекты будут рассматриваться. При этом $f_i(x_j)$, $i \in [1 \dots N]$, $j \in [1 \dots k]$ – значение i-го признака j-го объекта. Тогда признаковое описание тренировочных объектов задается в виде матрицы

$$D = || f_i(x_j) ||_{k \times N} = \begin{pmatrix} f_1(x_1) \dots & f_N(x_1) \\ f_1(x_k) \dots & f_N(x_k) \end{pmatrix}.$$

Этап обучения может быть формализован следующим образом. Метод обучения $\mu: (X \times Y)_k \rightarrow A$ по выборке $XY_k=(x_i,y_i)$, $i \in [1 \ldots k]$ строит алгоритм $a=\mu(XY_k)$:

$$\begin{pmatrix} f_1(x_1) \ldots & f_N(x_1) \\ f_1(x_k) \ldots & f_N(x_k) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ y_k \end{pmatrix} \xrightarrow{\mu} a$$

Этап применения обученной модели формализуется следующим образом. Алгоритм a для новых объектов $X'=\{x'_1 \ldots x'_k\}$ выдает ответы $y_i'=a(x'_i)$, $i \in [1 \ldots k]$:

$$\begin{pmatrix} f_1(x'_1) \ldots & f_N(x'_1) \\ f_1(x'_k) \ldots & f_N(x'_k) \end{pmatrix} \xrightarrow{a} \begin{pmatrix} y'_1 \\ y'_k \end{pmatrix}.$$

Общий алгоритм машинного обучения

С понятием обучения тесно связано понятие обобщающей способности. Обобщающая способность – это свойство модели отражать исходные данные в требуемые результаты ($X \rightarrow Y$) на всем множестве исходных данных (во всех сценариях, а не только на тренировочных примерах). Величину обобщения оценивают через обратную величину – отклонение или ошибку. Ошибка – это численно выраженная разница между ответом модели и требуемым (реальным) значением. В более общем смысле обобщающая способность – способность модели найти некий «закон природы», который будет описывать неизвестную нам и скрытую взаимосвязь входных и выходных данных. Таким образом, опираясь на понятие обобщающей способности, можно выделить основные проблемные вопросы машинного обучения:

1. Достаточно ли данных для нахождения в них полезных знаний?
2. Может ли в принципе данная модель обучиться на имеющихся данных?
3. Будет ли полученная модель приближать требуемый «закон природы» на всем возможном множестве X?

4. Насколько хорошо будет работать обученная модель или насколько часто и насколько сильно будет ошибаться модель на реальных (контрольных) данных?

По сути в машинном обучении решается задача приближения алгоритма к некоторому «закону природы». С математической точки зрения задача приближения является задачей оптимизации или минимизации ошибки:

$$E=|y-y'|\rightarrow min\ .$$

Таким образом, общий алгоритм решения задач в сфере машинного обучения будет состоять из следующих шагов:

1. Понимание задачи и исходных данных;
2. Формулировка решения на математическом языке (на этом шаге важно понять, что задача формализуема, а результаты работы модели могут быть проверены);
3. Предобработка данных и выделение ключевых признаков;
4. Построение модели (или моделей);
5. Обучение модели (моделей) и оценка качества;
6. Эксплуатация модели при достижении требуемого качества, либо возврат к одному из предыдущих шагов (перенастройка модели, добыча новых данных и т. п.).

Типы задач машинного обучения

Машинное обучение применяется для решения задач в следующих областях (основные сферы применения):

1. Медицинская диагностика.
2. Техника, в частности:
    2.1. Автоматизация и управление.
    2.2. Техническая диагностика.
    2.3. Робототехника.
    2.4. Компьютерное зрение.
    2.5. Распознавание речи.
3. Экономика, в частности:
    3.1. Кредитный скоринг (credit scoring).

3.2. Предсказание ухода клиентов (churm prediction).

3.3. Обнаружение мошенничества (fraud detection).

3.4. Биржевой технический анализ (technical analysis).

3.5. Биржевой надзор (market surveillance).

4. Офисная автоматизация, в частности:

4.1. Распознавание текста.

4.2. Обнаружение спама.

4.3. Категоризация документов.

4.4. Распознавание рукописного ввода.

Если же говорить об обобщенных типах задач машинного обучения, то можно выделить следующие:

1. Регрессия (или иногда встречается термин «аппроксимация»);
2. Классификация;
3. Кластеризация.

Помимо указанных видов существуют и другие, но остановимся на обозначенных, так как они наиболее распространены. Рассмотрим формальную постановку этих задач.

Задача регрессии – приближение неизвестной целевой зависимости на некотором множестве данных. Пусть X – множество данных – описаний некоторых объектов. Y – множество возможных ответов для X.

В задаче регрессии предполагается, что существует неизвестная целевая зависимость $y: X \to Y$, чьи значения известны только на объектах обучающей выборки $XY=\{(x_1,y_1)\dots (x_n,y_n)\}$, $x \in X$, $y \in Y$. Необходимо получить алгоритм $a: X \to Y$, приближающий целевую зависимость как на множестве XY, так и на X. То есть решить задачу регрессии – значит найти алгоритм, обладающий способностью к обобщению эмпирических фактов (способностью к выводу общих знаний из частных наблюдений, прецедентов).

Задача классификации – распределение некоторого множества объектов по заданному множеству групп (классов). При этом есть некоторое подмножество объектов, для которых распределение по классам известно, классовая принадлежность остальных – неизвестна.

Требуется построить алгоритм, который указывал бы классовую принадлежность для любого объекта из исходного множества.

Формально постановку задачи классификации можно описать следующим образом. Пусть X – множество данных – описаний некоторых объектов. Y – конечное множество классов, отмеченных метками. Существует неизвестная целевая зависимость – отображение $y: X \to Y$, чьи значения известны только на объектах обучающей выборки $XY = \{(x_1, y_1) \dots (x_n, y_n)\}$, $x \in X$, $y \in Y$. Необходимо получить алгоритм $a: X \to Y$, способный классифицировать произвольный объект $x \in X$.

Как можно заметить, данная задача схожа с предыдущей. Однако главная особенность задачи регрессии заключается в том, что функция $a: X \to Y$ является непрерывной вещественной функцией. Задача классификации отличается от этого тем, что Y – дискретное множество. Кроме того, в отличие от задачи аппроксимации у задачи классификации выделяют несколько типов. По количеству классов можно выделить:

1. Классификацию на два класса: множество Y содержит всего две метки.
2. Классификацию на множество классов: Y содержит от трех до нескольких тысяч меток.

По характеру разделения объектов на классы можно выделить:

1. Классификацию на непересекающиеся классы: один объект принадлежит только одному классу.
2. Классификацию на пересекающиеся классы: один объект может принадлежать нескольким классам.
3. Классификацию на нечеткие множества: объект принадлежит всем классам с определенной степенью принадлежности.

Задача кластеризации – разделение некоторого множества объектов на непересекающиеся группы (кластеры) таким образом, чтобы каждая группа состояла из схожих объектов, а объекты разных кластеров существенно отличались.

Формально постановку задачи кластеризации можно описать следующим образом. Пусть X – множество данных – описаний некоторых объектов. Y – множество кластеров, отмеченных метками. Определена функция расстояния между объектами из исходного множества X: $f(x, x')$, и есть некоторая обучающая выборка объектов $X_o = \{x_1 \dots x_n\}$, $x \in X$. Необходимо разбить обучающую выборку на кластеры, приписав каждому x номер кластера y_i, так, чтобы близкие по метрике f объекты принадлежали одному кластеру, а объекты разных кластеров существенно отличались по метрике f. То есть необходимо построить алгоритм $a: X \rightarrow Y$, который любому $x \in X$ ставит в соответствие номер кластера $y \in Y$. Причем, иногда множество Y известно заранее, но чаще все-таки ставится задача получить оптимальное число кластеров, исходя из характера данных. Оптимальность оценивается по какому-либо критерию качества кластеризации.

Задача кластеризации сложнее аппроксимации и классификации. Это обусловлено следующими причинами:

1. Нет однозначного критерия качества кластеризации. Существует ряд эвристических критериев, а также ряд бескритериальных алгоритмов, выполняющих вполне осмысленную кластеризацию, но дающих на одних и тех же данных разные результаты.
2. Число кластеров, как правило, заранее неизвестно и задается субъективно.
3. На результат кластеризации существенное влияние оказывает выбранная метрика расстояния, которая, как правило, также выбирается субъективно.

Однако, несмотря на описанные выше сложности, кластеризация помогает достичь следующие цели:

1. Улучшить понимание данных за счет выявления их кластерной структуры: разбиение объемной выборки на группы схожих объектов может упростить дальнейшую обработку

данных за счет применения к каждому кластеру своих методов анализа.

2. Осуществить сжатие данных. В данном случае подразумевается сокращение объемной выборки за счет работы с наиболее яркими представителями кластеров (групп схожих объектов).
3. Выявить новизну в массиве данных: обнаружить нетипичные объекты, которые не удается отнести ни к одному кластеру.
4. Решить задачу таксономии: построить древообразную иерархическую структуру, упорядочивающую исходные данные. Ее построение достигается за счет дробления крупных кластеров на более мелкие, которые в свою очередь также дробятся на еще более мелкие. Визуально таксономия отображается в виде графика – дендрограммы.

Способы обучения и оценки его качества

Как было сказано выше, основная характеристика систем, разрабатываемых с помощью методов машинного обучения, – способность к обучению. В зависимости от видов решаемых задач применяют различные алгоритмы реализации этой ключевой особенности. В рамках данного пособия рассмотрим три основных вида обучения, а также определим классы задач, подходящие для каждого из этих видов.

Первый тип – обучение с учителем. Формально он был описан ранее, поэтому здесь не будем на этом останавливаться подробно. Вкратце же обучение с учителем можно описать следующим образом: дано некоторое множество объектов и множество возможных реакций системы на эти объекты. При этом ответы и объекты связаны между собой некоторой неизвестной зависимостью. Есть конечная совокупность пар объект-ответ (прецедентов), называемая обучающей выборкой. На ее основе необходимо выявить алгоритм, который впоследствии для любого объекта из исходного множества даст достаточно точный ответ. Для измерения точности ответов используется один

из функционалов качества, как правило, завязанный на вычислении отклонения полученного ответа от ожидаемого, то есть вычислении ошибки. Рассмотрим некоторые их виды. В приведенных ниже формулах используются следующие обозначения: $XY=\{(x_1,y_1)\dots(x_n,y_n)\}$ – обучающая выборка, n – количество прецедентов, y_i – фактическое значение (ожидаемый ответ) в *i-м* прецеденте ($y_i \in XY$), $\hat{y}_i$ – выданный системой ответ для x_i ($x_i \in XY$).

1. Средняя ошибка представляет собой усреднение ошибок для каждого образца и вычисляется по формуле

$$\text{СО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}.$$

2. Средняя абсолютная ошибка представляет собой усреднение абсолютных ошибок на каждом шаге и вычисляется по формуле

$$\text{САО} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}.$$

3. Среднеквадратическая ошибка вычисляется как сумма средних квадратов ошибок. Формула:

$$\text{СКО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}.$$

4. Корень из среднеквадратической ошибки вычисляется по формуле

$$\text{КСКО} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}.$$

5. Средняя относительная ошибка вычисляется как среднее относительных ошибок:

$$\text{CO} = \frac{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)}{y_i}}{n} \times 100\%.$$

6. Средняя абсолютная относительная ошибка вычисляется как среднее относительных ошибок по модулю:

$$\text{MAPE} = \frac{\sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|}{n} \times 100\% .$$

7. Симметричная средняя абсолютная относительная ошибка вычисляется как

$$\text{SMAPE} = \frac{\sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{(y_i + \hat{y}_i)/2} \right|}{n} \times 100\% .$$

У всех представленных мер качества есть свои достоинства и недостатки. Например, у первой и пятой недостаток заключается в том, что положительные и отрицательные ошибки аннулируют друг друга, поэтому в некоторых случаях они не являются достаточно хорошими индикаторами качества. В связи с этим чаще всего используется третья или четвертая меры.

Обучение с учителем используется при решении задач аппроксимации и классификации. В первом случае ответы являются действительными числами или векторами, во втором – выбираются из конечного множества меток-классов. Необходимо отметить, что приведенные выше формулы подходят только для случаев, когда ответ системы и требуемый ответ – действительные числа, получаемые при решении задачи аппроксимации. В задачах классификации же оценка качества чаще всего завязана на соотношении количеств правильно и неправильно отнесенных к классам объектов.

Второй тип обучения – обучение без учителя. Формально постановку задачи обучения без учителя можно описать следующим образом. Пусть X – множество данных – описаний некоторых

объектов. Необходимо найти множество Y, состоящее из взаимосвязей $f: (x, x')$ между объектами из X $(x, x' \in X, f \in Y)$. Качество выявления взаимосвязей проверяется некоторой метрикой, выбранной исходя из решаемой задачи.

Обучение без учителя используется для решения следующих типов задач:

1. Задача кластеризации.
2. Поиск правил ассоциации.
3. Сокращение размерности данных.
4. Визуализация данных.

В определенной степени каждая из последних трех задач является производной от первой или ее частным случаем. Рассмотрим подробнее формулировки названных задач.

Под задачей поиска правил ассоциаций подразумевается выявление в признаковых описаниях объектов (исходных данных) таких наборов и значений признаков, которые особенно часто (неслучайно часто) встречаются в исходных данных. Если же проводить аналогию с первой задачей, то каждое правило в данном случае может быть представлено как кластер.

Задача сокращения размерности данных состоит в следующем. Существует большой (значительно большой) объем признаковых описаний объектов. Причем этот объем обуславливается внушительным количеством измерений признакового пространства. Необходимо представить те же данные в пространстве меньшей размерности, при этом минимизировав потери информации. Группировка по кластерам как раз и будет одним из вариантов решения проблемы.

Задача визуализации данных является по сути частным случаем предыдущей: ее цель – представить исходные данные в отображаемом пространстве, то есть пространстве размерности 2 или 3.

Как следует из описанного выше, обучение без учителя в какой-то мере так или иначе сводится к кластеризации. Поэтому для оценки качества обучения данным способом, как правило, используют метрики качества кластеризации. Причем при их выборе учитывается,

что эти метрики не должны зависеть от исходных данных, а только от результатов разбиения. Все оценки качества можно разделить на внешние и внутренние. Первые используют внешнюю информацию об истинном разбиении объектов на кластеры, вторые опираются только на набор исходных данных, то есть данные метрики могут работать с неразмеченной выборкой, когда заранее не известно истинное разбиение объектов на группы. И именно с их помощью определяют оптимальное число кластеров.

Приведем в качестве примера метрики, выделенные компанией ODS [2]:

1. Adjusted RandIndex (*ARI*). Данная метрика относится к группе внешних: предполагается, что истинные метки объектов известны (например, заданы экспертом), однако от самих значений меток она не зависит. Только от разбиения выборки на кластеры.

Рассчитывается мера следующим образом: пусть n – число объектов в выборке, a – число пар объектов, имеющих одинаковые метки и находящихся в одном кластере, b – число пар объектов, имеющих различные метки и находящиеся в различных кластерах. Тогда можно посчитать долю объектов, для которых исходное и полученное в результате кластеризации разбиения согласованы:

$$\boldsymbol{RI} = \frac{\boldsymbol{2(a+b)}}{\boldsymbol{n(n-1)}} .$$

Полученная величина называется RandIndex (*RI*) и выражает схожесть двух разных кластеризаций одной и той же выборки. Чтобы этот индекс давал значения, близкие к нулю, для случайных кластеризаций при любом n и числе кластеров, необходимо произвести его нормирование, то есть получить AdjustedRandIndex:

$$\boldsymbol{ARI} = \frac{\mathbf{RI} - \mathbf{E[RI]}}{\boldsymbol{max}(\mathbf{RI}) - \mathbf{E[RI]}} ,$$

где E – математическое ожидание.

Мера *ARI* симметрична и не зависит от перестановок и значений меток. По сути этот индекс является мерой расстояний между различными разбиениями выборки. Его область значений – [-1,1]. Интерпретировать ее интервалы можно следующим образом: для «независимых» разбиений на кластеры – отрицательные значения, для случайных разбиений – близкие к нулю, для схожих разбиений – положительные значения, причем, *ARI*=1 говорит о совпадении разбиений.

2. Adjusted MutualInformation (*AMI*). Данная метрика схожа с предыдущей: она также симметрична и не зависит от значений и перестановок меток. Для ее определения используется функция энтропии. Разбиения выборки интерпретируются как дискретные вероятности: вероятность отнесения к кластеру равна доле объектов в нем. Как и в предыдущем случае, для данной метрики необходимо вычислить специальный индекс – MutualInformation (*MI*). Данная величина определяется как взаимная информация для двух распределений, соответствующих разбиениям выборки на кластеры. Интерпретировать это можно как долю информации, общей для обоих разбиений: насколько информация об одном из них уменьшает неопределенность относительно другого. Этот индекс рассчитывается по следующей формуле:

$$\boldsymbol{MI(X,Y)} = \sum_{\boldsymbol{y}\in\boldsymbol{Y}}\sum_{\boldsymbol{x}\in\boldsymbol{X}}\boldsymbol{p(xy)}\log\left(\frac{\boldsymbol{p(xy)}}{\boldsymbol{p(x)p(y)}}\right),$$

где *p(x, y)* – совместная функция распределения вероятностей для *X* и *Y; p(x)* и *p(y)* – функции распределения предельной вероятности для *X* и *Y* соответственно.

Областью значения индекса *AMI* является диапазон [0,1]. Интерпретируется он следующим образом: значения, близкие к нулю, говорят о независимости разбиений, а близкие к единице – об их схожести (или совпадении при *AMI*=1).

3. Гомогенность, полнота, V-мера. Данные метрики рассматривают разбиение выборки как дискретные распределения и определяются с использованием функции энтропии и условной энтропии.

Гомогенность определяет, насколько каждый кластер состоит из объектов одного класса, и рассчитывается по формуле

$$h = 1 - \frac{H(C|K)}{H(C)},$$

где K – результат кластеризации, C – истинное разбиение, H – функция энтропии. При этом

$$\boldsymbol{H(X) = -\sum_{i=1}^{n} P(x_i)\log_b P(x_i)},$$

$$\boldsymbol{H(X|Y) = \sum_{j} p(x_i y_j)\log \frac{p(y_j)}{p(x_i y_j)}}.$$

Полнота измеряет, насколько объекты одного класса относятся к одному кластеру и определяется по формуле

$$c = 1 - \frac{H(K|C)}{H(K)}.$$

Эти меры принимают значения в диапазоне [0,1]. Интерпретировать их можно следующим образом: чем больше значение, тем более точна кластеризация. Эти меры не являются симметричными и не являются нормализованными, в отличие от рассмотренных ранее, и поэтому они зависят от числа кластеров. Согласно ресурсу [2]: «Случайная кластеризация не будет давать нулевые показатели при большом числе классов и малом числе объектов. В этих случаях предпочтительнее использовать *ARI*. Однако при числе объектов более 1000 и числе кластеров менее 10 данная проблема не так явно выражена и может быть проигнорирована».

Чтобы учесть значения обеих величин, вводится симметричная *V*-мера, показывающая, насколько кластеризации схожи между собой. Ее расчет происходит по формуле

$$v = 2\frac{hc}{h+c}.$$

Силуэт. Данная метрика относится к типу внутренних: она не предполагает знания об истинном разбиении и первоначально рассчитывается для каждого объекта следующим образом. Пусть a – среднее расстояние от данного объекта до объектов из того же кластера,

b – среднее расстояние от данного объекта до объектов из ближайшего кластера (но не того, в котором находится сам объект). Тогда силуэтом данного объекта будет величина, вычисляемая по формуле

$$s = \frac{b - a}{\max(ab)}.$$

Силуэтом же всей выборки будет средняя величина силуэтов ее объектов. Интерпретировать метрику можно следующим образом: она показывает, насколько среднее расстояние до объектов своего кластера отличается от среднего расстояния до объектов других кластеров. Область значения данной величины – диапазон [-1,1]. При этом значения, близкие к левой границе, соответствуют плохим (разрозненным) кластеризациям; значения, близкие к середине, говорят о пересечении и наложении кластеров; значения, близкие к правой границе, соответствуют «плотным», четко выделенным кластерам. Согласно [2]: «чем больше силуэт, тем более четко выделены кластеры, и они представляют собой компактные, плотно сгруппированные облака точек».

Данная величина может быть использована для выбора оптимального числа кластеров, если оно заранее неизвестно: оно будет равно числу, максимизирующему значение силуэта. В отличие от рассмотренных ранее метрик, данная величина зависит от формы кластеров. Силуэт достигает больших значений при более выпуклых кластерах, которые получаются при помощи алгоритмов, основанных на восстановлении плотности распределения.

Типовые задачи при подготовке данных и обучении моделей

В разделе «Общий алгоритм машинного обучения» были представлены шаги разработки решения. Если задача типовая и не требует специального исследования или разработки специального метода, то существенная работа начинается с п. 3, который посвящен подготовке данных (обучающей и тестовой выборке). Исходные данные в подавляющем большинстве случаев требуют предварительной обработки перед тем как на них будет обучаться модель. Как правило, предварительная обработка подразумевает следующую работу: учет

пропусков, кодирование нечисловых данных, приведение данных к единому масштабу и стандартизация данных, разметка данных. Рассмотрим их подробнее.

Учет пропусков

В реальных задачах данные могут содержать пропуски. Это может быть вызвано тем, что клиенты не заполняли все поля в анкете или аккаунте, или тем, что не все параметры были оцифрованы за все время работы системы.

Таким образом, встает вопрос о том, как интерпретировать пропуски. Простейший вариант заключается в исключении объектов, имеющих неполные сведения (то есть удаление тех строк из матрицы признаков, в столбцах которых есть пропуски), или исключении признаков, содержащих неполные сведения (то есть удаления тех столбцов, в которых есть пропуски). Плюсы этого варианта в том, что он очень прост и его можно сразу опробовать на какой-либо модели. Минусы вполне очевидны. Если много объектов будут иметь пропуски, то мы рискуем удалить важные аспекты закономерности, которая сокрыта в данных, и, как следствие, получим низкое качество аппроксимации. Если мы удалим столбец, который содержит крайне важный признак, мы рискуем вовсе потерять ключевую информацию о разделении объектов.

Второй вариант борьбы с пропусками заключается в том, чтобы заменить их при помощи интерполяции. Это может быть среднее или медианное значение по столбцу. В случае если признак является функцией от времени, как и отдельные объекты, то можно интерполировать пропуски только по соседним временным значениям (например, если объект представляет собой вероятность покупки квартиры, а признак – среднюю заработную плату программиста в этот год, то за пропущенный год цифру можно приблизить по двум соседним).

Третий вариант можно также рассмотреть на примере с квартирой, который был описан выше. Если речь идет о каких-то открытых

экономических, социальных или других параметрах, то их можно найти в других источниках и тем самым дополнить данные.

Четвертый вариант заключается в том, чтобы закодировать пропуски специальным числовым значением (это может быть число, не встречающееся у других объектов) или категориальным значением (представление категориальных признаков будет рассмотрено ниже). Такой подход, как минимум, позволит внести информацию о том, что эти объекты по этому признаку отличаются от других (то есть мы не сообщим точной информации, но не удалим объекты полностью, как в первом подходе).

Пятый подход заключается в привлечении эксперта в соответствующей теме, который сможет сказать, как именно лучше всего интерполировать данные (на основе специфичных математических моделей, которые, вероятно, существуют или могут подойти). Например, для каких-то признаков можно сгенерировать псевдослучайные значения, подчиняющиеся некому распределению с учетом других известных признаков. Сюда же можно отнести генерацию синтетических данных (то есть искусственных) на основании собственного понимания предметной области. Долгий и тщательный анализ данных может прояснить поведение скрытого параметра, а также выявить способы, как сэмулировать его с определенной достоверностью. Однако этот подход опасен тем, что неверное понимание данных приведет к тому, что мы заложим в данные другие закономерности и потом их же и найдем при помощи методов машинного обучения. Иными словами, вместо решения исходной задачи мы решим искусственно созданную.

В сложных случаях на практике применяется некоторая комбинация всех вышеперечисленных подходов. Какие-то объекты или столбцы полностью исключаются, потому что их слишком сложно интерполировать или сэмулировать. Как правило, это объекты или признаки с большим числом пропусков (например, где пропусков больше, чем реальных значений). Какие-то признаки помечаются

особым кодом, какие-то данные находятся извне, а какие-то эмулируются синтетическими данными.

Многие нюансы зависят от специфики задачи. Но если никакие идеи не работают, то, вероятно, данных просто недостаточно и нужно получать еще непосредственно исходные данные (проводить физические эксперименты, опрашивать клиентов, измерять какие-то статистические метрики поведения пользователей на веб-сайтах и тому подобное).

Замечание: кроме пропусков данные могут содержать ошибки и выбросы (аномальные отклонения), вызванные неверным заполнением данных или ошибкой измерения. Решение этих проблем выходит за рамки данного учебного пособия и рекомендуется к самостоятельному изучению.

Кодирование нечисловых признаков

Очевидно, что далеко не все признаки объектов естественно описываются численным значением. Если говорить о размерах объекта или стоимости какого-то товара, то эти признаки, несомненно, будут числовыми. Если же речь идет о цвете, типе товара (категории) или вообще о текстовом описании некоторого объекта, то подобные признаки, как правило, поступают неоцифрованными.

Нечисловые признаки с неупорядоченными значениями (в которых между значениями не определена дистанция, то есть нельзя сказать, что больше или меньше) называют категориальными, или номинальными. Типичный подход к их обработке – кодирование категориального признака с m возможными значениями с помощью m бинарных признаков. Каждый бинарный признак соответствует одному из возможных значений категориального признака и является индикатором того, что на данном объекте он принимает данное значение. Такой подход иногда называют one-hot-кодированием. Например, у нас есть три варианта материала изделия: дерево, пластик, сталь. Причем мы не знаем наверняка, что лучше, а что хуже и как это материал влияет на итоговое качество товара. Тогда вместо

того, чтобы закодировать набор матриалов в один столбец как значения 1, 2, 3, one-hot-кодирование даст три столбца и следующие коды: 001, 010, 100. Заметьте, что это не двоичное кодирование.

Если речь идет не просто о категориях, а о целых предложениях или больших текстах на естественном языке, то задача существенно усложняется. Кодирование текста побуквенно в большинстве случаев ничего не даст, так как разнообразие слов и смыслов настолько велико, что на таком уровне абстракции модель не построит нужную функцию. Обработка и понимание текста – это во многом направление для исследований.

Тем не менее на сегодняшний день существует ряд подходов, которые позволяют решать реальные задачи. Первый вариант – отнестись к тексту как к категориям. Допустим, в текстовых данных всего встречается 10 тыс. уникальных слов (не считая союзов, предлогов и тому подобное). Тогда каждое отдельное текстовое описание (присущее конкретному объекту) есть такая категория, где встречаются соответствующие слова, то есть каждый текст будет кодироваться в вектор из 10 тыс. элементов, где на месте соответствующих слов-элементов будут стоять единицы, а на месте слов, которых нет в данном тексте – нули. В итоге получим довольно разреженную матрицу (состоящую в основном из нулей) и при этом довольно большую. Из-за этого возникает проблема ее хранения и обработки. На сегодняшний день существует ряд техник оптимизации работы с разреженными матрицами как на фундаментальном уровне (например, сингулярное разложение), так и на уровне библиотек (например, в пакете scipy).

Однако учитывать все слова часто бывает избыточно и даже бессмысленно. Поэтому на практике применяется подсчет статистических характеристик текста, таких как TF-IDF. Подробнее о статистическом анализе текстов можно прочитать в [3].

Сегодня также актуально кодирование текста методом word2vec на основе машинного обучения. Ключевой особенностью метода является то, что большой массив слов отображается в вещественные

векторы небольшой размерности (100-200 элементов), причем, похожие слова имеют близкие друг к другу векторы (то есть вводится дистанция между словами).

Приведение данных к единому масштабу и стандартизация

«Сырые» данные имеют разный масштаб и разное распределение по каждому признаку. Например, какой-то химический показатель смеси может иметь значения в диапазоне от 0.0001 до 0.2, а другой показатель от –100 до 100. Или, скажем, возраст клиентов может быть от 16 до 40, причем гораздо больше клиентов имеет возраст от 18 до 25, иными словами, математическое ожидание смещено относительно центра распределения. Подобные различия в признаках могут вносить существенную ошибку для множества моделей (например, для регрессии, нейронных сетей), и потому требуется привести все признаки к единому виду.

Существует некоторая путаница в терминах «стандартизация» и «нормализация». Очень часто под стандартизацией и нормализацией понимаются разные вещи, а иногда стандартизацию рассматривают как часть нормализации. Поэтому важно понять общую суть и цель этих методов.

Стандартизация данных – это процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

Нормализация данных – это процесс масштабирования вектора каждого признака, то есть приведение его к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

Так, стандартизация матрицы X:

[[1., –1.,2.],
[2., 0., 0.],
[0., 1., –1.]],

даст следующий результат:

[[0. –1.22 1.34]
[1.22   0.   –0.27]
[–1.22 1.22 –1.07]]

Видно, что значения вектора сместились (выровнялись относительного единого центра в нуле), а также произошло выравнивание разброса. Уже такого преобразования достаточно, чтобы повысить качество данных. Однако видно, что значения разных векторов не будут в одинаковом диапазоне (от –1 до 1, например). Они будут лишь иметь стандартный разброс в рамках вектора\столбца\признака.

Для того чтобы достичь одинакового масштаба всех векторов, необходима нормализация. Есть несколько видов норм и, соответственно, нормализации.

Самый очевидный и простой метод – это max норма. Чтобы все значения лежали в одном диапазоне, нужно найти максимальное из возможных значений и все остальные поделить на него. Таким образом, максимальное значение будет единицей, а все остальные лягут в диапазон от 0 до 1. Но это при условии, что нет отрицательных значений.

Менее очевидный способ – это L1 норма и нормализация. Формула следующая:

$$x'_i = \frac{x_i}{\|x\|_1} = \frac{x_i}{\sum_j |x_j|},$$

где $\|x\|_1$ и есть L1 норма, а вся формула целиком отображает процесс нормализации вектора x.

Еще один способ – это L2 норма. Формула нормализации вектора x:

$$x'_i = \frac{x_i}{\|x\|_2} = \frac{x_i}{\left(\sum_j x_j^2\right)^{1/2}},$$

где $\|x\|_2$ и есть L2 норма, а вся формула целиком отображает процесс нормализации вектора x.

Исследователи говорят, что лучшие результаты дают L2 и max нормализация, хотя любой вариант лучше, чем использование исходных данных.

По поводу плюсов-минусов этих способов можно сказать следующее: max нормализация не дает запас на неизвестные новые значения (то есть если заранее неизвестен весь диапазон данных, лучше не использовать эту норму), а L2 норма вычислительно дольше, но чаще всего дает оптимальный результат. L1 норма может дать слишком большой запас при большом разбросе данных, что может удалить нужную информацию из данных.

Разметка данных

Разметка данных – последняя из рассматриваемых нами (по порядку, но не по важности) операция с данными. Если речь идет о том, чтобы обучить модель прогнозировать будущие показатели по прошлому опыту (например, прогноз средней выручки по количеству посетителей магазинов), то такие данные, как правило, приходят с известной целевой переменной Y (то есть средней выручкой). Эти данные необходимо будет обработать в соответствии с пунктами выше, но их не потребуется дополнительно размечать. Однако не редкость, когда специалисту по анализу данных необходимо будет самостоятельно размечать выборку. Эта потребность может возникнуть из-за отсутствия размеченных исходных данных, так и может быть обусловлена экспериментами, возникающими в ходе решения общей задачи. Например, для распознавания визуальных образов машин или светофоров потребуется создать специальный файл, в котором будут проставлены ассоциации образов по имени файлов (то есть 001.png – «светофор»). В реальном бизнесе эту задачу может решать и не специалист по анализу данных или машинному обучению, а другие сотрудники (например, со стороны заказчика), но важно отметить, что этот этап может также возникнуть в ходе разработки решения, и его нельзя избежать в случае обучения с учителем.

Однако стоит отметить, что в случаях обучения и без учителя может потребоваться частичная разметка данных с целью тестирования и оценки качества модели, так как в противном случае мы просто получаем «черный ящик» без понимания того, как мы решили поставленную задачу.

Переобучение

Нами были рассмотрены типовые задачи по подготовке данных, теперь рассмотрим одну из основных проблем алгоритмов машинного обучения – переобучение.

Мы помним, что одной из важных характеристик алгоритмов машинного обучения является обобщающая способность. Однако с ней связаны еще два понятия: недообучения и переобучения. При подготовке данного пункта использовался материал (в том числе иллюстративный) из источника [4].

Недообучение возникает при обучении по прецедентам и характеризуется тем, что алгоритм не дает удовлетворительно малой средней ошибки на обучающем множестве. Как правило, это явление появлется вследствие использования недостаточно сложных моделей.

Противоположное этому явлению – переобучение. Его суть состоит в том, что вероятность ошибки натренированного алгоритма на объектах тренировочной выборки оказывается существенно меньше, чем на объектах тестовой. Чаще всего переобучение появляется из-за использования слишком сложных моделей.

Рассмотрим график, иллюстрирующий эффект переобучения (рисунок 5).

Точки на графике с рисунка 5 соответствуют разным способам обучения, и каждая из них получена усреднением по большому числу разбиений исходной выборки объемом в 72 образца на тестовую и обучающую части. Как видно из рисунка, точки имеют постоянное смещение вверх относительно диагонали графика. То есть наблюдается эффект переобучения: ошибки на тестовой выборке появляются чаще, чем на обучающей.

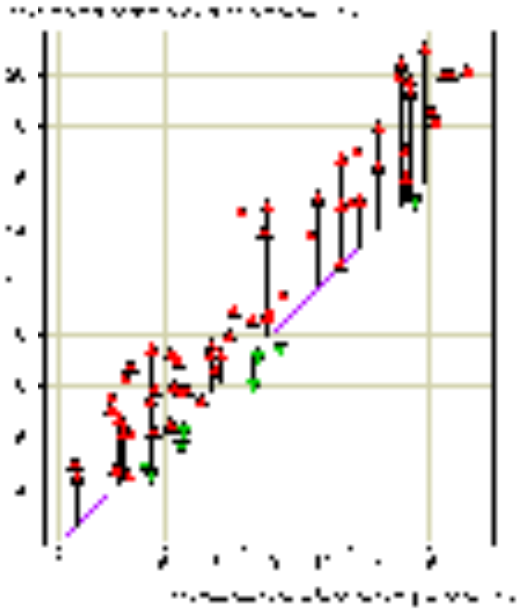


Рисунок 5. Иллюстрация переобучения

Чаще всего при построении алгоритмов обучения используется метод минимизации эмпирического риска (средней ошибки алгоритма на обучающей выборке). Его суть состоит в том, чтобы для текущей модели подобрать алгоритм, минимизирующий значение средней ошибки на данной обучающей выборке.

С переобучением метода минимизации эмпирического риска связаны три утверждения, объясняющие его причину:

1. Минимизация эмпирического риска не является гарантией малой вероятности ошибки на тестовых данных. Можно легко построить алгоритм, который минимизирует эмпирический риск до нуля, однако не будет способен к обучению. Суть состоит в том, что этот алгоритм запоминает обучающую выборку, потом сравнивает запомненный образец с предъявляемым. При совпадении предъявленного объекта и образца обучающей выборки алгоритм выдаст правильный ответ, иначе – выведется произвольный. То есть эмпирический риск равен нулю, однако обобщающей способности у алгоритма нет.

2. Согласно [4]: «Переобучение появляется именно вследствие минимизации эмпирического риска. Пусть задано конечное множество из D алгоритмов, которые допускают ошибки независимо и с одинаковой вероятностью. Число ошибок любого из этих

алгоритмов на заданной обучающей выборке подчиняется одному и тому же биномиальному распределению. Минимум эмпирического риска – это случайная величина, равная минимуму из D независимых одинаково распределенных биномиальных случайных величин, ожидаемое значение которой уменьшается с ростом D. Соответственно, с ростом D увеличивается переобученность – разность вероятности ошибки и частоты ошибок на обучении.

В данном модельном примере легко построить доверительный интервал переобученности, так как функция распределения минимума известна. Однако в реальной ситуации алгоритмы имеют различные вероятности ошибок, не являются независимыми, а множество алгоритмов, из которого выбирается лучший, может быть бесконечным. По этим причинам вывод количественных оценок переобученности является сложной задачей, которой занимается теория вычислительного обучения. До сих пор остается открытой проблема сильной завышенности верхних оценок вероятности переобучения».

3. Переобучение появляется в связи с избыточной сложностью модели. Всегда можно найти оптимальное значение сложности модели, при котором переобучение будет минимальным. Для примера приведем несколько графиков, на которых будет видна зависимость переобучения от сложности модели. При степени 2 полинома (рисунок 6) модель является недообученной. При степени 40 (рисунок 7) – переобученной и неустойчивой, а вот степень 20 (рисунок 8) – оптимальна.

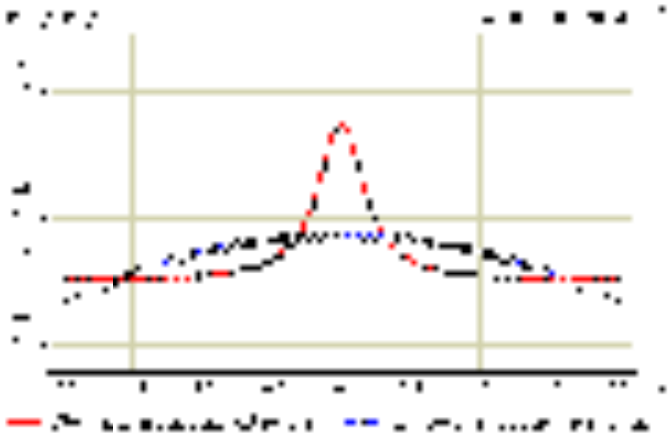


Рисунок 6. Недообученная модель

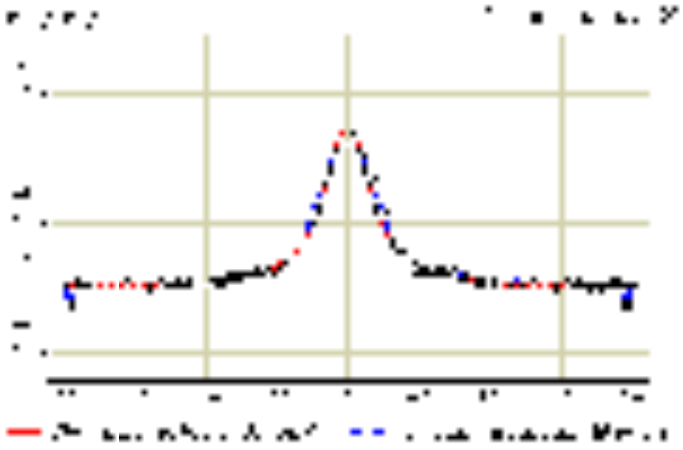


Рисунок 7. Оптимальная модель

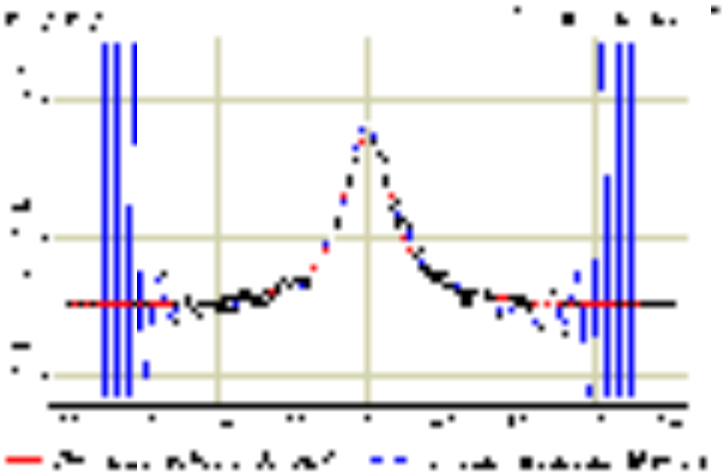


Рисунок 8. Переобученная модель

Одним из способов измерения вероятности переобучения является эмпирический метод Монте-Карло, или метод скользящего контроля. В литературе также встречаются названия кросс-проверка, или кросс-валидация. Суть его в следующем: производится некоторое количество разбиений исходной выборки на обучающую и контрольную. Для каждого разбиения происходит обучение алгоритма на обучающей подвыборке и оценка средней ошибки на контрольной. Затем вычисляется оценка скользящего среднего, как средняя по всем разбиениям величина вычисленной ошибки. Для независимой выборки этот показатель дает несмещенную оценку вероятности ошибки. Метод скользящего контроля является стандартным способом сравнения и оценивания алгоритмов классификации, регрессии, прогнозирования.

При использовании кросс-валидации можно сделать следующие выводы:

1. Если ошибка большая на большинстве участков, то скорее всего проблема в модели.

2. Если данные обучающей выборки характеризуются сильно смещенными математическим ожиданием и дисперсией, то уровень обобщения будет низким, что может быть связано с переобучением.

3. Если найдены сильные отклонения на определенных подвыборках, то, вероятно, проблема в этих участках данных или модель недообучается.

4. При малом объеме обучающей выборки кросс-валидация может стать способом борьбы с переобучением.

Если же говорить о прямых способах борьбы с переобучением, то можно выделить следующие:

1. Упрощение модели.

2. Подготовка большего числа обучающих данных (возможно, с помощью генерации).

3. Регуляризация.

Остановимся подробнее на последнем. Регуляризация представляет собой добавление некоторой дополнительной информации к условию минимизации ошибки. Выполняется это, чтобы решить некорректно поставленную задачу или предотвратить переобучение. Чаще всего добавляемая информация принимает вид штрафа за сложность модели. Например, введенные ограничения по норме векторного пространства или гладкости результирующей функции. Или же, с байесовской точки зрения, добавленные априорные распределения на параметры модели.

Согласно [5]: «Существуют следующие основные виды регуляризации:

L1-регуляризация (англ. Lasso regression):

$$\text{RSSR}(w) = \frac{1}{2}\sum_{i=1}^{n}(f(x_i, w) - y_i)^2 + \lambda\sum_{j=1}^{m}|w_j|,$$

где w – вектор весов полинома;

λ – коэффициент регуляризации.

L2, или Регуляризация Тихонова (в английской литературе – ridge regression или Tikhonov regularization), для интегральных уравнений позволяет балансировать между соответствием данным и маленькой нормой решения:

$$L(y, \hat{y}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \frac{\lambda}{2}\|w\|^2.$$

где $\|W\|^2$ – квадратичная норма вектора весов (сумма квадратов каждого веса).

Иными словами, переобучение в большинстве случаев проявляется в том, что в получающихся многочленах слишком большие коэффициенты. Соответственно и бороться с этим можно довольно естественным способом: нужно просто добавить в целевую функцию штраф, который бы наказывал модель за слишком большие коэффициенты».

По поводу применения той или иной регуляризации приведем материал источника [6]: «Регуляризацию можно применять с любым методом МО-классификации, который основан на математическом уравнении. Примеры включают линейную, логистическую регрессию и нейронные сети. Поскольку это уменьшает величину весовых значений в модели, регуляризацию иногда называют сокращением весов. Основное преимущество применения регуляризации в том, что оно часто приводит к созданию более точной модели. Главный недостаток заключается во введении дополнительного параметра, значение которого нужно определить, – весового значения регуляризации. В случае логистической регрессии это не слишком серьезно, так как в этом алгоритм обычно используется лишь параметр скорости обучения, но при использовании более сложного метода классификации, в частности нейронных сетей, добавление еще одного так называемого гиперпараметра может потребовать массы дополнительной работы для подбора комбинации значений двух параметров.

L1- и L2-регуляризация – процессы схожие. Какой же из них лучше? В принципе, исследователями сформулированы кое-какие теоретические правила насчет того, какая регуляризация лучше для определенных задач, но на практике придется поэкспериментировать, чтобы найти, какой тип регуляризации лучше в вашем случае и стоит ли вообще использовать какую-либо регуляризацию.

Применение L1-регуляризации иногда может давать полезный побочный эффект, вызывающий стремление одного или более весовых значений к 0.0, а это означает, что соответствующий признак не оказывает значимого влияния на результирующий, то есть включение его в модель требуется. Это одна из форм того, что называют «селекцией признаков». В отличие от L1, L2-регуляризация ограничивает весовые значения модели, но обычно не приводит к полному обнулению этих значений. Поэтому может показаться, что L1-регуляризация лучше L2-регуляризации. Однако недостаток применения L1-регуляризации в том, что этот метод не так-то просто использовать с некоторыми алгоритмами машинного обучения. Например, с теми, в которых используются численные методы для вычисления так называемого градиента. L2-регуляризацию можно использовать с любым типом алгоритма обучения.

Таким образом, можно сделать вывод, что L1-регуляризация иногда дает полезный побочный эффект удаления ненужных признаков, присваивая связанным с ними весам значение 0.0, но L1-регуляризация стабильно работает не со всеми формами обучения. L2-регуляризация работает со всеми формами обучения, но не обеспечивает неявной селекции функций. На практике же следует использовать метод проб и ошибок, чтобы определить, какая форма регуляризации (если она вообще нужна) лучше для конкретной задачи».

Рассмотренные методы борьбы с переобучением подойдут для регрессии или нейронных сетей. Для борьбы же с переобучением в моделях Деревьев Решений используют pruning (так называемое усечение) – удаление наименее информативных узлов для упрощения модели. Дело в том, что сразу нельзя построить оптимальное

(маленькое) дерево, так как в дереве малого размера будет мало информативных параметров, чтобы корректно обработать все входные образы (модель будет неполной, неинформативной). Поэтому сперва делают большое, но информативное дерево, а затем удаляют наименее информативные узлы и подтягивают дерево, уменьшая его размер\глубину.

Модели и алгоритмы машинного обучения

Как было сказано во введении, машинное обучение находится на стыке математической статистики, численных методов оптимизации, теории вероятностей и дискретного анализа. Эта дисциплина комбинирует и использует различные методы в построении математических моделей объектов и явлений для последующего их изучения и добычи знаний. В данном разделе предлагается рассмотреть, основные элементы математики, использующиеся в машинном обучении, а также изучить методы анализа данных, которые заложили основу рассматриваемой дисциплины.

Первое, что необходимо отметить, – любая дисциплина, использующая математический аппарат, так или иначе занимается математическим моделированием – заменой реального объекта его абстрактным, идеализированным представлением и использованием полученного представления для изучения объекта и добычи знаний. То есть основное, с чем работает машинное обучение, – это математические модели. Их можно разделить на различные типы по некоторым признакам. С точки зрения математического вида функции, составляющей модель, выделяют линейные и нелинейные модели. По количеству переменных в модели они делятся на сосредоточенные и распределенные. С точки зрения учета случайности модели делятся на детерминированные и стохастические, а с точки зрения изменчивости во времени – на статические и динамические. По природе используемых в модели параметров и переменных их можно разделить на дискретные и непрерывные.

В общем виде математическая модель может быть представлена следующим образом:

$$Y=F(X,W_c,W_d,W_s),$$

где Y – выходные данные; X – входные данные; F – некоторая функция; W_c – константные параметры модели; W_d – динамические параметры модели; W_s – статические параметры модели.

При построении моделей элементы множеств *W*, как правило, заранее неизвестны и требуют нахождения. Есть два способа решения этой задачи: аналитический и численный. Первый предполагает нахождение корней, второй – приближение результата к некоторому удовлетворительному значению. Удовлетворительность в данном случае оценивается некоторым заранее определенным критерием. При этом для решения задачи численным методом нередко происходит ее переформулировка для определения и введения условия, определяющего качество решения. Необходимо отметить, что численные методы чаще всего опираются на отклонение получаемого результата от желаемого и стремятся минимизировать это отклонение. Причем данный подход будет работать, даже если неизвестна точная формула для получения выходного значения, однако обязательно существует способ измерения отклонения, а также можно найти значения, которые он должен принимать.

Появление численных методов оптимизации было обусловлено, во-первых, тем, что не все задачи можно решить аналитически. Классический пример подобной задачи – гравитационная задача *N* тел. Согласно Википедии [1], ее формулировка звучит так: «В пустоте находится *N* материальных точек, массы которых известны $\{m_i\}$. Пусть попарное взаимодействие точек подчинено закону тяготения Ньютона, и пусть силы гравитации аддитивны. Пусть известны начальные на момент времени $t=0$ положения и скорости каждой точки $\mathbf{r}_i|_{t=0} = \mathbf{r}_{i0}$, $\mathbf{v}_i|_{t=0} = \mathbf{v}_{i0}$. Требуется найти положения точек для всех последующих моментов времени». И согласно тому же источнику [1]: «На данный момент в общем виде задача *N* тел для $N>3$ может быть решена только численно, причем для $N=3$ ряды Зундмана даже при современном уровне компьютеров использовать практически невозможно».

Во-вторых, даже возможные аналитические решения систем могут обладать существенной сложностью. Матричные решения имеют вычислительную сложность *O(N^3)*, тогда как численные – линейную.

В-третьих, при противоречивости или мультиколлинеарности данных аналитические решения могут давать неопределенный результат. В отличие от них численные методы всегда сходятся пусть и к локальному, но все же определенному решению.

В основном, при решении задач минимизации ошибки, в машинном обучении используются градиентные методы, составляющие особый класс алгоритмов оптимизации. Градиент в данном случае – это (согласно Википедии [1]): «вектор, своим направлением указывающий направление наибольшего возрастания некоторой величины φ, значение которой меняется от одной точки пространства к другой (скалярного поля), а по величине (модулю) равный скорости роста этой величины в этом направлении.

Например, если взять в качестве φ высоту поверхности земли над уровнем моря, то ее градиент в каждой точке поверхности будет показывать «направление самого крутого подъема», и своей величиной характеризовать крутизну склона.

С математической точки зрения на градиент можно смотреть как на:

1. Коэффициент линейности изменения значения функции многих переменных от изменения значения аргумента.

2. Вектор в пространстве области определения скалярной функции многих переменных, составленный из частных производных.

3. Содержимое Матрицы Якоби. Ее строки содержат градиенты составных скалярных функций, из которых состоит векторная функция многих переменных.

Пространство, на котором определена функция и ее градиент, может быть как обычным трехмерным пространством, так и пространством любой другой размерности, любой физической природы или чисто абстрактным (безразмерным)».

Метод градиентного спуска – нахождение локального экстремума (минимума или максимума) функции путем движения вдоль градиента в направлении наискорейшего спуска, задаваемого антиградиентом. Существуют следующие типы градиентного спуска:

1. С постоянным шагом.
2. С дроблением шага.
3. Наискорейшего спуска.
4. Стохастический.

Более подробно о градиентном спуске мы поговорим далее, здесь же рассмотрим аналитические методы, используемые в машинном обучении, так как именно они составляют первое поколение алгоритмов, применяемых для анализа данных.

Методы теории вероятностей

Первый срез методов, который необходимо рассмотреть, – методы теории вероятностей. Теория вероятностей – раздел математики, в котором изучаются случайные величины и события, их свойства и возможные операции над ними. Таким образом, ключевое понятие, лежащее в основе этой дисциплины, – вероятность события P_i. Опираясь на него, можно дать определение полной группы событий. Она определяется как система случайных событий, которая обладает следующими свойствами:

1. В результате случайного эксперимента непременно произойдет одно и только одно из составляющих ее событий.
2. Сумма вероятностей всех событий полной группы равна 1.

Достаточно часто при исследовании данных (выборок) и подсчете определенных характеристик выборки считаются полными группами событий.

С помощью методов теории вероятностей можно проводить как простые, так и сложные операции по анализу данных. Среди простых можно выделить подсчеты вероятностных характеристик выборки: медианы, математического ожидания, дисперсии, а также величины среднеквадратического отклонения. Рассмотрим их подробнее.

Медиана – число, характеризующее выборку по среднему из ее значений. То есть, если все данные выборки различны, и она упорядочена по возрастанию, то ровно половина из элементов выборки

будет меньше медианы, и ровно половина – больше. Формально величину можно выразить следующим образом:

Если $X=\{x_{1\ldots}x_n\}$ – характеризуемая выборка, то x_k – медиана, если $x_j<x_k$, при $j \in [1,k)$ и $x_k<x_i$, при $i \in (k,n]$ и $k=n/2$.

Рассмотрим пример. Пусть выборка X = {1, 2, 3, 4, 5, 6, 7, 8, 9}, тогда ее медианой будет число 5.

Математическое ожидание – среднее значение вероятностных элементов выборки. Формально она рассчитывается так:

$$M|X|=\sum x_i p_i \ .$$

Рассмотрим пример. Пусть выборка X = {1, 2, 3, 4, 5, 6, 7, 8, 9}, тогда ее математическим ожиданием будет величина, равная 5. Вычисляется она следующим образом: выборка рассматривается как полная группа событий с равными вероятностями. То есть p_i для каждого элемента равно 0,11. Тогда 1×0,11+2×0,11+3×0,11…=5.

Дисперсия – мера разброса элементов выборки относительно ее математического ожидания. Рассчитывается данная величина по формуле

$$D|X|=\sum(x_i - M|X|)^2 p_i \ .$$

При этом p_i рассчитывается как $1/(n-1)$, где n – количество элементов выборки. Это выполняется для того, чтобы не учитывать отклонение самого математического ожидания от себя. То есть, для выборки X = {1, 2, 3, 4, 5, 6, 7, 8, 9} дисперсия будет рассчитываться следующим образом:

(1–5) ×(1–5) ×0,125+(2–5) ×(2–5) ×0,125+(3–5) ×(3–5) ×0,125…=7,5.

Среднеквадратическое отклонение – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания. Формула расчета

$$\sigma = \sqrt{\frac{1}{n}\sum(x_i - \bar{x})^2},$$

где n – количество элементов в выборке; $\bar{x}$ – математическое ожидание.

Интерпретировать данную величину можно следующим образом: чем больше значение среднеквадратического отклонения, тем

больший разброс значений в представленном множестве (относительно его средней величины). Малое значение среднеквадратического отклонения говорит о том, что элементы выборки сгруппированы вокруг ее среднего значения. Если говорить о более практическом применении величины, то в экономике, например, она характеризует доходность портфеля и его риск.

Однако рассмотренные показатели характеризуют случайную величину лишь с какой-то одной стороны. Наиболее же полно и исчерпывающе ее описывает закон распределения – функция, определяющая для выборки X вероятность попадания в некоторый интервал или вероятность получения определенного значения x_i. Если какой-либо закон распределения описывает случайную величину, то говорят, что она ему подчиняется или по нему распределена. Иными словами, закон распределения описывает область значений случайной величины и вероятности их получения.

Среди законов распределения наиболее часто используется закон нормального распределения или распределения Гаусса, который характеризует большинство процессов, встречающихся в мире. Отсюда и произошло его название (нормальное). Закон (плотность распределения вероятности) описывается следующей формулой:

$$\boldsymbol{f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}},$$

где σ – среднеквадратическое отклонение; μ – математическое ожидание.

С математической точки зрения можно заметить, что данная функция зависит от двух параметров: математического ожидания и среднеквадратического отклонения, поэтому по сути данный закон представляет собой семейство распределений. В качестве примера приведем изображение из источника [7], показывающее графики нормальных распределений с разными параметрами (рисунок 9).

Обратите внимание!

- *График с параметрами σ2=1 и μ=0 соотвествует стандартному нормальному распределению.*

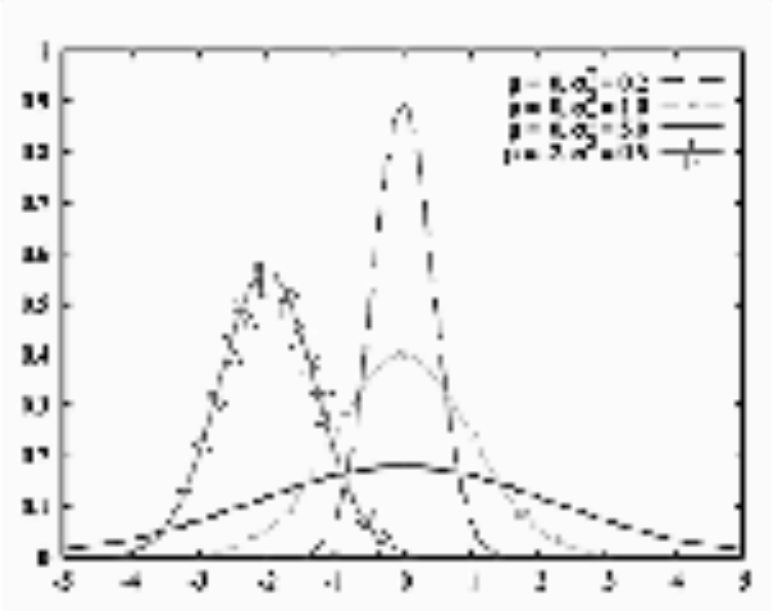


Рисунок 9. Плотность нормального распределения

Согласно Википедии [1]: «Важное значение нормального распределения во многих областях науки (например, в математической статистике и статистической физике) вытекает из центральной предельной теоремы теории вероятностей. Если результат наблюдения является суммой многих случайных слабо взаимозависимых величин, каждая из которых вносит малый вклад относительно общей суммы, то при увеличении числа слагаемых распределение центрированного и нормированного результата стремится к нормальному. Этот закон теории вероятностей имеет следствием широкое распространение нормального распределения, что и стало одной из причин его наименования».

Кроме описанных выше величин и характеристик для анализа данных очень часто используется одна из основных теорем теории вероятностей – теорема Байеса.

Она позволяет определить вероятность наступления какого-либо события при условии, что произошло другое событие, статистически

взаимосвязанное с исследуемым. Ключевыми понятиями в данной теореме являются понятия априорной и апостериорной вероятностей. Рассмотрим их подробнее.

Априорная вероятность – назначенная событию вероятность, при условии отсутствия знаний, поддерживающих его наступление. Апостериорная вероятность – назначенная событию вероятность при условии наличия знаний, поддерживающих его наступление и полученных опытным путем.

Теперь перейдем непосредственно к теореме Байеса. Формулируется она следующим образом:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

где $P(A|B)$ – апостериорная вероятность справедливости гипотезы A при наступлении B; $P(B|A)$ – вероятность наступления события B при истинности гипотезы A; $P(A)$ – априорная вероятность гипотезы A; $P(B)$ – вероятность наступления события B.

Метод Байеса имеет свои достоинства и недостатки. К первому можно отнести возможность использования экспертных знаний, возможность точного описания явления и хорошую интерпретируемость результатов. К недостаткам же относится следующее:

1. Метод не ставит целью минимизацию ошибки классификации.
2. Метод требует работы эксперта.
3. Сильная зависимость результатов от выбора модели.
4. Плохая работа при малом количестве и высокой размерности данных.
5. Метод дает плохое обобщение, особенно на высокоуровневых признаках.
6. Метод дает плохие результаты при взаимозависимости признаков.

Таким образом, можно сделать вывод, что данный метод хорошо работает в тех случаях, где признаки и результат сильно завязаны на их частотных характеристиках.

Деревья решений

Следующий метод анализа данных, который нам необходимо рассмотреть, – применение деревьев решений. Дерево решений – средство поддержки принятия решений для прогнозных моделей. Суть его работы заключается в последовательном разбиении множества данных на непересекающиеся классы, которые в свою очередь также подвергаются разбиению по каким-либо критериям с оценкой эффективности разбиения. Как правило, дерево решений состоит из «узлов», «листьев» и «веток». «Ветки» содержат записи атрибутов, от которых зависит целевая функция, «листья» – значения целевой функции, а «узлы» – остальные атрибуты, по которым происходит классификация. Чаще всего выделяют два типа деревьев: для классификации (в этом случае предсказываемый результат – класс, которому принадлежат данные) и для регрессии (результат – прогнозируемое значение целевой функции).

Обобщенный алгоритм построения дерева решений по обучающей выборке состоит из следующих шагов:

1. Берем следующий атрибут и помещаем его в корень.
2. Для всех значений этого атрибута – оставляем в «листьях» данной «ветки» только те значения, которые соответствуют определенному условию.
3. Продолжаем строить дерево среди оставленных на предыдущем шаге «листьев».

Для выбора следующего атрибута может быть использован один из следующих основных алгоритмов:

1. ID3. Атрибут выбирается на основе прироста информации и минимизации энтропии. Напомним, что под энтропией в данном случае подразумевается мера неупорядоченности системы. Чем меньше ее величина, тем более упорядочены составляющие системы.
2. C4.5 – улучшенная версия предыдущего алгоритма, в которой используется нормализованный прирост информации.

3. CART – алгоритм, используемый для построения бинарных деревьев, производящий разбиение на основе модальных значений признаков.

Рассмотрим работу по построению дерева решений с использованием первого алгоритма. Более подробно об этом можно прочитать в [8, 9].

Кратко алгоритм ID3 может быть описан тремя шагами:

1. Посчитать энтропию разбиваемого множества.
2. Выбрать признак с минимальной энтропией и, соответственно, максимальной информационной выгодой.
3. На основе полученного признака создать узел дерева и повторить процедуру.

Рассмотрим построения дерева решений для классификации следующего множества из семи объектов: {красный круг, зеленый квадрат, красный квадрат, зеленый треугольник, зеленый круг, красный треугольник, красный прямоугольник}. Как мы видим, каждый из объектов характеризуется двумя признаками: цвет и форма, то есть именно по ним мы будем проводить разбиение. Для меры энтропии выберем формулу энтропии Шеннона:

$$H = -\sum_{i=1}^{N} p_i \times \log p_i$$

В таблице 2 представлены расчеты энтропии всей системы. Подсчеты энтропии для каждого из признаков приведены в таблицах 3 и 4, соответственно.

Таблица 2. Подсчет энтропии системы

Объект	**pi**	**log(pi)**	**pi*log(pi)**
красный квадрат	0,142857	-0,8451	-0,12073
красный прямоугольник	0,142857	-0,8451	-0,12073
красный круг	0,142857	-0,8451	-0,12073
зеленый квадрат	0,142857	-0,8451	-0,12073
зеленый треугольник	0,142857	-0,8451	-0,12073
зеленый круг	0,142857	-0,8451	-0,12073
красный треугольник	0,142857	-0,8451	-0,12073
Энтропия			0,845098

Таблица 3. Подсчет энтропии для признака «Цвет»

Объект	pi	log(pi)	pi*log(pi)
красный	0,571429	-0,24304	-0,13888
зеленый	0,428571	-0,36798	-0,1577
Энтропия			0,296583

Таблица 4. Подсчет энтропии для признака «Форма»

Объект	pi	log(pi)	pi*log(pi)
круг	0,285714	-0,54407	-0,15545
квадрат	0,285714	-0,54407	-0,15545
треугольник	0,285714	-0,54407	-0,15545
прямоугольник	0,142857	-0,8451	-0,12073
Энтропия			0,587072

Как мы видим, первым признаком, на основе которого будет происходить разбиение, станет цвет, а вторым – форма. Классификация исходного множества с помощью построенного дерева показана на рисунке 10.

Рисунок 10. Классификация фигур с помощью дерева

Стоит отметить, что чаще всего деревья решений – бинарные. В узлах они содержат условия, а ветви соответствуют истинности или ложности этого условия. Поэтому, если дерево с рисунка 10 преобразовать в классическое дерево решений, то получится изображение, представленное на рисунке 11.

Теперь рассмотрим использование этого метода для решения задачи из источника [10]: спрогнозируем исход матча для футбольной команды исходя из следующих параметров:

- выше ли находится соперник по турнирной таблице;
- дома ли играется матч;
- пропускает ли матч кто-либо из лидеров команды;
- идет ли дождь.

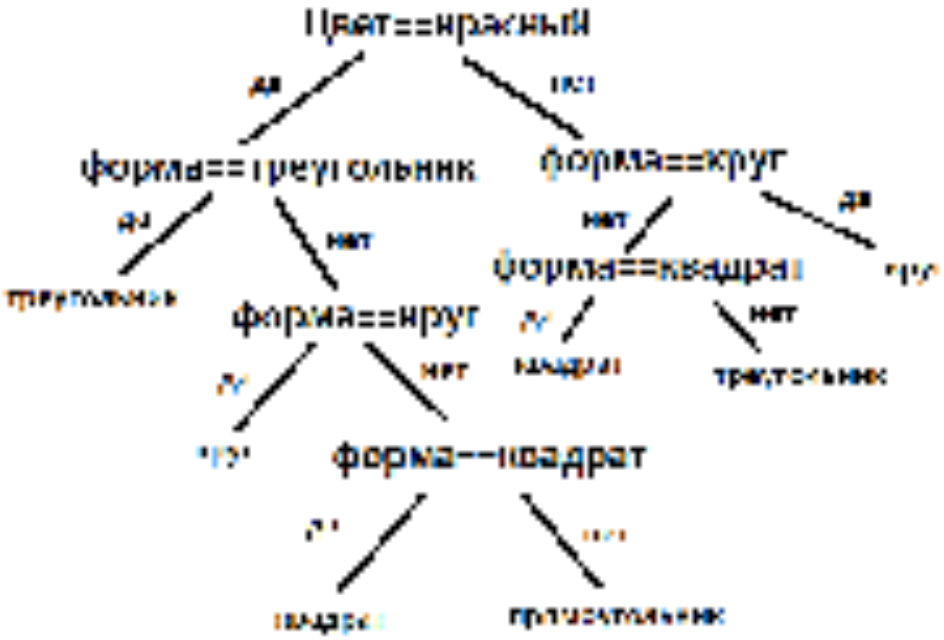


Рисунок 11. Классическое дерево решений

Прогноз предлагается выполнить на основе статистики, представленной в таблице 5.

Таблица 5. Статистические данные для задачи

Соперник	**Играем**	**Лидеры**	**Дождь**	**Победа**
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	???

Первое, что мы рассчитываем, – энтропию для признаков. Результаты можно увидеть в таблице 6, причем данные приведены в округленном виде.

Таблица 6. Энтропия для признаков

Соперник	Играем	Лидеры	Дождь	Победа
0,29	0,26	0,29	0,29	0,29

Согласно таблице 6, первый признак, по которому будет происходить деление, – место игры. По нашим данным получатся два множества: со значением «дома» и со значением «в гостях». Если мы рассмотрим второе множество, то увидим, что следующий целевой признак с нулевой энтропией – исход матча: все элементы множества «в гостях» имеют значение поля победа «нет». Следовательно, по нашим статистическим данным с помощью деревьев решений мы прогнозируем поражение.

Если говорить о достоинствах деревьев решений, то можно выделить следующие. Во-первых, простота понимания и интерпретации. Во-вторых, минимальные требования к подготовке данных, а также способность работы с большими объемами данных. В-третьих, метод одинаково хорошо работает с разными видами признаков. В-четвертых, является надежным методом и позволяет оценить модель статистическими тестами. К недостаткам же можно отнести следующее:

1. Достаточно сложно построить оптимальное дерево решений.
2. Подверженность переобучению.
3. Не для всех задач может быть получено решение удовлетворительного качества.

Статистические модели и методы

Мы рассмотрели методы машинного обучения, предоставляемые теорией вероятностей, а также модель дерева решений. Теперь обратимся к статистике и кратко опишем ее модели и методы, которые могут быть использованы для решения задач машинного обучения, а именно методы регрессионного анализа, подробно рассмотренные в следующих источниках: [11,12,13].

В контексте машинного обучения под регрессионным анализом понимается процесс построения математической модели, описывающей зависимость некоторой целевой характеристики объекта или процесса от других его характеристик. Например, зависимость числа новых клиентов от величины зарплаты работающего на улице промоутера.

В задаче регрессионного анализа всегда есть обучающая выборка, состоящая из входных параметров и откликов, а также начальная параметрическая модель, в самом простом случае – линейная, однако не обязательно таковая. Для задачи из примера эта модель может иметь вид $y=\beta_0+\beta_1\times x$, где x – размер зарплаты промоутера; y – количество новых клиентов; β_0 и β_1 – параметры модели. Задача регрессии – оценить их, то есть найти такие значения β_0 и β_1, чтобы полученная модель отражала зависимость между входом и выходом с требуемой точностью.

После получения адекватной модели мы можем решать задачу прогнозирования, подставляя в полученную формулу величину x и вычисляя величину y (с удовлетворяющей нас погрешностью). Приведенный выше пример модели – модель парной линейной регрессии, но помимо нее существует и множественная, и нелинейная регрессии. Начнем рассмотрение с самого простого.

Допустим, мы хотим описать зависимость между двумя факторами моделью вида $y=\beta_0+\beta_1\times x$. Первое, что необходимо учесть, – построенная линия никогда не будет точно проходить по опытным точкам, поэтому истинный вид регрессионной модели будет $y=\beta_0+\beta_1\times x+e$, где e – ошибки наблюдений. Второе – прежде чем переходить к оцениванию параметров модели, целесообразно построить диаграмму рассеяния, чтобы убедиться, что выбранная модель действительно может описать зависимость между факторами. На диаграмме рассеяния каждой паре «зависимый-влияющий параметр» соответствует точка на плоскости. Как правило, зависимый фактор откладывается по оси ординат, а второй – по оси абсцисс. Модель парной линейной регрессии графически представляет собой линию, следова-

тельно, использовать ее для описания зависимости целесообразно, если на диаграмме рассеяния точки располагаются вокруг (и достаточно близко) какой-либо прямой. Отклонение реальных точек от модельных – остатки или ошибки наблюдений, которые обозначаются *e*. Пример диаграммы рассеяния показан на рисунке 12.

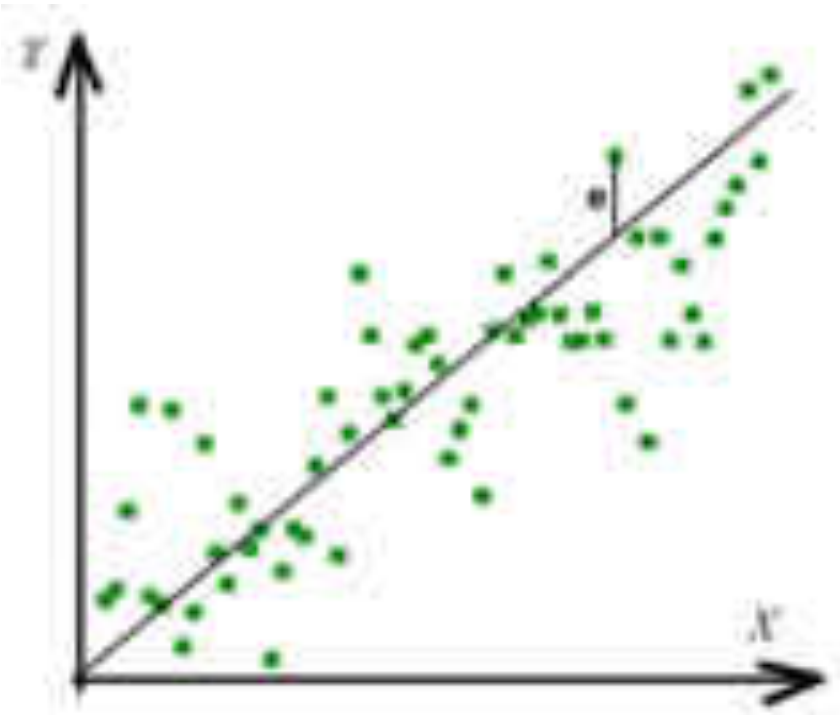


Рисунок 12. Диаграмма рассеяния

Если после анализа диаграммы рассеяния принимается решение использовать линейную парную регрессию для моделирования зависимости, то следующим шагом будет нахождение параметров модели β_0 и β_1. Для решения этой задачи в девяносто девяти процентах случаев используется метод наименьших квадратов, предложенный Гауссом более двухсот лет назад. Суть данного метода заключается в минимизации суммы квадратов отклонений опытных данных от модельных. Формально эта задача описывается так:

$$Q = \sum e_i^2 \rightarrow min$$

В данном случае e_i вычисляется следующим образом:

$$e_i = \beta_0 + \beta_1 * x_i - y_i,$$

где y_i – реальное выходное значение для входного значения x_i.

Чтобы решить указанную задачу оптимизации, решают систему уравнений:

$$\left.\begin{aligned}\frac{\partial Q}{\partial \beta_0} &= 0\\ \frac{\partial Q}{\partial \beta_1} &= 0\end{aligned}\right\}$$

Найдя необходимые частные производные и выполнив преобразования по упрощению выражений, получают нормальную систему уравнений для парной линейной регрессии:

$$\left.\begin{aligned} n * \beta_0 + \beta_1 * \sum x_i - \sum y_i &= 0\\ \beta_0 * \sum x_i + \beta_1 * \sum x_i^2 - \sum x_i y_i &= 0\end{aligned}\right\}$$

Рассмотрим для примера построение парной линейной регрессии для задачи зависимости ежемесячного количества новых клиентов от почасовой зарплаты промоутера по данным, представленным в таблице 7.

Таблица 7. Данные для анализа

Зарплата (*x*)	100	120	130	150
Количество клиентов (*y*)	70	100	120	140

Подставим данные в систему уравнений и получим следующий ее вид:

$$\left.\begin{aligned} 4 \times \beta_0 + \beta_1 \times 500 - 430 &= 0\\ \beta_0 \times 500 + \beta_1 \times 63800 - 55600 &= 0\end{aligned}\right\}$$

Выразим β_0 из первого уравнения:

$$\beta_0 = 107,5 - \beta_1 \times 125$$

Подставим во второе и найдем β_1:

$$53750 - 62500 \times \beta_1 + \beta_1 \times 63800 - 55600 = 0$$

$$\beta_1 = \frac{1850}{1300}.$$

В итоге получим уравнение регрессии:

$$y = 1,423\,1x - 70,385\ .$$

Полученные коэффициенты можно интерпретировать следующим образом. β_0 – значение при x=0. То есть, если у нас не будет работать промоутер (то есть мы не будем платить ему зарплату), ежемесячно мы прогнозируем отток 70 клиентов. Содержательная интерпретация второго коэффициента следующая: каждый рубль в зарплате промоутера дает 1,4 нового клиента в месяц.

Необходимо отметить, что коэффициенты для парной линейной регрессии можно найти средствами табличного процессора Excel. Для этого строится обычная точечная диаграмма, а затем отображается линия тренда и уравнение, как показано на рисунке 13.

Рисунок 13. Работа с парной линейной регрессией в Excel

Там же можно отобразить величину достоверности аппроксимации (или коэффициент детерминации R^2), которая показывает, насколько модельные значения близки к реальным. Чем ближе эта величина к 1, тем лучше модель описывает реальность. То есть, по сути, это критерий качества модели. Рассчитывается он по формуле

$$R^2 = \frac{\sum \tilde{y}_i^2 - n\bar{y}^2}{\sum y_i^2 - n\bar{y}^2},$$

где $\bar{y}=\frac{\sum y_i}{n}$ – среднее значение реальных результатов, $\tilde{y}$ – прогнозное значение.

Областью значения данной величины будет отрезок от 0 до 1.

Кроме расчета величины достоверности аппроксимации есть еще несколько способов оценить качество регрессионной модели. Однако в данном пособии не будем подробно останавливаться на этих методах, а лишь перечислим их:

1. Анализ диаграммы рассеяния;
2. Проверка статистических гипотез о значимости модели;
3. Анализ остатков.

По третьему пункту поясним: метод наименьших квадратов справедлив, если остатки обладают следующими свойствами:

1. Нормальным распределением.
2. Взаимонезависимостью.
3. Нулевым математическим ожиданием.
4. Постоянной дисперсией.

То есть, если остатки разбросаны хаотично, то метод наименьших квадратов можно использовать для решения поставленной задачи.

Если не хаотично, то нужно смотреть: если остатки растут при росте *x*, то есть их график напоминает диаграмму рассеяния, показанную на рисунке 12, это говорит о гетероскедастичности остатков. В данном случае необходимо использовать взвешенный метод наименьших квадратов. Если график остатков похож на отрезок параболы, то скорее всего была выбрана неправильная модель и необходимо выбрать другую. Если график остатков напоминает, например, синусоиду, то есть они циклически колеблются вверх-вниз, то это свидетельствует об автокорреляции остатков, которая может быть оценена количественно с помощью критерия Дарбина-Уотсона. В этом случае метод наименьших квадратов неприменим, и необходимо использовать другие подходы.

Как было сказано выше, парная линейная регрессия не всегда хорошо описывает данные. Тогда зависимость можно попробовать смоделировать либо кусочно-линейной моделью, то есть построить несколько разных линейных моделей для разных диапазонов значений величины *x*, либо использовать нелинейные модели: гиперболическую, параболическую, степенную и обратную.

В общем случае при втором подходе оптимизационная задача формулируется так:

$$Q = \sum(f(x_i) - y_i)^2 \to min,$$

где $f(x_i)$ – некоторая нелинейная функция.

Для ее решения можно воспользоваться, например, методом Ньютона-Рафсона. Но в некоторых случаях при работе с парной нелинейной регрессией используют прием преобразования модели к линейному виду.

Рассмотрим способ преобразования более подробно. Например, мы выбрали гиперболическую модель, задаваемую уравнением вида

$$y = \beta_0 + \beta_1/x\ .$$

Тогда, если мы введем замену z=1/*x*, то модель преобразуется к привычному нам линейному виду.

Для параболической модели вида

$$y = \beta_0 + \beta_1 \times x + \beta_2 * x^2$$

замена не выполняется. При ее решении используется подход, аналогичный случаю линейной модели, но в итоговой системе будет три уравнения.

Обратная модель вида

$$y = 1/(\beta_0 + \beta_1 * x)$$

приводится к линейной через замену z=1/y.

Степенная модель вида

$$y = \beta_0 \times x^{\beta_1}$$

приводится к линейной логарифмированием:

$$\ln(y) = \ln(\beta_0) + \beta_1 \ln(x).$$

Вводя замены, получаем уравнение:

$$z = \beta_0{}' + \beta_1 \times t.$$

Однако, выполняя подобные преобразования, мы рискуем исказить взаимосвязи и понизить адекватность модели, поэтому линеризацию нужно применять крайне аккуратно. К тому же данный подход работает не для всех моделей. В таком случае, как было сказано выше, требуется использовать численный метод Ньютона-Рафсона. Опишем кратко его идею, не вдаваясь в подробности. Поиск решения в нем происходит посредствам построения последовательных приближений, то есть на основе простой итерации. Первым шагом задается стартовое приближение около предполагаемого корня. Затем в точке приближения строится касательная к графику функции, для которой находится пересечение с осью абсцисс. Эта точка считается следующим приближением, для которого повторяется процесс. Алгоритм продолжает свою работу до тех пор, пока не будет достигнута необходимая точность.

Мы рассмотрели различные модели парной регрессии, то есть случаи, когда отклик зависит от одного фактора. Однако в некоторых задачах на выходную переменную (отклик) влияет несколько факторов. В таком случае эта зависимость описывается множественной регрессией. Чаще всего в этом случае используются линейные модели. Общий вид зависимости описывается следующим образом:

$$y_i = \beta_0 + \beta_1 \times x_{1i} + \cdots + \beta_k \times x_{ki} + \varepsilon_i,$$

где *i=1…n, n* – количество наблюдений; *k* – количество факторов.

При работе с множественной регрессией, как правило, переходят к матричной записи системы:

$$Y=X\times\beta+\varepsilon,$$

где $Y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$ – вектор откликов; $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_n \end{pmatrix}$ – вектор ошибок;

$\beta = \begin{pmatrix} \beta_1 \\ \dots \\ \beta_n \end{pmatrix}$ – вектор параметров; $X = \begin{pmatrix} 1 & x_{11} \dots & x_{k1} \\ 1 & x_{12} \dots & x_{k2} \\ \dots & \dots \dots & \dots \\ 1 & x_{1n} \dots & x_{kn} \end{pmatrix}$ – регрессионная матрица; *n* – количество наблюдений; *k* – количество факторов.

В данном случае задача формулируется следующим образом: для известных X и Y необходимо найти такие β, чтобы $Q=\sum \varepsilon_i^2 \rightarrow \min$.

В матричном виде последнее выражение можно переписать так:

$$Q = \varepsilon^T \varepsilon.$$

При этом

$$\varepsilon = Y - X\beta.$$

Тогда аналогом нормальной системы уравнений парной линейной регрессии в нашем случае будет следующее выражение:

$$\tilde{\beta} = (X^T X)^{-1} X^T Y.$$

Однако при работе с этой формулой могут возникнуть проблемы, связанные с тем, что обратную матрицу $(X^T X)^{-1}$ сложно посчитать. Как правило, это происходит, если ее столбцы сильно коррелированы. Например, если столбцы практически линейно зависят один от другого (имеет место мультиколлинеарность). Если коэффициент корреляции столбцов будет больше 0,8, то обратную матрицу уже не посчитать. Для решения этой проблемы либо исключают один из взаимозависимых факторов, либо прибегают к использованию гребневой регрессии (Ridge regression). Ее суть состоит в том, что для решения проблемы плохой обусловленности матрицы $(X^T X)$ к ней добавляется некое число λ. То есть обращается матрица $(X^T X + \lambda I)$. Полученные с помощью гребневой регрессии оценки параметров являются смещенными (в отличие от МНК-оценок), но доказано, что существует такое λ при котором оценки гребневой регрессии более эффективны, чем МНК-оценки. Однако четких рекомендация относительно выбора числа λ нет.

Продолжая тему проблем множественной регрессии, необходимо отметить еще одну. В случае непарной регрессии диаграмма рассеяния не применима для отслеживания выбросов, так как пространство будет многомерно. Для решения этой проблемы используют робастные методы. Основную информацию о них можно найти в источнике [14].

Если же говорить об общем алгоритме работы с множественной регрессией, то можно выделить следующие шаги:

1. Подготовка исходных данных, как правило, в виде таблицы (см. таблицу 8):

Таблица 8. Шаблон подготовки данных

№ опыта	***y***	x_1	x_2	**…**	x_k
1					
…					
n					

2. Оценивание параметров модели по формуле

$$\tilde{\beta} = (X^T X)^{-1} X^T Y,$$

или при использовании гребневой регрессии – по формуле

$$\tilde{\beta} = (X^T X + \lambda I)^{-1} X^T Y).$$

3. Проверка значимости модели с использованием критерия Фишера. Если по критерию Фишера линейная модель множественной регрессии оказалась незначима, то можно перейти к неполной квадратичной модели и проверить ее. Для двух факторов данная модель записывается следующим образом:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 .$$

Далее производят замену $\beta_3 = \beta_{12}$ и $x_3 = x_1 x_2$ и проверяют значимость получившейся модели. Если и она не значима, то переходят к полной квадратичной модели, затем (при ее незначимости) – к неполной кубической и полной кубической. Если же и последняя оказывается незначимой, то переходят к степенной, но здесь возрастает риск возникновения мультиколлинеарности. В принципе, даже кубическая модель уже дает высокую мультиколлинеарность. В случае, когда не удается найти ни одной значимой модели, признается, что регрессионными методами поставленную задачу решить нельзя, и ищется иной способ решения.

4. Проверка значимости каждого фактора по критерию Стьюдента. Если какой-то фактор оказывается незначимым, то его

убирают из расчета и повторяют его заново. Если незначимых факторов несколько, то убирают по одному.

5. Оценка качества модели с помощью коэффициента детерминации. В зависимости от области задачи приемлемы различные значения коэффициента детерминации. Например, для техники значение 0,9 будет пороговым. Все, что выше его, – хорошее, ниже – не очень. Для экономики порогом будет значение 0,5.

Теперь рассмотрим еще один особый вид регрессии – логистическую регрессию. Данный вид используется, если значение отклика (*y)* должно быть ограничено каким-либо диапазоном. Например, если *y* – вероятность некоторого события, то она должна лежать строго в диапазоне от 0 до 1. Рассмотренные ранее модели никак этого не учитывают, поэтому в подобных задачах используют логистическую регрессию. Чаще всего она применяется в задачах бинарной классификации. При $y < 0{,}5$ – один класс, иначе – другой.

В основе данного вида регрессии лежит следующая функция:

$$y = \frac{1}{1+e^{-x}}.$$

Ее график представлен на рисунке 14.

Соответствующая этой функции регрессионная модель будет иметь вид

$$y = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_k x_k)}}.$$

Теоретически, выполнив преобразование линерализации, ее можно привести к линейной модели множественной регрессии, произведя следующую замену:

$$z = -\ln\left(\frac{1}{y-1}\right).$$

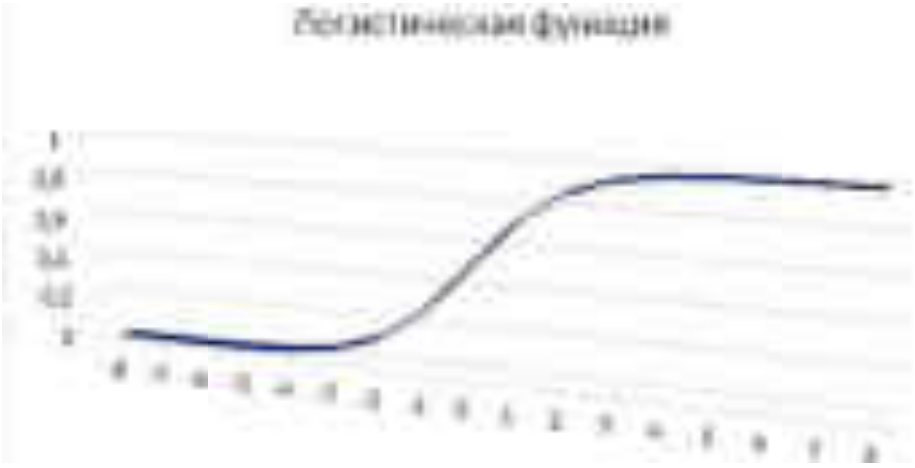


Рисунок 14. График логистической функции

Но так как в процессе преобразований нарушаются основные предпосылки МНК, то в практике этот подход не используется. Вместо этого для оценки параметров модели применяют метод максимального правдоподобия, с которым вам предлагается ознакомиться самостоятельно при необходимости.

Мы закончили рассмотрение статистических методов анализа данных. Теперь перейдем к более абстрактным моделям и алгоритмам, а именно к нечеткой логике.

Модели и методы нечеткой логики

Природа нечетких объектов обусловлена использованием экспертных оценок, которым свойственна неопределенность класса нечеткости. В отличие от стохастической неопределенности, нечеткость затрудняет или даже исключает применение статистических методов и моделей, но может быть использована для принятия предметно-ориентированных решений на основе приближенных рассуждений человека. Основные проблемы, решаемые в нечеткой логике, связаны с моделированием интеллектуальных операций приближенных рассуждений человека (эксперта), а также объектов, над которыми эти операции выполняются:

1. Объектами интеллектуальных операций, используемых в приближенных рассуждениях человека, являются переменные нового

класса – лингвистические переменные, значениями которых являются нечеткие множества. Важным является тот факт, что наименования лингвистической переменной и ее значений должны соответствовать словам, которые использует человек при решении прикладных задач. Таким образом, операндами и результатом интеллектуальных операций являются значения особого вида – *нечеткие множества*.

2. Основные интеллектуальные операции строятся с помощью *операций нечеткой логики*.

3. Алгоритмы вычисления нечетких значений предназначены для манипулирования со значениями, представленными нечеткими множествами на основе операций нечеткой логики, поэтому они классифицируются как нечеткие системы логического вывода. Часто используют сокращенную форму обозначенного класса моделей – *нечеткие модели или нечеткие системы.*

Нечеткие множества

Теория нечетких множеств, введенная Л. Заде [15], – это раздел прикладной математики, посвященный методам анализа неопределенных данных, в которых описание неопределенностей реальных явлений и процессов проводится с помощью понятия о множествах, не имеющих четких границ.

В дальнейшем для указания неопределенности экспертных оценок будем использовать эквивалентное выражение – лингвистическая неопределенность. Л. Заде предложил по аналогии с теорией вероятностей использовать в качестве математической модели лингвистической неопределенности объекта $x \in X$ функцию вида

$$Y = \mu\,(x,B),$$

где Y – результат вычисления функции, выражающий меру неопределенности (нечеткости) для конкретного объекта $x \in X$;

μ – непрерывная функция, такая, что $\mu: X \to [0,1]$. Содержательно функция μ определяет распределение неопределенности на X;

X – область определения функции μ. Область определения задается упорядоченным множеством значений произвольной природы, называемым *универсальным множеством (или универсумом*). Носителем функции $\mu(x,B)$ является подмножество $w \subset X$, на котором функция $\mu(x,B)$ принимает значение, отличное от нуля. В качестве универсального множества обычно задается множество действительных чисел;

B – вектор параметров функции, обычно числовых.

Функциональная модель лингвистической неопределенности получила название *нечеткого множества*, так как указанная функция μ рассматривается как характеристическая функция, определенная на множестве объектов X. Таким образом, с математической точки зрения, нечеткое множество моделируется параметрической функцией особого класса, называемого классом *функций принадлежности*. В том случае, если значения функции принадлежности нечеткого множества представлены точными числовыми значениями, такие нечеткие множества относят к *нечетким множествам типа 1*. Если значения функции принадлежности нечеткого множества моделируются другими нечеткими множествами, то такое нечеткое множество относят к *нечетким множествам типа 2*.

На практике используют несколько способов задания функции принадлежности. Среди них выделим следующие:

Структурный способ. Данная форма определения нечетких множеств основана на табличном представлении функций. В случае, если известен вектор параметров B, табличное представление функции принадлежности может быть задано явно путем табулирования функции $Y = \mu(x,B)$ на множестве значений w, являющемся ее носителем. При неизвестном векторе параметров B – путем прямого перечисления множества пар в виде

$$Y = \{\mu_1/x_1, \mu_2/x_2, .., \mu_n/x_n\}.$$

Данная форма удобна для графического отображения нечеткого множества и используется часто в тех случаях, когда затруднительно

задать математический вид функции $Y = \mu(x,B)$, например, если X не является множеством чисел.

Рассмотрим пример записи нечеткого множества в явной форме.

Пусть $w = \{x_1, x_2, x_3, x_4, x_5\}$, A – нечеткое множество, для которого $\mu_A(x_1) = 0{,}3; \mu_A(x_2) = 0; \mu_A(x_3) = 1; \mu_A(x_4) = 0{,}6; \mu_A(x_5) = 0{,}9$.

Тогда A можно представить в виде

$$A = \{0{,}3/x_1;\ 0/x_2;\ 1/x_3;\ 0{,}6/x_4;\ 0{,}9/x_5\}.$$

Функциональный способ. При этом предполагается, что форма функции принадлежности, моделирующей нечеткое множество, известна и определена на множестве действительных чисел X. Для представления $Y = \mu(x,B)$ используют различные функции (показанные, например, на рисунке 15: а) – треугольная, б) – трапецеидальная, в) – Гауссова).

Гауссова функция принадлежности описывается вектором параметров $B=\{\sigma, c\}$ и формулой

$$\mu(x, B) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right),$$

где c – среднее значение;

σ – среднее квадратичное отклонение.

Треугольная функция принадлежности характеризуется тройкой чисел, $B=\{a,b\ , c\}$, и вычисляется по формуле

$$\mu(x, B) = \begin{cases} \dfrac{x-a}{b-a}, a \le x \le b \\ \dfrac{c-x}{c-b}, b < x \le c \\ 0, x < a, x > c, \end{cases}$$

где b – задает координату вершины треугольника;

a, c – определяют основание треугольника.

По аналогии задается и трапецеидальная функция принадлежности, которая характеризуется четверкой чисел $B = \{a, b, c, d\}$:

$$\mu(x,B)=\begin{cases}\dfrac{x-a}{b-a}, a\le x\le b\\ 1, b<x<c\\ \dfrac{d-x}{d-c}, c\le x\le d\\ 0, x<a, x>d.\end{cases}$$

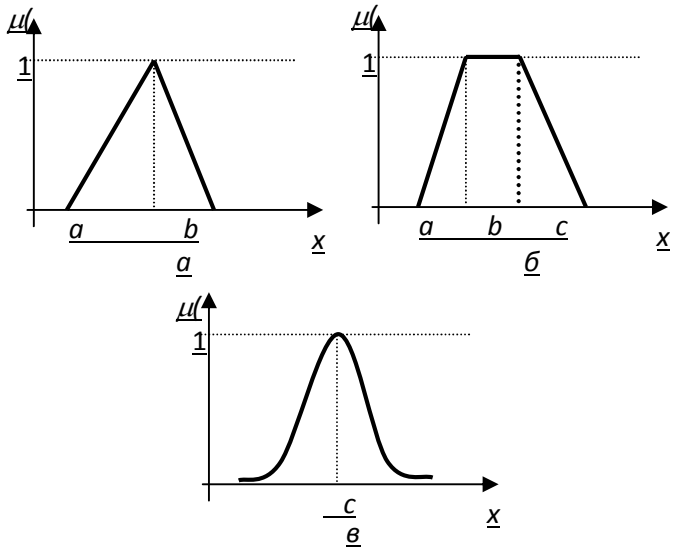


Рисунок 15. Типовые формы функций принадлежности

На практике часто параметр *B* явно не указывается для обеспечения более компактной записи функции принадлежности, то есть используется функциональная запись вида $Y=\mu(X)$ вместо $Y=\mu(x,B)$. При дальнейшем изложении в учебном пособии будем использовать компактную запись функции принадлежности.

С каждой функцией принадлежности $\mu(X)$ сопоставляется лингвистическое обозначение нечеткого множества (лингвистический терм). Тогда функция принадлежности нечеткого множества *Z* (функциональный способ) будет иметь следующую запись $Z=\mu_Z(X)$. Расширив традиционное понятие множества, Л. Заде построил «математический мостик» при описании свойств понятий в виде нечетких множеств между числом *x*, свойством *Z*, выраженным лингвистически, и степенью соответствия числа *x* свойству *Z* в виде функции

$\mu_Z(x)$. Фактически обозначение нечеткого множества через Z позволяет именовать функцию принадлежности $\mu_Z(X)$, в общем случае параметрическую, лингвистическими терминами, то есть оперировать с ней как со значениями лингвистической переменной. Часто эти два понятия – Z и $\mu_Z(X)$ – рассматриваются как эквивалентные.

Пусть $X = \{x\}$ – совокупность объектов (универсальное множество), обозначаемых через x. Пусть на X определены три нечетких множества: *A=«низкое», B=«удовлетворительное», C=«хорошее»,* обозначающие имена свойств понятия «качество». Тогда, следуя структурному способу для всех $x \in X$, нечеткое множество A может быть задано совокупностью упорядоченных пар $A = \{x, \mu_A(x)\}$, нечеткое множество B – совокупностью упорядоченных пар $B = \{x, \mu_B(x)\}$, нечеткое множество C – совокупностью упорядоченных пар $C = \{x, \mu_C(x)\}$. Используя функциональный способ записи, приведенный выше, нечеткие множества будут представлены следующим образом:

$$A = \mu_A(x),\ B = \mu_B(x),\ C = \mu_C(x).$$

Лингвистические переменные

Лингвистическая переменная – это переменная, значениями которой являются слова или высказывания естественного или искусственного языка.

Согласно [16]: «Поскольку слова в общем смысле менее точны, чем числа, понятие лингвистической переменной дает возможность приближенно описывать явления, которые настолько сложны, что не поддаются описанию в общепринятых количественных терминах… высокая точность несовместима с высокой сложностью. Таким образом, быть может, именно по этой причине обычные методы анализа систем и моделирования на ЭВМ, основанные на точной обработке численных данных, по существу не способны охватить огромную сложность процессов человеческого мышления и принятия решений. Отсюда напрашивается вывод о том, что для получения

существенных выводов о поведении гуманистических систем придется, по-видимому, отказаться от высоких стандартов точности и строгости, которые мы, как правило, ожидаем при математическом анализе четко определенных механистических систем, и относиться более терпимо к иным подходам, которые являются приближенными по своей природе».

Любая переменная описывается множеством допустимых значений, а лингвистические понятия описываются набором присущих им свойств. Л. Заде расширил понятие обычной лингвистической переменной, допустив, что в качестве ее значений (термов) выступают нечеткие переменные [16]. Пример лингвистической переменной, заимствованный из [17], представлен на рисунке 16.

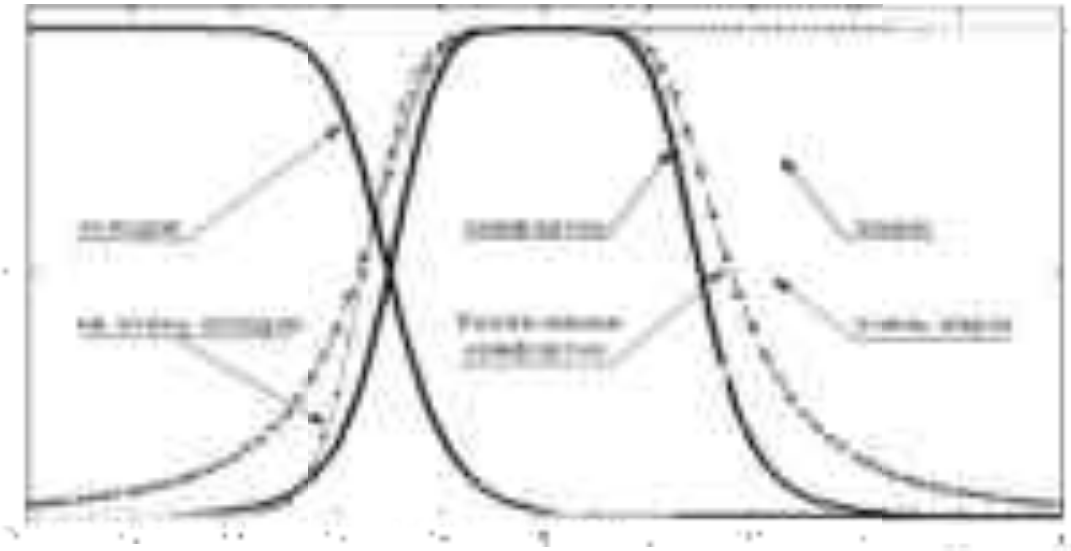


Рисунок 16. Пример лингвистической переменной «Температура»

Формально лингвистическая переменная описывается набором

$$< Name, \widetilde{X}, X, G, P >,$$

где *Name*– наименование лингвистической переменной;

X – универсальное множество объектов x;

$\widetilde{X}$ – базовое терм-множество, образующее совокупность термов лингвистической переменной, например, $\widetilde{X}$ ={«Отличный», «Хороший», «Плохой», «Удовлетворительный» и др.};

G – синтаксические правила вывода (порождения) новых термов $\tilde{X}^*$, не входящих в базовое терм-множество, задаваемые обычно на основе контекстно-свободной грамматики;

P – семантические правила, контекстно-зависимый способ вычисления смысла на основе функций принадлежности каждого терма из $\tilde{X} \cup \tilde{X}^*$.

Операции нечеткой логики

Нечеткая логика – это логика, оперирующая нечеткими высказываниями и рассуждениями на базе частичной истинности.

В основе операций нечеткой логики лежит понятие нечеткого множества, выраженного функцией принадлежности. Поэтому операндами и результатами операций нечеткой логики являются также функции, определяющие новые нечеткие множества.

В нечеткой логике для моделирования основных логических связок И ($\wedge$), ИЛИ ($\vee$) над нечеткими множествами используют триангулярные нормы [18].

Триангулярной нормой (t-нормой) называют отображение $T:[0,1]\times[0,1]\to[0,1]$, удовлетворяющее следующим условиям:

T(0, 0)=0; T(x, 1)= x; T(1, x)= x – ограниченность;

$T(x,y)\le T(a,b)$, если $x\le a$, $y\le b$ – монотонность;

$T(x,y)=T(y,x)$ – коммутативность;

$T(x,T(y,z))\le T(T(x,y),z)$ – ассоциативность.

Триангулярной конормой (s-конормой) называют отображение $S:[0,1]\times[0,1]\to[0,1]$, удовлетворяющее следующим условиям:

S(1, 1)=1; S(x, 0)= x; S(0, x)= x – ограниченность;

$S(x,y)\ge S(a,b)$, если $x\ge a$, $y\ge b$ – монотонность;

$S(x,y)=S(y,x)$ – коммутативность;

$S(x,S(y,z))\le S(S(x,y),z)$ – ассоциативность.

t-норма и s-конорма в определенном смысле являются двойственными понятиями. Эти функции могут быть получены друг из

друга, например, с помощью инволютивного отрицания и законов Де Моргана следующим образом:

$$S(x,y) = n(T(n(x),n(y))),\ T(x,y) = n(S(n(x),n(y))).$$

Простейшими примерами t-норм и s-конорм, взаимно связанных этими соотношениями для *n(x)=1-x*, являются следующие (таблица 9).

Таблица 9. Примеры t-норм и s-конорм

Формула	**Интерпретация**
T(x, y) = min{x,y}	*(минимум)*
S(x, y) = max{x,y}	*(максимум)*
T(x, y) = xy	*(произведение)*
S(x, y) = x+y-xy	*(вероятностная сумма)*
T(x, y)=max{x+y-1, 0}	*(t-норма Лукасевича)*
S(x, y)=min{x+y,1}	*(t-конорма Лукасевича ограниченная сумма)*

Основные операции с нечеткими множествами

1. Операция эквивалентности

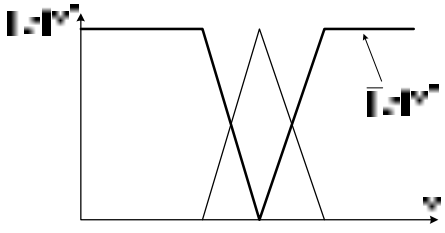
$$A \equiv B \Leftrightarrow \forall x \in X \ \ \mu_A(x) = \mu_B(x)$$

2. Операция включения

$$A \subseteq B \Leftrightarrow \mu_A(x) \le \mu_B(x), \forall x \in X$$

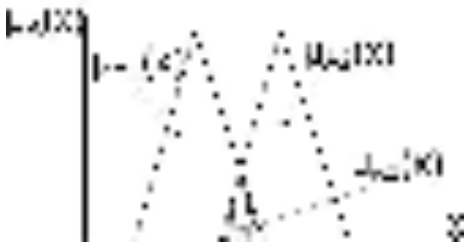
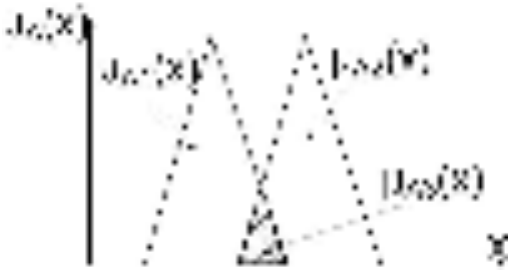
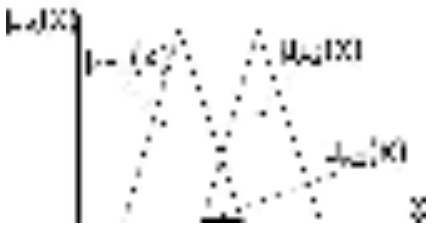
4. Нечеткая операция «НЕ» (дополнение) показана в таблице 10.

Таблица 10. Нечеткое «НЕ»

Иллюстрация	**Формула**
	$\overline{\mu_A}(x) = 1 - \mu_A(x)$ – нечеткая операция НЕ по Заде $\overline{\mu_A}(x) = \frac{1-\mu_A(x)}{1+\lambda \cdot \mu_A(x)}$ – НЕ по Сугено

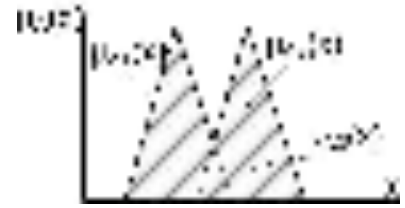
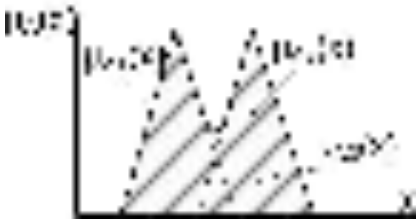
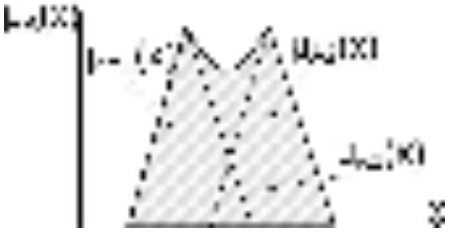
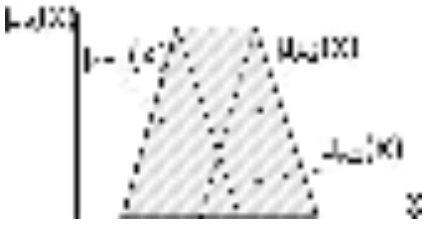
5. Нечеткая операция «И» показана в таблице 11.

Таблица 11. Нечеткое «И»

Иллюстрация	Формула
логическое произведение (Заде)	
	$\mu_{A_3}(x) = \mu_{A_1 \wedge A_2}$ $= \min(\mu_{A_1}(x), \mu_{A_2}(x))$ min – конъюкция
алгебраическое произведение (Бандлер, Коходт)	
	$\mu_{A_3}(x) = \mu_{A_1}(x) \cdot \mu_{A_2}(x)$
граничное произведение (Лукасевич, Гринс)	
	$\mu_{A_3}(x) = \mu_{A_1 \otimes A_2}$ $= \max(\mu_{A_1}(x) + \mu_{A_2}(x) - 1, 0)$
драстическое произведение (Вебер)	
	$\mu_{A_3}(x) = \mu_{A_1}(x) \triangle \mu_{A_2}(x) = \begin{cases} \mu_{A_1}(x), если\ \mu_{A_2}(x) \\ \mu_{A_2}(x), если\ \mu_{A_1}(x) \\ 0, иначе \end{cases}$ $0 \le \mu_{A_1 \triangle A_2} \le \mu_{A_1 \otimes A_2} \le \mu_{A_1 \cdot A_2} \le \mu_{A_1 \wedge A_2}$

6. Нечеткая операция «ИЛИ» показана в таблице 12.

Таблица 12. Нечеткое «ИЛИ»

Иллюстрация	**Формула**
логическая сумма (Заде)	
	$\mu_{A_3}(x) = \mu_{A_1 \vee A_2} = \max(\mu_{A_1}(x), \mu_{A_2}(x))$ max-дизъюнкция
алгебраическая сумма	
	$\mu_{A_3}(x) = \mu_{A_1}(x) + \mu_{A_2}(x) - \mu_{A_1}(x) \cdot \mu_{A_2}(x),$ $где\ \forall x \in X$
граничная сумма	
	$\mu_{A_3}(x) = \mu_{A_1 \oplus A_2} = \min(\mu_{A_1}(x) + \mu_{A_2}(x), 1)$
драстическая сумма	
	$\mu_{A_3}(x) =$ $\mu_{A_1}(x) \nabla \mu_{A_2}(x) = \begin{cases} \mu_{A_1}(x), если\ \mu_{A_2}(x) = 0 \\ \mu_{A_2}(x), если\ \mu_{A_1}(x) = 0 \\ 1, иначе \end{cases}$

По существу, все человеческие понятия являются нечеткими, так как они получаются в результате группировки (clumping) точек или объектов, объединяемых по сходству. Тогда нечеткость подобных

групп (clumps) есть прямое следствие нечеткости понятия сходства. Простыми примерами таких групп являются понятия «средний возраст», «деловая часть города», «немного облачно», «бестолковый» и др. Данную группу в нечеткой логике называют *«гранулой» (granule).* В естественном языке (ЕЯ) слова играют роль меток гранул и служат для сжатия данных. Сжатие данных с помощью слов является ключевым аспектом человеческих рассуждений и формирования понятий.

В нечеткой логике гранулирование информации лежит в основе понятий лингвистической переменной и нечетких правил типа «ЕСЛИ-ТО», задаваемой операцией импликации.

Операция импликации

В качестве основного математического инструмента при определении импликации A→B для нечетких множеств *A* и *B* используют композиционное правило Л. Заде [16], являющееся обобщением правила modusponens (таблица 13).

Таблица 13. Импликация

Описание	**Формула**
Предпосылка	$A \to B$
Событие	A^*
Вывод	$A^* \circ (A \to B)$

Пусть *U* и *V* – два универсальных множества с базовыми переменными *u* и *v* соответственно. Пусть *A* и *F* – нечеткие подмножества множеств *U* и *U×V*. Тогда композиционное правило вывода утверждает, что из нечетких множеств *A* и *F* следует нечеткое множество $B = A \circ F$. Функция принадлежности результата вычисляется с помощью триангулятных норм следующим образом:

$$\mu_B(v) = S(T(\mu_A(u), \mu_F(u,v)),$$

при моделировании s-конормы и t-нормы операциями (∨) *max* и (∧) *min* соответственно:

$$\mu_B(v) = \bigvee_{u \in U} ((\mu_A(u) \wedge \mu_F(u, v)) .$$

Другие формулы, реализующие операцию импликации, и математические объекты теории нечетких множеств и нечеткой логики приведены в учебном пособии [19].

Формализация нечеткой импликации позволила задать правила «ЕСЛИ-ТО» в виде нечетких продукционных правил и заложило основу нечеткого моделирования опыта и знаний экспертов, выраженных в виде приближенных зависимостей.

Нечеткие системы

Целью построения нечетких систем сложных явлений является приближенное описание зависимости (аппроксимация некоторой функции) $Y = f(X)$,

где Y – выходная лингвистическая переменная;

X – вектор входных лингвистических переменных. Его размерность – n;

f – зависимость между X и Y, описываемая совокупностью нечетких правил.

В основе нечетких систем лежат совокупность нечетких правил «ЕСЛИ-ТО», описывающих зависимости между нечеткими переменными предметной области, композиционное правило вывода и способ вычисления значений нечетких переменных (способ нечеткого вывода).

Системой нечеткого логического вывода в теории нечетких множеств и нечеткой логики называется модель, которая описывает поведение систем на естественном (или близком к естественному) языке в виде приближенных рассуждений на основе композиционного правила вывода.

В систему нечеткого логического вывода входят следующие объекты (см. рисунок 17):

- совокупность нечетких правил (база правил);
- набор функций принадлежностей базы нечетких переменных (база переменных);

- блок фаззификации;
- блок дефаззификации;
- блок вывода.

Рисунок 17. Система нечеткого вывода

База правил хранит множество логических правил вывода, а также их порядок (иерархическую структуру) применения. База нечетких переменных содержит названия лингвистических термов и параметры их функций принадлежности. База правил вместе с базой нечетких переменных образуют *базу знаний* (БЗ) системы нечеткого вывода.

Простейшие системы нечеткого логического вывода основаны на правилах вида:

R_i: *Если X есть* A_i *и Y есть* B_i, *то Z есть* C_i,

R_i: *Если X есть* A_i *и Y есть* B_i, *то* $z=f_i(x,y)$,

где X, Y – входные нечеткие переменные;

Z – выходная нечеткая переменная;

A_i, B_i, – входные значения (функции принадлежности);

C_i – выходные нечеткие значения (функции принадлежности);

f_i – некоторые вещественные функции.

При этом должны соблюдаться следующие условия:

- Существует хотя бы одно правило для каждого лингвистического терма выходной переменной.

- Для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

Распространены пять способов реализации нечеткого логического вывода [18].

Схема 1: Алгоритм Мамдани (Mamdani). Импликация моделируется минимумом, а агрегация – максимумом.

Схема 2: Алгоритм Цукамото (Tsukamoto). Исходные посылки – как у предыдущего алгоритма, но предполагается, что функции принадлежности являются монотонными.

Схема 3. Алгоритм Суджено (Sugeno). Алгоритм предполагает, что правые части правил вывода представлены в виде линейных функций.

Схема 4. Алгоритм Ларсена (Larsen). В алгоритме Ларсена нечеткая импликация моделируется с использованием операции умножения.

Схема 5. Упрощенный алгоритм нечеткого вывода. Исходные правила в данном случае задаются в виде

Если X есть A_i *и Y есть* B_i*, то* $z = Z_i$,

где Z_i – четкое значение.

Рассмотрим алгоритм нечеткого вывода по схеме Мамдани для базы правил вида R_i*: Если X есть* A_i *и Y есть* B_i*, то Z есть* C_i*.*, $i=[1,r]$.

1. Фаззификация.

Определяются степени истинности по функциям принадлежности для левых частей каждого правила:

$a_i = \mu_{A_i}(X)$

$b_i = \mu_{B_i}(Y)$,

где a_i – степень принадлежности X к A_i;

b_i – степень принадлежности Y к B_i;

$i= [1,r]$;

r – количество правил.

2. Импликация.

Определяется сила каждого правила, t-нормой является логический минимум:

$\alpha_i = \min(a_i, b_i)$.

Модифицируются функции принадлежности переменной z в каждом правиле:

$\mu'_i(z) = \min(\alpha_i, \mu_i(z))$,

где $\mu_i(z)$ – функция принадлежности переменной Z_i.

3. Агрегация.

Объединение выходов каждого правила логическим максимумом (s-конорма):

$\mu'(z) = \max_i(\mu_i(z))$

4. Деффазификация.

Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности. Однако пригодность этого способа ограничивается одноэкстремальными функциями принадлежности. Для многоэкстремальных функций на практике часто используется метод центра тяжести.

Дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле

$$x^0 = \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx}.$$

Физическим аналогом этой формулы является нахождение центра тяжести плоской фигуры, ограниченной осями координат и графиком функции принадлежности нечеткого множества. В случае дискретного универсального множества дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле

$$x^0 = \frac{\sum_{i=1}^{k} x_i \cdot \mu(x_i)}{\sum_{i=1}^{k} \mu(x_i)}.$$

На рисунке 18 графически представлен процесс нечеткого вывода по алгоритму Мамдани.

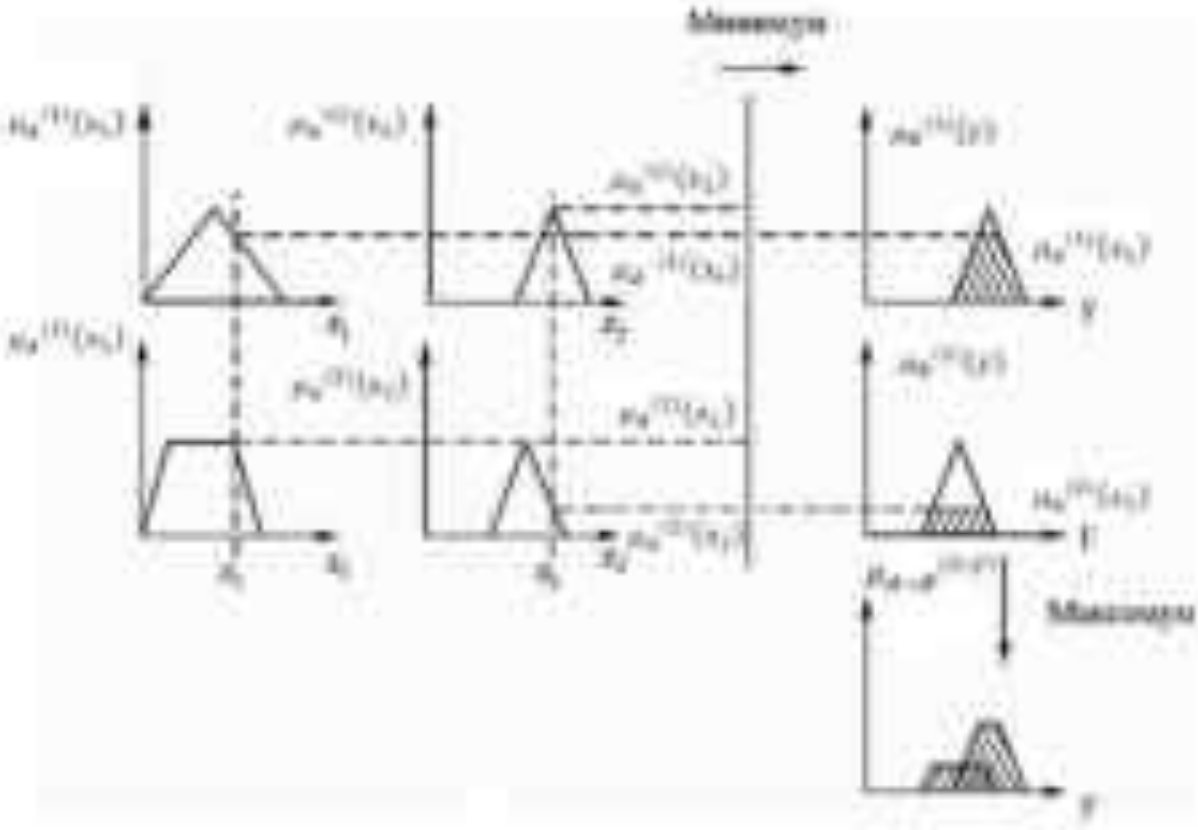


Рисунок 18. Алгоритм нечеткого вывода Мамдани

Пусть дана система управления с двумя правилами нечеткого управления:

Правило 1: IF x is A1 AND y is B1 THEN z is C1;

Правило 2: IF x is A2 AND y is B2 THEN z is C2.

Предположим, что величины x_0 и y_0, считываемые датчиком, являются четкими входными величинами для лингвистических переменных *x* и *y*, и что заданы следующие функции принадлежности для нечетких подмножеств *A1, A2, B1, B2, C1, C2* этих переменных:

$$\mu_{A1}(x) = \begin{cases} \frac{x-2}{3}, 2 \le x \le 5; \\ \frac{8-x}{3}, 5 \le x \le 8; \end{cases} \quad \mu_{A2}(x) = \begin{cases} \frac{x-3}{3}, 3 \le x \le 6; \\ \frac{9-x}{3}, 6 \le x \le 9; \end{cases}$$

$$\mu_{B1}(y) = \begin{cases} \frac{y-5}{3}, 5 \le y \le 8; \\ \frac{11-y}{3}, 8 \le y \le 11; \end{cases} \quad \mu_{B2}(x) = \begin{cases} \frac{y-4}{3}, 4 \le y \le 7; \\ \frac{10-y}{3}, 7 \le y \le 10; \end{cases}$$

$$\mu_{C1}(z) = \begin{cases} \frac{z-2}{3}, 1 \le z \le 4; \\ \frac{7-z}{3}, 4 \le z \le 7; \end{cases} \quad \mu_{C2}(z) = \begin{cases} \frac{z-3}{3}, 3 \le z \le 6; \\ \frac{9-z}{3}, 6 \le z \le 9; \end{cases}$$

Предположим, что в момент времени t_1 были считаны значения датчиков $x_0(t_1) = 4$ *и* $y_0(t_1) = 8$. Проиллюстрируем, как при этом будет вычисляться величина выходного сигнала.

Первым шагом находим -срезы для первого и второго правила на основе заданных функций принадлежности (с учетом значений x_0 и y_0). С этой целью вычисляем величины функций принадлежности в заданных точках для первого и второго правил:

$$\mu_{A1}(x_0=4)=2/3 \quad \text{и} \quad \mu_{B1}(y_0=8)=1;$$
$$\mu_{A2}(x_0=4)=1/3 \quad \text{и} \quad \mu_{B2}(y_0=8)=2/3.$$

Затем, в соответствии с правилом вывода по Мамдани (выбор минимального значения функций принадлежности), определяем:

$$\alpha_1 = min(\mu_{A1}(x_0), \mu_{B1}(y_0)) = min(2/3, 1) = 2/3;$$
$$\alpha_2 = min(\mu_{A2}(x_0), \mu_{B2}(y_0)) = min(1/3, 2/3) = 1/3.$$

Результат применения вычисленных значений α_1 и α_2 показан на рисунке 19. Окончательный результат получается путем объединения найденных значений функций принадлежности с использованием оператора максимума (с учетом стратегии вывода по Мамдани). Результирующая функция принадлежности также представлена на рисунке 19.

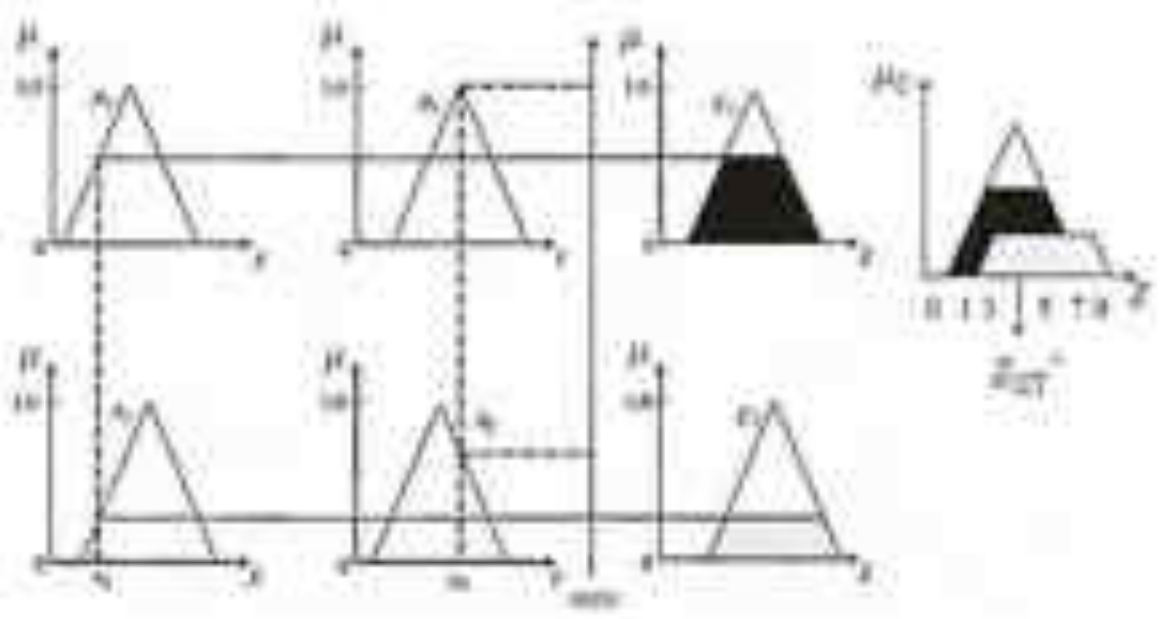


Рисунок 19. Иллюстрация нечеткого вывода по Мамдани

Для вычисления искомой выходной величины Z проводим дефаззификацию нечеткой величины μ_C. По методу центра тяжести получаем

$$Z_{\text{ЦТ}}{}^{*} = \frac{2\left(\frac{1}{3}\right) + 3\left(\frac{2}{3}\right) + 4\left(\frac{2}{3}\right) + 5\left(\frac{2}{3}\right) + 6\left(\frac{1}{3}\right) + 7\left(\frac{1}{3}\right) + 8\left(\frac{1}{3}\right)}{\left(\frac{1}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{1}{3}\right) + \left(\frac{1}{3}\right) + \left(\frac{1}{3}\right)} = 4{,}7.$$

Нечеткая логика в анализе временных рядов

Предположим, что задан процесс, состояния которого описываются *n* значениями одной переменной. Пусть в результате наблюдения получен временной ряд этой переменной, представляющий последовательность упорядоченных в равноотстоящие моменты времени пар $\{x_i, t_i\}$, таких, что $\forall x_i \in X, X \subset R^1, t_i \in N, i \in [1, n]$. Значение x_i – *уровень* временного ряда. Тогда введем понятие нечеткого временного ряда.

Нечетким временным рядом (НВР) называют упорядоченную в равноотстоящие моменты времени последовательность наблюдений над некоторым процессом, состояния которого изменяются во вре-

мени, если значение состояния процесса в момент t_i может быть выражено с помощью нечеткой метки $\tilde{x}_i$.

Нечеткая метка $\tilde{x}_i$ может быть сформирована непосредственно экспертом в форме

$$\mu_{\tilde{x}_i}(w, x_i),$$

где $\tilde{x}_i \in \tilde{X}$, $\tilde{X}$ – множество нечетких меток;

w – носитель (интервал на X) нечеткой метки $\tilde{x}_i$, $x_i \in w$;

$\mu_{\tilde{x}_i}(w, x_i) \in [0,1]$ – функция принадлежности нечеткой метки $\tilde{x}_i$ уровню временного ряда x_i, обычно треугольной формы.

Носитель нечеткой метки $\tilde{x}_i$ – это четкое множество $w \subseteq B$ таких точек $x_i \in w$, для которых $\mu_{\tilde{x}_i}(w) > 0$, где $B \subset X$ – базовое множество нечетких меток $\tilde{X}$.

Таким образом, нечеткий временной ряд формируется в результате интервального качественного оценивания уровней числового ВР. Интервалы-носители нечетких меток, образованные на множестве *X,* обязательно пересекаются. Качественный аспект нечеткой метке придает функция $\mu_{\tilde{x}_i}(w) \in [0,1]$.

На рисунке 20 изображен абстрактный нечеткий временной ряд, где каждой нечеткой метке $\tilde{x}_i$ соответствует нечеткое множество, задаваемое функцией принадлежности.

Рисунок 20. Абстрактный нечеткий временной ряд

В отличие от традиционного временного ряда значениями нечеткого ВР являются нечеткие множества, а не действительные значения уровней ВР.

В 1993 году ученые Сонг (Song) и Чиссом (Chissom) [20] предложили нечеткие модели детерминированных (time-variant) и авторегрессионных (time-invariant) временных рядов первого порядка (fist-order) и применили разработанные модели для прогнозирования количества регистрирующихся студентов университета штата Алабама (США), фаззифицировав предварительно четкий временной ряд. Это было первое применение нечетких моделей при моделировании ВР и первое определение моделей нечетких временных рядов.

Пусть $X_t,(t=1,...) \subset R^1$ – универсальное множество, на котором определены нечеткие множества $y_t^i,(i=1,2,...)$ и Y_t – коллекция $y_t^i,(i=1,2,...)$. Тогда Y_t называется нечетким ВР.

На практике в большинстве временных рядов последовательные наблюдения зависимы, так что:

$$R=\{(y_t,y_{t-1}),(y_{t-1},y_{t-2})...\} \subseteq Y_t \times Y_{t-1},$$

где Y_t, Y_{t-1} обозначают переменные;

y_t, y_{t-1} – наблюдаемые значения этих переменных.

Наиболее частой моделью зависимости является явная функция:

$$f: Y_{t-1} \to Y_t,$$

представленная линейной функцией:

$$y_t = f(y_{t-1},\phi,\varepsilon) = \phi y_{t-1} + \varepsilon_t,$$

где ε_t – случайная ошибка, шум.

В случае нечеткого временного ряда в качестве модели авторегрессии используется нечеткое разностное уравнение:

$$y_t^j = y_{t-1}^i \circ R_{ij}(t,t-1),$$

$$y_t^i \in Y_t,\ \ y_{t-1}^i \in Y_{t-1},\ \ i \in I,\ \ j \in J,$$

где ○ – обозначает операцию композиции из теории нечетких множеств;

$R(t,t-1)=\bigcup\limits_{i,j}R_{ij}(t,t-1)$ – система нечетких отношений, которая символически может быть записана в виде $Y_t \to Y_{t-1}$.

Систему отношений R в выражении $Y_t = Y_{t-1} \circ R(t,t-1)$ называют *моделью нечеткого временного ряда* первого порядка. Данная модель – важный частный случай общей модели порядка p:

$$Y_t = (Y_{t-1} \times Y_{t-2} \times ... \times Y_{t-p}) \circ R(t,t-p),$$

$$R(t,t-p) = \max\limits_{p} \left\{ \min\limits_{j,i_1,i_2,..i_p} \left\{ y_t^j, y_{t-1}^{i_1}, ... y_{t-p}^{i_p} \right\} \right\}.$$

Метод моделирования нечетких временных рядов

Моделирование нечетких временных рядов в соответствии с нечеткой моделью, предложенной в работе [20], состоит в реализации следующих шагов:

1. Определение нечетких переменных – разбиение данных на множество интервалов (носителей нечетких множеств), определение лингвистических значений нечетких множеств и их функций принадлежности.
2. Формирование логических отношений $Y_t \to Y_{t-1}$.
3. Фаззификация входных данных – определение степени принадлежности входных данных входным нечетким переменным.
4. Вычисление результата применения нечеткого правила $R_{ij}(t,t-1)$ для каждой импликации.
5. Вычисление результирующего отношения R как объединения $\bigcup\limits_{i,j}R_{ij}(t,t-1)$.
6. Применение полученной модели к входным данным и получение выходных нечетких результатов.
7. Дефаззификация нечетких результатов.

Одной из проблем в нечетком моделировании ВР является отсутствие четких рекомендаций на первом этапе построения модели

по выбору количества и параметров нечетких множеств, моделирующих входные и выходные переменные, в частности по определению их носителей (длины интервалов). Данные задачи выполняются экспертом, и, как показывают исследования, от выбора интервалов сильно зависит результат исследования.

Пример моделирования временного ряда в нечетком подходе

Приведем пример нечеткого моделирования временного ряда для прогнозирования прямых валютных котировок ЦБ USD/RUB за июнь 2005 года, описанный в работе [21] и представляющий модификацию метода Сонга, отличающуюся (а) использованием изменений (приращений) данных прошлого вместо реальных числовых значений (регистрации или валютного курса), и (б) вычислением отношений R_j для предсказания будущих состояний.

Рассматриваемый в работе метод был изначально успешно применен к временному ряду, характеризующему количество поступающих в Алабамский университет, которые являются бенчмаркингом при сравнении методов моделирования нечетких временных рядов.

Анализ результативности предлагаемого метода по показателю точности средней относительной ошибки аппроксимации для 6 нечетких множеств (MAPE=2,42) показал, что предложенный метод превышает аналогичный показатель для этого ВР, полученный методом Сонга и Чена (рисунок 21).

Применительно к проблеме прогнозирования валютного курса USD/RUB пошаговое описание предлагаемого метода нечеткого моделирования ВР можно свести к следующему:

Шаг 1: Задание области определения (универсального множества U) проблемы, исходя из вычисленных приращений валютного курса в течение рассматриваемого интервала времени.

Имеются следующие данные. Наибольшее положительное приращение курса доллара по отношению к российскому рублю наблю-

дается в феврале 2002 года, т. е. по сравнению с январским значением рост составляет 0.3679 (более 36 копеек/месяц). В ноябре-декабре 2004 года происходит самое значительное за трехлетний период наблюдений падение котировки доллара практически на 70 копеек (–0.6949). В результате, с целью упрощения последующего разбиения на равновеликие интервалы, полученные граничные значения (–0.6949 и +0.3679) слегка корректируются. Тогда, например, в случае использования шести подынтервалов U может быть представлена отрезком [–0.7, –0.5].

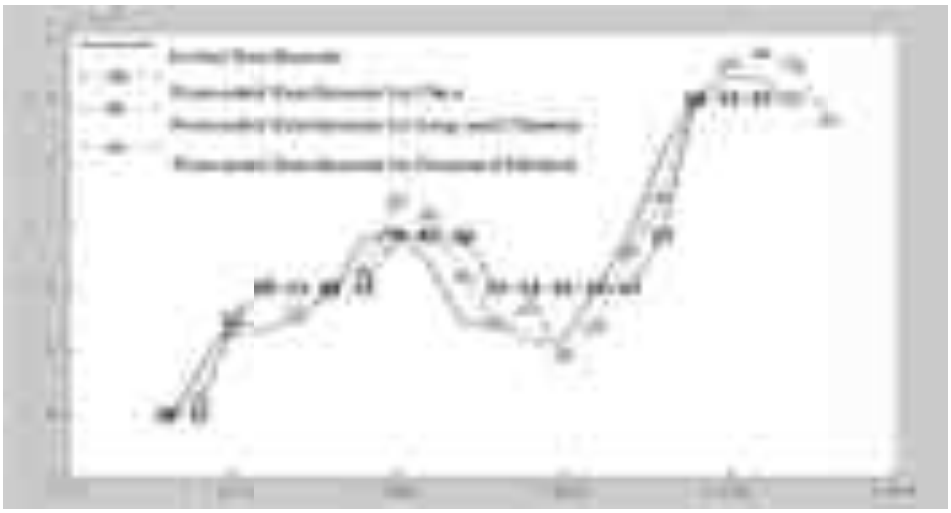


Рисунок 21. Сравнение результатов нечеткого моделирования

Шаг 2: Разбиение множества U на интервалы одинаковой длины.

Если мы оперируем с шестью нечеткими множествами, то область определения делится на 6 интервалов u_i, $i = \overline{1,6}$, $\mathrm{u}_1 = [-0.7, -0.5]$, $\mathrm{u}_2 = [-0.5, -0.3]$, ... , $\mathrm{u}_6 = [0.3, 0.5]$ (в действительности количество нечетких множеств не обязательно должно совпадать с числом интервалов разбиения).

Шаг 3: Определение нечетких множеств A_i.

Предположим, что лингвистическая переменная «изменение валютного курса» характеризуется терм-множеством, образуемым следующими значениями: A_1 (значительное уменьшение), A_2

(уменьшение), A_3 (без изменений/флэт), A_4 (увеличение), A_5 (значительное увеличение), A_6 (очень большое увеличение).

Для шести построенных выше интервалов u_i, $i = \overline{1,6}$, факт принадлежности каждого конкретного u_i определенному множеству A_j, $j = \overline{1,6}$ выражается действительным числом из единичного интервала [0,1] (предполагается, что элементы, отсутствующие в представлении множеств A_j, характеризуются нулевой степенью принадлежности):

$$A_1 = \{1/u_1 + 0.5/u_2\}$$
$$A_2 = \{0.5/u_1 + 1/u_2 + 0.5/u_3\}$$
$$A_3 = \{0.5/u_2 + 1/u_3 + 0.5/u_4\}$$
$$A_4 = \{0.5/u_3 + 1/u_4 + 0.5/u_5\}$$
$$A_5 = \{0.5/u_4 + 1/u_5 + 0.5/u_6\}$$
$$A_6 = \{0.5/u_5 + 1/u_6\},$$

где $u_i \subset U$ – элементы универсума U, а число, стоящее в числителе каждого элемента нечеткого множества, представляет собой степень принадлежности $\mu(u_i)$ этого элемента нечеткому множеству A_j, $j = \overline{1,6}$.

Шаг 4: Фаззификация приращений, полученных на шаге 1.

Считаем, что если приращение года t есть $p \in u_i$, и существует лингвистическое значение (нечеткое множество A_j) с максимальной степенью принадлежности, приходящейся на элемент u_i, тогда p фаззифицируется как A_j. Например, приращение за март 2002 по сравнению с предыдущим месяцем составляет +0.2476 – это значение попадает в интервал u_5, и фаззифицированное приращение становится равным A_5. Аналогичным образом производятся попарные сравнения каждого последующего и предыдущего месяцев, приводящие к формированию последовательности A_6 (февраль 2002), A_5

(март 2002), A_5 (апрель 2002), A_4 (май 2002), A_5 (июнь 2002), A_5 (июль 2002), A_4 (август 2002) и т. д.

Шаг 5: Формирование логических отношений $A_i \to A_j$.

Для построения последовательности логических отношений, рассматриваем попарно последовательные фаззифицированные приращения (февраль – март, март – апрель, и т. д.), определенные на шаге 4. Исключая повторяющиеся комбинации, окончательный список отношений принимает вид

$$A_1 \to A_2,\ A_1 \to A_4$$
$$A_2 \to A_1,\ A_2 \to A_2,\ A_2 \to A_3,\ A_2 \to A_4,\ A_2 \to A_5$$
$$A_3 \to A_2,\ A_3 \to A_3,\ A_3 \to A_4$$
$$A_4 \to A_2,\ A_4 \to A_3,\ A_4 \to A_4,\ A_4 \to A_5$$
$$A_5 \to A_3,\ A_5 \to A_4,\ A_5 \to A_5,\ A_5 \to A_6$$
$$A_6 \to A_5,\ A_6 \to A_4.$$

Мы предполагаем, что нечеткое импликативное отношение $D = B \to C$ для произвольных векторов B и C интерпретируется как нечеткая импликация Мамдани, следовательно, элементы матрицы D вычисляются по формуле $d_{ij} = b_i^T \times c_j = \min\left(b_i, c_j\right)$, где b_i и c_j – элементы векторов B и C, соответственно.

Шаг 6: Объединение логических отношений (шаг 5), имеющих одинаковые левые части, в группы, и вычисление отношений R_i, $i = \overline{1,6}$ для каждой сформированной группы. Можно обратить внимание на то, что группы отношений уже практически построены (см. шаг 5), и выглядят они следующим образом:

$$A_1 \to A_2,\ A_4$$
$$A_2 \to A_1,\ A_2,\ A_3,\ A_4,\ A_5$$
$$A_3 \to A_2,\ A_3,\ A_4$$
$$A_4 \to A_2,\ A_3,\ A_4,\ A_5$$
$$A_5 \to A_3,\ A_4,\ A_5,\ A_6$$
$$A_6 \to A_5,\ A_4.$$

Результирующие отношения $R_i, i=\overline{1,6}$, представляют собой объединения логических отношений, попавших в *i*-ю группу:

$$R_1 = A_1^T \times A_2 \cup A_1^T \times A_4$$
$$R_2 = A_2^T \times A_1 \cup A_2^T \times A_2 \cup A_2^T \times A_3 \cup A_2^T \times A_4 \cup A_2^T \times A_5$$
$$R_3 = A_3^T \times A_2 \cup A_3^T \times A_3 \cup A_3^T \times A_4$$
$$R_4 = A_4^T \times A_2 \cup A_4^T \times A_3 \cup A_4^T \times A_4 \cup A_4^T \times A_5$$
$$R_5 = A_5^T \times A_3 \cup A_5^T \times A_4 \cup A_5^T \times A_5 \cup A_5^T \times A_6$$
$$R_6 = A_6^T \times A_4 \cup A_6^T \times A_5.$$

Шаг 7: Прогнозирование и дефаззификация получаемых результатов.

Вычисленные отношения R_i используются в модели прогнозирования

$$A_i = A_{i-1} \circ R_i,$$

где A_i – нечеткое множество, выражающее прогнозное приращение месяца i;

A_{i-1} – известное приращение предшествующего $(i-1)$-го месяца (если $A_{i-1} = A_j$, то $R_i = R_j$, $j=\overline{1,6}$);

$\circ$ – обозначает композиционный «max-min» оператор.

Например, приращение валютного курса за февраль 2004 года при известном приращении (–0.5714) за январь месяц того же года вычисляется по формуле $F(02.2004) = A_1 \circ R_1$, где R_1 имеет вид, показанный в первой строке, а A_i – фаззифицированное приращение января 2004 года.

Шаг 8: Вычисление прогнозных валютных котировок USD/RUB.

Этот этап предусматривает преобразование полученных на шаге 7 нечетких прогнозных приращений в целые числа. В значительной степени такой процесс зависит от особенностей рассматриваемой

задачи, и одним из критериев выбора процедуры дефаззификации является ее вычислительная простота.

После того как получено числовое приращение для рассматриваемого месяца, оно суммируется с уже имеющимся значением обменного курса предыдущего месяца. Рассмотренный метод нечеткого моделирования может быть отнесен к числу полуавтоматических процедур, поскольку большинство выполняемых шагов, включая построение универсума на основании множества исходных данных задачи, могут быть эффективно воплощены в программной форме. Однако участие аналитика (эксперта) при формировании интервалов разбиения и соответствующих нечетких множеств играет также огромную роль.

Извлечение знаний из временных рядов

Исследования временных рядов, в том числе нечетких, в последние десятилетия оформились в виде отдельного направления, называемого *интеллектуальным анализом данных,* или *DataMining*, в котором анализ временных рядов получил название *интеллектуального анализа временных рядов,* или *TimeSeries DataMining (TSDM).*

Человеческое знание основано на образах и формулируется лингвистически. Вычисления со словами и образами дают методологию для управления информацией и для разработки систем, основанных на знаниях. *DataMining* интегрирует методы, основанные на образах, дает возможность для извлечения информации из баз данных в лингвистической форме, подходящей для их использования в методах принятия решений. Методы и технологии извлечения знаний с использованием временных рядов должны оперировать паттернами временных рядов, отыскивать ассоциации между ними и извлекать знания (рисунок 22). То есть необходимо представление результатов *DataMining* в форме, используемой в человеческих знаниях.

На основе новой методологии *DataMining* решается расширенная совокупность задач анализа временных рядов, определенных в работе [22]:

1) сегментация – разбиение ВР на значимые сегменты;

2) кластеризация – поиск группировок ВР или их паттернов;

3) классификация – назначение ВР или их паттернам одного из заранее определенных классов;

4) индексирование – построение индексов для эффективного выполнения запросов к базам данных ВР;

5) резюмирование (summarization) – формирование краткого описания ВР, содержащего существенные черты с точки зрения решаемой задачи;

6) обнаружение аномалий – поиск новых, нетипичных паттернов ВР;

7) частотный анализ – поиск часто проявляющихся паттернов ВР;

8) прогнозирование – получение очередного значения ВР на основе истории ВР;

9) извлечение ассоциативных правил – поиск правил, относящихся к паттернам ВР.

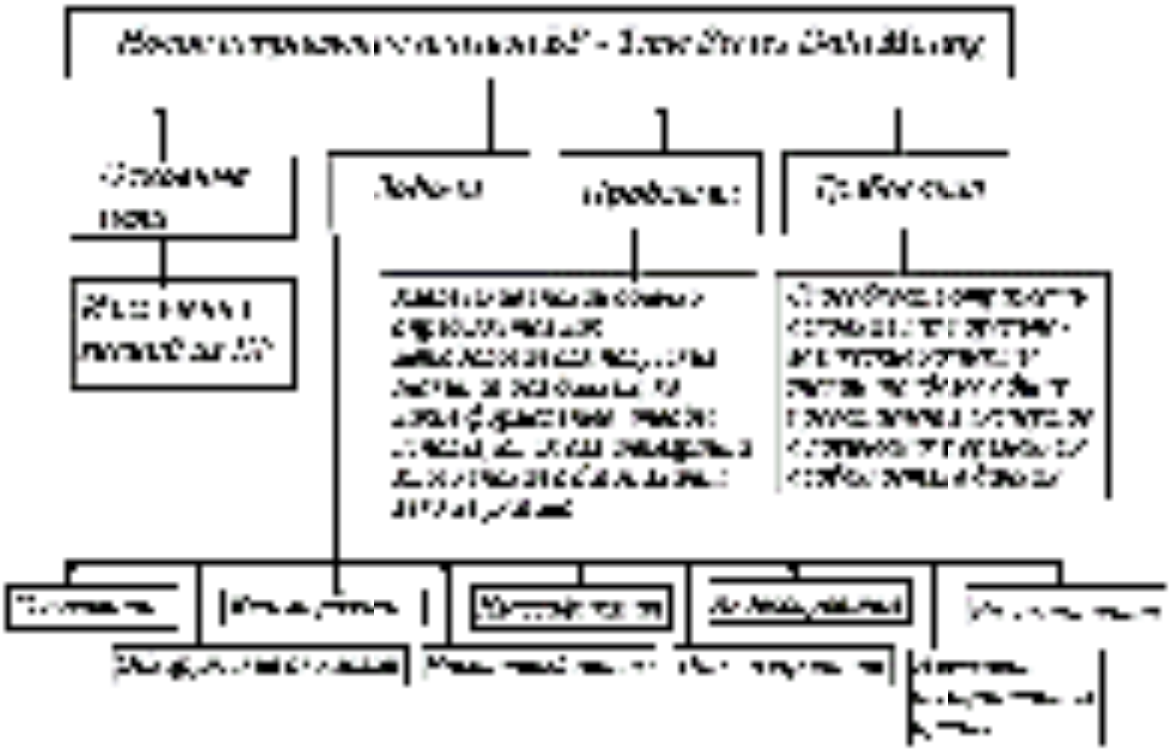


Рисунок 22. Задачи DataMining временных рядов

Традиционное выделение паттернов ВР было связано с выделением участков с постоянным знаком первой и второй производной: возрастающий и выпуклый, убывающий и гладкий и др. Различные шкалы и методы нечетких вычислений Л.Заде использовались для описания паттернов линейных трендов: рост, падение, резкий рост, медленное падение и т. д. Параметрические методы выпукло-гладкой модификации линейных функций и нечеткая грануляция выпукло-гладких паттернов позволили получить лингвистическое описание для ВР, подобное следующему: медленно убывающий и строго гладкий.

В рамках описанного выше направления TSDM акцент делается на поиск и извлечение правил из ВР, при этом полагаются на следующие основные принципы:

- Поиск правил нацелен на получение понимаемых результатов и не обязательно самых точных прогнозов;
- Важнейшим шагом на пути извлечения интерпретируемых знаний является порождение описаний фрагментов ВР в форме темпоральных образов, допускающих естественно-языковое толкование.

Требование к модели представления – способность отражать основные темпорально-логические концепты знаний (наиболее общие представления экспертов о логических и временных особенностях в данных).

Виды темпорально-логических концептов следующие:

- концепт временной продолжительности – присутствие определенного паттерна или признака ВР на определенном интервале времени;
- концепт очередности – порядок следования паттернов ВР во времени;
- концепт одновременности – совпадение во времени темпоральных событий (паттернов различных ВР);

- концепт нечеткости – нечеткость выраженности темпоральных событий и отношений.

В качестве инструментов анализа предлагается использовать нечеткие нейронные сети.

Существует множество методов прогнозирования временного ряда. Однако не создан лучший в любой ситуации метод: каждый из них имеет свои достоинства и недостатки. Поэтому одним из наиболее перспективных научных направлений стало создание комбинированных моделей.

Комбинированная модель прогнозирования – модель прогнозирования, состоящая из нескольких индивидуальных (частных) моделей, называемых базовым набором моделей. Использование комбинированной модели может повысить точность получаемого прогноза по ряду причин:

- Попытка выбрать одну модель для временного ряда с изменяющимся уровнем и динамическими свойствами приводит к выбору усредненной модели;
- При быстром изменении уровней ряда и его динамических свойств невозможно быстро производить анализ динамики и заменять одну модель прогнозирования другой;
- Любой отвергнутый из-за неоптимальности прогноз почти всегда содержит полезную независимую информацию;
- Слабые стороны одного метода прогнозирования возможно преодолеть, используя преимущества другого метода.

На кафедре «Информационные системы» Ульяновского государственного технического университета была разработана информационная система «Combination of fuzzy and exponential models». Данная система позволяет получать прогноз для временного ряда путем агрегации индивидуальных прогнозов моделей из базового набора. На момент создания системы базовый набор содержал 29 индивидуальных моделей и их модификаций, относящихся к экспоненциальным и нечетким моделям прогнозирования временных

рядов. Разработанная информационная система одержала победу на конкурсе «Computational Intelligence in Forecasting» в 2015 году [23].

Точность агрегированного прогноза напрямую зависит от выбранных из базового набора моделей. В связи с этим можно утверждать, что информационная система «Combination of fuzzy and exponential models» имеет два существенных недостатка:

- Пользователь вынужден самостоятельно выбирать модели из базового набора для каждого прогнозируемого временного ряда;
- Для всех временных рядов используется один агрегирующий метод.

Таким образом, целесообразно использовать методы машинного обучения для выбора моделей из базового набора и для выбора агрегирующего метода. Указанные задачи можно решить с помощью нейронной сети, выбирающей методы на основании значений метрик прогнозируемого временного ряда.

Для выбора метрик необходимо выбрать совокупность значимых для прогнозирования временного ряда характеристик, таких как: длина, степень выраженности тренда, степень выраженности сезонности, величина дисперсии, наличие стационарности.

При выборе используемых метрик временного ряда необходимо выполнение следующих условий:

- Метрика соответствует одной из основных характеристик временного ряда;
- Совокупность метрик максимально и разносторонне опиисывает временной ряд;
- Значение метрики принадлежит интервалу от 0 до 1.

При выборе методов прогнозирования из базового набора обучающая выборка нейронной сети представляет собой набор временных рядов, представленных в виде совокупности метрик, и соответствующих значений ошибки прогнозирования для каждой модели. Нейронная сеть, таким образом, обучается на основании значений метрик вычислять предполагаемые значения ошибки для каждой

модели из базового набора, что позволяет определять, какие модели из базового набора эффективнее применить для прогнозирования значений текущего временного ряда.

Для создания комбинированной модели прогнозирования на основании использования методов машинного обучения необходимо решить ряд задач:

- Определить состав моделей, входящих в базовый набор;
- Выявить ключевые для задачи прогнозирования метрики временного ряда;
- Определить структуру и способ обучения нейронных сетей выбора методов прогнозирования из базового набора и выбора метода агрегации.

Нечеткое сглаживание временного ряда

Очень часто при анализе временных рядов переходят от исследования самих значений ряда к исследованию тренда. Одним из методов его выделения является метод F-преобразования.

Нечеткое сглаживание временных рядов на основе нечеткого преобразования (F-преобразования) – методика, разработанная Перфильевой [24], которая может быть отнесена к методикам нечеткого приближения.

F-преобразование предполагает задание нечеткого разбиения универсального множества. В качестве последнего выбирается конечный интервал [a,b] действительной прямой. Зафиксируем значение n $(n > 2)$ узлов $x_1,...,x_n$ на [a,b] и предположим, что $x_1 \prec ... \prec x_n$, причем $a = x_1, b = x_n$.

Под нечетким разбиением [a,b] будем понимать совокупность n функций $A_1,...,A_n : [a,b] \to [0,1]$, удовлетворяющих следующим свойствам:

- $A_k : [a,b] \to [0,1], A_k(x_k) = 1$;

- $A_k(x) = 0$ если $x \notin (x_{k-1}, x_{k+1})$, где для единообразия обозначения мы положим $x_0 = a, x_{n+1} = b$;
- $A_k(x)$ непрерывна;
- $A_k(x), k = 2,...,n$ строго возрастает на $[x_{k-1}, x_k]$ и строго убывает на $[x_k, x_{k+1}]$;
- $\sum A_k(x) = 1$ для всех $x \in [a,b]$.

Функции $A_1,...,A_n$ называются базисными функциями. Базисные функции $A_1,...,A_n$ могут служить также функциями принадлежности нечетких подмножеств $A_1,...,A_n$ (обозначения функций и множеств унифицированы). Отметим, что форма базисных функций может быть уточнена дополнительно и согласована с такими требованиями к модели, как, например, гладкость.

Следующие формулы представляют нечеткое разбиение отрезка $[x_1, x_n]$, полученное совокупностью функций:

$$A_1(x) = \begin{cases} 1 - \dfrac{(x - x_1)}{h_1}, x \in [x_1, x_2] \\ 0, иначе \end{cases}$$

$$A_k(x) = \begin{cases} \dfrac{(x - x_k)}{h_{k-1}}, x \in [x_{k-1}, x_k], \\ 1 - \dfrac{(x - x_k)}{h_k}, x \in [x_k, x_{k+1}], \\ 0, иначе \end{cases}$$

$$A_{n-1}(x) = \begin{cases} 1 - \dfrac{(x - x_{n-1})}{h_{n-1}}, x \in [x_{n-1}, x_n] \\ 0, иначе \end{cases}$$

где $k = 1,...,n-1$ и $h_k = x_{k+1} - x_k$.

Предположим, что функция *f* имеет своей областью определения множество $P = \{p_1,..., p_l\} \subset [a,b]$, где $l>n$. Множество P считается плотным относительно нечеткого разбиения $A_1,...,A_n$, если выполнено условие:

$$(\forall k)(\forall j) A_k(p_j) > 0 \quad .$$

Пусть $A_k(p_j) = a_{kj}, k = 1,...,n; j = 1,...,l$, тогда матрица $A_{n\times l} = a_{kj}$ называется матрицей нечеткого разбиения для *P*, для которой справедливы свойства:

$$(\forall k)(\forall j) a_{kj} \in [0,1]$$

$$(\forall j)\sum_{k=1}^{n} a_{kj} = 1 .$$

F-преобразованием вектора *f*, определяемым матрицей нечеткого разбиения *A*, назовем вектор $F_n[f]$, где $F_n[f] = (F_1...F_n)$ и $F_i = \dfrac{\sum_{j=1}^{l} a_{ij} f_j}{\sum_{i=1}^{l} a_{ij}}$.

Координаты вектора $F_n[f]$ назовем соответственно компонентами F-преобразования. Обозначим $a_i = \sum_{j=1}^{l} a_{ij}, i = 1,..., n$; тогда $(a_1F_1,..., a_nF_n)^T = A \times f$. Компоненты F-преобразования являются точками минимума функции, задающей критерий взвешенного среднеквадратичного отклонения.

Пусть $F_n[f]$ есть F-преобразование *f*, определяемое $A_{n\times l} = a_{kj}$. Обратным F-преобразованием $F_n[f]$ назовем вектор $f_{F,n}$, вычисляемый по формуле $f_{F,n}^T = F_n[f] \times A$. Можно доказать, что если *n* возрастает, тогда $f_{F,n}(p_j)$ сходится $f(p_j), j = 1,..., N$.

F-преобразование имеет (кроме прочих) следующие свойства, важные для использования в качестве сглаживания временных рядов:

- У него прекрасные фильтрующие свойства;
- Его легко вычислять
- F-преобразование стабильно относительно выбора точек $p_1,...,p_N$. Это означает, что при выборе других точек p_k (и, возможно, изменяя их число *N*), результирующая функция $f_{F,n}$ значительно не меняется.

Прогнозирование тренда и прогнозирование числового представления временного ряда производится раздельно. Для этого необходимо вычислить так называемые остатки – разность между временным рядом и его трендом:

$$R = \{(f(p_j) - F_k)\}, \text{где } j : A_k(j) > 0 .$$

Полученный вектор *R*-вектор остатков для *k*-й компоненты тренда. Прогноз тренда и векторов остатков реализуется по формуле линейной комбинации. Прогноз компоненты тренда:

$$F_{k+1} = \alpha F_k + \beta F_{k-1}...$$

и прогноз вектора остатков:

$$R_{k+1} = \alpha R_k + \beta R_{k-1}... .$$

Для получения прогноза находится решение системы уравнений:

$$\begin{cases} F_{k-1} = \alpha F_{k-2} + \beta F_{k-3}... \\ F_k = \alpha F_{k-1} + \beta F_{k-2}... \end{cases}$$

Аналогично строится прогноз вектора остатков.

Также для построения прогноза может быть использована нейронная сеть, например, многослойный перцептрон или сеть Кохонена с выходной звездой Гроссберга. Вид нейронной сети определяет вид входных данных: на вход многослойного перцептрона подаются только абсолютные значения ($F_k, F_{k-1,...}$), на вход сети Кохонена подаются значения точек тренда, вычисленные относительно друг друга ($F_k - F_{k-1}, F_{k-1} - F_{k-2},....$).

На качество прогноза влияет количество нейронов во внутреннем слое. После получения прогнозных значений тренда и вектора остатков возможно построение числового представления прогноза временного ряда. Для этого производится сложение прогнозной компоненты тренда и прогнозного вектора остатков.

Точность прогноза зависит от количества точек временного ряда, участвующих в обучении: чем больше их, тем меньше ошибка (рисунок 23). Как видно из рисунка, вывод справедлив и для MAPE, и для SMAPE (линии на графике практически совпадают).

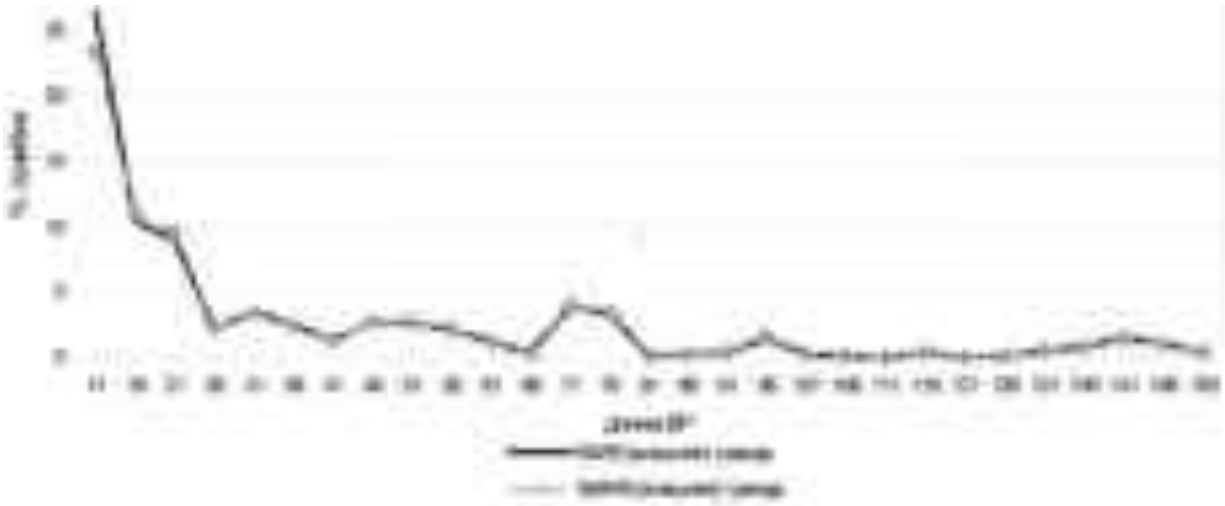


Рисунок 23. Зависимость процента ошибок от длины обучающей выборки

Процесс получения прогноза тренда и остатков зависит от некоторых параметров, которые в модели не удается задать жестко. Выбор данных параметров в значительной степени влияет на качество прогноза. Определим их:

- Степень авторегрессии при построении прогноза тренда (область значений);
- Метод, которым производится прогноз:
    - решение системы линейных уравнений;
    - нейронная сеть с абсолютными значениями предыдущих компонент на входе;
    - нейронная сеть с разностями между предыдущими компонентами на входе;

    - нейронная сеть с абсолютными значениями предыдущих компонент, а также разности между ними на входе;
- количество точек, покрываемых базисной функцией;
- сезонность (история отстоит на k точек от прогноза).

Сочетание параметров, при котором получается наилучший прогноз, получается перебором.

Нечеткая регрессия

В области прикладной статистики, анализа временных рядов и принятия решений в условиях неопределенности накоплен богатый опыт исследований и существует множество моделей, начиная от простейших линейных регрессионных моделей поиска тренда временного ряда и заканчивая сложными многоуровневыми авторегрессионными и адаптационными моделями. Регрессионный анализ, основанный на методе наименьших квадратов (Least-square), является очень удобным методом построения моделей, позволяющим численно оценивать зависимость интересующего исследователя параметра от воздействующих на него факторов. При анализе зависимости нечетких оценок от воздействующих факторов зачастую исследователям приходится иметь дело с важной информацией, которая не может быть задана точно. Некоторые наблюдения могут быть описаны только лингвистическими выражениями (типа «удовлетворительный», «хороший» и «превосходный»). Для таких данных аппаратом формализации может служить теория нечетких множеств. Были разработаны различные нечеткие регрессионные модели, основой которых является модель нечеткой линейной регрессии.

В нечеткой регрессионной модели параметры представляются триангулярными нечеткими числами и являются коэффициентами в нечеткой линейной функции. Неопределенность (vagueness) системы представляется суммарным разбросом («шириной») параметров (нечетких коэффициентов).

Построение модели состоит в нахождении оптимальных в некотором смысле коэффициентов с учетом нечеткой информации об объекте и субъективных представлений исследователя.

Базовые предположения нечеткой регрессии заключаются в том, что остатки, полученные как разность между наблюдениями и их оценками, продуцируются не случайными ошибками измерения, а неопределенностями (типа нечеткость) при вычислении параметров модели.

Большинство работ, посвященных нечеткой регрессии, были основаны на следующих базовых определениях.

Пусть дано множество наблюдений: $(y_j, x_{j1}, ...x_{jn}), j = 1,..., m$, необходимо найти нечеткую модель по следующей форме:

$$\tilde{Y} = A_0^{\sim} + A_1^{\sim} x_1 + ... + A_n^{\sim} x_n,$$

где $A_i(a_i^c, s_i^L, s_i^R), i = 1,..., n$ – триангулярные нечеткие числа;

a_i^c – среднее значение $A_i^{\sim}$;

s_i^L, s_i^R – показывают левый и правый разброс треугольной функции принадлежности соответственно.

Используются два критерия определения нечетких коэффициентов модели:

1. Для всех наблюдений принадлежность значения y_j к его нечеткой оценке $Y_j^{\sim}$ должна быть как минимум $Y_j^{\sim}(y_j) \geq h, j = 1,..., m$, где h – уровень доверия, выбранный лицом, принимающим решения.
2. Общая нечеткость предсказываемого значения зависимой переменной должна быть минимизирована. Это может быть достигнуто минимизацией суммы разбросов нечетких чисел для всех наборов данных.

Итак, проблему настройки нечеткой модели с заданными данными $(y_j, x_{j1},...x_{jn}), j = 1,..., m$ можно решить как эквивалентную задачу линейного программирования. То есть найти $a_-^c = (a_0^c,...a_n^c)$, $s_-^L = (s_0^L,...s_n^L)$, $s_-^R = (s_0^R,...s_n^R)$, которые минимизируют выражение:

$$Z = m(s_0^L + s_0^R)s_0^L + \sum_{I=1}^{n}\left[\left(s_i^L + s_i^R\right)\sum_{j=1}^{m}\left|x_{ij}\right|\right].$$

Чтобы оценить качество настройки нечеткой регрессии, используют метод наименьших квадратов. Для нечеткой регрессии среднеквадратичное отклонение (MSE) определяется следующим образом:

$$MSE = \frac{1}{m}\sum_{j=1}^{n}[y_j - def(Y_j)]^2 ,$$

где $def(Y_j)$ – дефаззифицированное значение зависимой переменной.

Исследователями были выделены различные варианты методов на основе классификации «вход – выход»: «четкий вход – четкий выход» метод CICO (Crisp-Input and CrispOutput), «нечеткий вход – нечеткий выход» метод FIFO (Fuzzy-Inputs and Fuzzy-Outputs) и смешанные данные – метод CIFO (Crisp-Inputs and Fuzzy-Outputs).

ACL-шкала и нечеткая кластеризация объектов

Для задания отношений между элементами нечеткого временного ряда мы можем использовать специальную лингвистическую шкалу в качестве инструмента как абсолютного, так и сравнительного нечеткого оценивания – ACL-шкала (Absolute&Comparative Linguistic) [25]. Абсолютные оценки, полученные по ACL-шкале, будут соответствовать нечетким меткам уровней ВР, а сравнительные оценки – нечетким тенденциям НВР. Опишем ACL-шкалу [25] как алгебраическую систему вида

$$Sx = <Name\_Sx, \tilde{X}, N, X, G, P, TTend, RTend>,$$

где *Name_ Sx* – имя ACL-шкалы; $\tilde{X}$ – базовое терм-множество абсолютных лингвистических оценок (лингвистическое название градаций); *N* – мощность базового терм-множества шкалы; *X* – универсальное множество, на котором определена шкала; *G* – синтаксические правила вывода (порождения) цепочек оценочных высказываний (производные термов, не входящих в базовое терм-множество); *P* – семантические правила, определяющие функции принадлежности для каждого терма (задаются обычно экспертно); $TTend(\tilde{x}_i,\tilde{x}_j)$ – лингвистическое отношение, фиксирующее тип изменения между двумя оценками $\tilde{x}_i,\tilde{x}_j$ шкалы; $RTend(\tilde{x}_i,\tilde{x}_j)$ – лингвистическое отношение, фиксирующее интенсивность различия между двумя оценками $\tilde{x}_i,\tilde{x}_j$ шкалы.

Первое, что нужно сделать для построения шкалы, – определить базовое терм-множество. Например, определим множество мощностью 5, содержащее в себе оценки уровней тренда исходного временного ряда: {малое, ниже среднего, среднее, выше среднего, большое}. Мощность множества обуславливается необходимостью вербализации каждого понятия шкалы, таким образом, чтобы она была понятна для человека и семантически содержательна.

Далее необходимо выбрать способ построения ACL-шкалы. Этим способом может стать неравномерное разбиение зафиксированной области значения величины с помощью алгоритма кластеризации. Достоинством такого подхода является то, что разбиение происходит без участия эксперта (тогда как в остальных возможных способах обязательно участие эксперта).

При безэкспертном разбиении задача формулируется следующим образом: дано множество из *L* точек, необходимо разбить его на *k* групп. Здесь *L* – количество элементов ВР (зафиксированное множество значений), *k* – количество термов на ACL-шкале, в нашем случае равное *N*. Как мы видим по формулировке задачи, это задача кластеризации. Кластеризация – это процесс разбиения объектов на группы по степени их схожести между собой. В отличие от

классификации, где есть заданная структура групп и признаки, на основе которых объекты в эти группы помещаются, в кластеризации структура разбиения, а также характеристические признаки являются не входными, а выходными данными. Кластеризация может быть четкой и нечеткой. При четкой кластеризации предполагается, что каждый объект принадлежит только одному кластеру, при нечеткой объект принадлежит всем кластерам, но с разной степенью принадлежности. При нечеткой кластеризации мы сможем использовать собственно степени принадлежности объектов ВР кластерам для получения термов шкалы. В итоге вся зафиксированная область разобьется на N взаимопересекающихся кластеров, упорядоченных по возрастанию значений их центров. Взаимопересечение получается за счет нечеткости кластеризации.

Кластеризация может выполняться с помощью нейронных сетей, генетических алгоритмов или с помощью собственно алгоритмов кластеризации. Например, для нечеткого разбиения некого множества точек на заданное количество кластеров можно использовать fcm-алгоритм кластеризации. Рассмотрим его подробнее.

FCM-алгоритм (Fuzzy Classifier Means) кластеризации применяется для нечеткого разбиения некого множества объектов на заданное количество кластеров. Целью алгоритма кластеризации является автоматическое разбиение множества объектов, которые задаются векторами признаков в пространстве признаков. Этот алгоритм предполагает, что объекты принадлежат всем кластерам с определенной степенью принадлежности, которая определяется расстоянием от объекта до соответствующих кластерных центров. Данный алгоритм итерационно вычисляет центры кластеров и новые степени принадлежности объектов. При этом он основан на минимизации целевой функции по мере расстояния объектов от центров кластеров:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{C} (u_{ij})^m \left\| x_i - c_j \right\| \quad ,$$

где N – количество объектов; C – количество кластеров; u_{ij} – степень принадлежности объекта i кластеру j; m – любое действительное число, большее 1; x_i– i-й объект набора объектов; c_j– j-й кластер набора кластеров; $||x_i - c_j||$ – норма, характеризующая расстояние от центра кластера j до объекта i.

Объектами кластеризации при анализе текстов, например, могут являться термины. Вектор признаков объектов кластеризации в данном случае содержит только значение какой-либо статистической метрики (например, частоты термина в тексте).

Алгоритм нечеткой кластеризации выполняется по шагам. Рассмотрим каждый из шагов подробнее, приведя необходимые модификации целевой функции для упрощения дальнейшего программирования.

Первый шаг – инициализация. На этом шаге задаются параметры кластеризации и инициализируется первоначальная матрица принадлежности объектов кластерам. К параметрам относятся следующие величины. Во-первых, степень нечеткости кластеризации – параметр m. От выбора этого параметра зависит значение функции принадлежности точки кластеру. Обычно он выбирается в пределах ~1.5..2. Чем больше его значение, тем более «нечеткая» кластеризация. В приведенных ниже примерах кода эмпирически было выбрано значение = 1.6. Во-вторых, выбирается мера расстояний $||x_i - c_j||$. Она характеризует степень близости точки к центру кластера. Если вектор характеристик состоит только из одного значения и задача, по сути, сводится к выделению значимых интервалов на прямой, то мера расстояния может иметь вид

$$||x_i - c_j|| = |x_i - c_j|.$$

Эта мера характеризует удаленность точки от центра кластера на прямой.

Третий параметр, который выбирается при инициализации – параметр сходимости алгоритма ε (уровень точности). Когда разность значений целевых функций текущей и предыдущей итераций достиг-

нет этого уровня, считается, что кластеризация завершена. Обычно этот параметр равен 0.001.

Четвертый параметр задается во избежание зависания алгоритма при невозможности достижения уровня точности. Это количество итераций алгоритма. Например, 10 000.

После выбора параметров генерируется случайным образом первоначальная матрица принадлежности объектов кластерам и происходит переход ко второму шагу алгоритма.

На шаге 2 происходит вычисление центров кластеров следующим образом:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{1.6} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{1.6}},$$

где c_j – центр j-го кластера; N – количество объектов; x_i – значение i-го объекта; u_{ij} – степень принадлежности объекта i кластеру j.

На третьем шаге формируется новая матрица принадлежности с учетом вычисленных на предыдущем шаге центров кластеров:

$$u_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_l\|}\right)^{3.33}},$$

где u_{ij} – степень принадлежности объекта i кластеру j; c – количество кластеров; c_j – вектор центра j-го кластера; c_l – вектор центра l-го кластера.

При этом если для некоторого кластера j и некоторого объекта i расстояние $|x_i - c_j|=0$, тогда полагаем, что степень принадлежности u_{ij} равна 1, а для всех остальных кластеров степень принадлежности этого объекта равна нулю.

Далее на четвертом шаге вычисляется значение целевой функции, и полученное значение сравнивается со значением на преды-

дущей итерации. Если разность не превышает заданного в параметрах кластеризации значения ε, считаем, что кластеризация завершена. В противном случае переходим ко второму шагу алгоритма.

Мы рассмотрели методы нечеткой логики, теперь перейдем к следующей модели – искусственных нейронных сетей (ИНС).

Искусственные нейронные сети

Для подготовки данного раздела использовался материал (в том числе иллюстративный) из следующего источника: [26].

Особенности нейронных сетей

Прежде чем перейти к описанию того, что собой представляет модель искусственной нейронной сети, акцентируем внимание на особенностях данного подхода, а в частности его достоинствах и недостатках. Это основные моменты, которые обязательно необходимо понять и запомнить.

Достоинства ИНС:

1. Нелинейность модели, что дает возможность аппроксимировать любые нелинейные функции. Та же задача с помощью полиномиальной модели не всегда решается, либо же ее решение становится низким по качеству.

2. Локальности восприятия, заключающаяся в том, что каждый нейрон получает не весь входной вектор. Это позволяет ей сегментировать данные и работать с более сложными ситуациями.

3. Каскад слоев. В сочетании с пунктом 2 это дает способность воспринимать более абстрактные признаки.

4. Лучше работает с мультиколлинеарностью и комбинациями признаков.

5. Нейронная сеть дает несколько механизмов для контроля переобучения.

6. Нейронная сеть способна дообучаться при непротиворечивости новых образов.

Недостатки:

Нейронная сеть не интерпретируема. Поэтому не ясна логика преобразования входных данных в выходные. Не всегда можно убедиться в стабильности решения на всей области определения.

Математика процессов и некоторых аспектов функционирования еще недостаточно изучена. Часто количество нейронов и слоев приходиться подбирать экспериментально для конкретной задачи, потому что определенной методики нет.

Нет аналитического решения, а решение численными методами градиентного спуска может приводить к попаданию в локальные минимумы ошибки.

Для ИНС очень часто требуется большая выборка. Так для распознавания речи одного языка может потребоваться от 5 до 10 тыс. часов размеченных записей. Для Глубоких сетей необходимо еще большее число объектов обучающей выборки, иначе будет постоянно происходить переобучение.

Обучение большой ИНС требует больших вычислительных ресурсов и специального оборудования GPU.

Поскольку на сегодняшний день нейронные сети очень модная тема, следует отметить, что ИНС, так же как любые обобщающие модели машинного обучения, чувствительна к противоречиям в данных и, конечно же, на текущем этапе развития НС не могут содержать противоречивые «представления» об объекте, проводить какие-то внутренние размышления и перестановки. Иными словами, из неоткуда информацию ИНС не возьмет (самостоятельно не проделает ряд выводов).

Так одним из главных конкурентов ИНС в ряде задач являются решающие деревья и ансамбли над решающими деревьями (например, XGBoost). Исключением являются задачи машинного зрения и работы с изображениями вообще, в которых ИНС занимаются сегодня лидирующие позиции.

Определение модели искусственной нейронной сети

Согласно Википедии [1]: «Искусственная нейронная сеть – математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети, созданные У. Мак-Каллоком и У. Питтсом. После разработки алгоритмов обуче-ния получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.».

Искусственные нейроны сильно упрощены по сравнению с их биологическими прототипами. Каждый из них имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим нейронам. Способность решать довольно сложные задачи обуславливается тем, что все нейроны соединены в достаточно большую сеть с управляемым взаимодействием. Схему простой нейронной сети можно увидеть на рисунке 24. Литерой «I» обозначены входные нейроны, «S» – скрытые нейроны, «O» – выходной нейрон. Соединяющие линии – связи между нейронами, или синаптические связи. Именно синаптические связи играют важнейшую роль в модели ИНС, так как в основе обучения лежит механизм корректировок силы этих связей.

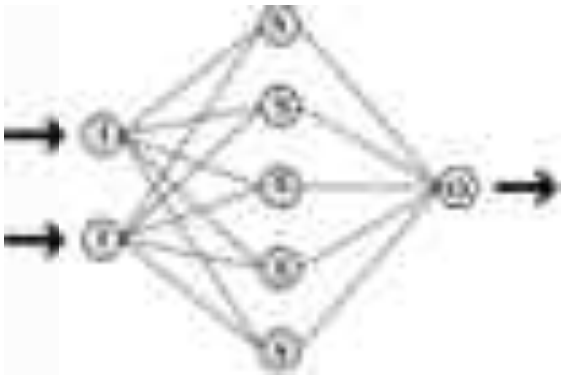


Рисунок 24. Схема простой нейросети

Нейронную сеть можно рассмотреть с разных точек зрения. Например, в машинном обучении, нейронная сеть – частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С точки зрения математики, обученная нейронная сеть – решенная многопараметрическая задача нелинейной оптимизации. Кибернетика представляет нейронную сеть как «черный ящик» или произвольную передаточную функцию, которую можно получить при обучении сети и затем воспроизводить при использовании сети. С точки же зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Очень часто именно про нейронные сети говорится, что они не программируются в привычном смысле этого слова, они обучаются и что возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Конечно же, такие трактовки и объяснения нельзя назвать точными и их можно допускать только при первом знакомстве с машинным обучением вообще.

Термин «обучение» необходимо трактовать лишь математически. В таком случае и Регрессия, и Деревья также обучаются, хотя в любой модели происходит одно и то же – корректировка весов.

Однако шумиха вокруг нейронных сетей порождает массу мифов, принижающих остальные методы, изученные нами ранее.

В чем-то эта модель напоминает уже изученные модели, такие как Регрессия и Деревья решений. В каком-то смысле Нейронные сети являются комбинацией этих методов (на уровне концепции).

В основе математической модели вычислений в ИНС лежит взвешенное суммирование, то есть, по сути, уже знакомое вам полиномиальное уравнение (в котором коэффициенты уже принято называть весами). Веса этого полинома и отражают силу синаптической связи между нейронами. Именно синаптические связи играют важнейшую роль в модели ИНС, так как в основе обучения лежит

механизм корректировок силы этих связей (подобно биологическому прототипу).

Все также вычисления завязаны на понятии ошибки и на том принципе, что корректное обобщение (отображение X=>Y) строится посредством уменьшения этой ошибки. Таким образом, они являются логическим продолжением прошлых тем.

Однако стоит отметить, что в отличие от методов Байеса и Регрессии нейронные сети изначально разрабатывались исходя из кибернетического подхода к моделям и информации. То есть нейронные модели – это некая адаптивная система, которая может прийти из одного состояния в требуемое, но при этом имеется некая случайная составляющая, которая вносит нестабильность в решения. Также нет хорошей компактной формулы, описывающей работу сети, – с самого начала суть вычислений в нейронных сетях была основана на численном (постепенном\итеративном) подходе к уменьшению ошибки.

Первая формальная модель и первая реализация нейронной сети

Согласно [27]: «Первой формальной моделью нейронных сетей (НС) и нейрона вообще была модель МакКаллока-Питтса, уточненная и развитая Клини. Впервые было установлено, что НС могут выполнять любые логические операции и вообще любые преобразования, реализуемые дискретными устройствами с конечной памятью. Эта модель легла в основу теории логических сетей и конечных автоматов и активно использовалась психологами и нейрофизиологами при моделировании некоторых локальных процессов нервной деятельности. В силу своей дискретности она вполне согласуется с компьютерной парадигмой и, более того, служит ее «нейронным фундаментом».

Пусть имеется n входных величин $x_1,\ldots,x_n$ бинарных признаков, описывающих объект x. Значения этих признаков будем трактовать как величины импульсов, поступающих на вход нейрона через n входных синапсов. Будем считать, что, попадая в нейрон, импульсы складываются с весами $\omega_1,\ldots,\omega_n$.

Если вес положительный, то соответствующий синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный импульс превышает заданный порог активации ω_0, то нейрон возбуждается и выдает на выходе 1, иначе выдается 0.

Таким образом, нейрон вычисляет n-арную булеву функцию

$$a(x) = \varphi(\sum_{j=1}^{n} w_j x^j - w_0),$$

где $\varphi(z) = [z \geq 0]$ – ступенчатая функция Хевисайда.

В теории нейронных сетей функцию φ, преобразующую значение суммарного импульса в выходное значение нейрона, принято называть функцией активации. Таким образом, модель МакКаллока-Питтса эквивалентна пороговому линейному классификатору. Схема ее вычислений показана на рисунке 25.

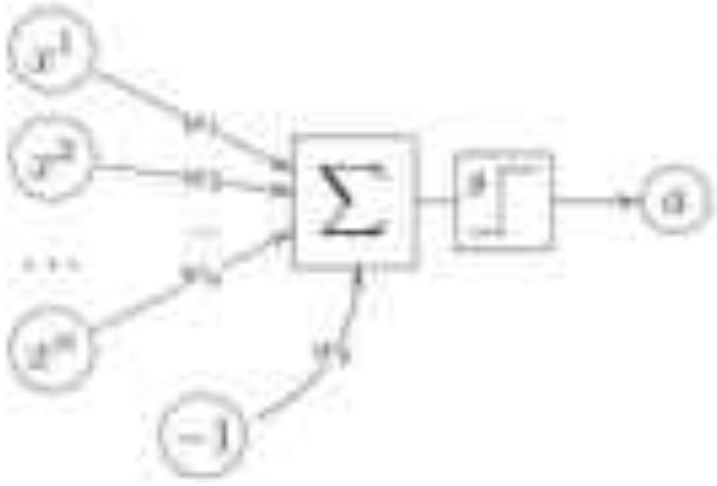


Рисунок 25. Схема вычислений в модели нейрона МакКаллока-Питтса

Получается, что нейрон производит взвешивание входных признаков (то есть рассчитывает результат с учетом их важности\веса) подобно регрессии. Однако важно понимать, что это пока всего лишь один нейрон.

Теоретические основы нейроматематики были заложены в начале 40-х годов, и в 1943 году У. МакКаллок и его ученик У. Питтс сформулировали основные положения теории деятельности головного мозга.

Ими были получены следующие результаты:

- разработана модель нейрона как простейшего процессорного элемента, выполняющего вычисление переходной функции от скалярного произведения вектора входных сигналов и вектора весовых коэффициентов;
- предложена конструкция сети таких элементов для выполнения логических и арифметических операций;
- сделано основополагающее предположение о том, что такая сеть способна обучаться, распознавать образы, обобщать полученную информацию.

Недостатком данной модели является то, что в качестве функции активации используется функция Хевисайда или униполярная пороговая функция (рисунок 26). В формализме, предложенном МакКаллоком и Питтсом, нейроны имеют состояния 0, 1 и пороговую логику перехода из состояния в состояние. Каждый нейрон в сети определяет взвешенную сумму состояний всех других нейронов и сравнивает ее с порогом, чтобы определить свое собственное состояние.

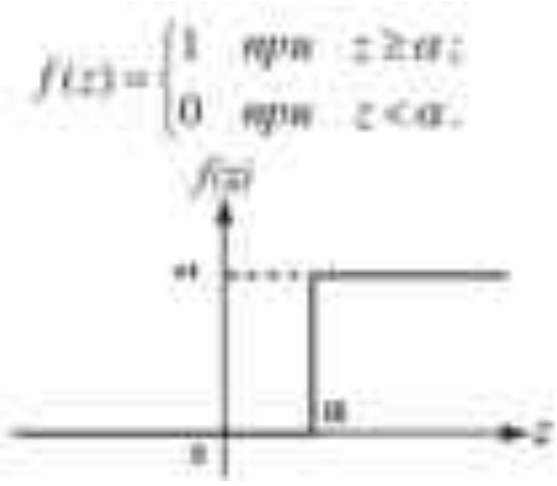


Рисунок 26. Пороговая функция, или функция Хевисайда

Пороговый вид функции не предоставляет нейронной сети достаточную гибкость при обучении и настройке на заданную задачу. Если значение вычисленного скалярного произведения, даже незначительно, не достигает до заданного порога, то выходной сигнал не

формируется вовсе, и нейрон «не срабатывает». Это значит, что теряется интенсивность выходного сигнала (аксона) данного нейрона и, следовательно, формируется невысокое значение уровня на взвешенных входах в следующем слое нейронов.

К тому же модель не учитывает многих особенностей работы реальных нейронов (импульсного характера активности, нелинейности суммирования входной информации, рефрактерности).

Несмотря на то, что за прошедшие годы нейроматематика ушла далеко вперед, многие утверждения МакКаллока остаются актуальными и поныне. В частности, при большом разнообразии моделей нейронов принцип их действия, заложенный МакКаллоком и Питтсом, остается неизменным».

Первой реализацией модели нейронной сети была созданная в 1960 электронная машина «Марк-1». Идею предложил Фрэнк Розенблатт в 1957. Согласно Википедии [1]: «Несмотря на то, что это первая модель и концепция, тем ни менее многие идеи современных нейронных сетей во многом совпадают с концепцией персептрона. Так выше была рассмотрена модель упрощенного искусственного нейрона, способного производить вычисления. Персептрон по своей сути является сетью таких нейронов. Из чего и предполагалось получать механизм для более сложных вычислений.

Несмотря на свою простоту, персептрон способен обучаться и решать довольно сложные задачи. Основная математическая задача, с которой он справляется, – это линейное разделение любых нелинейных множеств, так называемое обеспечение линейной сепарабельности. Логическая схема персептрона с тремя выходами представлена на рисунке 27.

Персептрон состоит из трех типов элементов, а именно: поступающие от датчиков сигналы передаются ассоциативным элементам, а затем реагирующим элементам. Таким образом, персептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе.

В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов. Согласно современной терминологии, персептроны могут быть классифицированы как искусственные нейронные сети:

- с одним скрытым слоем;
- с пороговой передаточной функцией;
- с прямым распространением сигнала».

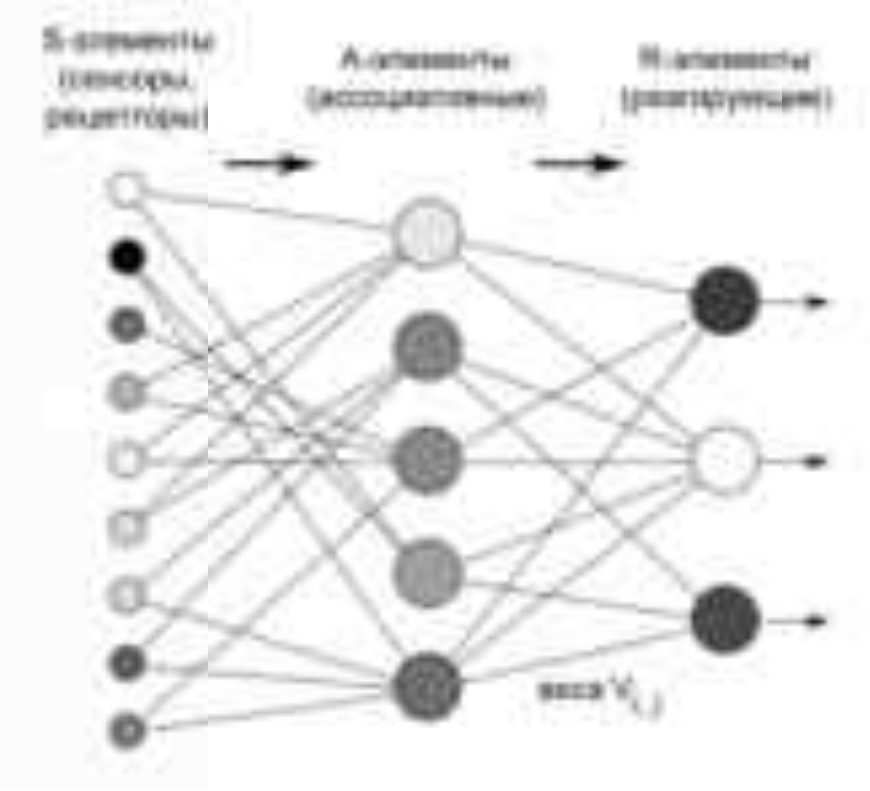


Рисунок 27. Логическая схема персептрона с тремя выходами

Многослойный персептрон (MLP)

Данный тип\архитектура является самой распространенной архитектурой искусственных нейронных сетей и в каком-то смысле классической. Очень часто, когда речь идет о нейронных сетях вообще, то имеется в виду именно многослойная сеть прямого распространения. Многие задачи классификации, аппроксимации и управления, которые решаются с помощью нейронных сетей, реша-

ются именно этим типом сети. Кроме того, принципы обучения, разработанные для этого типа сети, впоследствии стали применяться и для других типов. Так что в каком-то смысле это базовая архитектура для других типов сетей.

Если подходить к определению строго, то не совсем корректно называть такой тип сети Персептроном, ведь она отличается от него не только количеством слоев, но и тем, что:

- Нейроны имеют не ступенчатую функцию активации Хевисайда, а сигмоидальную функцию.
- Обучение производится не по правилу Хебба, а с помощью обратного распространения ошибки.

Поэтому такую архитектуру называют либо просто многослойной сетью, либо сетью прямого распространения или многослойной сетью прямого распространения. Но в то же время можно встретить и более удобное название MLP, что значит многослойный персептрон. Такое название определяет, что это все-таки не просто сеть. Структура MLP представлена на рисунке 28.

Замечание: выше уже упоминалось о том, что терминология в машинном обучении и в нейросетевых технологиях в частности еще не до конца систематизирована. Поэтому необходимо разбираться в сути моделей, допуская определенную вариативность названий. Однако со временем это перестает доставлять какие-либо трудности.

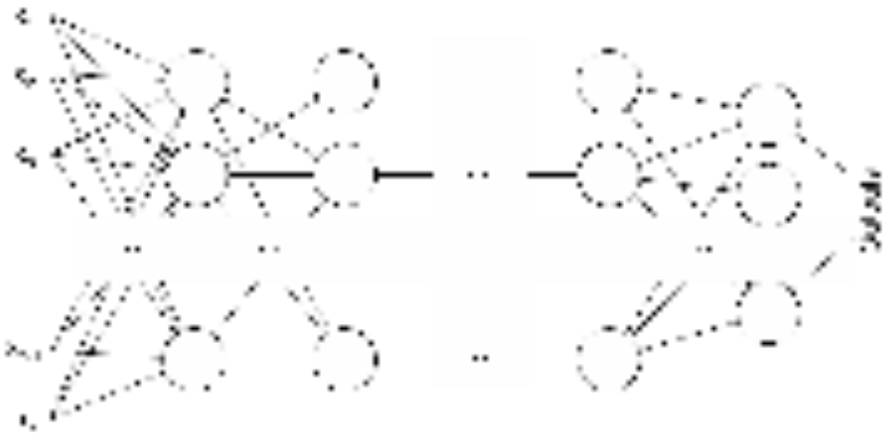


Рисунок 28. Структура многослойного персептрона

На рисунке 29 представлен график сигмоидальной функции активации.

Рисунок 29. График сигмоидальной функции

Сигмоидальная функция активации обладает следующими преимуществами по сравнению с пороговой функцией Хевисайда:

Нелинейность. Обеспечивает модели возможность обрабатывать (воспринимать) нелинейные закономерности в данных.

Фиксированные ограничения выхода. Это особенность позволяет масштабировать данные от слоя к слою.

Непрерывность и Дифференцируемость. Данные свойства позволяют обойти главный недостаток пороговой функции – резкий переход. Поскольку обучение сети происходит посредством градиентного спуска, то движение по пороговой функции слишком резкое, в отличие от сигмоидальной функции – движение по градиенту которой плавное и, как следствие, нахождение минимума ошибки становится более стабильным и вероятным.

Замечание 1: Кроме того, что нелинейная функция активации позволяет модели обрабатывать нелинейные закономерности в данных, она выполняет и другую важную задачу. Нелинейные выходы одного слоя нейронов позволяют подключить к нему второй, третий и т. п. слои. Если сделать много слоев в Персептроне с линейной функцией активации или с пороговой (суть в том, что она не непрерывна), то можно свести все эти слои всего лишь к одному слою

(с математической точки зрения последовательность взвешенных сумм (полиномов) можно легко свести к одному другому полиному).

Замечание 2: Подобно регрессионным моделям обучение модели ИНС подразумевает правку только весовых коэффициентов. Никакие другие процессы или изменения в модели не происходят.

Алгоритм обратного распространения ошибки как специальный метод градиентного спуска, разработанный именно для ИНС, будет подробнее разобран далее.

Сверточные (ConvolutionalNeuralNet) и Глубокие (DeepNet) Сети

Одна из базовых способностей людей и животных заключается в том, что мы легко распознаем визуальные образы под любым углом и в любом положении. Буква «А» останется для нас буквой «А» в какой бы области видимости наших глаз мы ее не увидели. Конечно же, при условии того, что выдержан определенный порог зашумленности, освещенности и т. п. Даже если текст расположен вверх ногами, то довольно быстро можно приноровиться его читать. А уж небольшие наклоны или разные шрифты на рекламных щитах вообще не представляют для нас проблемы.

А вот системы по распознаванию текста, к сожалению, не обладают такими возможностями. Либо они заточены распознавать определенный тип шрифта, либо они очень чувствительны к сдвигам или наклонам, либо к масштабу, либо вообще ко всему. И несмотря на то, что нейронные сети в качестве прототипа использовали принципы работы мозга, долгое время не было хорошего решения этой проблемы. Данный недостаток называется проблемой чувствительности к пространственным искажениям. А добиться нужно так называемой инвариативности к таким искажениям трех основных типов: смещениям, поворотам, масштабированию.

По поводу пространственных искажения и появления сверточной сети, в источнике [28] сказано следующее:

«Дело в том, что работа зрительной коры гораздо сложнее: в ней происходит анализ и цвета, и текстуры, и движения; информация от

двух глаз объединяется для осуществления стереозрения; работает множество других механизмов. Высшие зрительные функции все еще остаются загадкой. И самое главное, до сих пор не известно, как происходит обучение. До конца неясно даже, что в структуре зрительной системы заложено генетически, а что формируется под влиянием опыта. Хотя структура зрительной системы у человека продолжает формироваться до 4–5 лет, это может быть, как реализацией генетической программы, лишь немного адаптирующейся к окружению, так и детальным обучением, в результате которого создаются основные связи зрительного тракта.

Пытаясь разработать ИНС, которая была бы еще ближе к биологическому прототипу, японский ученый-компьютерщик Кунихика Фукусима предлагает принципиально новую модель Когнитрона в 1975 г., а в 1980 г. модификацию этой модели – «Неокогнитрон». Главное отличие этих сетей от MLP заключалось в том, что слои нейронов упаковывались не в линию, а в двумерную плоскость, чтобы информация циркулировала не только от слоя к слою, но еще сохраняла определенную пространственную ориентацию (рисунок 30). Кроме того, в этих сетях использовались принципы самоорганизации, то есть обучение происходило без учителя».

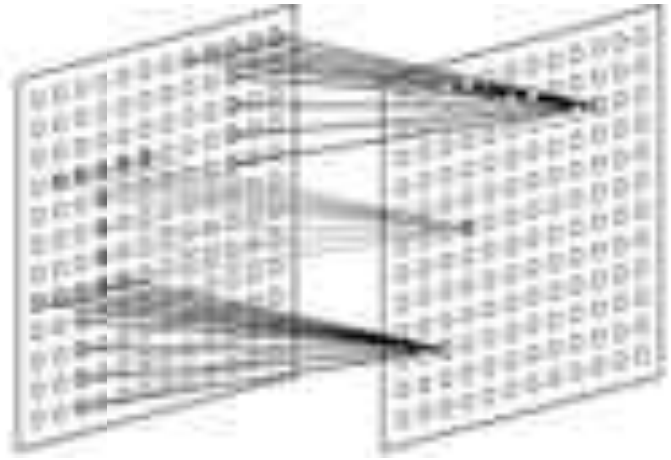


Рисунок 30. Рецептивные поля (квадратные плоскости) простых клеток, настроенных на поиск выбранного паттерна в разных позициях

В итоге подобная архитектура дала хорошие результаты при распознавании символов и даже рукописного текста относительно классических нейронных сетей. Однако по сравнению другими специализированными алгоритмами (не машинного обучения) на тот момент неокогнитрон «не дотягивал». Требовалось усложнять модель, но тогда были явные недостатки по скорости обучения и работы.

Однако общая концепция пространственно-ориентированных карт была очень ценной. И французский ученый и специалист в области обработки сигналов и компьютерного зрения Ян ЛеКун предлагает более упрощенную модель сети. Он убрал из неокогнитрона функции, которые нужны были только для того, чтобы быть похожим на реальный мозг. Он также показал, как можно использовать метод обратного распространения ошибки для обучения сетей, архитектура которых, как и у неокогнитрона, отдаленно напоминает строение коры мозга. И этот способ обучения, уже хорошо проверенный на тот момент, оказался куда эффективнее самоорганизации сетей. Так в 1995 появились нейронные сети Сверточного типа.

Структура сверточных сетей представлена на рисунке 31.

Нейроны здесь также упакованы не только в слои, но и двумерные карты. Информация циркулирует слева на право по нейронам такого же типа (как и в MLP). Но главная особенность заключается в том, что сеть не полносвязная, то есть каждый нейрон имеет свою небольшую область видимости (на верхнем рисунке область видимости показана пунктирными линиями). Такое локальное восприятие и обобщение от слоя к слою и дает решение проблемы чувствительности к пространственным искажениям, о которых говорилось выше. Иными словами, Сверточная Нейронная Сеть (СНС) способна обрабатывать пространственную топологию.

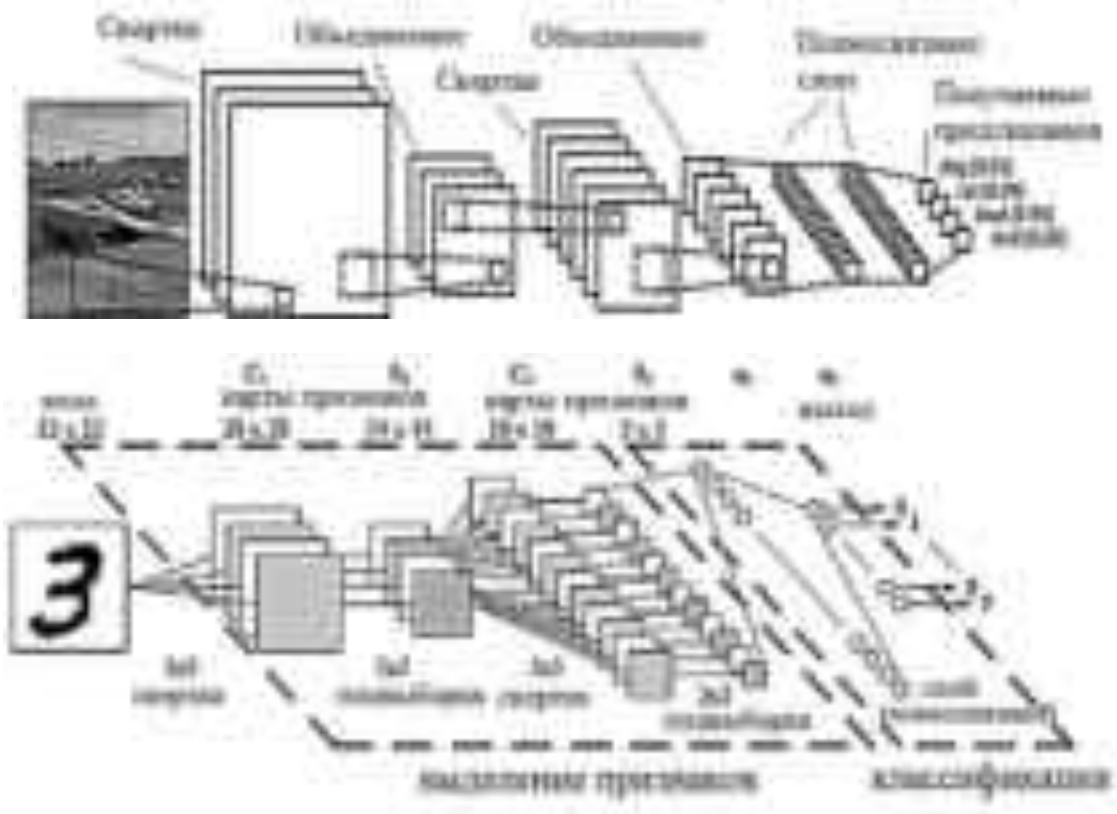


Рисунок 31. Структурная схема Сверточных нейронных сетей

Именно эта архитектура ИНС легла в основу так называемых Глубоких сетей, что породило понятие глубокого обучения (Deep Learning). Именно сеть этой архитектуры произвела такую шумиху в 2007 г., приблизив качество распознавания текста к человеческому уровню. Безусловно, данный метод не лишен недостатков, но во многих задачах именно визуального распознавания СНС является лучшим решением.

Глубокие сети, по сути, являются уже четвертой вехой развития когнитрона и их главная особенность заключается в многослойности (рисунок 32).

Но кроме этого важную роль играет разнообразное комбинирование определенных блоков карт. В глубоких сетях очень много разных модулей и ответвлений (может быть и такое, что сеть имеет несколько входов на разных уровнях).

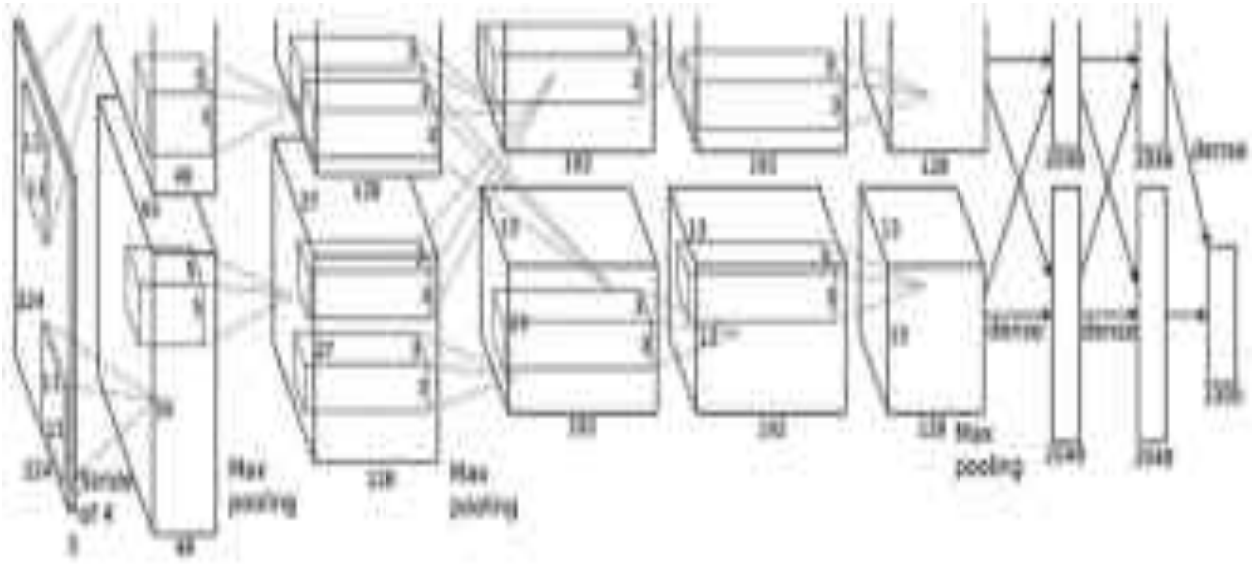


Рисунок 32. Схематичное представление Глубокой сети

Однако наращивание слоев приводит к целому ряду трудностей, как с объемом вычислений, так и с обучением сети, из чего складываются особенности, присущие именно глубокому обучению.

Карты (ART, SFAM)

Данный тип нейронных сетей разрабатывался как решение проблемы «пластичности-стабильности». Как известно, взрослый человек может понять какую-либо информацию с одного или нескольких объяснений. Речь, конечно, не идет о сложных вещах вроде курса математического анализа, но, тем не менее, мозг человека способен понять какие-то концепции даже за одно объяснение. Если переводить это на язык машинного обучения, то это означает, что нам не нужно многократное предъявление схожих примеров, нам хватит одного-двух объектов обучающей выборки. Конечно, мозг способен на такие вещи не только или, правильнее будет сказать, не столько благодаря быстро обучающимся нейронам, а потому что, воспринимая чье-то объяснение (например, когда вы читаете инструкцию к телевизору) мозг уже имеет представление о многих вещах. Иными словами, мозг уже обучен, у него есть предыдущий опыт. Кроме того,

разум способен держать контекст и т. п. Если объяснять упрощенно, то мозг дообучается, а не переобучается в таких случаях.

Но даже если не принимать во внимание контекст и априорный опыт, то возможно ли сделать обучение сети столь же пластичным, ну или близким к этому? Первая задача разработчиков сетей ART заключалась в том, чтобы сделать процесс обучения быстрым, чтобы не проходить итеративный градиентный спуск для правки весов. Эта проблема и есть проблема пластичности ИНС.

Но была и вторая задача. Опять же, вдохновляясь биологическим прототипом, разработчики понимали, что если человек узнает, что по какому-то вопросу у него на протяжении некоторого времени была неверная точка зрения, то он достаточно просто скорректирует отдельные представления о мире. У него не разрушится вся картина мира, он не сойдет с ума и не потеряет другие знания. Конечно, тут мы не берем в расчет психологические факторы, которые помешают ему это сделать.

Но если не рассматривать такие ситуации, то можно принять тот факт, что мозг очень пластичен, то есть может обучится\дообучиться с малого количества раз и в то же время стабилен – одни знания могут быть заменены на другие без разрушения другой информации.

В то же время информация в классических ИНС (в MLP) распределяется по весам всей сети. Иными словами, нет одного четкого места, которое бы отвечало за определенный блок информации. Таким образом, если дообучение происходит на объектах тех же классов, которые использовались во время обучения, то многослойную сеть прямого распространения можно дообучить (но опять же медленно). Но если взять обученную сеть на одних классах и попытаться до обучить на новых, то вся старая информация начнет разрушаться.

Например, ИНС типа MLP была обучена для классификации квадратов и кругов по визуальным образам. Допустим, что сеть стала показывать хорошие показатели качества после предъявления различных образов квадратов и кругов в количестве 10 тыс. штук. Если

позже дообучить сеть на новых вариантах квадратов и кругов, то сеть может еще улучшить показатели. Но если начать обучать ее еще и на треугольниках, то под этот новый класс не выделится определенное место в памяти сети, а распределенная по весам информация, наоборот, начнет меняться, разрушая обученное состояние.

Это и есть «проблема стабильности ИНС». Она была второй задачей разработчиков сетей типа ARTMAP (Творческие карты, или просто Карты).

Первые варианты Карт (ARTMAP 1, ARTMAP 2) были очень сложными, избыточными и во многом были инженерным творением в виде электрических схем, а не алгоритмом.

После возвращения интереса к нейронным сетям этот тип сетей был существенно оптимизирован и модифицирован. Один из самых успешных и эффективных вариантов называется Simplified FUZZY ARTMAP (упрощенная нечеткая Карта). Под нечеткостью здесь понимается то, что вычисления основаны на нечеткой логике. Детали работы нечеткости в этом подразделе не приводятся.

Структура такой сети представлена на рисунке 33.

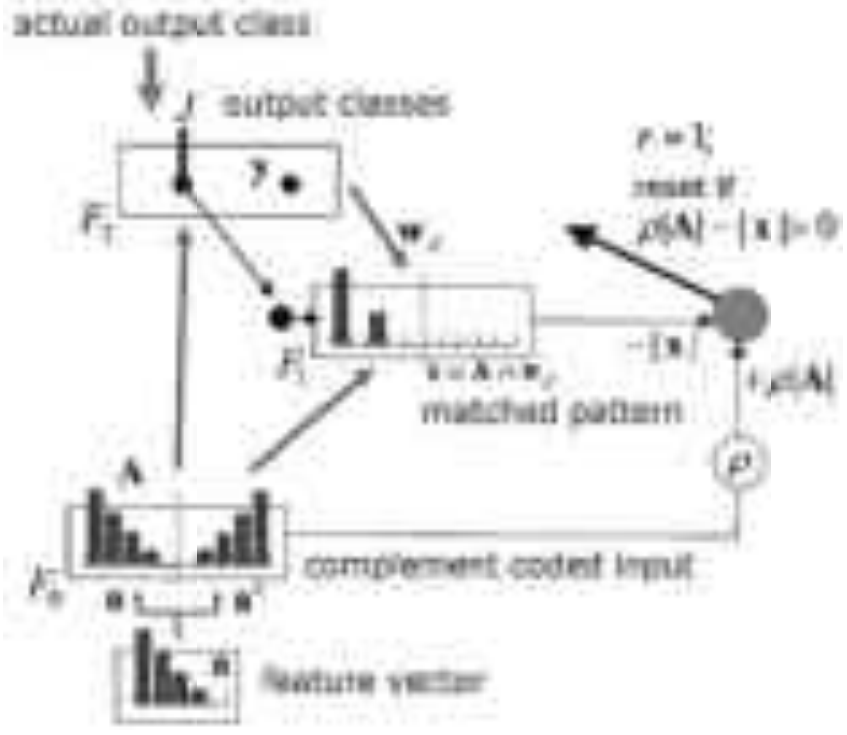


Рисунок 33. Структура основных модулей системы ARTMAP

Собственно, правильнее будет назвать это системой, содержащей нейроны, так как в структуре есть множество сторонних блоков (узлов), которые не очень красиво вписываются в биологическую концепцию мозга (хотя и не противоречат ей).

Итак, рассмотрим представленную на рисунке структуру снизу-вверх. В самом низу имеем вектор признаков «a» (как и во всех других методах ML). Данный вектор обязательно должен быть нормализован (отмасштабирован), то есть сеть не просто очень чувствительна к ненормализованным данным, а вообще не работает без нормализации. Соответственно если природа каких-то признаков неизвестна и не ясно, какой диапазон может быть в данных, то применение этой сети затрудняется (нужно специальным образом контролировать данные на вход) или вовсе отменяется.

Далее нормализованный вектор *a* дополняется комплементарной (обратной) парой a^c: там, где в исходном векторе было значение 0, в обратной паре будет 1, и наоборот (при условии нормализации от 0 до 1). Для этого и была необходима нормализация.

Далее, полученный вектор сравнивается со всеми шаблонами в базе нейронной сети. В данной случае это именно база, а не просто распределенная информация. Один нейрон в такой архитектуре представляет собой одну единицу информации, один конкретный шаблон\образ. То есть один нейрон выражает какой-то образ обучающей выборки (например, квадрат или круг). При обучении каждый поступающий вектор сравнивается с уже имеющейся базой образов, которые сохранены в нейронах сети.

Если близких образов нет, то образ сохраняется как новый, то есть создается новый нейрон и вектор входного образа полностью копируется в веса нового нейрона. Так с одного раза происходит полное корректное запоминание. При этом за нейроном закрепляется метка y (то есть это по-прежнему обучение с учителем, и для каждого входного вектора имеется ответ).

Если в базе уже имеется образ, похожий на входной образ, то происходит следующее. Образ в базе, выраженный нейроном и его

весами, подтягивается\приближается ко входному. Допустим, у сети в базе уже имеется некое представление о квадрате, и мы подаем еще один похожий образ квадрата, тогда после подтягивания в базе будет храниться нечто среднее, какой-то усредненный образ этих двух квадратов. Соответственно надо четко понимать, по какими принципам рассчитывается это усреднение. Ведь это все-таки не искусственный интеллект, и по одному образу человека и дельфина, сеть не усреднит их в образ млекопитающих. Надо понимать, что это лишь векторное усреднение. То есть если входной образ (0.3, 0.7), а образ в базе (0.35, 0.8), то итоговый будет (0.325, 0.75). Что с точки зрения гипотезы компактности позволяет решать задачу обобщения.

Замечание: напомним, что гипотеза компактности говорит нам о том, что объекты одного класса обладают близкими по значению признаками (или какой-либо комбинацией признаков), из-за чего их вообще можно отделить от объектов другого класса. Если пойти еще дальше, то можно сказать, что объекты сходных классов располагаются в некотором N-мерном пространстве признаков также близко. Тут можно сказать, что объекты потому и принадлежат какому-то одному смысловому классу, потому что по какому-то признаку или группе (комбинации) похожи.

Итак, данная сеть способна сохранять образ с одного предъявления, что решает проблему пластичности. В сети сразу появляется образ с меткой класса, к которому принадлежит этот образ. Таким образом, вместо минимизации ошибки данная сеть обучается путем создания похожих прототипов и сравнением входного образа с прототипами. В рабочем режиме сеть выдает ответ Y в виде метки того нейрона, образ которого сильнее всего похож на текущий входной образ.

При этом данная сеть также решает задачу стабильности, так как теперь есть четкие хранилища информации и система организации новых нейронов. Так что даже при предъявлении новых классов сеть будет стабильно работать.

Данный тип сети хорошо зарекомендовал себя в медицине и при распознавании звуковых сигналов. Однако есть и ряд минусов данной сети:

- при равной степени обобщения сеть ARTMAP потребует существенно большее количество памяти и вычислений, чем MLP, если речь идет о входных векторах большой размерности и очень больших обучающих выборках;
- сеть не способна учитывать топологию пространства и сложные абстрактные признаки (так как слоев по сути нет).

Рекуррентные сети (Recurrent Neural Network)

Рекуррентные нейронные сети (RecurrentNeuralNetwork; RNN) – вид нейронных сетей, в которых имеется обратная связь. При этом под обратной связью подразумевается связь от логически более удаленного элемента к менее удаленному (рисунок 34). Наличие обратных связей позволяет запоминать и воспроизводить целые последовательности реакций на один стимул.

В сетях такого типа возникает эффект памяти и способности воспринимать не только статичный образ, но и динамику образов (так как есть возможность учитывать историю через обратную связь).

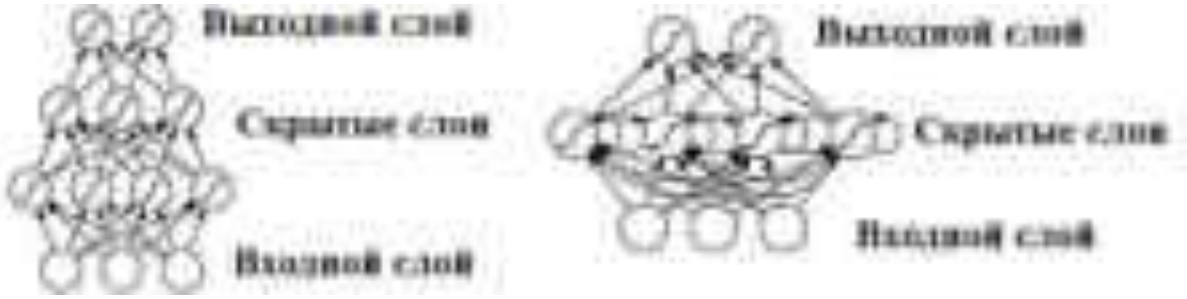


Рисунок 34. Сеть прямого распространения (слева) и рекуррентная сеть (справа)

Согласно Википедии [1]: «Долгая краткосрочная память (англ. Longshort-termmemory; LSTM) – разновидность архитектуры рекуррентных нейронных сетей (RNN), предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером. Как и большинство рекур-

рентных нейронных сетей, LSTM-сеть является универсальной в том смысле, что при достаточном числе элементов сети она может выполнить любое вычисление, на которое способен обычный компьютер, для чего необходима соответствующая матрица весов, которая может рассматриваться как программа. В отличие от традиционных рекуррентных нейронных сетей, LSTM-сеть хорошо приспособлена к обучению на задачах классификации, обработки и прогнозирования временных рядов в случаях, когда важные события разделены временными лагами с неопределенной продолжительностью и границами.

Относительная невосприимчивость к длительности временных разрывов дает LSTM преимущество по отношению к альтернативным рекуррентным нейронным сетям, скрытым Марковским моделям и другим методам обучения для последовательностей в различных сферах применения. Из множества достижений LSTM-сетей можно выделить наилучшие результаты в распознавании несегментированного слитного рукописного текста и победу в 2009 году на соревнованиях по распознаванию рукописного текста (ICDAR). LSTM-сети также используются в задачах распознавания речи, например, LSTM-сеть была основным компонентом сети, которая в 2013 году достигла рекордного порога ошибки в 17,7% в задаче распознавания фонем на классическом корпусе естественной речи TIMIT. По состоянию на 2016 год, ведущие технологические компании, включая Google, Apple, Microsoft и Baidu, используют LSTM-сети в качестве фундаментального компонента новых продуктов».

Самоорганизующиеся карты (Self-organization map, SOM)

Согласно ресурсу [29]: «Такие сети представляют собой соревновательную нейронную сеть с обучением без учителя, выполняющую задачу визуализации и кластеризации. Является методом проецирования многомерного пространства в пространство с более низкой размерностью (чаще всего, двумерное), применяется также для решения задач моделирования, прогнозирования и др. Является одной из версий нейронных сетей Кохонена. Самоорганизующиеся

карты Кохонена служат, в первую очередь, для визуализации и первоначального («разведывательного») анализа данных.

Геометрическая суть алгоритма такая, что близкие объекты в многомерном пространстве (схожие животные по тысяче признаков) будут расположены близко и на двумерной карте, где объекты будут представлены точками. Собственно, обучение сети это и есть процесс укладки таких точек (итеративного оттягивания таких точек от начальных позиций, см. рисунок 35).

Сигнал в сеть Кохонена поступает сразу на все нейроны, веса соответствующих синапсов интерпретируются как координаты положения узла, и выходной сигнал формируется по принципу «победитель забирает все», то есть ненулевой выходной сигнал имеет нейрон, ближайший (в смысле весов синапсов) к подаваемому на вход объекту. В процессе обучения веса синапсов настраиваются таким образом, чтобы узлы решетки «располагались» в местах локальных сгущений данных, то есть описывали кластерную структуру облака данных, с другой стороны, связи между нейронами соответствуют отношениям соседства между соответствующими кластерами в пространстве признаков».

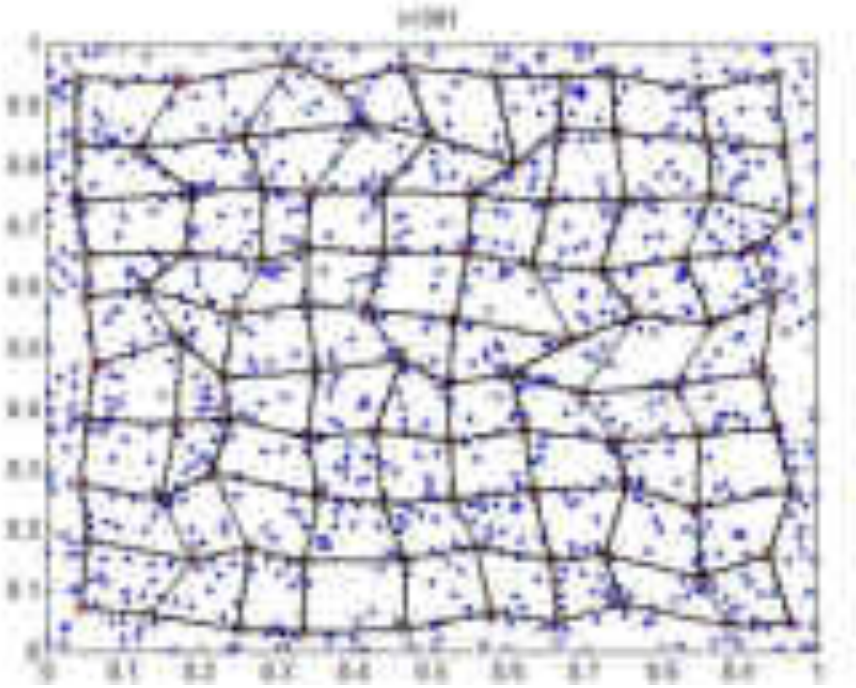


Рисунок 35. Визуализация двумерной плоскости с точками (проекциями) объектов в SOM

Автокодировщики (AutoEncoder)

Согласно Википедии [1]: «Автокодировщики (AutoEncoder) – специальная архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки. Простейшая архитектура автокодировщика – сеть прямого распространения, без обратных связей, наиболее схожая с персептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от персептрона, выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой (см. рисунок 36).

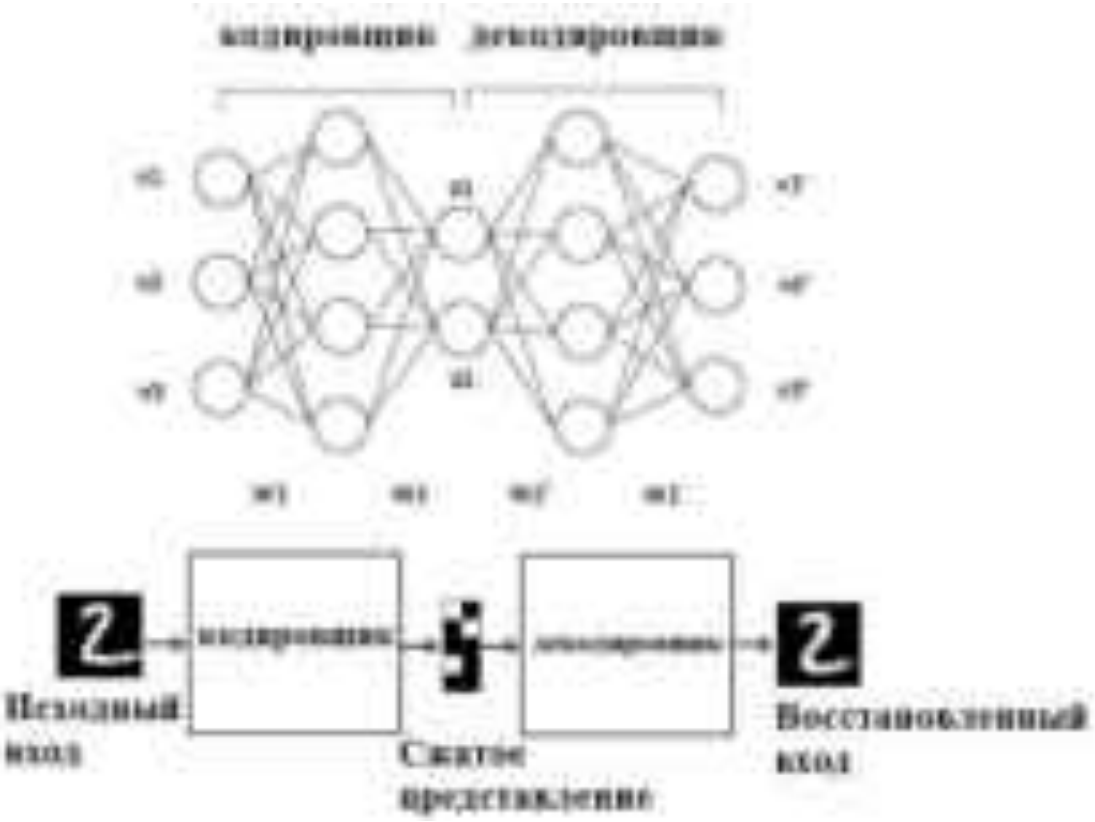


Рисунок 36. Схема сети типа Автокодировщик

Основной принцип работы и обучения сети автокодировщика – получить на выходном слое отклик, наиболее близкий к входному. Чтобы решение не оказалось тривиальным, на промежуточный слой автокодировщика накладывают ограничения: промежуточный слой должен быть или меньшей размерности, чем входной и выходной

слои, или искусственно ограничивается количество одновременно активных нейронов промежуточного слоя – разреженная активация. Эти ограничения заставляют нейросеть искать обобщения и корреляцию в поступающих на вход данных, выполняя при этом их сжатие. Таким образом, нейросеть автоматически обучается выделять из входных данных общие признаки, которые кодируются в значениях весов сети. Так, при обучении сети на наборе различных входных изображений, нейросеть может самостоятельно обучиться распознавать линии и полосы под различными углами.

Какое-то время автокодировщики применялись каскадно для обучения глубоких (многослойных) сетей, а именно для предварительного обучения глубокой сети без учителя. Для этого слои обучаются друг за другом, начиная с первых. К каждому новому необученному слою на время обучения подключается дополнительный выходной слой, дополняющий сеть до архитектуры автокодировщика, после чего на вход сети подается набор данных для обучения. Веса необученного слоя и дополнительного слоя автокодировщика обучаются при помощи метода обратного распространения ошибки. Затем слой автокодировщика отключается и создается новый, соответствующий следующему необученному слою сети. На вход сети снова подается тот же набор данных, обученные первые слои сети остаются без изменений и работают в качестве входных для очередного обучаемого автокодировщика слоя. Так обучение продолжается для всех слоев сети за исключением последних. Последние слои сети обычно обучаются без использования автокодировщика при помощи того же метода обратного распространения ошибки и на маркированных данных (обучение с учителем).

В последнее время автокодировщики мало используются для описанного «жадного» послойного предобучения глубоких нейронных сетей. После того как этот метод был предложен в 2006 г. Джеффри Хинтоном и Русланом Салахутдиновым, достаточно быстро оказалось, что новые методы инициализации случайными весами с определенным видом распределения оказываются эффективными

для улучшения сходимости. А предложенная в 2014 г. пакетная нормализация позволила обучать еще более глубокие сети, предложенный же в конце 2015 г. метод остаточного обучения позволил обучать сети произвольной глубины».

Основными практическими приложениями автокодировщиков остаются уменьшение шума в данных, а также уменьшение размерности многомерных данных для визуализации. Кроме того, с определенными оговорками, касающимися размерности и разреженности данных, автокодировщики могут позволять получать проекции многомерных данных, которые оказываются лучше тех, что дает метод главных компонент либо какой-либо другой классический метод. Помимо этого можно использовать такие сети, как алгоритм сжатия данных.

Крайне важно отметить, что если же взять архитектуру автокодировщика, но обучать его с учителем, то есть Y будет отличаться от X, то получится сеть для преобразования одного изображения в другое изображение через некоторую сложную функцию. На основе данного преобразования строятся различные фильтры изображений и видео (например, наложение различных эффектов) или же можно решать сложные задачи сегментации изображения. На сегодняшний день подобные архитектуры очень распространены и хорошо решают задачи анализа сцены для беспилотного автомобиля.

Импульсные (Спайковые) сети

Согласно Википедии [1]: «Первая научная модель импульсной нейронной сети была предложена Аланом Ходжкином и Эндрю Хаксли в 1952 году. Эта модель описывала, как потенциалы действия возникают и распространяются. Импульсы, однако, как правило, не передаются непосредственно между нейронами. Связь требует обмена химическими веществами, которые называются нейротрансмиттерами, в синаптической щели.

С точки зрения теории информации, проблема заключается в отсутствии модели, которая бы объясняла, как кодируется инфор-

мация и декодируются серии последовательностей импульсов, то есть потенциалы действия. Для нейробиологии все еще открытым является ответ на вопрос: нейроны связываются с помощью частотного или временного кодирования? С помощью временного кодирования один импульсный нейрон может заменять сотни скрытых элементов частотной нейронной сети.

Что же касается отличия спайковых сетей от классических, то тут необходимо отметить следующее. Если основное отличие сверточных сетей от классических заключается в структуре и организации связей, то импульсные сети, напротив, по структуре похожи на MLP. Их главное отличие заключается в том, что нейроны обмениваются короткими (у биологических нейронов – около 1-2 мс) импульсами одинаковой амплитуды (у биологических нейронов – около 100 мВ). Является самой реалистичной, с точки зрения физиологии, моделью ИНС (см. рисунок 37)».

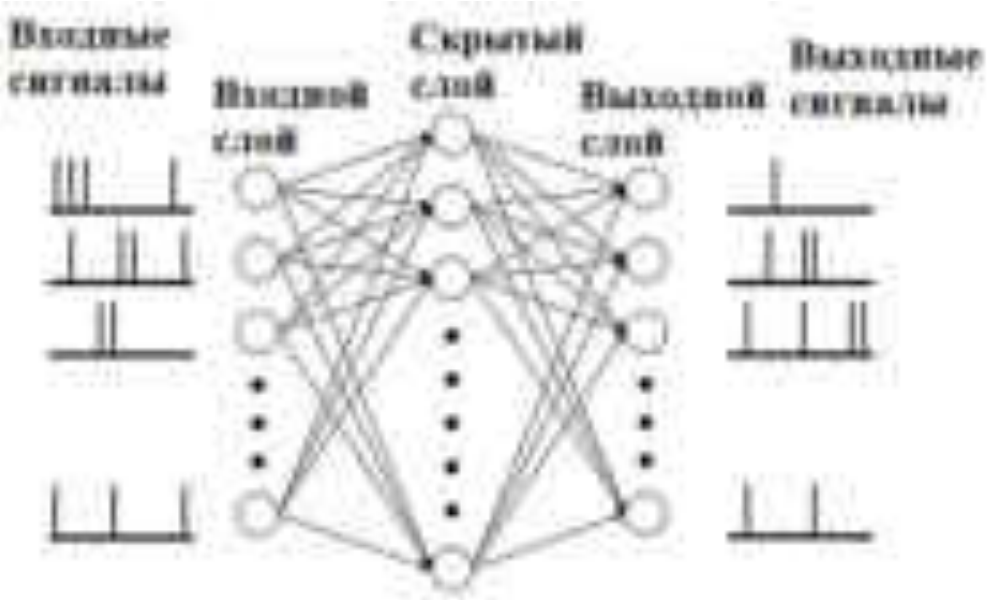


Рисунок 37. Схематичное представление структуры Импульсной нейронной сети

Данный тип сети еще находиться на стадии активного исследования. Тем ни менее ИмНС уже успешно применялась для динамического управления мобильным роботом. В качестве основных ее преимуществ можно назвать:

- Перспективность в обработке динамической информации (например, видеопотока, временных рядов и т. п.).
- Более плотное и эффективное кодирование информации по сравнению с MLP.
- Более эффективное расходование электроэнергии при физической реализации.
- Высокая степень эффективности параллельного выполнения при физической реализации.

К недостаткам можно отнести слабую изученность, сложность математической модели и вычислительную сложность при выполнении на обычном железе. А главный недостаток заключается в отсутствии хорошего алгоритма обучения.

Причины бурного развития ИНС сегодня

Казалось бы, раз ИНС такой мощный инструмент\метод и первая нейронная сеть была предложена так давно, то почему только сейчас появляется столько технологий на нейронных сетях? Тому есть две причины:

Чтобы исследовать глубокие ИНС, нужно проводить эксперименты, а они могли длиться по месяцу, а если и меньше, то нужно было все равно проводить десятки экспериментов. Таким образом, исследовать сети было сложно.

Поэтому появление соответствующих параллельных вычислителей и общее увеличение мощности компьютеров изменило эту ситуацию. Кроме того, появились обычные видео карты (GPU), на которых любой мог запускать сложные вычисления. Поэтому появилось много исследователей, которые проводили эксперименты. То есть, как ни странно, развитие тормозилось не из-за принципиальных возможностей ИНС, а из-за того, что экспериментировать было долго.

Функция активации «Relu» упростила функцию сетей в целом и сделала вычисления Глубоких сетей еще быстрее.

Психологический фактор. Извилистая история нейронных сетей и долгое пребывание в состоянии заморозки создали дурную славу нейронным сетям, и исследователи очень неохотно занимались этой сферой.

Возрастание интереса сразу сложило разные кусочки мозаики, и Сверточные сети получили вторую жизнь, а их успех дал еще больший толчок.

Стоит также отметить, что сегодня наблюдается обратный эффект. Пресса и Интернет часто преувеличивают возможности ИНС или допускают неточности, из-за чего может сложиться впечатление, что нейронные сети – это искусственный интеллект.

Борьба с переобучением в ИНС

Тема переобучения была подробно рассмотрена ранее, поэтому просто перечислим способы борьбы с этим эффектом, подходящие для большинства типов ИНС:

- Кросс-валидация. Данный метод не зависит от модели, это общий подход к тренировке моделей, помогающий контролировать и бороться с переобучением особенно на малых выборках.
- Уменьшение числа слоев\нейронов. Уменьшение слоев и нейронов явно влияет на сложность модели, однако необходимо искать компромисс между обобщающей способностью нейронов и степенью абстрактности признаков.
- Добавление L2 регуляризации. Аналогично Регрессии этот дополнительный метод штрафует высокие веса модели.
- Локальность восприятия. Задавая рецептивное поле, восприятие нейрона можно также уменьшить не только количество параметров или сложность модели, но, и более того, – определить влияющие факторы, что позволит нейрону решать локальные задачи и складывать их в глобальное решение. Это крайне важная особенность ИНС.

Обратное распространение ошибки

Итак, как и у любой другой модели в машинном обучении, у ИНС есть 2 этапа работы:

- обучение;
- использование, моделирование или получение отклика, прогноза (все это подразумевает расчет выхода по обученной модели).

При этом использование сети называют прямым распространением (forwardpropagation), потому что сигнал со входа сети распространяется к выходу (рисунок 38), то есть слева направо.

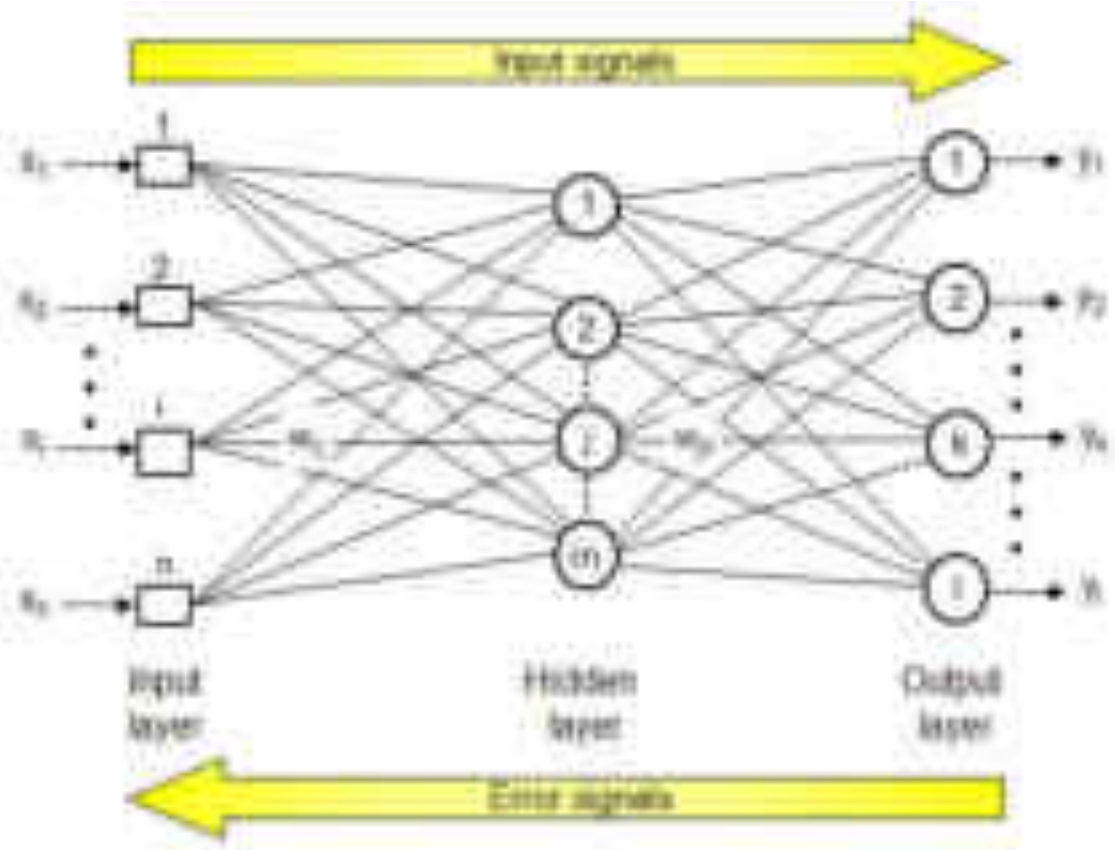


Рисунок 38. Обобщенная схема распространения сигналов по MLP

На рисунке 39 изображена принципиальная схема прямого распространения сигнала со входа на один нейрон. В случае нескольких нейронов в слое схожие вычисления будут проводиться для каждого нейрона. После чего сигнал двинется к следующему слою, и если в нем также несколько нейронов, то там будут сходные вычисления

для каждого нейрона и так далее. Каждый нейрон берет взвешенную сумму своих входов, считает функцию активации и выдает выход для следующего слоя и т. д.

Обучение называют обратным распространением (backpropagation) ошибки, потому что информация идет наоборот, с выхода сети, и информация эта – об ошибке, а не о входном образе, как при прямом распространении.

В модель нейрона на рисунке 39 включен пороговый элемент (bias), который обозначен символом b_k. Эта величина отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации. По сути, это свободный коэффициент при некотором признаке в нулевой степени (как и у Регрессии).

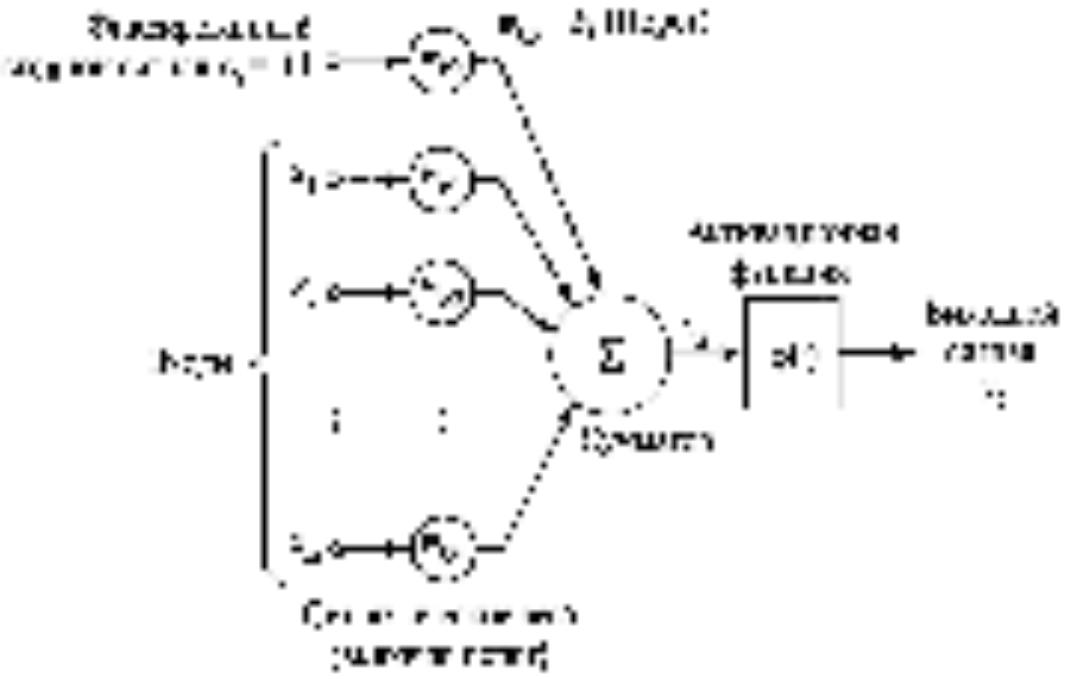


Рисунок 39. Принципиальная схема прямого распространения сигнала

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений:

$U_k = \sum_i^m w_{ki} x_i$,

$y_k = \varphi(U_k)$,

где U_k – индуцированное локальное поле, или потенциал активации.

Серия таких уравнений, вычисленных последовательно, в конце концов даст результат всей сети. Если в последнем слое сети один нейрон, то соответственно получим скаляр или число, а если нейронов несколько, то получим вектор значений для каждого входного образа X.

Теперь рассмотрим алгоритм обратного распространения ошибки для обучения искусственной нейронной сети. Он разработан в первую очередь для сетей MLP-типа. Но также подходит для сверточных сетей, автокодировщиков и, при определенных модификациях, для рекуррентных сетей.

Для начала введем определения:

Обучить сеть – минимизировать ошибку, а именно минимизировать разницу между выходом сети (реакцией на вход) и требуемой реакцией, то есть это обучение с учителем.

Минимизация ошибки производится путем итеративных правок\корректировок весов ИНС. Корректируются в ИНС только веса и больше ничего.

Минимизация ошибки происходит для каждого отдельно взятого входного образа. Повторяя такую минимизацию для каждого образа много раз, получаем общую минимизацию. Количество проходов по тренировочной выборке называется эпохами.

Начнем рассмотрение алгоритма с последнего (выходного) слоя сети (рисунок 40).

Сперва предположим, что выходной слой содержит только один нейрон. Тогда:

E, e – ошибка выхода сети для текущей пары (X, Y);

Y, y – требуемый выход для соответствующего входного образа X;

o = φ(U) – выход нейрона, рассчитывается как сигмоида от U;

U – результат взвешенной суммы, или индуцированное локальное поле;

Wij – вес, соединяющий некоторый нейрон слоя i с некоторым нейроном слоя j.

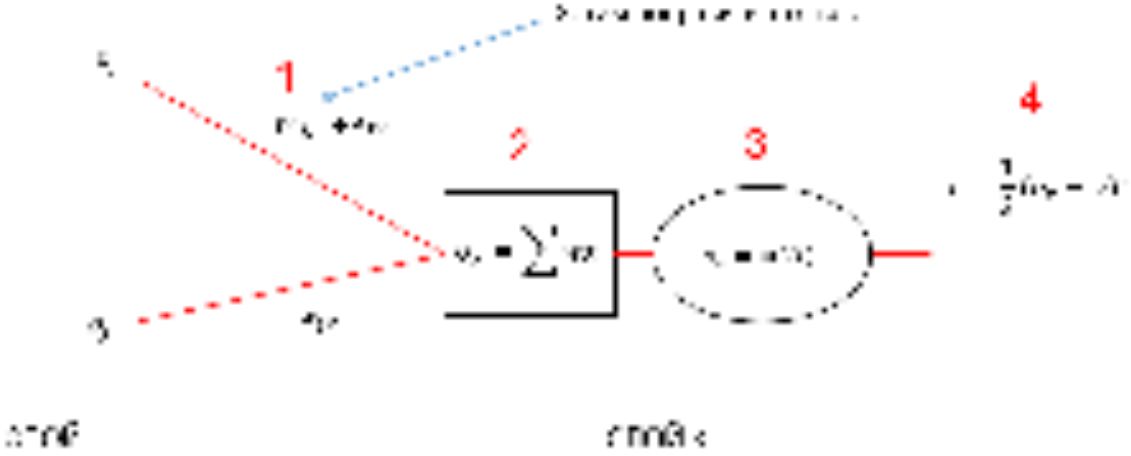


Рисунок 40. Представлена схема последнего слоя сети MLP

Индексы при символах означают индексы слоев. Так как в каждом слое имеется несколько нейронов, то о каком нейроне идет речь, когда пишется oj? О любом произвольном в этом слое, так как формула обобщенная и не вносит конкретики о номере нейрона внутри слоя.

Первая идея алгоритма\метода заключается в том, что мы двигаемся по градиенту ошибки, как и во всех численных градиентных методах. Мы не знаем, где точное решение, но если итеративно понемногу уменьшать ошибку для каждого входного образа, то мы рано или поздно придем к некому равновесному состоянию. Не обязательно минимуму ошибки, но к такому моменту, когда дальнейшие правки уже не будут коренным образом менять ситуацию, а система войдет либо в полную заморозку (изменений вообще не будет), либо в некий колебательный процесс около некоторой матрицы весов.

Собственно, двигаясь по антиградиенту ошибки $-\nabla F$, мы как раз и производим корректировки весов на некие дельта (в данном случае у весов нет никаких индексов, потому что это общая концепция правок):

$$w^{t+1} = w^t - \alpha \Delta w,$$

$$w^{t+1} = w^t - \alpha \nabla F(w^t),$$

$$\nabla F(w^t) = \frac{\partial E}{\partial w^t},$$

где w^{t+1} – вес в следующий момент времени;

w^t – вес в текущий момент времени;

Δw – правка веса на текущем шаге, в сторону уменьшения ошибки на текущем шаге;

α – размер шага, скорость движения по антиградиенту или сила правок весов.

Еще раз о том, почему используется именно градиент. Потому что нельзя рассчитать точное значение правок для всех весов сразу, ведь мы не знаем вклад каждого веса в ошибку, мы лишь знаем значение ошибки и направление. Чтобы понять суть идеи, представьте, что некая правка $\pm\Delta w$ некого веса w даст нам увеличение или уменьшение ошибки на выходе сети (см. рисунок 40). Получается, что небольшие правки весов приводят к некоторым небольшим изменениям конечной ошибки сети. Возможно ли найти функциональную зависимость между этими небольшими изменениями? Что есть отношение небольшого изменения ошибки к изменению какого-либо веса? Это производная $\frac{\partial E}{\partial w}$.

Таким образом, найдя аналитическое выражение производной ошибки по любому весу, мы определим характер воздействия правок этого веса на ошибку. А в конкретных точках этой функции (при конкретном входном образе и всех других параметрах сети) мы сможем точно подсчитать значение функции производной, то есть сможем оценить знак и значение dE, на которое будет меняться ошибка при наших правках dw. А значит, мы можем выбрать такую правку, чтобы уменьшить эту самую E. При этом все остальные веса, кроме того который правится в данный момент, замораживаем.

Итак, для последнего уровня все относительно просто. Чтобы найти производную $\frac{\partial E}{\partial w}$, необходимо взять серию производных по каждой вложенной функции (по цепному правилу дифференцирования):

$$w_{jk}^{t+1} = w_{jk}^{t} - \alpha\frac{\partial E}{\partial w_{jk}} =$$

$$= w_{jk}^{t} - \alpha\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial U_k}\frac{\partial U_k}{\partial w_{jk}} = w_{jk}^{t} - \alpha(o_k - y)o_k(1 - o_k)o_j.$$

Но что делать с внутренними слоями? Дело в том, что для нейронов внутренних слоев мы не можем явно рассчитать не то что значение правки, но и значение ошибки. Ведь чтобы рассчитать значение ошибки, надо знать, какое значение выхода должно быть у каждого нейрона внутреннего слоя. А мы не можем этого знать, ведь не знаем конкретный вклад каждого нейрона в ошибку. Мы знаем производную, то есть ближайший характер изменений ошибки от правок весов, но не можем посчитать нужные значения для произвольных точек.

Но что если пойти таким же путем, как и раньше? Допустим, что некая правка $\pm\Delta w$ некого веса w (теперь уже на скрытом слое) даст нам увеличение или уменьшение ошибки на выходе сети (см. рисунок 41). Получается, что небольшие правки этого веса тоже приводят к некоторым небольшим изменениям конечной ошибки сети при условии, что все остальные веса сети константны в рамках этого временного среза.

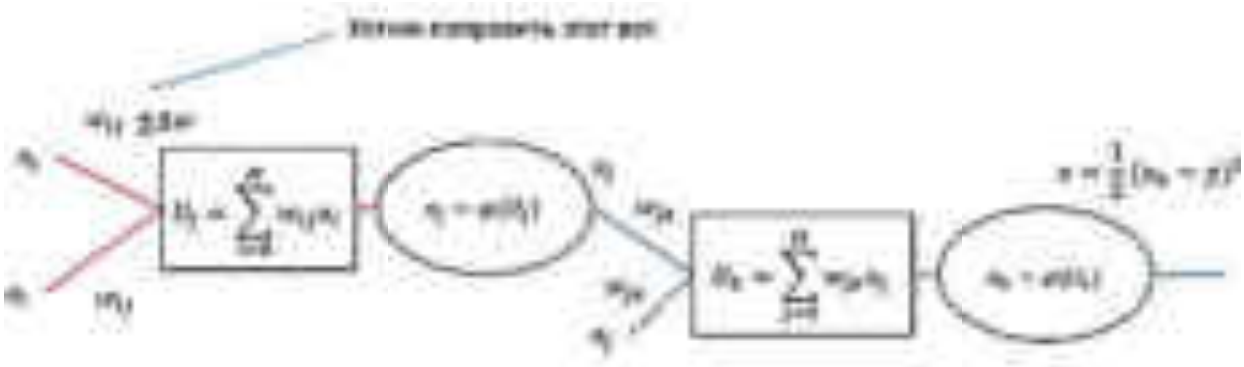
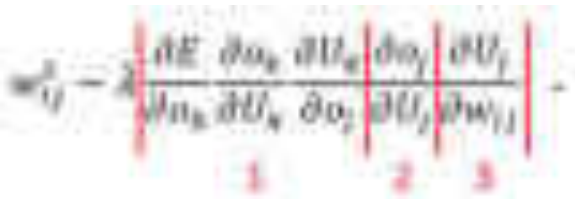


Рисунок 41. Схема последних двух слоев сети

Поэтому можно взять производную выходной ошибки по этому нейрону. Это будет такая же цепочка производных (только вес w_{ij} теперь будет константой, а не переменной, поэтому он и останется после взятия производной по o_j, а берем мы именно по o_j, чтобы пройти (протиснуть информацию) дальше).

$$w_{ij}^{t+1} = w_{ij}^{t} - \lambda\frac{\partial E}{\partial w_{ij}} = w_{ij}^{t} - \lambda\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial U_k}\frac{\partial U_k}{\partial o_j}\frac{\partial o_j}{\partial U_j}\frac{\partial U_j}{\partial w_{ij}}.$$

Теперь можно подставить каждую производную и получить формулу корректировки веса в скрытом слое. Однако выше мы делали серьезное допущение, что в последнем слое будет только один нейрон. А это может быть не так. И нас интересует именно этот общий случай и для любого слоя. Таким образом, из этой формулы надо выделить закономерность, которую можно использовать для произвольного слоя:



В формуле градиента ошибки по весу скрытого слоя присутствуют 3 составляющих. 1 – это составляющая следующего слоя, 2 – это составляющая текущего слоя и 3 – составляющая предыдущего слоя. То есть изменение выходной ошибки складывается из влияния этих составляющих, если все остальные элементы среды заморозить. А раз так, то можно сделать два вывода:

Первая составляющая в случае нескольких нейронов в следующем слое будет не одна. Понятно, что текущий нейрон j-го слоя распространяет свою ошибку на все связанные с ним нейроны слоя k. А значит, надо просуммировать вклад, который оказывает корректировка веса по всем путям. Или, иными словами, просуммировать производные по разным связям нейронов. Тогда первый блок можно будет переписать так:

$$\sum_{k=0}^{M} \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial U_k} \frac{\partial U_k}{\partial o_j} = (o_k - y)\varphi'_k w_{jk}.$$

Для сколь угодно далекого слоя можно не считать полный градиент, а брать вклад следующего слоя за основу. Ведь любой произвольный слой в первую очередь делает вклад именно в следующий слой. Таким образом, учитывая промежуточные вклады или точнее влияние всех промежуточных вкладов, мы сможем оценить итоговое влияние на ошибку. Опять же речь идет о небольших локальных приращениях $\pm\Delta w$ по конкретному w. Значит, можно выделить первую составляющую формулы в отдельную передаточную

величину вклада ошибки δ . Также ее называют производной ошибки по взвешенной сумме, по локальному индуцированному полю (по U). Тогда для последнего слоя:

$\delta_k = (o_k - y) o_k (1 - o_k)$.

А для любого другого скрытого слоя:

$\delta_j = (\sum \delta_k w_{jk}) o_j (1 - o_j)$.

Получается, что сигма рассчитывается как совокупность всех вкладов в ошибку (начиная с конца). И именно эта величина является той информацией, которая выражает ошибку от слоя к слою и распространяется обратно. Учитывая оба пункта одновременно, формулы для корректировок можно переписать следующим образом:

$\Delta w = -a\delta_k o_j = -a(o_k - y) o_k (1 - o_k) o_j$ – если корректируется вес последнего слоя.

$\Delta w = -a\delta_j o_i = a(\sum \delta_k w_{jk}) o_j (1 - o_j) o_i$ – если корректируется вес скрытого слоя.

Общий алгоритм можно представить так:

- Инициализировать веса случайным образом;
- Рассчитать выход сети прямым распространением сигнала;
- Рассчитать ошибку на выходе;
- Рассчитать корректировки весов текущего слоя по ошибке следующего;
- Обновить веса.

Условия завершения алгоритма обратного распространения ошибки:

- Требуемая величина ошибки;
- Максимальное количество итераций;
- Последние N проходов веса не правились больше чем на T.

Замечание 1: шаг градиентного спуска или скорость градиентного спуска a крайне важный параметр и нужен не только ради формальности. Дело в том, что при $a = 1$ движение по градиенту ошибки будет слишком быстрым, т.к. каждая корректировка отдельно взятого веса производиться с учетом заморозки всех остальных весов.

А следующий вес корректируется на основании исходного значения ошибки (вычисленного еще до правки первого веса). В противном случае движение по градиенту будет плавным, но это очень расточительно с точки зрения вычислительных ресурсов (такой алгоритм будет очень медленным). Кроме того, точный пересчет в рамках одного образа и не нужен, так как полное уменьшение ошибки для одного образа не гарантирует факт того, что уменьшиться общая ошибка. А уменьшить необходимо именно общую ошибку, а значит в рамках одного образа корректировки должны быть небольшими. Уточним, что на практике часто принимают $a = 0.01$.

Замечание 2: конечно, рассмотренный алгоритм не лишен недостатков. Некоторые из них очевидны и их возможно устранить. Существует множество модификаций градиентного спуска, например: Adadelta, Adam и т. п. Есть и другие методики улучшения обучения. Один из самых мощных и простых – это Batching. Но все эти методы выходят за рамки пособия, так как при необходимой фундаментальной подготовки вы можете изучить их особенности самостоятельно.

Замечание 3: алгоритм обратно распространения ошибки линеен с точки зрения вычислительной сложности.

Замечание 4: более подробное рассмотрение модификаций алгоритма обратного распространения выходит за рамки этого пособия, как и рассмотрение альтернативных функций активации (таких как ReLU, LeakyReLU, Parametric ReLU, RandomizedReLU). Поэтому вам предлагается изучить эти вопросы самостоятельно.

Нечеткие нейронные сети

Нечеткой нейронной сетью (НС) обычно называют четкую нейросеть, которая построена на основе многослойной архитектуры с использованием специальных «И»-, «ИЛИ»-нейронов.

Нечеткая нейросеть функционирует стандартным образом на основе четких действительных чисел, нечеткой является только интерпретация результатов.

Нечеткие нейронные сети осуществляют выводы на основе аппарата нечеткой логики, а параметры функций принадлежности настраиваются с использованием алгоритмов обучения НС. Поэтому для подбора параметров таких сетей применим метод обратного распространения ошибки, изначально предложенный для обучения многослойного персептрона. Нечеткая нейронная сеть, как правило, состоит из четырех слоев: слоя фаззификации входных переменных, слоя агрегирования значений активации условия, слоя агрегирования нечетких правил и выходного слоя.

Наибольшее распространение в настоящее время получили архитектуры нечеткой НС вида ANFIS и TSK [18]. Доказано, что такие сети являются универсальными аппроксиматорами. Быстрые алгоритмы обучения и интерпретируемость накопленных знаний – эти факторы сделали сегодня нечеткие нейронные сети одним из самых перспективных и эффективных инструментов мягких вычислений.

Преимущества нечетких нейронных сетей

Основным преимуществом технологии нейрокомпьютинга служит возможность выразить зависимость «выход-вход» без предварительной аналитической работы по выявлению правил, а на основе обучения на примерах. Недостатком нейросетей является невозможность объяснить выходной результат, так как значения распределены по нейронам в виде значений коэффициентов весов. Основной трудностью в применении нечетких экспертных систем служит необходимость явно сформулировать правила проблемной области в форме правил. В нечетких экспертных системах легко построить объяснение результата в форме протокола рассуждений. Поэтому в настоящее время создаются гибридные технологии.

Примером гибридной технологии служит реализация базы нечетких правил на основе нейросети. База нечетких правил для двух входных переменных имеет следующую структуру:

$R_i: if\ x_{1i}\ \ is\ \ A_{1i}\ \ and\ x_{2i}\ \ is\ \ A_{2i}\ \ then\ z_i\ \ is\ \ C_i.$

Простой реализацией базы нечетких правил служит интерпретация базы правил как таблицы определения некоторой функции, то есть базу правил можно представить обучающей выборкой:

$$\{((A_{1i}, A_{2i}), C_i)\}.$$

Например, обучающая выборка в нечетких терминах может быть сформулирована следующим образом:

{((малое, большое), около нуля)}.

Чтобы совместить две технологии: технологию нечетких систем и технологию нейрокомпьютинга, необходимо предложить способ четкого дискретного представления непрерывных функций принадлежности. Чтобы представить в четких данных непрерывные функции принадлежности, выберем максимально большой интервал $[x_1,x_2]$, в котором представлены все нечеткие множества условных частей правил. Разбиваем интервал с равным шагом, тогда любое нечеткое значение представляется четким вектором. Другой способ представления нечеткого понятия в вид четких данных состоит в представлении нечеткого множества в виде совокупности α-срезов.

α-срезом называется четкое множество, включающее все элементы x некоторого нечеткого множества X, принадлежность которых больше равна α.

$$\mu_X(x) \geq \alpha .$$

Каждое α-подмножество представляется двумя числами – левой и правой границей α^L, α^R, то есть α-срезы четко представляют непрерывную функцию принадлежности.

Изменение элемента нейросети для адаптации к нечетким системам может касаться выбора функции активации, реализации операций сложения и умножения, так как в нечеткой логике сложение моделируется любой треугольной конормой (*max, a + b – a×b...*), а операция умножения треугольной нормой (*min, a×b ,...*).

И-нейроном (AND-нейроном) называется нейрон, в котором умножение веса на вход моделируется конормой *S(w,x)*, а сложение – нормой *T(w,x)*.

ИЛИ-нейроном (OR-нейроном) называется нейрон, в котором умножение веса и входа моделируется нормой $T(w,x)$, а сложение взвешенных весов конормой $S(w,y)$ $Y = S(T(w_1,x_1),T(w_2,x_2))$.

Пример: $(max(min(w_1,x_1),min(w_2,x_2)))$.

В качестве функции активации обычно используют функцию

$F(x) = 1/(1+exp(b(x-a)))$.

Нечеткой нейросетью называют четкую нейросеть, которая построена на основе многослойной архитектуры с использованием «И-», «ИЛИ-нейронов».

Нечеткая нейросеть функционирует стандартным образом на основе четких действительных чисел. Нечеткой является только интерпретация результатов. При создании гибридной технологии, кроме объединения систем по данным, можно использовать нейрокомпьютинг для решения частной подзадачи нечетких экспертных систем, а именно настройки параметров функции принадлежности. Функции принадлежности можно сформировать двумя способами: методом экспертной оценки или на основе статистики. Гибридные технологии предлагают третий способ: в качестве функции принадлежности выбирается параметризованная функция формы (например, параметризованная Гауссова кривая), параметры которой настраиваются с помощью нейросетей. Настройка параметров может быть получена в рамках алгоритма обратного распространения ошибки.

Пусть задана следующая система нечетких правил:

If x_1 *is* A_{1i} *and ...* x_n *is* A_{ni} *then* z_i *is* C_i ,

где A_i – нечеткие числа; C_i– действительные числа. Значение $\alpha_j=\Pi_i\alpha_i$ – сила или достоверность правила, $i = 1,..,n$ – номер входной переменной, $j = 1,..,m$ – количество правил.

$$Z = \frac{\sum_{j=1}^{m} a_j z_j}{\sum_{j=1}^{m} a_j},$$

где Z – вычисленное значение выхода.

Допустим, что разработана нейросеть с n входами и одним выходом. Каким образом такая НС может аппроксимировать базу нечетких правил? Любая совокупность нечетких правил может рассматриваться как нелинейное соответствие, заданное таблицей определения $\{(x,y)\}$, где x – вектор входа; y – желаемое значение выхода; а z – значение, вычисляемое нейросетью. Тогда можно определить текущую ошибку:

$E^K=1/2\times(z^K-y^K)^2$.

То есть можно применить стандартный алгоритм коррекции ошибки на основании данного определения

$Z(t+1) = Z(t)- \zeta\times (\partial E^K/\partial Z)$,

где ζ – уровень обучения; $\partial E^K/\partial Z$ – направление градиента снижения ошибки.

$Z(t+1) = Z(t)- \zeta\times (Z^K-Y^K) \times \alpha_j / (\alpha_1 +\alpha_2 +...+\alpha_m)$.

При применении стандартного алгоритма обратного распространения ошибки для того, чтобы настроить выход, необходимо изменить параметры функции принадлежности условных частей, то есть обучение сети позволит настроить функцию принадлежности с точки зрения обучающей выборки. При практической реализации системы нечетких правил важным является вопрос о типичных представителях нечетких значений в правилах. Большинство нечетких понятий, представленных лингвистическими переменными, выражает свои значения с помощью количественных нечетких множеств: NB – отрицательное большое; NM – отрицательное среднее; NS – отрицательное малое; ZE – около нуля; PS – положительное малое; PM – положительное среднее; PB – положительное большое.

Пример использования нечеткой нейронной сети

Рассмотрим нечеткие нейронные сети на примере нечеткого регулятора для стиральной машины [18], то есть построим fuzzy-neuro контроллер. Выберем для демонстрации технологии нечетких нейронных сетей архитектуру ANFIS (Adaptive Network Based Fuzzy Inference System). Основа интеграции нейронных сетей и систем нечеткого вывода заключается в том, что оба метода представляют

нелинейное отношение в пространстве входы-выходы. Важная задача нечеткого моделирования – это настройка функций принадлежности, являющаяся по существу задачей оптимизации. Как нейронные сети, так и генетические алгоритмы используются для ее решения. Самый простой подход требует назначить определенную параметризованную функцию формы в качестве функции принадлежности и подобрать параметры на основе обучения нейронной сети. Рассмотрим простой пример с тремя нечеткими правилами вывода в базе знаний.

R_1: ЕСЛИ <количество_белья> is <много>

И <температура_воды> is <высокая> И <загрязненность> is <высокая>

ТО <длительность> is <высокая>;

R_2: ЕСЛИ <количество_белья> is <много>

И <температура_воды> is <высокая> И <загрязненность> is <низкая>

ТО <длительность> is <низкая;

R_3: ЕСЛИ <количество_белья> is <мало>

И <температура_воды> is <низкая> И <загрязненность> is <низкая>

ТО <длительность> is <малая>;

Зададим следующие функции формы для высказываний. Пусть высказывание <количество_белья> is <мало> соответствует функции $L_1(x)$ и высказывание <количество_белья> is <много> – функции $H_1(x)$.

$L_1(x)= 1/(1+exp(b_1(x-c_1))$,

$H_1(x)= 1/(1+exp(-b_1(x-c_1))$,

причем $L_1(x) + H_1(x) =1$.

Аналогично для высказывания <температура_воды> is <высокая> будем использовать функцию $L_2(t)$ и для <температура_воды> is <низкая> – функцию $H_2(t)$.

$L_2(t)= 1/(1+exp(b_2(t-c_2))$,

$H_2(t)= 1/(1+exp(-b_2(t-c_2))$,

причем$L_2(t) + H_2(t) =1$.

Для высказывания <загрязненность> is <высокая> определим функцию $L_3(z)$ и для <загрязненность> is <низкая> – функцию $H_3(z)$.

$L_3(x)= 1/(1+exp(b_3(z-c_3))$,

$H_3(x)= 1/(1+exp(-b_3(z-c_3))$,

$L_3(z) + H_3(z) =1$.

Для выходов <длительность> is <высокая> и <длительность> is <малая> аналогично определим функции

$L_4(y)= 1/(1+exp(b_4(y-c_4))$,

$H_4(y)= 1/(1+exp(-b_4(y-c_4))$,

то есть $L_4(x) + H_4(x) =1$.

Для четких значений <количество_белья>, <температура_воды> и <загрязненность>: A_1, A_2, A_3 определим релевантность (силу) правил α:

$\alpha_1 = H_1 \wedge H_2 \wedge H_3$,

$\alpha_2 = H_1 \wedge H_2 \wedge L_3$,

$\alpha_3 = L_1 \wedge L_2 \wedge L_3$.

Выходы по каждому из правил определяется с помощью обратных функций принадлежности правых частей правил.

$Y_1 = H_4^{-1}(\alpha_1)$,

$Y_2 = H_4^{-1}(\alpha_2)$,

$Y_2 = L_4^{-1}(\alpha_3)$.

Общий выход из системы нечетких правил определяется как

$y_0 = (\alpha_1 \times y_1 + \alpha_2 \times y_1 + \alpha_3 \times y_1)/(\alpha_1 + \alpha_2 + \alpha_3)$.

Далее построим нечеткую нейронную сеть, идентичную системе нечеткого вывода, и обучим функции принадлежности анцедента и консеквента правил (рисунок 42).

Слой 1. Выходы узлов – это степени, в которых заданные входы удовлетворяют функциям принадлежности, ассоциированным с этими узлами.

Слой 2. Каждый узел вычисляет силу правила. Выход верхнего нейрона $\alpha_1 = H_1 \wedge H_2 \wedge H_3$, выход среднего нейрона $\alpha_2 = H_1 \wedge H_2 \wedge L_3$, а выход нижнего – $\alpha_3 = L_1 \wedge L_2 \wedge L_3$. Все узлы помечены Т, так как

можно выбрать любую Т-норму для моделирования логического И. Узлы этого слоя называются узлами правил.

Слой 3. Каждый узел помечен N, чтобы показать, что узлы нормализуют силу правил $\beta_i = \alpha_i / (\alpha_1 + \alpha_2 + \alpha_3)$.

Слой 4. Выход нейронов – это произведение нормализованной силы правила и индивидуального выхода соответствующего правила.

$\beta_1 Y_1 = \beta_1 H_4^{-1}(\alpha_1)$,

$\beta_2 Y_2 = \beta_2 H_4^{-1}(\alpha_2)$, $\beta_3 Y_2 = \beta_3 L_4^{-1}(\alpha_3)$.

Слой 5. Одиночный выходной нейрон вычисляет выход сети

$y_0 = \beta_1 Y_1 + \beta_2 Y_2 + \beta_3 Y_2$.

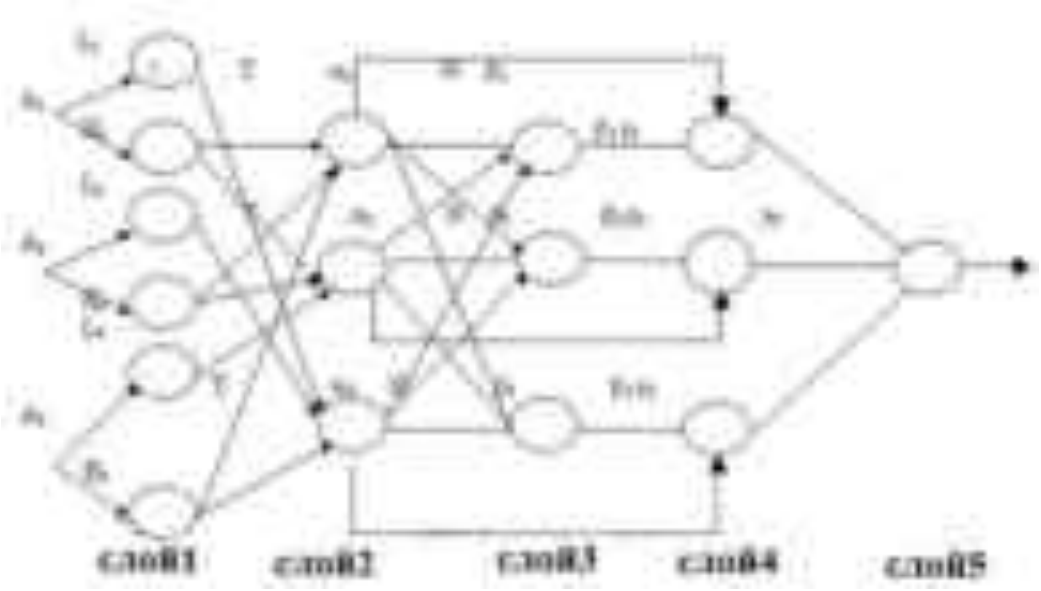


Рисунок 42. Пример нечеткой нейронной сети

Алгоритм обучения для нечеткой нейронной сети примера

Пусть задана четкая обучающая выборка

$\{(x_1,t_1, z_1 y_1), \ldots (x_k, t_k, z_k y_k)\}$, где x,t,z – входные условия количества белья, температуры воды и загрязненности, а y – длительность стирки. Ошибку на k-м образце определим как обычно:

$E^k = 1\backslash 2(O_i - Y_i)^2)$.

Используем традиционный градиентный метод для обучения параметров левой и правой частей нечетких правил. Покажем, как можно настроить параметры функции формы.

$b_4(t+1) = b_4(t) - \zeta \times (\partial E/\partial b_4)$,

$c_1(t+1) = c_1(t) - \zeta \times (\partial E/\partial c_1)$

.....

$c_4(t+1) = c_4(t) - \zeta \times (\partial E/\partial c_4)$.

Используя данные соотношения как правила изменения весов в алгоритме обратного распространения ошибки, можно настроить параметры функций принадлежности в ходе обучения нечеткой нейронной сети.

Генетические алгоритмы

Генетические алгоритмы представляют собой адаптивные методы поиска, использующиеся для решения задач функциональной оптимизации. В их основе лежит идея природной эволюции, когда популяции развиваются в течение нескольких поколений и подчиняются законам естественного отбора. Лучшая особь выбирается по принципу «выживает наиболее приспособленный». Как мы помним, он был открыт Чарльзом Дарвином. Согласно этим принципам генетические алгоритмы способны «развивать» решения реальных задач при соответствующем формальном описании. Причем в отличие от эволюции, генетические алгоритмы моделируют лишь существенные для развития процессы.

В природе действует такой механизм, что наиболее приспособленные особи имеют больше шансов на воспроизведения потомства. Причем комбинация наиболее хороших характеристик может привести к появлению максимально приспособленного потомка. Генетические алгоритмы используют прямую аналогию с этим механизмом. Множество возможных решений проблемы – это популяция. Каждое решение оценивается по степени его «приспособленности» для решения поставленной задачи.

Сфера применения генетических алгоритмов весьма обширна. Они могут использоваться при создании таких вычислительных структур, как автоматы или сети сортировки. В машинном обучении их

можно использовать для проектирования нейронных сетей или в управлении роботами. Их можно использовать для моделирования процессов в различных областях (биологические, социальные, когнитивные системы). Однако наиболее популярное приложение – оптимизация многопараметрических функций, когда нужно найти оптимальное значение, зависящее от некоторых входных параметров.

В своей работе генетический алгоритм работает с хромосомами, которые состоят из генов. Чаще всего ген – это 0 или 1, говорящая о включении или не включении признака, характеризующего решение, в хромосому. Соответственно, хромосома – это битовая строка, описывающая решение. Но иногда возможны и другие кодировки в зависимости от задачи.

При работе с генетическим алгоритмом нам необходимо определить следующее:

Кодировку хромосомы (что будет представлять собой ген, и как гены будут участвовать в характеристике решения).

Пространство гипотез (популяцию), из которых мы должны выбрать лучшую.

Функцию приспособленности, оценивающую хромосомы.

Набор и вид генетических операций (скрещивание, мутацию).

Критерий остановки алгоритма (либо желаемое оптимальное значение, либо количество шагов эволюции популяции).

Для примера рассмотрим решение задачи с помощью генетического алгоритма. Формулировка задачи: необходимо составить наиболее сбалансированный рацион питания, удовлетворяющий определенным медицинским требованиям. Известен перечень продуктов из N наименований, каждый из которых имеет такие характеристики, как содержание жиров, белков, углеводов, калорий, а также известны рамки допустимого их содержания. Подходящим считается рацион, который лучше всего удовлетворяет определенным медицинским требованиям. Например, если рекомендуемое содержание жиров – 50, белков – 60, углеводов – 70, клетчатки – 80, витамина B – 90, витамина C – 100, то лучшим будет считаться рацион, чье суммарное

отклонение минимально, но чья десятая часть, например, укладывается в рамки от 40 до 80.

Первое, что нам необходимо сделать, – выбрать кодировку хромосомы. Так как основная наша задача – подобрать список продуктов, то хромосомой может быть весь перечень из N продуктов, а 0 или 1 в ней будут говорить о включении или не включении продуктов в рацион. Для лучшего понимания рассмотрим конкретный пример. Пусть у нас есть 4 продукта, каждый из которых можно описать набором из 6 числовых параметров-характеристик. Тогда по сути получим массив чисел:

Список_объектов={Объект$_1$={20,30,40,50,60,70},

Объект$_2$={40, 40, 40, 40, 40, 40},

Объект$_3$={30, 30, 30, 30, 30, 30},

Объект$_4$={20, 30, 40, 40, 30, 30}}.

Тогда хромосома может иметь вид:

1010

Это говорит о том, что в рацион включаются первый и третий объекты. Тогда популяция может иметь вид:

0100

1010

1100

0101

Теперь определим функцию приспособленности для оценки каждой хромосомы в нашей задаче. Пусть у нас есть набор эталонных значений для каждой из характеристик объектов:

Эталон= {50,60,70,80,90,100}.

Тогда функция приспособленности будет иметь вид:

$$F = \sum_{j=1}^{N} \left(\frac{\sum_{i=1}^{m} \text{Эталон}_i - \text{Объект}_{ij}}{10} \right) \times Hrom_j \ ,$$

где $Hrom_j$– соответствующий ген в хромосоме.

Для оценки приспособленности хромосомы необходимо также проверить, укладывается ли значение функции в рамки от 40 до 80 (добавлением соответствующих условий), а для выявления наиболее приспособленной особи необходимо найти минимальную подходящую *F* в популяции.

Обратите внимание!

- *Определение вида функции приспособленности – чрезвычайно важная задача, потому что от правильности ее определения будет во многом зависеть успешность решения.*

Теперь рассмотрим генетические операции мутации и скрещивания. Если говорить формально, то мутация – это изменение одного или нескольких генов хромосомы вследствие случайного влияния. В контексте нашей задачи это может быть принудительное включение некоего продукта в набор, изменение значения вхождения продукта в набор на противоположное, принудительное исключение некоего продукта из набора и т. д. То есть, по сути, мутация – это изменение значения одного или нескольких генов хромосомы на противоположный или четко заданный. Вид и критерий применения мутации выбираются эмпирически. Допустим, для нашей задачи мы решили, что мутация будет изменять случайный ген на противоположный. Тогда, если исходная хромосома имела вид 0010, то мутировавшая может быть такой: 1010.

Операция скрещивания, или кроссовер, – операция, которая получает из двух хромосом одну, используя заданную маску. По сути из каждой хромосомы «вырезается» кусок, который помещается в новую. Существует несколько видов кроссовера. Одноточечный кроссовер: «разрез» хромосомы происходит только в одной точке, и новая особь получается путем соединения первой части первой хромосомы и второй части второй хромосомы. Двухточечный кроссовер: есть две точки «разреза», и новая хромосома получается из двух частей первой хромосомы и одной части второй.

Одноточечный кроссовер:

Хромосома$_1$ = 1001, Хромосома$_2$=1100. Точка разреза = 2. Маска потомка: 1часть_Хромосома$_1$+2часть_Хромосома$_2$. Потомок=(10)(00).

Двухточечный кроссовер:

Хромосома$_1$ = 1001, Хромосома$_2$=1100. Точки разреза = 1,3. Маска потомка: 1часть_Хромосома$_1$ + 2часть_Хромосома$_2$ + +3часть_Хромосома$_1$. Потомок=(1)(10)(1).

Теперь разберемся с таким вопросом, как отбор хромосом для воспроизведения потомства и выживание в популяции. Обычно в популяцию выбирается s наиболее приспособленных особей, которые и дают в ней потомство. Для того чтобы выбрать хромосомы в популяцию, можно использовать разные методы: турнирный, ранговый или метод рулетки. Рассмотрим эти методы подробнее.

Турнирный метод: задаем некую фиксированную вероятность выживаемости хромосомы, обозначив ее p. Случайно выбираем две особи. С вероятностью p выживает наиболее приспособлена, а 1-p – менее приспособленная.

Метод рулетки: вычисляем удельную приспособленность каждой особи относительно суммарной приспособленности популяции. Эту величину используем в качестве значения вероятности выживания.

Ранговый метод: выживает s наиболее приспособленных особей.

Таким образом, общую схему генетического алгоритма можно описать так:

Генерация начальной популяции из N особей.

Оценка приспособленности особей.

Пока не сработало условие выхода, делаем следующее:

Выберем s особей для новой популяции и кроссовера.

Выполним кроссовер и мутацию.

Оценим приспособленность итоговой популяции.

Нечеткие системы с генетической настройкой

Настройка функций принадлежности с помощью нейронной сети или генетического алгоритмы (ГА) устраняет принципиальную слабость теории нечетких систем – субъективность функций принадлежности. Применение нейронных сетей к настройке функций

принадлежности позволяет рассматривать окончательную форму функции как аппроксимацию обучающей выборки. Такую же задачу можно решить с помощью ГА, как метода стохастической оптимизации. Генетической нечеткой системой называют нечеткую систему, функции принадлежности и база правил которой спроектирована с помощью генетического алгоритма.

Применить ГА – это значит выбрать единицу кодирования, то есть хромосому; уточнить эволюционные операторы рекомбинации, мутации и селекции; сформировать функцию адаптивности (fitness-function, performanceindex). В настоящее время ГА используют либо для настройки функций принадлежности (базы данных), либо для формирования базы правил, либо для одновременного формирования и функций принадлежности и правил (а именно, базы знаний). В соответствии с объектами оптимизации выбирают единицу кодирования – хромосому. Для настройки функций принадлежности (ФП) за хромосому выбирают одно правило (Мичиганский подход), для настройки базы правил за хромосому выбирают вариант базы правил (Питтсбургский подход, подход итеративного обучения правил). В соответствии с кодированием уточняются правила генерации новых хромосом. Функция адаптивности представляет собой механизм нечеткого вывода, который для каждого варианта базы правил строит либо управление для нечеткого контроллера, либо экспертное заключение для диагностической экспертизы. Как видно из механизма нечеткого вывода, все нечеткие правила вносят вклад в окончательный результат, то есть правила сотрудничают. Но при отборе правил (хромосом) для генерации новых правил ГА накапливает правила, внесшие максимальный вклад в общий результат, то есть хромосомы конкурируют. В этом случае имеет место проблема «конкуренции и кооперации» в генетических нечетких системах (acompetition vs. a cooperation). Решение проблемы в каждом конкретном случае гене-тической нечеткой системы (ГНС) строится эвристически, например, при подходе итеративного обучения правил используют два этапа оптимизации. На первом шаге правила конку-

рируют за право войти в базу правил, а на втором – взаимодействуют при формировании общего результата.

Нечеткие нейронные сети с генетическим проектированием

Рассмотрим возможности генетических вычислений как средства структурной оптимизации нечетких нейронных сетей. Генетические вычисления можно применить на этапе проектирования нейронной сети. Возможности нейронных сетей интерполировать значения временных рядов могут быть широко использованы для оценки поведения макроэкономических показателей, в том числе индексов деловой активности или уровня ценных бумаг. Задача построения нейронного предиктора связана с принятием решений во многих точках проектирования. Необходимо исследовать входные данные и решить, по какому отрезку входных данных рационально делать предсказания, сколько точек в будущем разумно предсказать. Пространство перебора решений организации входных и выходных данных огромно для развитых рынков ценных бумаг, накопивших статистические сведения за десятки лет. Следующей точкой решения служит выбор типа нейронной сети: многослойный персептрон, радиально базисная сеть, вероятностная нейронная сеть, сеть регрессии и т. д. Для выбранного типа НС необходимо определить количество скрытых слоев, количество нейронов в них, виды функций активации и другие. Для каждой сети необходимо выбрать алгоритм обучения и его параметры, например, использование моментов (уровень моментов), уровень обученности, целевой уровень накопленной ошибки и т. д. Для нечеткой нейронной сети необходимо определить архитектуру сети, параметры функций принадлежности. В результате пространство решений при формировании нейронной сети становится необозримым для исследователя, и целесообразно применить ГА как средство эволюционного проектирования.

Генетическая оптимизация F-преобразования временных рядов

Задачи переборного типа с меньшими затратами могут быть решены путем применения эволюционных алгоритмов. Предлагается следующий классический алгоритм генетической оптимизации:

Имеется функция $F(x_1, \ldots, x_4)$ – оценка построения прогноза тренда (должна быть минимизирована), где

x_1 – степень авторегрессии при построении прогноза тренда;

x_2 – метод, которым производится прогноз;

x_3 – количество точек, покрываемых базисной функцией;

x_4 – прогноз на основании истории из предыдущих N точек.

Для выявления сезонности следует строить прогноз, отодвигая историю на t точек назад:

$$F_k = \alpha F_{k-t} + \beta F_{k-t-1} + \ldots$$

Алгоритм генетической оптимизации

1. Хромосомы имеют битовое представление (кодируем в коде Грея для получения отличия соседних хромосом в 1 бите). При этом в начале производится нормировка значений параметров к интервалу [0, 1].
2. Оператор скрещивания (возможен случайный выбор между данными вариантами оператора скрещивания).
3. Сформировать случайно начальную популяцию, состоящую из k особей $B_0 = \{A_1, A_2, \ldots, A_k\}$. При этом k определить эмпирически.
4. Вычислить приспособленность каждой особи $F_{Ai} = fit(A_i)$, $i=1 \ldots k$.
5. Выбрать двух особей A_c из популяции. $A_c = Get(B_t)$.
6. Произвести скрещивание (выбрав один из предложенных вариантов) и получить новую особь.
7. Произвести мутацию генов с некоторой вероятностью.

8. Оценить полученную особь функцией приспособленности и добавить в популяцию взамен худшего родителя (таким образом сохраняется количество особей популяции).

9. Выполнить пункты 5-8 m раз.

10. Увеличить счетчик эпох.

11. Проверить условие останова (предельное количество эпох или предельное количество особей одного генотипа в популяции).

РАЗРАБОТКА ПРИЛОЖЕНИЙ В СФЕРЕ МАШИННОГО ОБУЧЕНИЯ

Для реализации полноценного решения в сфере ПО необходимо учесть аппаратную и программную составляющую. Под аппаратной составляющей подразумевается следующее. Машинное обучение в ряде случаев требует подбор специального оборудования (или конфигурирование специальных серверов), рассчитанного на массивные вычислительные нагрузки или на тяжелые параллельные вычисления. Причем отличительной чертой можно назвать то, что мощное оборудование может потребоваться не только в производственной стадии проекта, но даже на ранних стадиях разработки для проведения экспериментов и соответствующего исследования моделей. Без мощного оборудования требуемые модели могут работать в 10-60 раз медленнее или просто не запуститься ввиду нехватки памяти (или видеопамяти).

В случае разработки встраеваемых решений может потребоваться сильная оптимизация кода и квалификация в написании кода под специальное оборудование (например, программируемые логические интегральные схемы). В случае разработки распределенных систем потребуется также применение специальных программных средств для организации распределенных вычислений.

Эти специализированные темы в данном пособии не рассматриваются.

Однако если речь не идет о разработке специализированного программно-аппаратного комплекса или встраевоемого решения, то наращивание вычислительных узлов на сегодняшний день является не столько инженерной проблемой сколько экономической.

Что же касается программной составляющей, то здесь стоит упомянуть следующее. На сегодняшний день огромное число различных алгоритмов уже реализовано в виде библиотек и пакетов, поэтому нет необходимости писать алгоритм нейронной сети или

агоритм решающих деревьев. За исключением опять же специализированных задач или исследовательских проектов.

Теоретически разработку систем МО можно вести на любом языке программирования, но при отсутствии специализированных библиотек сложность разработки может возрасти. Однако необходимо учесть, что специализированные средства создания моделей МО могут быть сложно применимы при реализации полноценных приложений. Поэтому нередко используют комбинацию технологий, когда предобработку данных и саму модель реализуют на одном языке программирования, а затем работающий прототип встраивают в приложение, написанное с использованием другой технологии. Одна из возможных архитектур реализации представлена на рисунке 43.

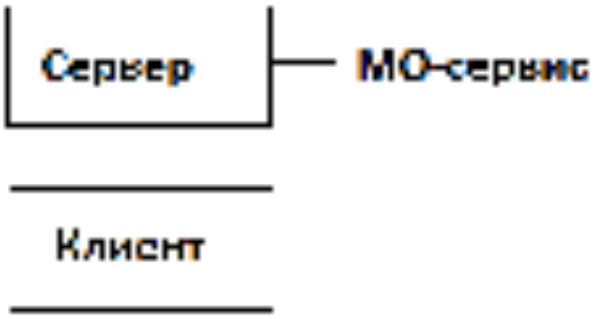


Рисунок 43. Возможная архитектура системы МО

Язык Python является одним из самых популярных языков для разработки в сфере машинного обучения и анализа данных ввиду очень простого синтаксиса, большого числа библиотек и развитого сообщества. Несмотря на то, что это интерпретируемый язык и выполнение кода на нем довольно медленное, большая часть библиотек реализована на быстрых языках типа Си, C++ и т. п. Поэтому на Python приходиться писать лишь высокоуровневое обращение к API библиотек. Что делает разработку на Python удобной, а решения – не уступающими по скорости со специальной реализацией на Си. В силу вышеизложенного будем работать именно с Python.

Однако прежде чем перейти непосредственно к Python, необходимо сказать, что при разработке систем МО, как правило, приходится пройти несколько типовых этапов. Таким образом, можно сформулировать общий алгоритм создания систем МО, который может быть изменен в силу особенностей конкретного проекта.

Алгоритм состоит из следующих шагов:

1. Сбор и предобработка данных;

2. Выбор метода решения, создание и настройка модели(подбор параметров, реализация прототипа);

3. Тестирование модели и совершенствование ее до достижения необходимой точности. При невозможности достижения точности – возврат к шагу 2;

4. Сохранение модели и интеграция ее в оболочку взаимодействия с пользователем;

5. Эксплуатация и сопровождение полученной системы и ее модернизация (при необходимости).

Основы работы с Python

Теперь перейдем к рассмотрению возможностей, предоставляемых Python для разработки систем МО. Общие сведения об этом языке можно посмотреть здесь: [30], а справочную информацию по синтаксису – здесь: [31]. В нашей же книге мы рассмотрим лишь то, что касается возможностей Python в плане создания приложений в сфере машинного обучения. Все примеры кода подготовлены на Python версии 2.7, но они могут быть адаптированы и под более старшие версии языка.

Как было сказано выше, сейчас Python является одним из наиболее распространенных языков программирования. Одним из его преимуществ является большое количество пакетов, решающих самые разные задачи. В данном пособии мы рекомендуем использовать библиотеки Pandas, NumPy и SciPy, которые существенно упрощают чтение, хранение и обработку данных. Вы также познако-

митесь с пакетом Scikit-Learn, в котором реализованы многие алгоритмы машинного обучения.

Необходимо отметить, что основными отличиями Python являются:

- Отсутствие завершающих символов в конце строки (точек, запятых и т. п.), что делает написание линейных конструкций очень «приятным» и быстрым занятием (а большинство программ для машинного обучения – это все-таки линейная математика, а не сложные многоуровневые системы).
- Нестрогая типизация. Не нужно объявлять тип данных при объявлении переменной, что опять же ускоряет процесс разработки модели.
- Язык Python интерпретируемый, кросс-платформенный и обладает хорошими средствами отладки.

Для разработки приложений нам нужно следующее:

- Язык, среда для разработки кода (IDE) и исполняемая среда. В нашем случае это Anaconda и PyCharm.
- Набор основных библиотек: Scikit-learn, Numpy, Pandas, Matplotlib, Theano, Keras.

Ниже представлены инструкции по установке и настройке компонент. Важный момент заключается в том, что все компоненты для работы в своем исходном состоянии «не родные» для систем Windows, поэтому в первую очередь эти инструкции относятся к пользователям Windows.

Инструкции по установке основных компонент:

1. Скачайте и установите AnacondaEnvironment (среда свободно распространяется на официальном сайте [32]). Anaconda необходима по большей части не как IDE, а как среда, в которой будет сразу установлено множество необходимых библиотек (развернутых соответствующим образом под Windows), таких как pip, numpy, matplotlib и еще пара десятков других. Также будет установлен и сам Python 2.7.

Обратите внимание!

- *Для корректной работы всех компонент аккаунт пользователя Windows, из под которого будет проводиться установка, должен содержать только латинские символы в своем имени.*

2. Рекомендуется установить PyCharm как основную IDE для разработки (хотя можно пользоваться самой SpiderAnaconda, которая будет установлена на предыдущем шаге) или JupyterNotebook. Однако если вы новичок в программировании или в Python, то рекомендуется именно PyCharm, так как эта среда сильно облегчает программирование и навигацию по коду. CommunityEdition распространяется бесплатно и скачать его можно здесь: [33].

3. Для проверки того, что все компоненты установлены корректно, запустите IDE, создайте новый файл (команда Alt+Insert или через меню File и пункт New) под именем testc кодом на Python, как показано на рисунке 44. Вставьте в файл код из листинга 1 и выполните его через пункт меню Run, как показано на рисунке 45.

Листинг 1

```
from sklearn import preprocessing
import numpy as np
X = np.array([[ 1., -1., 2.], [ 2., 0., 0.], [ 0., 1., -1.]])
X_scaled = preprocessing.scale(X)
print(X_scaled)
```

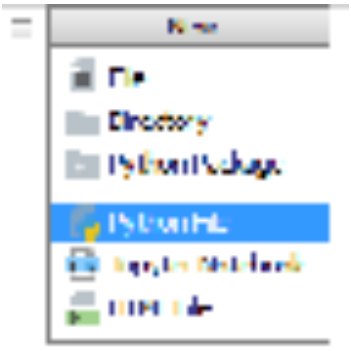


Рисунок 44. Новый файл Python

Обратите внимание!

- *Если в первом пункте меню Run нет имени вашего файла, то выберите третий пункт и в открывшемся окошке найдите имя вашего файла.*

Рисунок 45. Запуск кода на выполнение

Если все установлено верно, то в консоли вы должны увидеть результат, представленный на рисунке 46.

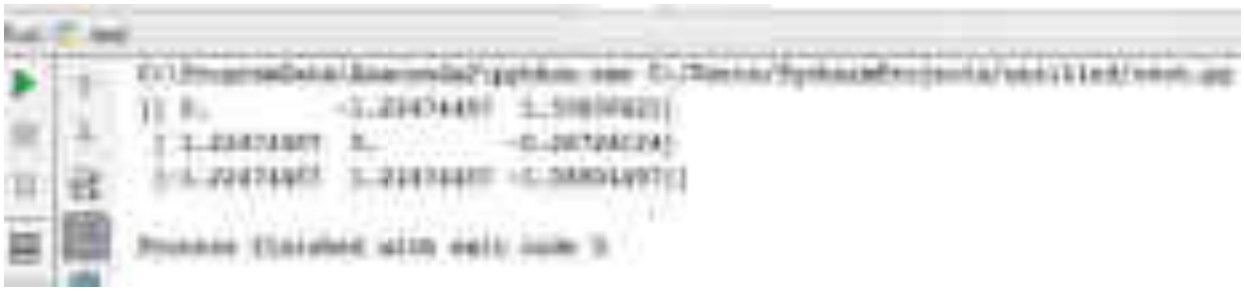


Рисунок 46. Корректная работа кода

Кроме описанного выше, нам понадобится установить и настроить еще ряд компонентов и библиотек. Инструкции по их установке следующие:

1. Скачайте и установите TDM-GCC (специальное средство компиляции для ОС Windows);
2. Откройте консоль Anacondaprompt или обычную командную строку Windows через команду cmd;
3. Выполните в консоли conda update conda;
4. Выполните в консоли conda update --all;
5. Выполните в консоли conda install mingw libpython;

6. Выполните в консоли conda install Theano;

7. Выполните в консоли pip install keras;

Обратите внимание!

- *Keras может быть настроен на tensorflowbackend, тогда надо внести изменения в файл C:/Users/AccauntName/.keras/keras.json и в строке backend=”tensorflow” написать theano;*

8. Запустите IDE, создайте новый файл с кодом на Python и выполните код из листинга 2 для проверки того, что все компоненты установлены корректно.

Листинг 2

```
import numpy as np
import sklearn.linear_model as lin
from keras.layers.core import Dense, Activation
from keras.models import Sequential
from keras.optimizers import RMSprop

x_train = np.array([[0.1, 0.3], [0.2, 0.2], [0.7, 0.8], [1.0, 0.9]])
y_train = np.array([0, 0, 1, 1]).reshape(4, 1)
x_test = np.array([[0, 0], [0.3, 0.3], [0.6, 0.7], [1, 1]])
model = Sequential()
model.add(Dense(2, init='lecun_uniform', input_shape=(2,)))
model.add(Activation('relu'))
model.add(Dense(1, init='lecun_uniform'))
model.add(Activation('linear'))
rms = RMSprop(lr=0.01)
model.compile(loss='mse', optimizer=rms)
epochs = 200
model.fit(x_train, y_train, batch_size=1, nb_epoch=epochs, verbose=0)
y_predict = model.predict(x_test, batch_size=1)
print(y_predict)
```

В результате в Output консоли не должно быть ошибок (код выхода равен 0), и должна будет распечататься информация, представленная в листинге 3.

Листинг 3

```
[[ 0.00339771]
[ 0.14396997]
[ 0.67507285]
[ 1.1803354 ]]
```

Мы установили основные, необходимые нам, компоненты. Теперь перейдем непосредственно к рассмотрению конкретных приемов работы с ними.

В данном пособии будет приведено описание только тех операторов и функций, которые непосредственно используются при решении конкретной задачи. Для получения прочей информации по Python рекомендуется воспользоваться справочником или же поисковой системой Google.

Обратите внимание!

- *Если у вас есть конкретный вопрос, например, о том, как выбрать последний элемент в одномерном или двумерном массиве, или о том, как отсортировать массив и т. п., то лучшее решение – написать этот вопрос в Google на английском;*
- *Если ваш уровень английского недостаточен, то просто сформулируйте вопрос на русском максимально коротко, затем вставьте в GoogleTranslate и затем – в поисковик. С вероятностью 90% на такие конкретные вопросы вы найдете очень конкретные примеры кода на сайте StackOverflow.*

Элементарные операции с данными

Рассмотрим работу по предобработке данных и поиску простых закономерностей в них средствами Python, а именно средствами пакетов Numpy и Pandas. Более подробно с их функциями можно ознакомиться, например, здесь: [34].

Для импорта модуля Numpy необходимо написать следующую строку кода (листинг 4).

Листинг 4

```
Import numpy as np
```

Обратите внимание!

- *Numpy импортируется с псевдонимом np, через который в дальнейшем будет обращение к модулю.*

Основными единицами данных в машинном обучении являются векторы и матрицы. С точки зрения программирования такие структуры представляют собой одномерные и двумерные массивы или специальные объекты-таблицы.

Обратите внимание!

- *Если в тексте написано просто – вектор или матрица, то значит идет речь о одномерном или двумерном массиве Numpy. В других случаях будет специально написано в каком формате представлены данные (например, DataFrame, о чем речь пойдет ниже).*

Для того чтобы рассмотреть способы работы с данными, нам необходимо получить данные для работы. Так как модели, разрабатываемые на Python, в 99% случаев используют полученные откуда-то данные или же генерируют тестовые, то мы опишем три способа получения данных: прямое объявление, случайная генерация и загрузка из csv-формата.

Начнем с первого. Объявить матрицу можно, используя код из листинга 5. Как видите, ничего сложного здесь нет: ни предварительного объявления данных, ни выделения памяти, ни объявления типа данных.

Листинг 5

```
Z = np.array([[4, 5, 0],
              [9, 9, 9]])
```

Теперь рассмотрим второй способ: сгенерируем случайную матрицу, состоящую из 6 строк и 5 столбцов. Элементы ее будут являться случайными числами из нормального распределения. Функция для генерации таких чисел: np.random.normal.

Ее параметры:

- loc: среднее нормального распределения (в нашем случае 1);
- scale: стандартное отклонение нормального распределения (его значение в нашем случае равно 10);
- size: размер матрицы (в нашем случае (6, 5)).

Код генерации матрицы X и ее распечатки представлены в листинге 6.

Листинг 6

```
X = np.random.normal(loc=1, scale=10, size=(6, 5))
print X
```

Далее рассмотрим способ загрузки данных из файла в csv-формате, который предназначен для хранения табличных данных. Как правило, в файлах этого формата столбцы разделяются запятой, а первая строка содержит их имена.

Допустим, у нас есть файл «titanic.csv», содержащий данные в виде, представленном на рисунке 47.

Обратите внимание!

- *Небольшие файлы csv (до 1 Гб) можно достаточно удобно посмотреть в Excel. Чтобы корректно загрузить CSV файл, нужно сначала открыть Excel, создать пустую книгу, а затем вызвать соответствующего мастера загрузки через вкладку «Данные»=>«Из текста»..*

Рисунок 47. Файл с данными

Для загрузки этих данных нам необходимы средства библиотеки pandas. Код ее импорта, а также код загрузки и распечатки загруженных данных представлены в листинге 7. Там же представлен код записи данных в csv-файл.

Листинг 7

```
# -*- coding: utf-8 -*-
import csv
import pandas
#чтение из файла
data = pandas.read_csv('titanic.csv', index_col='PassengerId')
print data
#запись в файл данных в исходном виде
data.to_csv('q.csv')
#запись в файл данных как массива значений
with open('test.csv', 'w') as csvfile:
  spamwriter = csv.writer(csvfile, delimiter=';',quotechar='|', quoting=csv.QUOTE_MINIMAL)
  spamwriter.writerow (data.as_matrix())
```

Обратите внимание!

- *Символ # - знак комментария.*
- *Для того чтобы закомментировать (или раскомментировать) несколько строк, необходимо выделить их и нажать сочетание клавиш Ctrl+правый слеш.*
- *Для того чтобы писать комментарии в Python на русском языке, необходимо добавить следующую строку в начало файла с кодом:# -*- coding: utf-8 -*-*

При выполнении кода из листинга 7 данные будут загружены в виде DataFrame, с помощью которого можно удобно работать с ними. Параметр index_col='PassengerId' означает, что колонка PassengerId задает нумерацию строк данного DataFrame.

DataFrame – первичная структура данных pandas, представляющая собой двумерную, изменяемую по размеру, потенциально гетерогенную структуру табличных данных с маркированными осями (строками и столбцами).

В виде DataFrame над данными удобно производить различные манипуляции сортировки, выборки, применения функций построчно или по колонкам и т. п. Однако DataFrame – это комплексный объект высокого уровня и его нельзя использовать напрямую для обучения моделей. Перед этим его нужно перевести в обычную цифровую матрицу, например, следующим образом (листинг 8), и дальше работать с ним как с обычным двумерным массивом данных.

Листинг 8

```
x_test= data[['ColumnName_1', 'ColumnName_2']].as_matrix()
```

Вернемся к листингу 7. В нем представлены два способа записи в файл. Первый – через метод самого DataFrame, второй – через метод библиотеки csv. Необходимо отметить, что второй способ может быть использован не только для работы с DataFrame, но и с любыми данными, которые надо записать в csv-файл.

Мы рассмотрели основные способы получения данных, теперь перейдем к работе с ними и продемонстрируем некоторые возможности библиотеки Numpy по операциям с матрицами на конкретных задачах из практики.

Допустим, перед нами стоит следующая задача: есть данные о ежедневной выручке по филиалам компании за первые 12 дней месяца, записанные в виде таблицы (рисунок 48) и сохраненные в файле «data.csv». Выведем номер филиала, преодолевшего по прибыли порог, равный 1000.

Первое, что нам необходимо сделать, – загрузить данные. Однако, чтобы pandas корректно обработал файл, нам нужно убрать из него кириллицу и убедиться, что разделители в csv действительно запятые. Для этого мы можем поместить файл в папку проекта pyCharm, двойным щелчком мыши на нем (в окне проекта) открыть его и привести к виду, показанному на рисунке 49. Теперь для чтения данных, преобразованию их в матрицу и распечатки полученного массива мы можем использовать код из листинга 9.

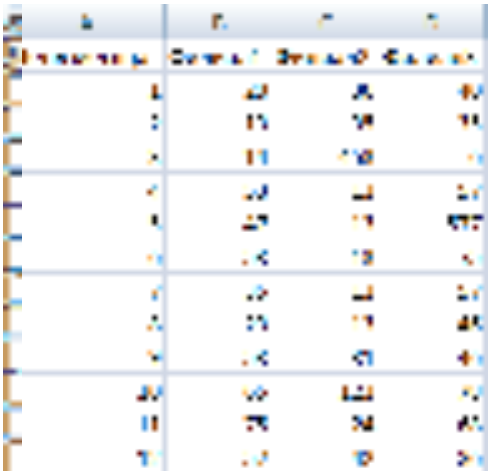


Рисунок 48. Файл с данными для задачи

Рисунок 49. Корректный файл с данными

Обратите внимание!

- *Если исходный файл создан через Excel, то разделителями могут быть точки с запятой. Заменить их на запятые можно при редактировании файла в PyCharm через цепочку пунктов меню «Edit»=>«Find» =>«Replace».*

Листинг 9

```
import pandas
data= pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
print x_test
```

Если все сделано верно, в консоли должен появиться результат, показанный на рисунке 50.

Рисунок 50. Загруженные данные

Обратите внимание!

- *При преобразовании к матрице мы указываем в кавычках имена столбцов с данными по филиалам и не указываем столбец Day, так как содержащиеся в нем данные для задачи не нужны.*

Теперь перейдем непосредственно к решению поставленной задачи. С точки зрения работы с матрицей, для получения ответа нам

необходимо посчитать сумму по каждому столбцу и вывести номера тех, чья сумма превысит 1000.

Функция для подсчета суммы: np.sum. В качестве параметров она принимает матрицу, для которой необходимо посчитать сумму и измерение (строки или столбцы), которое необходимо суммировать. Измерение (axis) задается цифрой (для двумерной матрицы 0 – строки, 1 – столбцы).Если этот параметр не задать, то результат функции будет рассчитан для всей матрицы целиком. Результатом выполнения операции будет массив с соответствующими суммами. Библиотека Numpy предоставляет возможности применения к матрицам (и массивам) логических операций, причем применяемых поэлементно. Соответственно, результатом такой операции будет матрица такого же размера, в ячейках которой будет записано либо True, либо False (удовлетворяет текущий элемент условию или нет). Индексы элементов со значением True можно получить с помощью функции np.nonzero. Функция в качестве параметра принимает матрицу, в которой необходимо отыскать ненулевые элементы. Заметим, что в нашем случае мы можем сразу передать логическое выражение, составленное из массива сумм и самого условия, тогда элементы со значением False будут интерпретироваться как нулевые. Код решения представлен в листинге 10. Результат работы кода – на рисунке 51. Ответ к задаче: третий филиал.

Листинг 10

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
#print x_test
r = np.sum(x_test, axis=0)
print (r)
print np.nonzero(r> 1000)
```

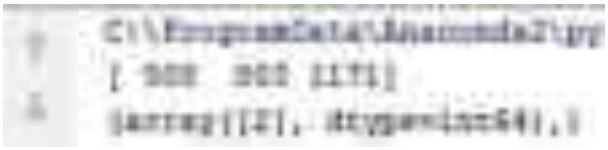


Рисунок 51. Решение задачи

Теперь решим следующую задачу: определим, в какой день суммарная выручка по филиалам превысила 500. Код решения представлен в листинге 11, а результат – на рисунке 52.

Рисунок 52. Решение задачи

Листинг 11

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.sum(x_test, axis=1)
print (r)
print np.nonzero(r>500)
```

Код нахождения филиала с максимальной средней выручкой представлен в листинге 12, а результат показан на рисунке 53.

Листинг 12

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.mean(x_test, axis=0)
print (r)
print np.nonzero(r== np.max(np.mean(x_test, axis=0)))
```

Рисунок 53. Номер филиала с максимальной средней выручкой

Номер филиала с максимальной величиной стандартного отклонения прибыли можно получить, используя код из листинга 13.

Далее нам необходимо более подробно рассмотреть работу со структурой DataFrame, но перед этим упомянем еще две полезные функции Numpy, которые могут понадобиться в последующем: функцию генерации единичной матрицы (np.eve) и функцию вертикальной стыковки матриц (np.vstack). Первая в качестве параметра принимает количество строк (оно же количество столбцов), вторая – матрицы, которые нужно объединить. В листинге 14 показан пример генерации и объединения двух единичных матриц.

Листинг 13

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.std(x_test, axis=0)
print (r)
print np.nonzero(r== np.max(np.std(x_test, axis=0)))
```

Листинг 14

```
import numpy as np
A = np.eye(3)
B = np.eye(3)
print A
print B
AB = np.vstack((A, B))
```

Работа с DataFrame

Мы рассмотрели основные моменты работы с матрицами, поэтому теперь вернемся к другой структуре данных: DataFrame. Как мы помним, она является таблицей, где колонки имеют заголовки, а строки, кроме номеров, могут иметь дополнительные индексы в виде цифр или имен (по сути, индексы представляют собой отдельную, специальную колонку в таблице DataFrame, которая не относится к данным). Для того чтобы посмотреть, что представляют собой данные, можно воспользоваться несколькими способами.

Если мы хотим распечатать первые 5 строк данных, то можем воспользоваться кодом из листинга 15, в случае, если указан только один индекс, или же воспользоваться методом head() DataFrame, как показано в листинге 16.

Листинг 15

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data[:5]
```

Листинг 16

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data.head()
```

Если же нас интересует содержимое конкретного столбца, то можно использовать квадратные скобки и название столбца, как показано в листинге 17.

Листинг 17

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data['Fil1']
```

Обратите внимание!

- *Код из листинга 16 не применим к индексному столбцу.*

Для подсчета некоторых статистик (количества, среднее, максимум, минимум) можно также использовать методы объекта DataFrame. В листинге 18 приведен пример кода, считающего количество повторов каждого из значений в столбце с именем 'Fil1'. Результат работы кода показан на рисунке 54. Значения суммы, минимума, максимума, среднего, количества и среднеквадратического отклонения вычисляются аналогично с использованием соответствующих функций (sum, min, max, mean, count, std).

Листинг 18

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data['Fil1'].value_counts()
```

Рисунок 54. Результат работы кода из листинга 18

Теперь рассмотрим еще несколько более сложных функций работы с DataFrame на примере некоторой таблицы, представленной на рисунке 55.

В листинге 19 представлены интересующие нас фрагменты кода с комментариями. Как можно заметить, возможности DataFrame весьма обширны, поэтому здесь приведены лишь основные моменты, а с остальным вам предлагается ознакомиться самостоятельно.

Рисунок 55. Структура таблицы с данными

Обратите внимание!

- *Для лучшего понимания кода листинга 19 вам рекомендуется почитать про Лямбда функции в Python, например, здесь: [35];*
- *Более подробно со списком методов DataFrame можно познакомиться в документации: [36];*
- *Ссылка на уроки по Pandas: [37].*

Листинг 19

```
#-*- coding: utf-8 -*-
# Выбор элементов DataFrame с использованием iloc
# Строки:
data.iloc[0] # выбирает первую строку таблицы.
data.iloc[1] # выбирает вторую строку таблицы
data.iloc[-1] # выбирает последнюю строку таблицы

# Столбцы:
data.iloc[:,0] # выбирает первую колонку таблицы
data.iloc[:,1] # выбирает вторую колонку таблицы
data.iloc[:,-1] # выбирает последнюю колонку таблицы

# Множественный выбор
data.iloc[0:5] # выбирает первые пять строк таблицы
data.iloc[:, 0:2] # выбирает первые две колонки таблицы со всеми строками

# Выбирает строки с индексами 'Andrade' и 'Veness' (в случае текстовых индексов),
#со всеми колонками между колонок с именем 'city' и 'email'
data.loc[['Andrade', 'Veness'], 'city':'email']
```

Окончание листинга 19

```
# Выбирает те же строки, только с колонками 'first_name', 'address' и 'city'
data.loc['Andrade':'Veness', ['first_name', 'address', 'city']]

# Выбирает строки с firstname равным Antonio и выбирает только колонки между
#колонками 'city' и 'email'
data.loc[data['first_name'] == 'Antonio', 'city':'email']

# Выбирает строки со всеми колонками, где в колонке email встречаются ячейки,
#заканчивающиеся на 'hotmail.com'
data.loc[data['email'].str.endswith("hotmail.com")]

#Выбирает строки, где first_name равно одному из следующих значений
data.loc[data['first_name'].isin(['France', 'Tyisha', 'Eric'])]

# Выбирает строки, где в колонке first_name встречается Antonio и в колонке email
#встречается gmail.com
data.loc[data['email'].str.endswith("gmail.com") & (data['first_name'] == 'Antonio')]

# Выбирает строки, у которых в колонке id есть значения от 100 до 200, и возвращает
#только колонки 'postal' и 'web'
data.loc[(data['id'] > 100) & (data['id'] <= 200), ['postal', 'web']]

#Лямбда функция, которая возвращает True/False переменную.
# Выбирает строки, где ячейка в колонке company_name имеет 4 слова.
data.loc[data['company_name'].apply(lambda x: len(x.split(' ')) == 4)]

#Распечатка заголовков
print list(data)
```

Предобработка данных. Стандартизация и нормализация

Мы рассмотрели основные моменты работы с данными, теперь перейдем к такому вопросу, как предобработка данных. Прежде чем обучать и использовать большинство моделей, необходимо привести исходные данные к единообразному виду (к единому диапазону). Единый диапазон определяется через математическое ожидание выборки и разброс (дисперсию). То есть нужно, например, чтобы все данные были в диапазоне от 0 до 1 или от –1 до 1. Как было сказано

в одном из предыдущих разделов, достигается такое приведение посредством двух процессов\методов: стандартизации данных и нормализации.

Можно провести стандартизацию самостоятельно, используя Numpy. Для этого вычтите из каждого столбца его среднее значение, а затем поделите на его стандартное отклонение. Или же можно использовать библиотеку scikit-learn, как показано в листинге 20.

Однако, чаще всего, кроме нормировки одной матрицы, потребуется нормировать несколько матриц по одним и тем же шкалам (для каждого столбца\признака будут свои коэффициенты смещения и масштабирования).

Так если тестовая\рабочая выборки поступают в систему онлайн, то их потребуется нормировать по тем же коэффициентам, что и тренировочную (рассчитанным на тренировочной выборке). Иначе нормировка будет некорректной.

Листинг 20

```
from sklearn import preprocessing
import numpy as np
X = np.random.normal(loc=1, scale=10, size=(6, 5))
X_scaled = preprocessing.scale(X)
print X
print X_scaled
```

Вместо того чтобы хранить матрицы коэффициентов нормировки, гораздо удобнее воспользоваться классом StandardScaler из scikit-learn. Сперва нужно «обучить» его на тренировочных данных (методом fit), а потом преобразовывать разные матрицы одинаковым образом. Работа с этим классом показана в листинге 21.

Листинг 21

```
from sklearn import preprocessing
import numpy as np
X = np.random.normal(loc=1, scale=10, size=(6, 5))
X_test = np.random.normal(loc=1, scale=10, size=(6, 5))
```

Окончание листинга 21

```
scaler = preprocessing.StandardScaler().fit(X)
train_data_scaled = scaler.transform(X)
test_data_scaled = scaler.transform(X_test)
print train_data_scaled
print test_data_scaled
```

Как было сказано в одном из предыдущих разделов, стандартизация смещает значения векторов (выравнивает относительного единого центра в нуле), а также производит выравнивание разброса. Однако значения разных векторов не будут в одинаковом диапазоне (от –1 до 1, например). Они будут лишь иметь стандартный разброс в рамках вектора\столбца\признака. Для того чтобы достичь одинакового масштаба всех векторов, необходима нормализация.

Средствами scikit-learn нормализацию можно выполнить следующим образом (листинг 22).

Листинг 22

```
#-*- coding: utf-8 -*-
from sklearn import preprocessing
import numpy as np
X=[[1., -1.,2.],
  [2., 0., 0.],
  [0., 1., -1.]]

#нормализация max-нормой
X_normalized_max = preprocessing.normalize(X, norm='max', axis=0)

#нормализация нормой l1
X_normalized_max = preprocessing.normalize(X, norm='l1', axis=0)

#нормализация нормой l2
X_normalized_max = preprocessing.normalize(X, norm='l2', axis=0)
```

Более подробную информацию о методах библиотеки scikit-learn по предобработке данных можно найти здесь: [38].

Работа с деревьями решений

Мы рассмотрели методы предобработки данных. Теперь перейдем к следующему – к деревьям решений. Как было сказано в одном из предыдущих разделов, решающие деревья относятся к классу логических методов. Их основная идея состоит в объединении определенного количества простых решающих правил, благодаря чему итоговый алгоритм является интерпретируемым. Как следует из названия, решающее дерево представляет собой бинарное дерево, в котором каждой вершине сопоставлено некоторое правило вида «j-й признак имеет значение меньше b». В листьях этого дерева записаны числа-предсказания. Чтобы получить ответ, нужно стартовать из корня и делать переходы либо в левое, либо в правое поддерево в зависимости от того, выполняется правило из текущей вершины или не выполняется.

Одна из особенностей решающих деревьев заключается в том, что они позволяют получать важности всех используемых признаков. Важность признака можно оценить на основе того, как сильно улучшился критерий качества благодаря использованию этого признака в вершинах дерева.

Рассмотрим данные о пассажирах «Титаника» (их можно скачать здесь: [39]). Решим на них задачу классификации, в которой по различным характеристикам пассажиров требуется найти у выживших пассажиров два наиболее важных признака (из четырех рассматриваемых: пол, класс, возраст, цена билета).

В библиотеке scikit-learn решающие деревья реализованы в классах sklearn.tree.DecisionTreeClassifier (для классификации) и sklearn.tree.DecisionTreeRegressor (для регрессии). Обучение модели производится с помощью функции fit. Найти важность признаков можно, имея уже обученный классификатор: его поле feature_importances_ содержит массив «важностей» признаков. Индекс в этом массиве соответствует индексу признака в данных. Стоит обратить внимание, что данные могут содержать пропуски.

Pandas выгружает такие значения как nan (not a number). Для того чтобы проверить, является ли число nan'ом, можно воспользоваться функцией numpy isnan.

Перейдем к решению задачи. Первое, что нам необходимо сделать, – внимательно посмотреть на данные перед загрузкой. Ранее мы уже предположили, что среди них могут быть пропуски, и определились с методом решения этой проблемы, но при изучении информации мы должны увидеть еще одну: признак Sex имеет строковые значения, тогда как все остальные – числовые. Следовательно, нам необходимо привести и его к числовому виду. Например, заменим male на 0, остальное – на 1.

Для этого определим функцию, которую применим к нашему массиву данных. В Python функции начинаются с ключевого слова def, и их операторы (тело функции) обязательно имеют отступ от начала строки. Применение функции к объекту выполняется оператором apply. Код функции представлен в листинге 23.

Листинг 23

```
def Sex_to_bool(sex):
    if sex == "male":
        return 0
    return 1
```

Алгоритм решения задачи будет следующим:

1. Загрузить выборку из файла titanic.csv с помощью пакета Pandas.
2. Оставить в выборке четыре признака: класс пассажира (Pclass), цену билета (Fare), возраст пассажира (Age) и его пол (Sex). Привести пол к числовому виду и убрать из выборки пустые значения.
3. Выделить целевую переменную (она записана в столбце Survived).

4. Обучить решающее дерево с параметром random_state=241 и остальными параметрами по умолчанию (речь идет о параметрах конструктора DecisionTreeClassifier).
5. Вывести важности признаков.

Код решения задачи (с комментариями) – в листинге 24. Результат работы – на рисунке 56.

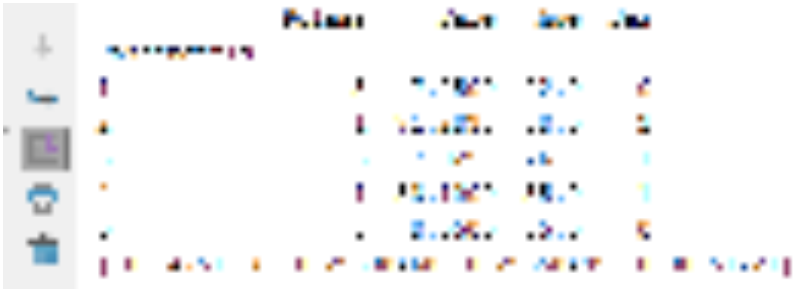


Рисунок 56. Важности признаков

Листинг 24

```
-*- coding: utf-8 -*-
import pandas
from sklearn.tree import DecisionTreeClassifier
import numpy as np

data = pandas.read_csv('titanic.csv', index_col='PassengerId')
#функция для приведения пола к числу
def Sex_to_bool(sex):
    if sex == "male":
        return 0
    return 1
#приведение пола к числу
data['Sex'] = data['Sex'].apply(Sex_to_bool)
#выгрузка непустых данных
data=data.loc[(np.isnan(data['Pclass'])==False) & (np.isnan(data['Fare'])==False)
&(np.isnan(data['Age'])==False) & (np.isnan(data['Sex'])==False)&
(np.isnan(data['Survived'])==False)]
#отбор нужных столбцов
corr = data[['Pclass', 'Fare', 'Age', 'Sex']]
#респечатка первых 5 строк данных
print corr.head()
```

Окончание листинга 24

```
#определение целевой переменной
y = data['Survived']
#создание и обучение дерева решений
clf = DecisionTreeClassifier(random_state=241)
clf.fit(corr, y)
#получение и распечатка важностей признаков
importances=clf.feature_importances_
print importances
```

Таким образом мы видим, что наиболее важными признаками будут пол и цена билета. Однако по описанному решению стоит сделать одно важное замечание. При преобразовании пола к числовому виду мы исказили характер данных: ранее объекты столбца «пол» не могли быть математически сравнимы между собой, а после наших преобразований эта характеристика у них появилась. Это привело к неоднозначности итогового решения: при male=0 массив ответа имеет вид: [0.14751816 0.29538468 0.25658495 0.30051221], а при female=0 – [0.14000522 0.30343647 0.2560461 0.30051221]. Конечно, в нашей ситуации изменение незначительное и не влияет на итоговый ответ, но в другой ситуации это может быть не так. Поэтому вам предлагается самостоятельно подумать, как избежать подобного искажения данных.

Работа с линейной регрессией

Мы рассмотрели работу с решающими деревьями, теперь же перейдем к рассмотрению регрессии. Начнем с линейных моделей, приведя в качестве примера решение задачи прогнозирования уровня зарплаты по данным вакансий.

Линейные методы хорошо подходят для работы с разреженными данными. К таковым относятся, например, тексты. Это можно объяснить высокой скоростью обучения и небольшим количеством параметров, благодаря чему удается избежать переобучения.

Линейная регрессия имеет несколько разновидностей в зависимости от того, какой регуляризатор используется. В рамках предлагаемой задачи мы будем использовать гребневую регрессию, где применяется квадратичный, или L2-регуляризатор. Эта модель реализована в классе sklearn.linear_model.Ridge. Более подробно о реализации гребневой регрессии можно почитать здесь: [40].

Исходные данные объемом 60 000 записей хранятся в файле «salary_train.csv», тестовые данные (2 записи) хранятся в файле «salary_test_mini.csv» (скачать их можно отсюда: [41]). Оба файла имеют вид, представленный на рисунке 57.

В данном случае столбцы A-C будут содержать значения входных признаков, а последний столбец – целевой переменной.

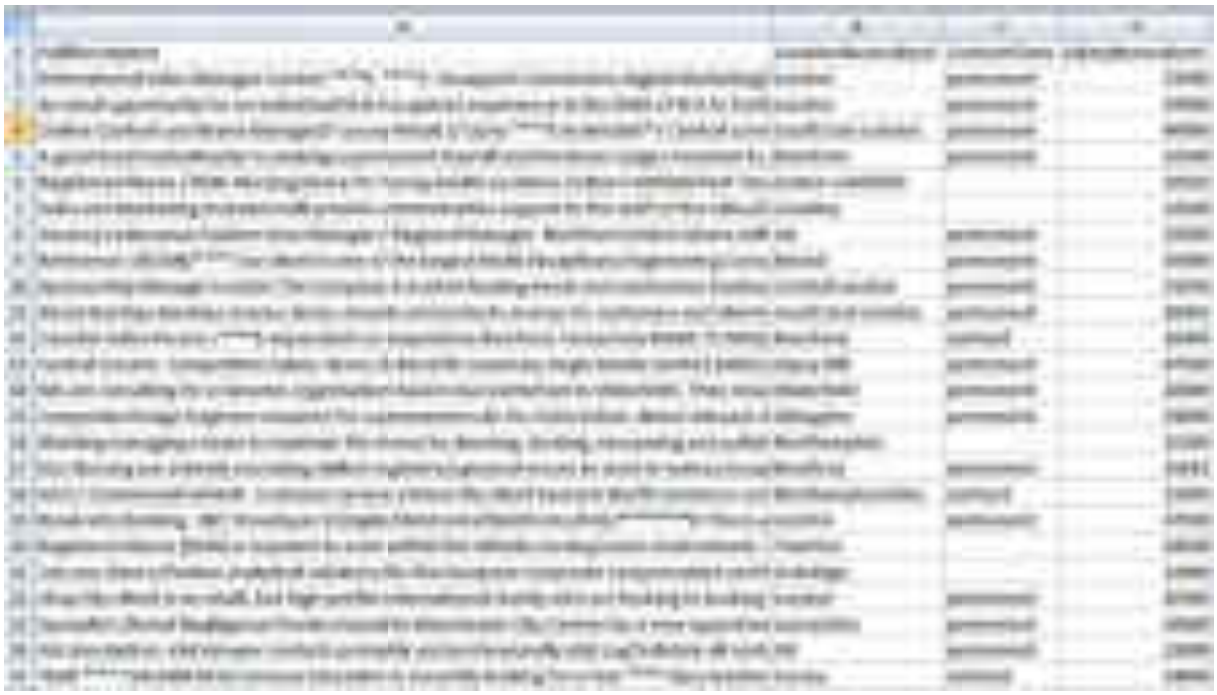


Рисунок 57. Исходные данные

Как можно заметить, первый столбец содержит объемный текст, который необходимо предобработать для загрузки в модель. Например, извлечь TF-IDF-признаки, воспользовавшись классом sklearn.feature_extraction.text.TfidfVectorizer, который преобразует массив текста в матрицу, содержащую TF-IDF-признаки. Более

подробно об этом классе можно почитать здесь: [42]. Использование его для решаемой задачи показан в листинге 25.

Листинг 25

```
vectorizer = TfidfVectorizer(min_df=10)
train_text_feature_matrix = vectorizer.fit_transform(data_train['FullDescription'])
idf = vectorizer.idf_
print dict(zip(vectorizer.get_feature_names(), idf))
```

В данном случае поле idf_ класса TfidfVectorizer содержит вектор с глобальными весовыми коэффициентами. Строка кода print dict(zip(vectorizer.get_feature_names(), idf)) выведет в консоль массив данных вида: {u'pre': 5.6339293213815242, u'paperless': 9.0576775285655771, u'peoplewithchemistry': 9.6042212349336467, u'rfps': 9.1117447498358519, u'yellow': 8.957594070008593...}.

Теперь, что касается оставшихся столбцов-признаков: LocationNormalized и ContractTime являются строковыми, и поэтому с ними нельзя работать напрямую. Применим к ним one-hot-кодирование. Для рассматриваемого нами языка Python оно реализовано в классе sklearn.feature_extraction.DictVectorizer, который преобразует списки сопоставленных пар признак-значение в массивы, с которыми может работать Numpy, или в разреженные матрицы (scipy.sparse-matrices) для использования с классами-«измерителями качества» scikit-learn (scikit-learnestimators). Когда значения признаков строковые, этот трансформер выполняет бинарное one-hot-кодирование: для каждого из возможных строковых значений признака строится признак с логическим значением, говорящий, принимает данный признак указанное строковое значение или нет. Например, пусть на входе есть признак «c», который может принимать значения «кот» и «дог». На выходе будет два признака: один – «c = кот», другой – «c = дог». Признаки, которые не представлены в обучающей выборке, будут иметь нулевое значение в результирующей матрице или результирующем массиве.

Подробнее о DictVectorizer можно прочитать здесь: [43]. Пример использования этого класса для указанных данных приведен в листинге 26.

Листинг 26

```
# -*- coding: utf-8 -*-
from sklearn.feature_extraction import DictVectorizer
.....
enc = DictVectorizer()
train_dic = data_train[['LocationNormalized', 'ContractTime']].to_dict('records')
test_dic = data_test[['LocationNormalized', 'ContractTime']].to_dict('records')
X_train_categ = enc.fit_transform(train_dic)
X_test_categ = enc.transform(test_dic)
```

При этом считается, что в data_train загружены данные обучающей выборки, а в data_test – данные, на которых будет проверяться работа модели.

В результате работы кода из листинга 26 train_dic (также как и test_dic) будет содержать множество записей, таких как: {['LocationNormalized': 'UK', 'ContractTime': 'permanent'], ['LocationNormalized': 'Bradford', 'ContractTime': 'permanent']}. X_train_categ будет содержать данные, представленные на рисунке 58.

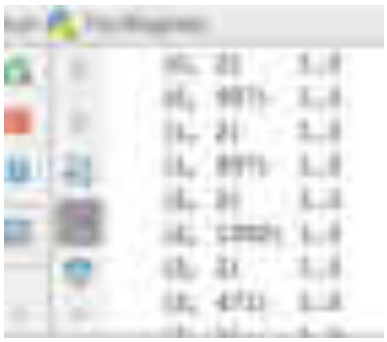


Рисунок 58. Преобразованные данные

Интерпретировать их можно следующим образом. X_train_categ – это по сути матрица, но только не в классическом формате. Если точнее, то X_train_categ – это разреженная матрица, которая в сокращенном виде описывает некую полную матрицу (назовем ее X_Full).

В X_train_categ указано, в каких позициях X_Full стоят единицы. Так (0, 2) 1.0 – означает, что в нулевой строке и втором столбце стоит единица. Во всех остальных ячейках матрицы X_Full, которые не обозначены в X_train_categ, стоят нули. X_Full – это матрица размера M на N, где M – число строк, которое определяется количеством объектов в выборке, а N – число уникальных слов в столбцах ['LocationNormalized', 'ContractTime'] исходной таблицы.

Теперь вернемся к предобработке данных. Так как в строковых данных есть пропуски, то нам потребуется заменять их на специальные строковые величины (например, 'nan'). Для этого можно использовать код из листинга 27.

Листинг 27

```
data_train['LocationNormalized'].fillna('nan', inplace=True)
```

С учетом изложенного выше, алгоритм решения поставленной задачи будет следующим:

1. Загрузить данные об описаниях вакансий и соответствующих годовых зарплатах из файла salary-train.csv.
2. Провести его предобработку:
   2.1. Привести тексты к нижнему регистру (text.lower()).
   2.2. Заменить все, кроме букв и цифр, на пробелы для облегчения дальнейшего разделение текста на слова. Для такой замены в строке text подходит следующий вызов: re.sub('[^a-zA-Z0-9]', ' ', text). Также можно воспользоваться методом replace у DataFrame, чтобы сразу преобразовать все тексты, например, так: train['FullDescription'] = train['FullDescription'].replace('[^a-zA-Z0-9]', ' ', regex = True)
   2.3. Применить TfidfVectorizer для преобразования текстов в векторы признаков. Оставить только те слова, которые встречаются хотя бы в 5 объектах. Для этого использовать параметр min_df у TfidfVectorizer.

2.4. Заменить пропуски в столбцах LocationNormalized и ContractTime на специальную строку 'nan' (листинг 27).

2.5. Применить DictVectorizer для получения one-hot-кодирования признаков LocationNormalized и ContractTime.

2.6. Объединить все полученные признаки в одну матрицу «объекты-признаки».

Обратите внимание!

- *Матрицы для текстов и категориальных признаков являются разреженными. Для объединения их столбцов нужно воспользоваться функцией scipy.sparse.hstack.*

2.7. Обучить гребневую регрессию с параметрами alpha=1 и random_state=241, помня, что целевая переменная записана в столбце SalaryNormalized. Обучение производится вызовом метода fit.

3. Построить прогноз для двух примеров из файла salary-test-mini.csv. Прогноз строится методом predict.

Код решения задачи (с комментариями) – в листинге 28. Результат работы – два прогнозных числа для двух записей из тестовой выборки: [56876.4573163 37557.00551031].

Листинг 28

```
# -*- coding: utf-8 -*-
import pandas
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Ridge
from scipy.sparse import hstack

#Загрузка данных
data_train = pandas.read_csv('salary-train.csv')
data_test = pandas.read_csv('salary-test-mini.csv')

# Приведение текста к нижнему регистру
data_train['FullDescription'] = data_train.FullDescription.str.lower()
data_test['FullDescription'] = data_test.FullDescription.str.lower()
```

Окончание листинга 28

```
# Удаление ненужных символов
data_train['FullDescription'] = data_train['FullDescription'].replace('[^a-zA-Z0-9]', ' ',
regex=True)
data_test['FullDescription'] = data_test['FullDescription'].replace('[^a-zA-Z0-9]', ' ',
regex=True)
# Преобразование текста в вектор признаков, используя TdifVectorizer из sklearn
vectorizer = TfidfVectorizer(min_df=10)
train_text_feature_matrix = vectorizer.fit_transform(data_train['FullDescription'])
test_text_feature_matrix = vectorizer.transform(data_test['FullDescription'])
# Заполнение пустых ячеек
data_train['LocationNormalized'].fillna('nan', inplace=True)
data_train['ContractTime'].fillna('nan', inplace=True)
data_test['LocationNormalized'].fillna('nan', inplace=True)
data_test['ContractTime'].fillna('nan', inplace=True)
# Преобразование категориальных признаков в числовые
enc = DictVectorizer()
train_dic = data_train[['LocationNormalized', 'ContractTime']].to_dict('records')
test_dic = data_test[['LocationNormalized', 'ContractTime']].to_dict('records')
X_train_categ = enc.fit_transform(train_dic)
X_test_categ = enc.transform(test_dic)
# Здесь по горизонтали объединяется разреженная матрица с признакими текста (из
#столбца FullDescription) и матрица с закодированными категориями (из столбцов
#Location Normalized и Contract Time)
x_train = hstack((train_text_feature_matrix, X_train_categ))
y_train = data_train['SalaryNormalized'].values
x_test = hstack((test_text_feature_matrix, X_test_categ))
#Создание и обучение регрессии
ridge_regression = Ridge(alpha=1, random_state=241)
ridge_regression.fit(x_train, y_train)
#Получение и распечатка прогнозных значений
y_test = ridge_regression.predict(x_test)
print y_test
```

Сохранение и загрузка обученной модели

Как вы могли заметить, работа программы идет довольно долго. Большую часть этого времени занимает процесс обучения, поэтому при работе с обученной моделью используют практику сохранения и загрузки ее параметров в\из файл(а). Рассмотрим следующий

пример. Пусть у нас есть некоторый массив данных о двух факторах и зависимой переменной, сохраненные в файле «data-logistic.csv» (скачать можно отсюда: [41]). Выполним следующее: обучим на этих данных модель линейной регрессии, сохраним ее в файл, создадим еще одну новую модель линейной регрессии, обучим ее на части данных, затем загрузим ранее обученную модель из файла и сравним результаты предсказания обеих моделей. Сохранение и загрузку мы будем выполнять с помощью модуля pickle, и для удобства вынесем код загрузки и сохранения в отдельные функции LoadFromObject и SaveToObject. Решение задачи показано в листинге 29, а ее результат – на рисунке 59.

Листинг 29

```
# -*- coding: utf-8 -*-

import pandas
import pandas.core.series as Series
import numpy as np
from scipy.sparse import hstack
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
import sys
import matplotlib.pyplotasplt
from sklearn import linear_model
import pickle

#Определяем функцию сохранения
def SaveToObject(obj, filename):
  with open(filename, 'wb') as output:
  pickle.dump(obj, output)

#Опредлеляем функцию загрузки
def LoadFromObject(filename):
   f = open(filename, 'rb')
   loaded_obj = pickle.load(f)
   f.close()
   return loaded_obj
```

Окончание листинга 29

```
# Загружаем данные
data_train = pandas.read_csv('data-logistic.csv')
X = data_train.ix[:,1:3].values
y = (data_train.ix[:,0].values + 1)/2
#Для лучешго понимания представления данных распечатайте формы массивов:
print X.shape
print y.shape
# Создание, обучение модели и получение ее прогноза
regression = linear_model.LogisticRegression()
regression.fit(X, y)
ans = regression.predict(X)
# Оценка качества созданной модели
metric = roc_auc_score(y, ans)
print "AUC Trained: ",  metric.real
#Сохранение и загрузка модели в файл с именем model
SaveToObject(regression, "model")
loaded_model = LoadFromObject("model")
#Создание, обучение новой модели, получение ее прогноза и оценка качества
not_fitted = linear_model.LogisticRegression()
not_fitted.fit(X[0:2,:], y[0:2])
ans = not_fitted.predict(X)
metric = roc_auc_score(y, ans)
print "AUC not Trained: ", metric.real
#получение прогноза и оценки качества загруженной модели
ans = loaded_model.predict(X)
metric = roc_auc_score(y, ans)
print "AUCLoaded: ", metric.real
```

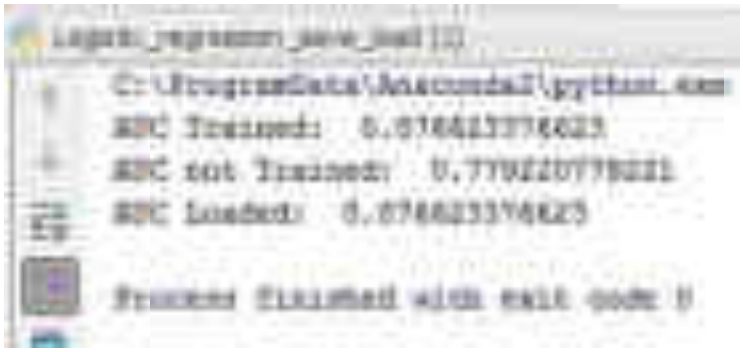


Рисунок 59. Результат работы кода из листинга 29

Как видно из рисунка 59, модель, обученная на части данных, дала более плохой результат, тогда как качество загруженной модели не изменилось. То есть можно говорить о том, что сохранение и загрузка прошли корректно.

Работа с логистической регрессией

Мы рассмотрели применение регрессии для решения задачи предсказания, теперь перейдем к задаче классификации и познакомимся с еще одной моделью – логистической регрессией. Это один из видов линейных классификаторов. Ее особенность заключается в том, что результатом является вероятность принадлежности объекта к классу N, тогда как большинство линейных классификаторов могут выдавать только номера классов.

Логистическая регрессия использует достаточно сложную функцию для оценки качества, которая не допускает записи решения в явном виде (в отличие от, например, линейной регрессии). Тем не менее логистическую регрессию можно настраивать с помощью градиентного спуска.

Для примера мы выполним следующее: сгенерируем некоторую случайную выборку и каждому ее элементу поставим в соответствие число 0 или 1, которое будет обозначать принадлежность к первому или второму классу. Затем обучим логистическую модель на этих данных и линейную модель. После – построим график и сравним результаты. Для генерации случайной выборки из 100 элементов используем следующий код из листинга 30.

Листинг 30

```
import numpy as np
…..
n_samples = 100
np.random.seed(0)
X = np.random.normal(size=n_samples)
```

Обратите внимание!

- *В дальнейших листингах операторы импорта библиотек будут появляться только при первом вызове импортируемых элементов, так как предполагается, что все представленные ниже листинги формируют единый код, который вам предлагается собрать в единый файл самостоятельно.*

Для формирования множества откликов Y определим функцию, реализующую правило: Y=1 для всех X>0 и Y=0 – в остальных случаях. Код функции можно увидеть в листинге 31.

Листинг 31

```
Y = (X > 0).astype(np.float)
```

Следующим шагом нам необходимо выполнить преобразования нашего одномерного массива X: во-первых, добавить небольшой разброс и смещение, чтобы избавиться от прямой функциональной зависимости. Во-вторых, нужно преобразовать его в двумерный массив. Последнее выполняется для функции обучения регрессии: на вход она требует данные в формате двумерного массива. Преобразование выполняется кодом, представленным листинге 32.

Листинг 32

```
X[X > 0] *= 4
X += .3 * np.random.normal(size=n_samples)
X = X[:, np.newaxis]
```

Теперь мы можем объявить и обучить модели регрессии, как показано в листинге 33.

Листинг 33

```
from sklearn import linear_model
log_reg = linear_model.LogisticRegression()
log_reg.fit(X, Y)
lin_reg = linear_model.LinearRegression()
lin_reg.fit(X, Y)
```

Подробнее о реализации логистической модели можно прочитать здесь: [44], а линейной – здесь: [45].

Следующим шагом нам необходимо сгенерировать тестовые данные и выполнить предсказание. Последнее будет выполняться методом predict. Он возвращает бинарные метки классов (0,1). Кроме этого существует еще метод predict_proba (для логистической регрессии). Он возвращает вероятность принадлежности объекта к классу 0 или 1 соответственно. Поскольку у нас два класса, то в результате predict_proba получается двумерный массив (один столбец отражает принадлежность к классу 0, а другой к классу 1). Для отображения мы выберем лишь первый столбец.

Обратите внимание!

- *Сумма элементов массивов (столбцов, векторов) с одним индексом равна единице по правилу полной вероятности.*

Код генерации тестовых данных и вызовы методов прогноза представлен в листинге 34.

Листинг 34

```
X_test = np.linspace(start=-5, stop=10, num=300)
X_test = X_test[:, np.newaxis]
y_log_label = log_reg.predict(X_test)
y_lin = lin_reg.predict(X_test)
y_log_probabilty = log_reg.predict_proba(X_test)[:,1]
```

Следующим шагом нам необходимо вывести график для отображения результатов. Создавать полотно для рисования графиков и отображать их мы будем с помощью библиотеки Matplotlib, как показано в листинге 35.

Листинг 35

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
....
```

Окончание листинга 35

```
# рисуем исходные точки
plt.figure(1, figsize=(4, 3))
plt.scatter(X.ravel(), y, color='black', zorder=20)
# рисуем график меток логистической регрессии
plt.plot(X_test, y_log_label, color='red', linewidth=3)
# рисуем график линейной регрессии
plt.plot(X_test, y_lin, linewidth=1)
# добавляем линию уровня, по которому точки будут относится к 1 или 2 классу
plt.axhline(.5, color='green')
# рисуем логистическую кривую для вероятностей принадежности объектов к
#классам
plt.plot(X_test, y_log_probabilty, color='blue', linewidth=3)
# настраиваем оси и выводим график
plt.ylabel('y')
plt.xlabel('X')
plt.xticks(range(-5, 10))
plt.yticks([0, 0.5, 1])
plt.ylim(-.25, 1.25)
plt.xlim(-4, 10)
plt.legend(('LabelLogisticRegressionModel', 'LinearRegressionModel',
'ProbabiltyLogRegression'), loc="lowerright", fontsize='small')
plt.show()
```

В результате после выполнения кода появится график, как показано на рисунке 60.

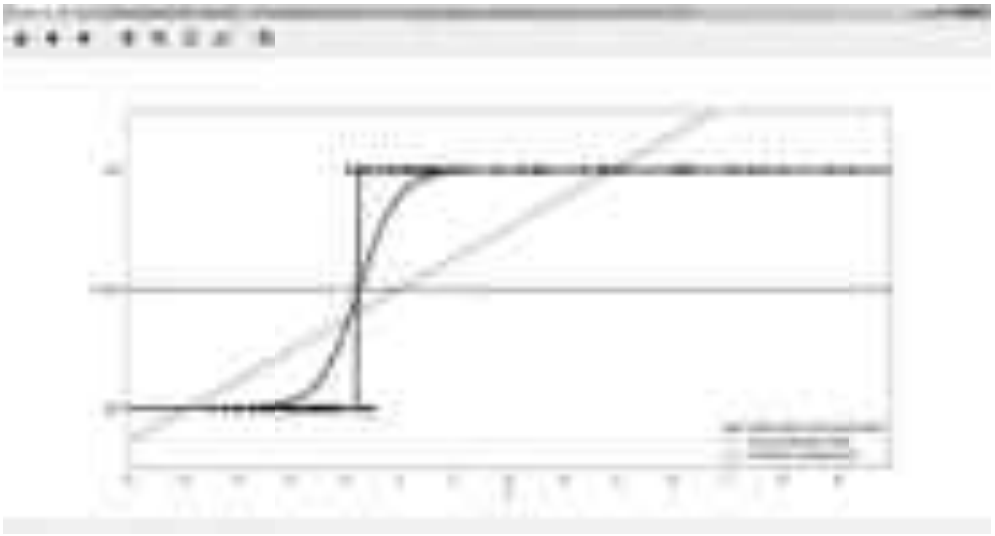


Рисунок 60. Графики регрессий

Для лучшего понимания вам рекомендуется воспроизвести код и посмотреть на график, который вам выведет программа. На нем должно быть видно, что красная кривая (результат логистического предсказания\классификации) лучше приближает распределение точек (лишь несколько точек вылетает). Красная линия построена лишь по меткам, полученным от log_reg.predict, поэтому она имеет такой резкий переход. Такой «округленный» результат легко использовать в дальнейшем, хотя в таком случае мы не получаем информацию о вероятностях. Линейная регрессия (голубая линия) приближает данные не так хорошо.

Важно понимать, что линейная регрессия отнесет все точки (x<1) к классу 0, так как результат функции линейной регрессии будет меньше 0.5. А все точки (X>1) будут отнесены к классу 1, так как результат функции линейной регрессии будет больше 0.5. Для этого на графике выведена разделяющая линия Y=0.5. Граница разделения классов по логистической регрессии проходит примерно в точке X=0.25, что разделяет классы гораздо точнее. Толстая синяя линия отображает вероятностную логистическую кривую, которая дает больше детальной информации о предсказании. Поскольку результат представляет собой не бинарные метки классов, а вероятность принадлежности к выбранном классу, то кривая получается гладкой.

Для точной оценки качества моделей воспользуемся функциями score и accuracy_score в scikit-learn. Поскольку линейная регрессия выдает вещественные значения, а y_test сгенерирована как бинарный массив, то для оценки качества нужно использовать функцию model.score(), которая способна читать и бинарные, и вещественные ответы. Для подсчета качества логистической регрессии можно воспользоваться измерителем качества классификации accuracy_score. Но перед этим нам необходимо получить реальную принадлежность данных тестовой выборки к классам. Сделать это можно с помощью определенной нами функции для обучающей выборки, как показано в листинге 36.

Листинг 36

```
y_test = (X_test > 0).astype(np.float)
```

Теперь можно вывести значения качества классификации для обеих моделей, как показано в листинге 37. Для линейной регрессии это выполняется ее методом score, а для логистической – методом accuracy_score из sklearn.metrics.

Листинг 37

```
from sklearn.metrics import accuracy_score
....
# оцениваем точность решений
lin_score = lin_reg.score(X_test, y_test)
print "Linear regression accuracy: ", lin_score
log_score = accuracy_score(y_log_label, y_test)
print "Logistic regression accuracy: ", log_score
```

В итоге в консоль будут выведены следующие результаты: Linearregression accuracy=0,52442, Logisticregression accuracy=0,98333. Они подтверждают, что логистическая регрессия выполнила классификацию более качественно.

Решение задачи ранжирования признаков

Следующее, что нам необходимо рассмотреть, – применение регрессионных моделей для определения важности признаков. Здесь в качестве примера возьмем регрессионную проблему Фридмана, когда на вход моделей подается 14 факторов, выход рассчитывается по формуле, использующей только пять факторов, но факторы 1-5, а также 10-14 взаимозависимы. Наша задача – посмотреть, как разные виды регрессий оценят важности факторов и какой из них будет иметь наибольшую среднюю значимость по всем моделям. Для работы мы возьмем три модели: линейную регрессию, гребневую и лассо (линейную регрессию с L1-регуляризатором). О реализации в scikit-learn последней модели можно подробнее прочитать здесь: [46].

Первое, что нам необходимо сделать для решения поставленной задачи, – подключить необходимые модули, как в листинге 38. Вторая строка кода из листинга 38 – импорт класса, выполняющего масштабирование данных до заданного диапазона (например, так, чтобы все значение находились в диапазоне от 0 до 1). Подробнее о реализации класса можно прочитать здесь: [47]. Необходимость его использования объясняется следующим: каждая модель регрессии дает оценки важности признаков в своем диапазоне. Для того чтобы найти признак с максимальной средней важностью по трем моделям, нам необходимо привести выданные ими оценки к одному виду, в чем и поможет нам указанный класс.

Листинг 38

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import MinMaxScaler
import numpy as np
```

После импорта необходимых классов мы должны сгенерировать исходные данные, как показано в листинге 39.

Листинг 39

```
#генерируем исходные данные: 750 строк-наблюдений и 14 столбцов-признаков
np.random.seed(0)
size = 750
X = np.random.uniform(0, 1, (size, 14))
#Задаем функцию-выход: регрессионную проблему Фридмана
Y = (10 * np.sin(np.pi*X[:,0]*X[:,1]) + 20*(X[:,2] - .5)**2 +
    10*X[:,3] + 5*X[:,4]**5 + np.random.normal(0,1))
#Добавляем зависимость признаков
X[:,10:] = X[:,:4] + np.random.normal(0, .025, (size,4))
```

Теперь создаем и обучаем модели (листинг 40).

Листинг 40

```
#линейная модель
lr = LinearRegression()
```

Окончание листинга 40

```
lr.fit(X, Y)
#гребневая модель
ridge = Ridge(alpha=7)
ridge.fit(X, Y)
#Лассо
lasso = Lasso(alpha=.05)
lasso.fit(X, Y)
```

Наконец, нам нужно выгрузить в единый массив размера 3×14 (количество_моделей и количество_признаков) все оценки моделей по признакам. Найти средние оценки и вывести результат в формате списка пар {номер_признака – средняя_оценка}, отсортированном по убыванию. Получить оценки признаков можно через поле coef_ у каждой из наших моделей. Для удобства отображения данных поместим их в конструкцию вида: [имя_модели : [{имя_признака : оценка},{имя_признака : оценка}]]. То есть верхним уровнем у нас будет словарь, где ключом будет имя модели. В нем будут располагаться три записи из четырнадцати пар каждая. Пример содержимого списка представлен в листинге 41.

Листинг 41

```
{'Lasso': {'x13': 0.0, 'x14': 0.0, 'x10': 0.0, 'x8': 0.0, 'x9': 0.0, 'x11': 0.0, 'x2': 0.72, 'x3': 0.0,
'x12': 0.0, 'x1': 0.69, 'x6': 0.0, 'x7': 0.0, 'x4': 1.0, 'x5': 0.29},
'Ridge': {'x13': 0.0, 'x14': 0.92, 'x10': 0.01, 'x8': 0.08, 'x9': 0.0, 'x11': 0.59, 'x2': 0.75, 'x3': 0.06,
'x12': 0.67, 'x1': 0.76, 'x6': 0.08, 'x7': 0.03, 'x4': 1.0, 'x5': 0.61},
'Linear reg': {'x13': 0.48, 'x14': 0.12, 'x10': 0.01, 'x8': 0.03, 'x9': 0.0, 'x11': 0.59, 'x2': 0.61, 'x3':
0.51, 'x12': 0.19, 'x1': 1.0, 'x6': 0.02, 'x7': 0.0, 'x4': 0.69, 'x5': 0.19}}
```

Первое, что нам необходимо сделать, чтобы реализовать описанное, – создать список вида ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14'], содержащий имена признаков. Выполнить это можно с помощью кода из листинга 42.

Листинг 42

```
names = ["x%s" % i for i in range(1,15)]
```

Затем нам необходимо объявить функцию, которая на вход будет получать сформированный выше список и список оценок по признакам. Выходом же функции должен быть словарь, составленный из попарно соотнесенных элементов списков (то есть словарь из пар имя_признака: оценка_признака, где первое – ключ, а второе – значение). Причем оценки внутри функции должны быть приведены к единому диапазону от 0 до 1 и округлены до сотых.

В теле функции нам потребуются следующие объекты. Во-первых, класс MinMaxScaler, о котором было сказано выше. Во-вторых, функция abs пакета Numpy для получения абсолютных значений оценок(модуля). В-третьих, функция map, позволяющая применить операцию округления (round) к каждому элементу массива посредствам оператора lambda. В-четвертых, нам понадобятся функции rashape и ravel. Первая – для переформирования входного массива признаков из одной строки и четырнадцати столбцов в четырнадцать строк и один столбец, вторая – для преобразования полученного двумерного массива в одномерный. В-пятых, нам потребуются функции zip и dict. Первая попарно соотнесет имена признаков и оценки, вторая – преобразует полученные пары в словарь. Примеры использования функций zip, map и lambda можно найти здесь: [48]. Код реализации описанных действий можно найти в листинге 43.

Листинг 43

```
def rank_to_dict(ranks, names):
    ranks = np.abs(ranks)

    minmax = MinMaxScaler()

    ranks = minmax.fit_transform(np.array(ranks).reshape(14,1)).ravel()
    ranks = map(lambda x: round(x, 2), ranks)
    return dict(zip(names, ranks))
```

Вызов функции для наших моделей представлен в листинге 44.

Листинг 44

```
ranks["Linear reg"] = rank_to_dict(lr.coef_, names)
ranks["Ridge"] = rank_to_dict(ridge.coef_, names)
ranks["Lasso"] = rank_to_dict(lasso.coef_, names)
```

Последним шагом мы должны сформировать среднее по каждому признаку, загрузив их в отдельный список. Затем отсортировать его по убыванию и вывести. Реализовать это можно, используя код из листинга 45.

Листинг 45

```
#Создаем пустой список для данных
mean = {}

#«Бежим» по списку ranks
for key, value in ranks.iteritems():
#«Пробегаемся» по списку значений ranks, которые являются парой имя:оценка
for item in value.iteritems():
#имя будет ключом для нашего mean
#если элемента с текущим ключем в mean нет - добавляем
if(item[0] not in mean):
mean[item[0]] = 0

#суммируем значения по каждому ключу-имени признака
      mean[item[0]] += item[1]

#находим среднее по каждому признаку
for key, value in mean.iteritems():
   res=value/len(ranks)
   mean[key] = round(res, 2)

#сортируем и распечатываем список
mean = sorted(mean.iteritems(), key=lambda (x, y): y, reverse=True)

print "MEAN"
print mean
```

Результат работы программы представлен в листинге 46.

Листинг 46

```
MEAN
[('x4', 0.9), ('x1', 0.82), ('x2', 0.69), ('x11', 0.39), ('x5', 0.36), ('x14', 0.35), ('x12', 0.29), ('x3', 0.19), ('x13', 0.16), ('x8', 0.04), ('x6', 0.03), ('x10', 0.01), ('x7', 0.01), ('x9', 0.0)]
```

Мы видим, что по средней оценке наиболее важным оказался четвертый признак, затем первый и потом второй. Сравним эти данные с показателями каждой отдельной модели. Для этого отсортируем массив ranks также по убыванию оценок и выведем его, используя код из листинга 47. Результат работы кода представлен в листинге 48.

Листинг 47

```
for key, value in ranks.iteritems():
    ranks[key] = sorted(value.iteritems(), key=lambda (x, y): y, reverse=True)
for key, value in ranks.iteritems():
    print key
    print value
```

Листинг 48

```
Lasso
[('x4', 1.0), ('x2', 0.72), ('x1', 0.69), ('x5', 0.29), ('x13', 0.0), ('x14', 0.0), ('x10', 0.0), ('x8', 0.0), ('x9', 0.0), ('x11', 0.0), ('x3', 0.0), ('x12', 0.0), ('x6', 0.0), ('x7', 0.0)]
Ridge
[('x4', 1.0), ('x14', 0.92), ('x1', 0.76), ('x2', 0.75), ('x12', 0.67), ('x5', 0.61), ('x11', 0.59), ('x8', 0.08), ('x6', 0.08), ('x3', 0.06), ('x7', 0.03), ('x10', 0.01), ('x13', 0.0), ('x9', 0.0)]
Linear reg
[('x1', 1.0), ('x4', 0.69), ('x2', 0.61), ('x11', 0.59), ('x3', 0.51), ('x13', 0.48), ('x12', 0.19), ('x5', 0.19), ('x14', 0.12), ('x8', 0.03), ('x6', 0.02), ('x10', 0.01), ('x9', 0.0), ('x7', 0.0)]
```

Как можно увидеть, метод Lasso отобрал признаки x1, x2, x4, x5 как значимые параметры, а все остальные совсем незначимые (так как они равны нулю). Что довольно близко к истине, так как параметры x1-x5 заданы как независимые случайные величины. Но взвешивание по такому методу довольно грубое, резкое.

Метод Ridge сделал более мягкое взвешивание. Но несмотря на то, что x3 также упущен, его влияние может быть учтено через скорре-

лированные параметры x14, x12, x11 (которые были также отобраны как важные). Поэтому данный метод может быть очень полезным.

Метод линейной регрессии также отдал предпочтение многим действительно важным параметрам (причем x3 здесь имеет гораздо больший вес).

В заключение можно сказать, что учет оценок от разных методов позволит более оптимально выделить значимые признаки и отделить от менее значимых. Но, конечно, без реальных экспериментов, проверяющих выбранное множество параметров, невозможно гарантировать идеального разбиения для всех задач.

Кроме рассмотренных выше моделей для отбора признаков еще можно использовать:

1. RandomizedLasso из sklearn.linear_model (оценки признаков будут находиться в поле scores_). Это так называемое рандомизированное лассо работает, разделяя тренировочные данные на подвыборки и вычисляя Лассо-оценки признаков, где штраф случайного подмножества коэффициентов масштабирован. Применяя такую двойную рандомизацию несколько раз, метод присваивает наивысшие оценки признакам, выбор которых повторялся в процессе рандомизации. Это называется «выбором стабильности». Если же сказать короче, то признаки, которые выбираются наиболее часто, считаются значимыми. Подробнее об этом классе можно почитать здесь: [49].

2. RFE из sklearn.feature_selection (оценки признаков будут находиться в поле ranking_, однако в отличие от всех остальных моделей, класс выдает не веса при коэффициентах регрессии, а именно ранг для каждого признака. Так наиболее важные признаки будут иметь ранг – «1», а менее важные признаки ранг больше «1». Коэффициенты остальных моделей тем важнее, чем больше их абсолютное значение). Класс RFE выполняет ранжирование признаков через рекурсивный отбор путем отсева признаков. Целью рекурсивного устранения элемента (RFE) является выбор признаков путем рекурсивного рассмотрения меньших и меньших наборов признаков, учитывая внешнюю оценку, которая присваивает признакам веса

(например, коэффициенты линейной модели). Сначала оценщик обучается на начальном наборе признаков и ассоциированных с ними весами. Затем признаки, абсолютные веса которых являются наименьшими, удаляются из текущего набора. Эта процедура рекурсивно повторяется на оставшемся множестве до тех пор, пока в конечном итоге не будет достигнуто желаемое количество признаков. То есть работа с этим классом строится, как показано в листинге 49. Подробнее о нем можно почитать здесь: [50].

Листинг 49

```
lr = LinearRegression()
lr.fit(X, Y)

rfe = RFE(lr)
rfe.fit(X,Y)
```

3. RandomForestRegressor из sklearn.ensemble (оценки признаков будут находиться в поле feature_importances_). Регрессор на основе Случайного леса – это мета-оценщик, который обучает несколько классификационных деревьев решений на разных подвыборках данных и использует усреднение для улучшения точности и контроля переобучения. Размер подвыборки всегда соответствует размеру входной выборки, но образцы из выборки перемешиваются (если параметр bootstrap=True, что является значением по умолчанию). Подробнее об этом классе можно посмотреть здесь: [51].

4. f_regression из sklearn.feature_selection (для получения оценок см.код из листинга 50. Оценки будут находиться в объекте f). Это быстрая линейная модель, позволяющая оценить эффект одного признака и применяющаяся последовательно к множеству признаков. В основе ее работы лежит вычисление взаимной корреляции между каждым признаком и выходной переменной. Более подробно можно посмотреть здесь: [52].

Листинг 50

```
f, pval  = f_regression(X, Y, center=True)
```

Работа с полиномиальной регрессией

Напоследок, заканчивая разговор о регрессионных моделях, мы познакомимся с полиномиальной регрессией, а также посмотрим на практике такое явление, как переобучение модели. Рассматривать мы это будем на следующем примере: сгенерируем точки функцией косинуса и попробуем аппроксимировать их линейной регрессией, полиномиальной регрессией со степенью 4 и 15, а также гребневой регрессией и созданными на основе нее полиномами 4 и 15 степеней. Затем рассчитаем точность моделей, выведем шесть графиков и сравним результаты.

Итак, первым шагом создадим обычную функция косинуса от аргумента, записанную в виде lambda-выражения (листинг 51). Особенности Python позволяют использовать в качестве аргумента не только числа, но и массивы\векторы (так что использование циклов не требуется).

Листинг 51

```
import numpy as np
true_fun = lambda X: np.cos(1.5 * np.pi * X)
```

Представленная в листинге 51 функция будет использоваться как истинная закономерность в некоторых данных. То есть именно то, что аналитики и ученые пытаются выяснить, проводя измерения некоторого объекта природы или бизнеса. Собственно, поэтому название функции – true_fun. В дальнейшем эту функцию можно будет вызывать подобно функции, которая определена через def, как показано в листинге 52.

Листинг 52

```
result= true_fun(something_data)
```

Далее, для корректного проведения эксперимента необходимо создать некую выборку, которая будет служить аналогом измерений истинной закономерности. Для этого мы создадим распределение\выборку

на основе объявленной ранее функции с небольшим случайным смещением по X и Y (листинг 53).

Листинг 53

```
np.random.seed(0)
n_samples = 30
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1
```

Полиномиальная регрессия задается в несколько шагов. Сперва необходимо определить коэффициенты полинома (которые автоматически генерируются с помощью функции PolynomialFeatures) и модель для основы (в нашем случае линейную или гребневую). Затем необходимо соединить вместе базовую регрессию и полиномиальные коэффициенты (вектор или матрицу преобразования) как последовательность трансформаций некоторых данных. Итоговый объект и будет являться полиномиальной моделью. Трансформацию данных можно выполнить с помощью объекта Pipeline из sklearn.pipeline (подробнее о нем можно почитать здесь: [53]).

Для удобства работы и сокращения кода сделаем следующее: объявим массив с нужными нам степенями полинома, как показано в листинге 54.

Листинг 54

```
degrees = [1, 4, 15]
```

Затем объявим функцию, которая будет принимать в качестве параметров номер степени (индекс элемента в массиве dergees), полотно, на котором нужно отобразить график, базовую модель и уровень графика на полотне (первый или второй ряд по вертикали). Внутри тела функции будет создаваться и обучаться нужная полиномиальная модель, результат работы которой будет выведен на возвращаемом функцией графике. Кроме точек на графике будет отображена точность модели, вычисленная методом кросс-валидации:

средние оценки качества по метрике «neg_mean_squared_error» (среднеквадратичная ошибка). Эти оценки будут получены с помощью объекта cross_val_score из sklearn.model_selection. Подробнее о кросс-валидации в scikit можно почитать здесь: [54, 55].

Объявление функции показано в листинге 56, с работой которого вам предлагается разобраться самостоятельно. Единственно, поясним назначение параметра cv в cross_val_score. Параметр cv определяет, на сколько групп будет разбита тренировочная выборка. Каждая группа будет содержать тренировочную часть А и тестовую Б. Причем эти множества (А и Б) не будут пересекаться. Так для следующих данных: входа X = np.array([[11, 2], [3, 14], [1, 2], [3, 4]]), выхода y = np.array([1, 2, 3, 4]) cross_val_score с параметром cv=2 даст следующее разбиение на 2 группы\итерации (листинг 55).

Листинг 55

```
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
```

Массив TRAIN содержит это индексы тех объектов, которые входят в тренировочную выборку. То есть TRAIN: [2 3] означает, что в тренировочную выборку войдут второй и третий объекты из X: [1,2], [3, 4]. Все остальные – в тестовую.

Листинг 56

```
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import cross_val_score
def MakeExample(index, plt, model,k):
   polynomial_features = PolynomialFeatures(degree=degrees[index],
                                            include_bias=False)
```

Окончание листинга 56

```
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", model)])
    pipeline.fit(X[:, np.newaxis], y)
    scores = cross_val_score(pipeline, X[:, np.newaxis], y,
                             scoring="neg_mean_squared_error", cv=10)
    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    if k == 1:
      plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    if k==0:
     plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(degrees[i], -scores.mean(),
scores.std()))
    else:
     plt.title("MSE = {:.2e}(+/- {:.2e})".format(-scores.mean(), scores.std()))
    return plt
```

Код вызова функции из листинга 56 (и, соответственно отрисовка графиков) представлен в листинге 57, а результат работы кода – на рисунке 61.

Листинг 57

```
plt.figure(figsize=(14, 6))
for i in range(len(degrees)):
    linear_regression = LinearRegression()
    plt.subplot(2, len(degrees), i+1)
    plt = MakeExample(i, plt, linear_regression,0)
for i in range(len(degrees)):
    ridge = Ridge(alpha=0.02)
    plt.subplot(2, len(degrees), len(degrees) + i + 1)
    plt = MakeExample(i, plt, ridge,1)
plt.show()
```

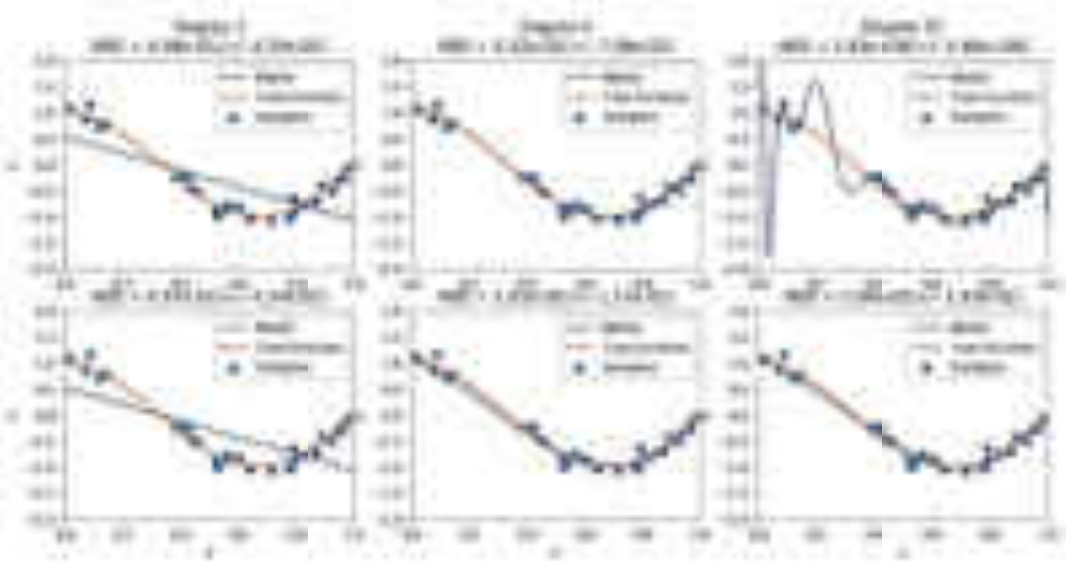


Рисунок 61. Результаты работы с разными моделями

Для лучшего понимания вам рекомендуется воспроизвести код и посмотреть на рисунок, выданный вашей программой. Если говорить о моделях, построенных на основе линейной регрессии, то, сравнивая первый и второй графики (полином 1 и 4 степени), можно сделать вывод, что повышение степени положительно влияет на качество модели. Но третий график (полином 15 степени) демонстрирует большую ошибку. Как мы видим, на тренировочной выборке модель показала хороший результат (кривая проходит через все точки выборки). Но на тестовой выборке модель «ведет себя плохо» (плохо приближает true_fun). Это и означает, что при повышении сложности модели есть риск перейти за критическую точку и переобучить модель. Однако, как мы видим на графиках гребневой регрессии, регуляризация позволяет улучшить качество решения.

Работа с простейшими моделями нейронных сетей

Следующее, с чем нам необходимо познакомиться, – работа с простейшими моделям нейронных сетей, которые содержатся в библиотеке Sklearn (в дальнейшем будет рассмотрена работа с другими библиотеками, которые содержат более сложные алго-ритмы и позволяют создавать другие нейроклассификаторы и модели). Однако Sklearn не специализируется на нейросетевых моделях и не содержит более сложные алгоритмы.

Начнем мы с обычного персептрона, являющегося одним из вариантов линейных моделей. Это распространенный класс моделей, которые отличаются своей простотой и скоростью работы. Их можно обучать за разумное время на очень больших объемах данных, и при этом они могут работать с любыми типами признаков: вещественными, категориальными, разреженными. Рассмотрим решение следующей задачи: пусть у нас есть данные вида, показанного на рисунке 62, и сохраненные в файлах perceptron-train.csv и perceptron-test.csv (скачать файлы можно отсюда: [41]). Целевая переменная записана в первом столбце, признаки – во втором и третьем. Выполним следующее: применим к этим данным однослойный персептрон (о реализации модели Perceptron более подробно здесь: [56]), многослойный персептрон (о реализации модели MLPClassifier более подробно здесь: [57]), сравним точности предсказания моделей, затем нормализуем данные и повторим сравнение. Причем данные мы будем прогонять 100 раз, и на каждом прогоне выполнять 2000 эпох обучения. В качестве показателей выведем минимальные, максимальные, средние значения accuracy и стандартное отклонение, полученные на 100 прогонах. А также выведем график, визуализирующий результаты.

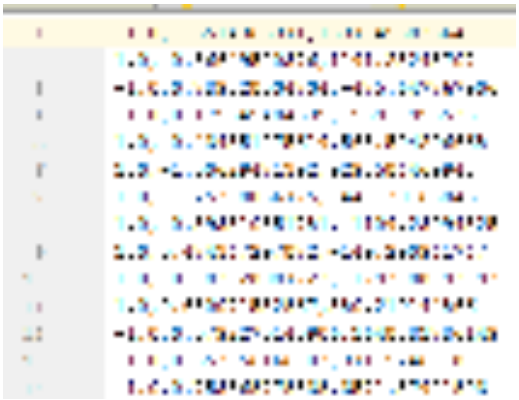


Рисунок 62. Данные для тестовой задачи

Поясним необходимость нормализации. Как и в случае с метрическими методами, качество линейных алгоритмов зависит от некоторых свойств данных. В частности, признаки должны иметь одина-

ковый масштаб. Если это не так, и масштаб одного признака сильно превосходит масштаб других, то качество может резко упасть.

Один из способов достижения нужного свойства признаков заключается в их стандартизации. Для этого берется набор значений признака на всех объектах, вычисляется их среднее значение и стандартное отклонение. После этого из всех значений признака вычитается среднее, и затем полученная разность делится на стандартное отклонение. В ходе нашего эксперимента мы выясним, улучшает ли нормализация качество работы модели. Код решения задачи (с комментариями) представлен в листинге 58. Результат программы, выдаваемый в консоль, представлен на рисунке 63, а построенный график – на рисунке 64.

Листинг 58

```
# -*- coding: utf-8 -*-
#Необходимые импорты
import numpy as np
import pandas
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#функция расчета медианы
def median(lst):
    return np.median(np.array(lst))
#Загрузка тренировочных данных
data_train = pandas.read_csv('perceptron-train.csv')
X_train = data_train.ix[:, 1:3].values
y_train = data_train.ix[:, 0].values
#Загрузка тестовых данных
data_test = pandas.read_csv('perceptron-test.csv')
X_test = data_test.ix[:, 1:3].values
y_test = data_test.ix[:, 0].values
```

Продолжение листинга 58

```
#Инициализация массива для счетчика итераций
rs = np.linspace(0,100,num=100)
#Инициализация списков для сохранения accuracy моделей
acc_p = []
acc_pn = []
acc_mlp = []
acc_mlpn = []
#Цикл прогона моделей
for i in rs:
    i = int(i)
#Распечатка номера итерации
   print "Random: ", i
#Создание модели персептрона
    clf = Perceptron(random_state=i, alpha=0.01, n_iter=2000)
#Обучение модели
   clf.fit(X_train, y_train)
#Получение прогноза
    predictions = clf.predict(X_test)
# Расчет показателя accuracy
    acc = accuracy_score(y_test, predictions)
#Распечатка результата
    print "Perceptron: ", acc
#Добавление оценки в список оценок для модели персептрона
    acc_p.append(acc)
#Нормализация данных
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
#Работа пресептрона с нормализованными данными
    clf = Perceptron(random_state=i, alpha=0.01, n_iter=2000)
    clf.fit(X_train_scaled, y_train)
    predictions = clf.predict(X_test_scaled)
    acc = accuracy_score(y_test, predictions)
    print "Perceptron with normalization: ", acc
    acc_pn.append(acc)
#Создание многослойного классификатора
    mlp = MLPClassifier(random_state=i, solver="sgd", activation="tanh", alpha=0.01,
hidden_layer_sizes=(2, ), max_iter=2000, tol=0.00000001)
```

Окончание листинга 58

```
    mlp.fit(X_train, y_train)
#Работа с ненормализованными данными
    predictions = mlp.predict(X_test)
    acc = accuracy_score(y_test, predictions)
    print "MLP: ",  acc
    acc_mlp.append(acc)
#Работа с нормализованными данными
    mlp = MLPClassifier(random_state=i, solver="sgd", activation="tanh", alpha=0.01,
hidden_layer_sizes=(2, ), max_iter=2000, tol=0.00000001)
    mlp.fit(X_train_scaled, y_train)
    predictions = mlp.predict(X_test_scaled)
    acc = accuracy_score(y_test, predictions)
    print "MLPwith Norm: ",  acc
    acc_mlpn.append(acc)
#Распечатка итоговых результатов
print "Perceptron: ", min(acc_p), median(acc_p), max(acc_p), np.std(acc_p)
print "Perceptron with Norm: ", min(acc_pn), median(acc_pn), max(acc_pn),
np.std(acc_pn)
print "MLP: ", min(acc_mlp), median(acc_mlp), max(acc_mlp), np.std(acc_mlp)
print "MLP with Norm: ", min(acc_mlpn), median(acc_mlpn), max(acc_mlpn),
np.std(acc_mlpn)
#Расчет минимума и максимума для графика
X = np.concatenate((X_train, X_test), axis=0)
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
#Построение графика
figure = plt.figure(figsize=(17, 9))
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
ax = plt.subplot(1, 1, 1)
# Точки из обучающей выборки
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright)
# Тестовые точки
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.6)
ax.set_xlim(x_min, x_max)
ax.set_ylim(y_min, y_max)
ax.set_xticks(())
ax.set_yticks(())
plt.show()
```

Рисунок 63. Результат работы программы

Воспроизведите код, посмотрите на полученный график. Он должен быть похож на изображенный на рисунке 64. Обратим внимание на один важный технический момент. На графике можно увидеть, что принадлежность к классу (то есть по сути Y) отображается не отдельным измерением, а цветом. Такой механизм позволяет рисовать двумерные графики вместо трехмерных, что проще и нагляднее.

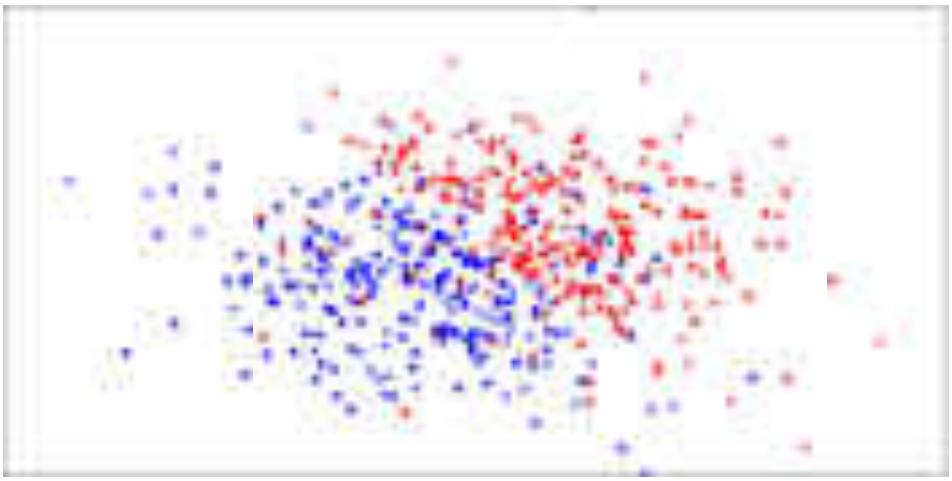


Рисунок 64. Итоговый график

Содержательный анализ результатов работы программы и доработку кода вам предлагается выполнить самостоятельно (см. практическое задание «Многослойный персептрон»).

Реализация алгоритма обучения нейронной сети

Данный пункт посвящен алгоритму обратного распространения ошибки как классическому подходу к обучению нейронных сетей. Поскольку нейронные сети сегодня – это важный класс моделей для построения сложных иерархических признаков, то возникает необходимость детального понимания протекающих в них процессов. Важность именно понимания нейронных сетей обуславливается еще тем, что сложнейшие на сегодняшний день сети еще плохо описаны математически, и требуется проделать много исследовательской работы в этом направлении.

Задача состоит в том, чтобы реализовать обучение полносвязной трехслойной сети прямого распространения на каких-то простых синтетических данных.

Если вы собираетесь заниматься именно исследованием в ML, то подобного рода задачи – это ваша обычная работа. Если вы собираетесь заниматься решением конкретных проблем бизнеса, то с такими задачами вам сталкиваться вряд ли придется, и основная работа будет посвящена подготовке и интерпретации данных. Подготовке данных было уже уделено много внимания, поэтому сейчас рассмотрим исследовательскую задачу. Однако прежде чем перейти непосредственно к ее реализации, рассмотрим применение матричной записи данных в ней.

Так, если вы хотите взвесить некоторые значения X по некоторым весам W и получить сумму, то не обязательно писать выражение $w_1 \times x_1 + w_2 \times x_2$. Ту же самую операцию можно реализовать в виде матричного умножения. Первая матрица A будет представлять из себя X, где столбцы – признаки, а строки – традиционно объекты (если вы обрабатываете один объект за раз, то это может быть матрица с одной строкой). Вторая матрица B будет представлять из себя матрицу весов, где каждая колонка будет отождествляться с нейроном следующего слоя, а каждая строка будет отвечать за конкретный вес (значение веса). Однако даже матрица с одной строкой

в Python будет представлена как двумерный массив. Вы можете поэкспериментировать самостоятельно, выводя различные массивы\матрицы в консоль. После перемножения таких двух матриц получим новую матрицу C, где каждая строка будет представлять собой взвешенный набор признаков $X (x_1, ..., x_n)$ по $W (w_1, ..., w_n)$. Схематично это выглядит так, как показано на рисунке 65 [1].

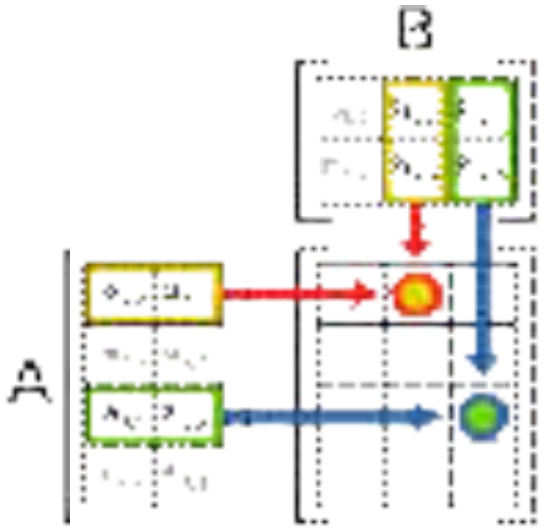


Рисунок 65. Умножение матриц

Необходимо отметить, что взвешивание значений с точки зрения матрицы происходит так, что из матрицы A берутся строки, а из матрицы B берутся колонки. Поэтому иногда массивы\матрицы нужно будет транспонировать. Напомним суть операции транспонирования, приведя в качестве примера рисунок 66.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Рисунок 66. Транспонирование матриц

Объявления функций активации и ее производной представлены в листинге 59.

Листинг 59

```
# -*- coding: utf-8 -*-
#Необходимый импорт
import numpy as np
#Определяем функцию активации
def activation(x):
    return 1 / (1 + np.exp(-x))
#Определяем производную от функции активации
def sigma_derivative(x):
    return x * (1 - x)
```

Реализация алгоритма обраного распространения ошибки с подробными комментариями приведена в листинге 60. Забегая вперед, скажем, что в практической части вам предлагается задача модификации данного кода и интерпретации его результатов.

Листинг 60

```
# X - это матрица, где столбцы это признаки,
# а строки это объекты выборки.
X = np.array([[0, 0, 1],
              [0, 1, 0],
              [1, 0, 0],
              [1, 1, 1]])
# Y - это матрица, где столбцы это признаки
# (в данном случае один целевой признак),
# а строки это объекты выборки.
y = np.array([[0],
              [1],
              [1],
              [0]])
# Зафиксируем генератор случайных чисел, чтобы получать каждый раз
#предсказуемый (одинаковый) результат
np.random.seed(4)
# Нотация слоев - L1, L2, L3.
# В каждом слое есть n1, n2, n3 нейронов.
# Строки в матрицах индексируются как i, столбцы как j.
# W_1_2 - это матрица весов между первым и вторым слоем.
# Строки определяют левый нейрон (т.е. нейрон первого слоя) и соответственно
количество строк равно n1.
```

Продолжение листинга 60

```
# Столбцы определяют правый нейрон (т.е. нейрон второго слоя) и соответственно количество столбцов равно n2.
# На пересечении i-й строки и j-го столбца получаем ячейку с конкретным весом, который
# связывает i-ый левый нейрон (слоя L1) и j-й правый нейрон (слоя L2)
W_1_2 = 2 * np.random.random((3, 2)) - 1
# W_2_3 - это матрица весов между вторым и третьим слоем. Все остальное как в W_1_2
W_2_3 = 2 * np.random.random((2, 1)) - 1
#скорость движения по антиградиенту
speed = 1.1
#цикл прогона модели
for j in range(100000):
    # lo, l1, l2 - матрицы определенного слоя сети. Каждая строка матрицы это реакция #на i-й объект входа. Каждая колонка матрицы это реакция j-го нейрона #соответствующего слоя на разные входные образы.
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной #реакцией j-го нейрона на конкретный вход.
    # Первый слой нейронов полностью принимает значения входа X (т.е. все реакции #единичные).
    l1 = X
    # Второй слой нейронов рассчитывается как функция активации по каждому #элементу матрицы U
    # По сути l2 это матрица 4 на 4, где в каждой ячейке результат активации нейрона #второго слоя для всех 4-х входных образов
    # Детально:
    # матрица U = np.dot(l0, W_0_1) является результатом
    # матричного произведения выходов нейронов предыдущего слоя на веса между 1 #и 2 слоем.
    # Строки матрицы U отвечают за конкретный входной образ (объект).
    # Столбцы матрицы U отвечают за нейроны правого слоя (L2).
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной #взвешенной суммой
    # для i-го входного образа и j-го нейрона слоя (l2). Иными словами, для каждого #нейрона слоя L2 и для каждого входа
    # считается U = W1X1 + W2X2 + W3X3 (поскольку входной нейрон содержит 3 #нейрона).
    l2 = activation(np.dot(l1, W_1_2))
    # тоже самое проделываем для Третьего слоя
```

Продолжение листинга 60

```
    # По сути l3 это матрица 4 на 1, где в каждой ячейке результат активации нейрона
#третьего слоя (а он один)
    #  для всех 4-х входных образов
    l3 = activation(np.dot(l2, W_2_3))
    # рассчитывает ошибку на выходе
    l3_error = y - l3
# рассчитывает модуль средней ошибки
    if (j % 10000) == 0:
        print "Error:" + str(np.mean(np.abs(l3_error)))
    # sigma - есть локальный градиент ошибки
    # l3_sigma рассчитывается как ошибка выхода всей сети на производную функции
#активации всех нейронов L3.
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной сигмой
    # для i-го входного образа и j-го нейрона слоя (l3). Иными словами, для каждого
#нейрона слоя L3
    # и для каждого входа рассчитывается производная по функции активации
#умноженная на ошибку.
    # То есть l3_sigma будет матрица 4 на 1 (поскольку в L3 только один нейрон).
    l3_sigma = l3_error * sigma_derivative(l3)
    print l3_sigma
    # Ошибка L2 слоя оценивается через взвешенную сигму слоя L3 по весам между
#L2 и L3
    # Поскольку l3_sigma это матрица 4 на 1 и матрица W_2_3 4 на 1, то последнюю
#матрицу
    # надо Транспонировать, чтобы выполнить правило умножения матриц и взвесить
#элементы l3_sigma
    # по элементам W_2_3.
  # Тогда итоговая матрица l1_error будет 4 на 4, где на пересечении i-й строки и j-го
#столбца
    # будет ячейка с конкретной ошибкой j-го нейрона слоя L2 для i-го входного
#образа.
    l2_error = l3_sigma.dot(W_2_3.T)
    print l2_error
    # l2_sigma рассчитывается как ошибка слоя L2 на производную функции активации
#всех нейронов L2. На пересечении i-й строки и j-го столбца получаем ячейку с
#конкретной сигмой для i-го входного образа и j-го нейрона слоя (L2). Иными
#словами, для каждого нейрона слоя L2 и для каждого входа рассчитывается
#производная по функции активации, умноженная на ошибку.
```

Окончание листинга 60

```
    # То есть l2_sigma будет матрица 4 на 4 (поскольку в L2 четыре нейрона).
    l2_sigma = l2_error * sigma_derivative(l2)
    print l2_sigma
   # обновляем веса
    # l2 это матрица 4 на 4, где в каждой ячейке результат активации нейрона второго
#слоя для всех 4-х входных образов (по строкам).
    # A l3_sigma это матрица 4 на 1, где в каждой строке локальный градиент ошибки
#для 4-х входных образов.
    # Чтобы взвесить столбец матрицы l2 по локальному градиенту (l3_sigma), нужно
#транспонировать матрицу l2, чтобы
    # результат активации каждого нейрона в слое L2 по каждому входному образу
#вытянулся в строку.
    # Тогда соответствующая строка умножится на столбик l3_sigma.
    # Заметьте, что транспонирование l3_sigma не даст результата, так как тогда в
#каждом столбце
    # l3_sigma будет по одному значению, а в каждой строке l2 по 4 значения, а значит
#такое перемножение матриц не пройдет. Применяем транспонирование к l2.
    W_2_3 += speed * l2.T.dot(l3_sigma)
    # аналогично W_2_3
    W_1_2 += speed * l1.T.dot(l2_sigma)
# Прямое распространение для тестовых данных
X_test = np.array([[0, 0, 0],
                   [0, 1, 1],
                   [1, 0, 1],
                   [1, 1, 0],
                   [0.5, 0.5, 0],
                   [0.5, 0.5, 1]])
# Y_test должен получиться [1, 0, 0, 1, 1, 0]
l1 = X_test
l2 = activation(np.dot(l1, W_1_2))
l3 = activation(np.dot(l2, W_2_3))
print l3
```

Регуляризация и сеть прямого распространения

В отличие от предыдущих пунктов здесь не будет кода целиком, а только общие рекомендации по выполнению экспериментов, позволяющих выявить влияние регуляризации на многослойную сеть прямого распространения. В ходе работы над пунктом вы научитесь

генерировать синтетические данные, проводить серии экспериментов, анализировать возможности моделей на разнотипных данных и сравнивать результаты экспериментов.

Необходимые для выполнения работы операторы import представлены в листинге 61.

Листинг 61

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
```

Первый шаг, который вам необходимо выполнить, – генерация данных. Библиотека Scikit-learn содержит множество наборов данных (sklearn.datasets). Среди этого набора есть реальные данные (выборки, которые можно загрузить при помощи функций – Loaders), так и специальные генераторы синтетических данных – Samplesgenerator. Подобные генераторы позволяют создать данные на лету по заданными распределениям, функциям, с возможностью добавления шума, случайных отклонений, смещений и т. п.

Есть несколько классических сценариев для проверки возможностей моделей в машинном обучении. Мы воспользуемся одним генератором линейных задач (datasets.make_classification) и двумя нелинейными генераторами (make_moons, make_circles). Отметим, что одной из важнейших характеристик данных является размерность. Важно понимать, что величина размерности данных соответствует количеству признаков, которые описывают исходные объекты. То есть каждая ось декартового пространства закрепляется за одним количественным признаком. Не все генераторы данных позволяют задавать размерность данных. Тем ни менее ниже приведены крайне важные распределения данных, которые иллюстрируют разные типы задач в машинном обучении. Соответственно тестирование моделей

на подобных данных позволяет без наличия реальных данных проверять какие-либо гипотезы.

Пример генераторов:

- make_classification – позволяет сгенерировать такие классы-признаки, которые будут линейно разделяться в n-мерном пространстве (см. рисунок 67. а).
- make_circles – позволяет сгенерировать такие классы-признаки, что признаки одного класса геометрически окружают признаки другого класса (см. рисунок 67.б).
- make_moons – позволяет сгенерировать такие классы-признаки, что признаки одного класса частично пересекаются с признаками другого класса (см. рисунок 67. в).

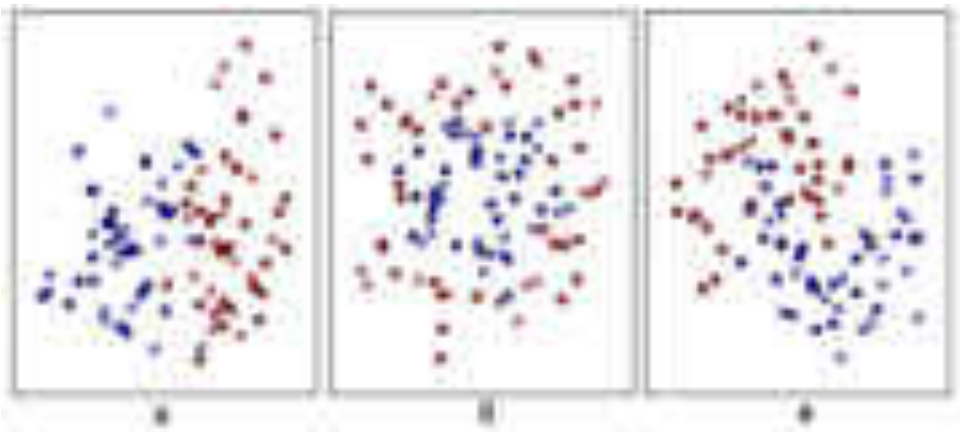


Рисунок 67. Виды распределений признаков, сгенерированных инструментами Scikit-learn

Самостоятельно найдите в Scikit информацию по различным генераторам и сгенерируйте синтетические данные для проведения экспериментов. Пример кода для создания данных трех разнотипных задач классификации приведен в листинге 62.

Для удобства дальнейшей работы рекомендуется объединить все наборы данных в список, как показано в листинге 63, тогда вы сможете организовать цикл по каждому элементу этого списка (что приведено в коде). В этом же листинге есть пример масштабирования данных и разбиения их на тренировочную тестовую выборку.

Листинг 62

```
X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
random_state=0, n_clusters_per_class=1)
rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_dataset = (X, y)
moon_dataset = make_moons(noise=0.3, random_state=0)
circles_dataset = make_circles(noise=0.2, factor=0.5, random_state=1)
```

Листинг 63

```
datasets = [moon_dataset, circles_dataset, linearly_dataset]
for ds_cnt, ds in enumerate(datasets):
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=42)
```

Функция train_test_split делит данные так, что тестовая выборка составляет 40% от исходного набора данных. Разделение происходит случайным образом (т.е. элементы берутся из исходной выборки не последовательно). В этом же цикле можно организовать обучение моделей, их тестирование и формирования графиков.

В теле цикла создайте коллекцию многослойных сетей (MLPClassifier) с различными коэффициентами регуляризации (alpha) и обучите каждую модель на каждом типе данных. Всего должно получиться 15 моделей (3 типа данных\задачи и 5 видов коэффициентов регуляризации). Для реализации используйте код из листинга 64.

Листинг 64

```
alphas = np.logspace(-5, 3, 5)
```

Следующим шагом отобразите исходные данные и результаты на графиках. Пример отображения исходных данных уже был приведен в предыдущих пунктах. Однако на этот раз необходимо построить не один график, а серию графиков в одном окне. В итоге должна получиться таблица из графиков (3 строки, 6 столбцов). Для этого изучите функцию subplot из библиотеки matplotlib. Пример кода для инициализации нового графика (если точнее, то подграфика в рамках одного полотна\окна) показан в листинге 65.

Листинг 65

```
current_subplot= plt.subplot(3, 5 + 1, i)
```

Показанным в листинге 65 способом мы можем получить один из графиков в полотне и заполнять его данными, а потом таким же образом получить следующий график (то есть объект current_subplot). В первую очередь на всех графиках необходимо отобразить исходные данные (точки\признаки). Можете отобразить тренировочную и тестовую выборку или только одну. Пример кода, в котором точки тестовой выборки рисуются полупрозрачными (alpha=0.6), приведен в листинге 66.

Листинг 66

```
current_subplot.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright)
current_subplot.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,alpha=0.6)
```

В листинге 67 cm и cm_bright это цветовые схемы для отрисовки графиков. Их можно взять как из готового набора, так и задать вручную. Это делается следующим образом (листинг 65):

Листинг 67

```
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
```

При отрисовке графиков рекомендуется придерживаться следующего. Пусть левый столбец графиков отображает только исходные данные и ничего больше. Остальные же столбцы должны отображать функции обученных моделей в пространстве исходных данных. Это необходимо, чтобы визуально оценить, как каждая модель разделила исходное пространство признаков. Вместо построения значений функции Y на отдельной оси отобразите значения Y в виде цветовых градиентов (чтобы не строить трехмерные графики). Пример визуализации показан на рисунке 68.

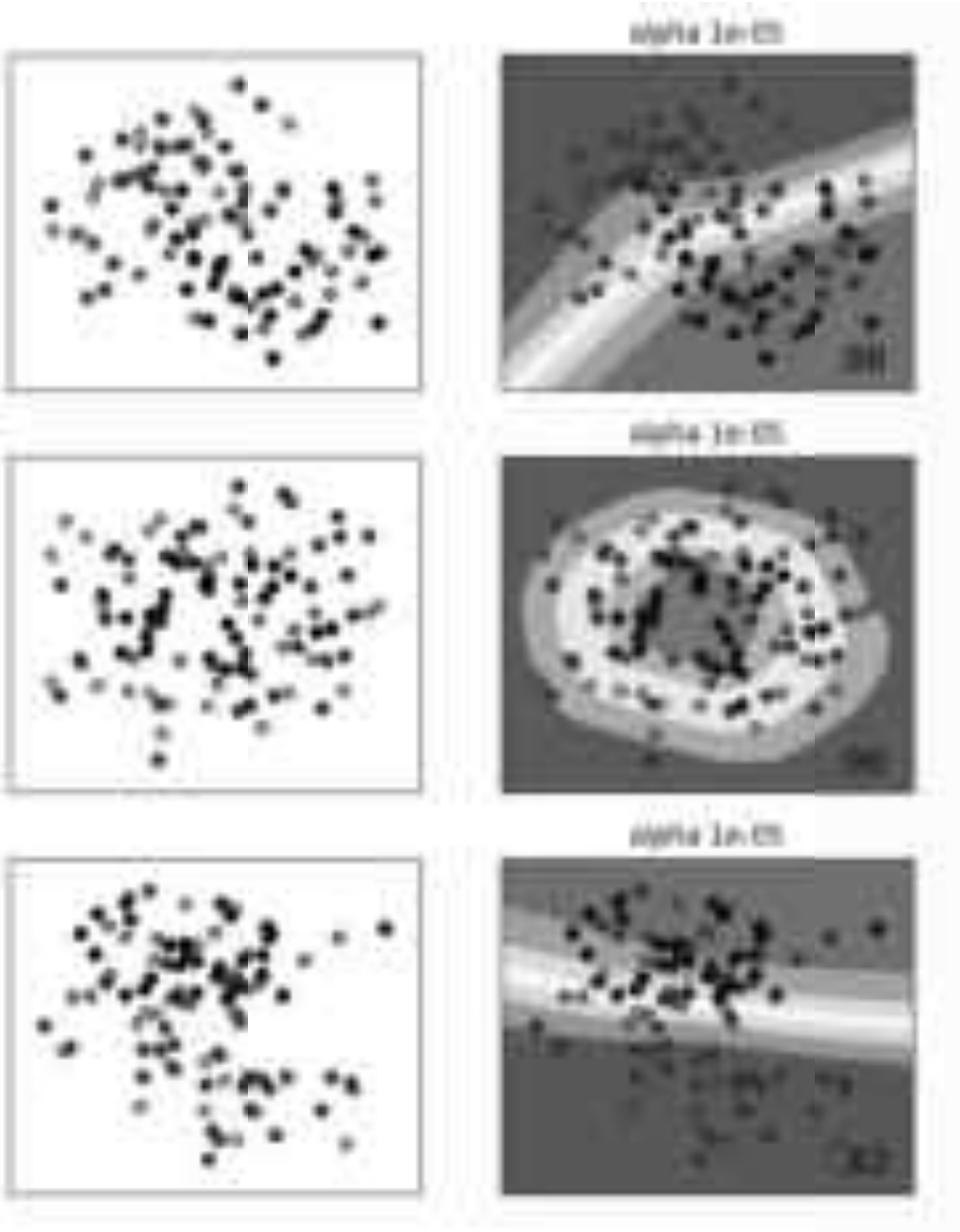


Рисунок 68. Пример визуализации качества моделей

Здесь слева представлена колонка с тремя графиками исходных распределений признаков (на каждом графике две группы точек исходных данных). Каждая строка отвечает за свою задачу классификации (moondataset, circledataset, linearseparabledataset). Правая колонка демонстрирует как конкретная модель многослойной сети MLP (с параметром alpha=0.00001) классифицирует исходное распределение. Значения модели в каждой точке признаков представлены цветовым градиентом. Это гораздо более удобный способ визуализации, чем построение трехмерных поверхностей.

Как мы видим, правый столбец отображает градиент. Чтобы получить градиент, необходимо определить множество точек, по которым будет строиться график. Это можно сделать при помощи регулярной сетки, как показано в листинге 68.

Листинг 68

```
h = .02  #шаг регулярной сетки
x0_min, x0_max = X[:, 0].min() - .5, X[:, 0].max() + .5
x1_min, x1_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx0, xx1 = np.meshgrid(np.arange(x0_min, x0_max, h), np.arange(x1_min, x1_max, h))
```

В листинге 69 представлено несколько вариантов кода для получения значений функции модели в указанных точках сетки.

Листинг 69

```
Z = clf.decision_function(np.c_[xx0.ravel(), xx1.ravel()])#1
Z = clf.predict_proba(np.c_[xx0.ravel(), xx1.ravel()])[:, 1]#2
Z = clf.predict(np.c_[xx0.ravel(),xx1.ravel()])#3
```

Сделаем некоторые пояснения к коду:

1. decision_function возвращает расстояние до разделяющей гиперплоскости в соответствующем измерении (проекционной оси).
2. predict_proba возвращает вероятность принадлежности к каждому классу (так для двух классов можно брать одну из вероятностей (один из столбцов двумерного массива) и исходя из значения раскрашивать область в соответствующий цвет).
3. Predict возвращается целочисленную оценку принадлежности к классу (как бинарная маска).

Третий вариант в данном случае выдаст округленные значения классов (0, 1 и т. д.), а функция predict_proba выдает сглаженную функцию вероятности класса. Чтобы понять разницу, попробуйте различные варианты.

Однако не все классификаторы имеют decision_function. Чтобы узнать, содержит ли текущий объект интересующую нас функцию, можно воспользоваться методом hasattr. Этот метод возвращает флаг,

указывающий на то, содержит ли объект указанный атрибут и нужен для обработки разных вариантов классификаторов (можно выстроить на этом if-else логику). Пример ее вызова показан в листинге 70.

Листинг 70

```
hasattr(clf, "decision_function")
```

Далее вам необходимо нарисовать результат, оси, заголовок и текст со значением точности модели. Выполнить это можно с помощью кода из листинга 71.

Листинг 71

```
Z = Z.reshape(xx0.shape)
current_subplot.contourf(xx0, xx1, Z, cmap=cm, alpha=.8)
current_subplot.set_xlim(xx0.min(),xx0.max())
current_subplot.set_ylim(xx0.min(),xx1.max())
current_subplot.set_xticks(())
current_subplot.set_yticks(())
current_subplot.set_title(name)
current_subplot.text(xx0.max() - .3, xx1.min() + .3, ('%.2f' % score).lstrip('0'),
size=15, horizontalalignment='right')
```

После отрисовки градиента добавьте на каждый график тестовую выборку (точки из каждой задачи для графика каждой модели). Это можно сделать таким же образом, как показано в листинге 72.

Листинг 72

```
current_subplot.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,alpha=0.6)
```

Не забудьте, что после построения каждого графика необходимо инкрементировать переменную i для функции subplot(), а также помните о необходимости применить функцию plt.show() в конце всех операций формирования графиков для вывода результата в окне ОС.

При желании вы можете настроить отступы графиков в окне просмотра при помощи следующего кода (листинг 73).

Листинг 73

```
figure.subplots_adjust(left=.02, right=.98)
```

Для задания заголовков и вывода текста на графиков можно использовать следующие функции из листинга 74.

Листинг 74

```
current_subplot.set_title("Input data")
current_subplot.text(xx0.max() - .3, xx1.min() + .3, ('%.2f' % score).lstrip('0'), size=15,
horizontalalignment='right')
```

Работа с библиотеками Keras и Theano. Настройка под Windows

Мы закончили рассмотрение простых моделей нейронных сетей и переходим к более сложным. В частности, к сверточным сетям. Однако поскольку в библиотеке scikit нет полноценной поддержки таких сетей, то в данном пункте мы рассмотрим установку и настройку библиотек Keras и Theano, потому что для пользователей Windows установка этих библиотек нетривиальна.

Для разворачивания Keras и Theano для CPU на Windows выполните следующее:

1. Откройте Anaconda prompt (командную строку Анаконды через меню «Пуск» или просто выполните стандартную команду cmd. Если все установлено верно, то Anaconda «перехватит» адресованные ей команды)
2. Выполните conda update conda
3. Выполните conda update --all
4. Выполните conda install mingw libpython
5. УстановитеTheano, pip install Theano
6. Установите Keras pip install keras
7. Выполните код для проверки корректности всех действий (он показан в листинге 75).

Листинг 75

```
import numpy as np
import sklearn.linear_model as lin
from keras.layers.core import Dense, Activation
from keras.models import Sequential
from keras.optimizers import RMSprop

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([0, 1, 1, 0]).reshape(4, 1)
x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

model = Sequential()
model.add(Dense(2, init='lecun_uniform', input_shape=(2,)))
model.add(Activation('relu'))
model.add(Dense(1, init='lecun_uniform'))
model.add(Activation('linear'))
rms = RMSprop(lr=0.01)
model.compile(loss='mse', optimizer=rms)

epochs = 200
model.fit(x_train, y_train, batch_size=1, nb_epoch=epochs, verbose=0)
y_predict = model.predict(x_test, batch_size=1)
print y_predict
```

Теперь опишем разворачивание Theano для GPU на Windows. Ведь обучение даже относительно простых сверточных сетей – крайне затратный вычислительный процесс, который может занять несколько часов на довольно мощном процессоре (здесь также стоит учитывать, что не все библиотеки и платформы поддерживают параллельное исполнение кода между ядрами процессоров, а значит, попросту не используют всю вычислительную мощь CPU).

Современные графические карты позволяют запускать очень много параллельных вычислений на аппаратном (а не только на программном) уровне. Лидером таких вычислений на сегодняшний день является компания NVidia, которая предлагает специальную платформу под названием CUDA. Это программно-аппаратная платформа, которая встроена в видеокарты и драйвера NVidia. Подобная

технология может ускорить вычисления во много раз (от 4 до 100). Конечно же, GPU вариант предпочтительнее не только в реальных проектах и промышленных системах, но и в процессе обучения, так как может существенно сэкономить время на эксперименты и выполнение учебных заданий.

Библиотека Theano может исполнять свои модели на графическом процессоре (GPU), который установлен в системе. Однако поддерживаются только графические процессоры от NVidia и только те, которые поддерживают режим CUDAComputeCapability 3.0 или выше. Узнать о том, какой режим поддерживает ваша видеокарта можно на официальном сайте NVidia.

Само разворачивание Theano состоит из следующих шагов:

1. Проверьте, что у вас есть свежий драйвер на видеокарту Nvidia. Необходима именно карта NvidiacCompute Capablity = 3.0 или выше (проверить Compute Capablity своей видеокарты можно тут [58]). Если карты от Nvidia нет, то этот сценарий реализовать не получиться и просто используйте CPU.
2. Скачайте и установите CUDA Toolkit 8.0 отсюда: [59].
3. Скачайте Cudnn 5.0 для CUDA Toolkit 8.0 отсюда [60]. Это библиотека для низкоуровневой поддержки нейронных сетей. Распакуйте архив Cudnn. Там будет несколько папок. В каждой папке по одному файлу. Эти файлы нужно переместить в место, куда был установлен CUDA Toolkit 8.0. Например, файл cudnn64_5.dll из папки cuda\bin положить в папку bin в место, куда установился CUDA Toolkit 8.0 и так далее.
4. Теперь код Theano может запускаться на GPU. Но его еще нужно скомпилировать. Для компиляции нужен VC++ компилятор от Майкрософта. Нужную версию компилятора можно получить, установив Visual Studio 2012.
5. Создайте текстовый файл с именем .theanorc, заполненный как показано в листинге 76. Описание приведено в листинге 77.

6. Созданный файл нужно положить в домашнюю или пользовательскую директорию ОС. Для Win это C:\Users\$UserName$. Чтобы четко определить директорию, можно выполнить следующий код (листинг 78).

Листинг 76

```
[global]
floatX=float32
device=gpu
mode=FAST_RUN
cnmem=0.7
 [cuda]
root=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5
 [nvcc]
flags=-LD:\Program_Files\Anaconda\libs
–compiler_bindir=C:\Program_Files_(x86)\Microsoft_Visual_Studio_11.0\VC\bin\
–fastmath=True
```

Листинг 77

```
[global]
Cnmem - объем памяти, который используется на видео карте
 [cuda]
Root - путь на корневую папку CUDAtoolkit. Например:
«root=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0»
 [nvcc]
Flags - указывает путь до библиотек Anaconda
compiler_bindir - путь до компилятора cl.exe из VisualStudio
Во всех параметрах [nvcc] не должно быть пробелов. Так «Program Files (x86)» нужно заменить на  «Program_Files_(x86)». Пример пути:
«D:\Program_Files\Visual_Studio_1\VC\bin\»
```

Листинг 78

```
os.environ['THEANO_FLAGS']="device=gpu0,lib.cnmem=0.5"
path=os.path.expanduser('~/.theanorc.txt')
```

7. Выполните проверочный код из листинга 79. Он должен вывести 'Used the gpu' и нулевой код ошибки.

Листинг 79

```
from theano import function, config, shared, sandbox
import theano.tensor as T
import numpy
import time
 vlen = 10 * 30 * 768  # 10 x #cores x # threads per core
iters = 10000
  rng = numpy.random.RandomState(22)
x = shared(numpy.asarray(rng.rand(vlen), config.floatX))
f = function([], T.exp(x))
print(f.maker.fgraph.toposort())
t0 = time.time()
for i in range(iters):
    r = f()
t1 = time.time()
print("Looping %d times took %f seconds" % (iters, t1 - t0))
print("Result is %s" % (r,))
if numpy.any([isinstance(x.op, T.Elemwise) for x in f.maker.fgraph.toposort()]):
    print('Used the cpu')
else:
    print('Used the gpu')
```

Получение данных средствами Keras

Подобно библиотеке Scikit-learn библиотека Keras также имеет встроенные функции для получения готового набора данных. Пример показан в листинге 80. На выходе у нас получаются стандартные массивы Numpy.

Листинг 80

```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Забежав немного вперед, скажем, что сверточные модели, представленные библиотекой Keras, принимают на вход немного непривычный формат данных для обучения. Так параметр Input_shape функции model.fit() имеет следующую форму: (samples, channels, rows, cols). Здесь samples – количество элементов в выборке (может

не указываться), channels – количество каналов (если речь идет об RGB-изображениях, то на каждую компоненту будет по матрице и этот параметр будет равен трем), rows и cols соответственно ширина и высота матрицы (изображения).

Исходя из этого, нам необходимо модифицировать полученные данные. Чтобы сменить форму массива Numpy, выданного load_data, воспользуйтесь функцией reshape, которая в качестве параметров принимает размер каждого измерения. Пример функции показан в листинге 81.

Листинг 81

```
X_train = X_train.Reshape(N, 1, r, c)
```

Кроме смены формы массива его элементы необходимо привести в диапазон от 0 до 1 для корректной работы с ним сверточной нейронной сетью. Сделайте это самостоятельно.

Над выборкой Y также необходимо провести преобразования. Целое число нужно перевести в бинарную маску (вектор) так, чтобы класс 2 из трех возможных классов представлял собой вектор [0,0,1] при отсчете от нуля. Подобное преобразование можно сделать при помощи функции np_utils.to_categorical.

Создание и обучение модели сверточной сети

Общая логика обучения модели такая же, как и в моделях библиотеки Scikit-Learn. Но поскольку Keras больше ориентирована на нейросетевые модели, то она позволяет более детально настраивать структуру и работу сети. В этом пункте мы рассмотрим лишь некоторые из возможностей.

Модели в Keras описываются как вычислительные графы. Поэтому, чтобы создать и обучить сверточную сеть, сначала необходимо создать последовательную модель (граф последовательных вычислений\обработчиков, «контейнер моделей»). Сделать это можно так, как показано в листинге 82.

Листинг 82

```
model = Sequential()
```

После создания граф необходимо заполнить рядом обработчиков. Описание всех возможных обработчиков есть в документации Keras, например, здесь: [61]. Ниже представлен список основных обработчиков, с которыми мы столкнемся в этом пункте.

Dense – одномерный слой обычных нейронов. По сути, этот обработчик вычисляет взвешенные суммы для каждого нейрона слоя, но не вычисляет для них функцию активации (так как она задается отдельным обработчиком). Функция, создающая обработчик, принимает на вход число нейронов. Использовать ее можно так, как показано в листинге 83.

Листинг 83

```
Dense(128)
```

ConvolutionND – сверточный слой нейронов. Данный обработчик подсчитывает взвешенные суммы для каждой карты и каждого нейрона в слое, но также не вычисляет для них функцию активации. Функция, создающая обработчик принимает на вход число карт в слое и размер ядра\фильтра, также определяется форма входа и режим обработки границ (valid означает, что все ядра поместятся в исходную матрицу и не будет выступающих границ, при этом карта будет меньше исходной матрицы; в случае same карта будет такого же размера, как и входная матрица). Можно задать и ряд других параметров, но с ними вам предлагается ознакомиться самостоятельно. Пример использования приведен в листинге 84.

Activation – обработчик функции активации. Задается после каждого слоя нейронов (Convolution или Dense), такие обработчики, как MaxPooling2D или Dropout, не являются слоями нейронов. Данный обработчик рассчитывает значение активации для взвешенных сумм предыдущего слоя. В качестве аргумента функция прини-

мает строку с названием собственно функции активации (relu, tanh, linear, sig, prelu). Пример использования – в листинге 85.

Листинг 84

```
Convolution2D(nb_filters, nb_conv, nb_conv,
border_mode='valid',
input_shape=(1, img_rows, img_cols)))
```

Листинг 85

```
Activation('relu')
```

MaxPoolingND – обработчик объединения по максимуму. Обработчик проходит по матрице, поступающей на вход, окном размера pool_size (например, 2X2) и подает на выходную матрицу одно максимальное значение из данной области входной матрицы. Иллюстрация работы функции – рисунок 69 [1].

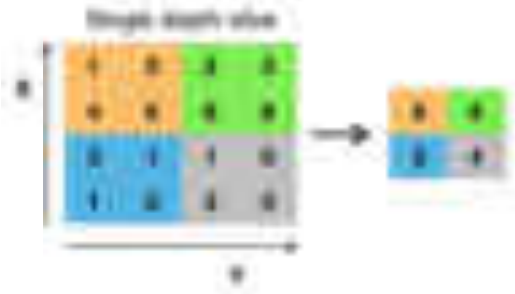


Рисунок 69. Иллюстрация действия обработчика MaxPooling2D

Как мы видим, из входной матрицы 4X4 формируется матрица 2X2 максимальных значений. Пример вызова функции приведен в листинге 86.

Листинг 86

```
MaxPooling2D(pool_size=(nb_pool, nb_pool))
```

Dropout – данный обработчик задает степень разрыва нейронных связей между соседними слоями. В качестве параметра функция принимает значение от 0 до 1, выражающее процент разрыва. В данном случае 25% связей между слоями выродятся в ноль и не будут меняться в процессе обучения.

Обратите внимание!

- *Такой возможности не было в библиотеке scikit-learn. Это важная настройка, так как она позволяет контролировать переобучение сети не только регуляризацией. Кроме того, таким способом можно добиться эффекта локальности.*

Вызов обработчика – в листинге 87.

Листинг 87

```
Dropout(0.25)
```

Flatten – выравнивающий обработчик. Преобразует связи от нейронов слоя свертки к одномерному слою обычных нейронов. Такой обработчик используется ближе к завершению сети, чтобы формировать выход модели. Пример вызова – в листинге 88.

Листинг 88

```
Flatten()
```

Если создать модель из двух сверточных слоев с функцией активации «relu», затем перейти к обычным слоям, добавить разрывы и завершите модель слоем из 10 нейронов с функцией активации «softmax», то данная функция сформирует выходной вектор таким образом, чтобы сумма всех элементов была равна единице, а отдельный элемент вектора выражал вероятность принадлежности входного образа к классу, который отображается этим элементом вектора.

Чтобы добавить тот или иной обработчик в вычислительный граф, используется функция add, как показано в листинге 89.

После формирования графа необходимо скомпилировать модель. Причем то же самое придется выполнить и после загрузки модели из файла, поскольку сохраняется лишь граф вычислений, а не готовый к исполнению объект. Пример компиляции модели приведен в листинге 90.

Листинг 89

```
model.add(<обработчик>)
```

Листинг 90

```
model.compile(loss='categorical_crossentropy',
optimizer='adadelta',
metrics=['accuracy'])
```

Обучается модель уже знакомым способом (с помощью функции fit, куда передаются тренировочные выборки X, Y). Однако в данных моделях имеет смысл рассмотреть еще несколько параметров:

- batch_size – определяет количество предъявляемых образов в рамках одного обновления весов (так если batch_size>1, то производиться обучение по группе, а не по одиночным образам). Данный параметр может повлиять на скорость обучения (особенно на CPU) и на объем требуемой памяти. Так небольшие группы по 50 образов могут потребовать меньше памяти и вычислительных ресурсов (слово «могут» используется потому, что есть и другие факторы, в том числе аппаратная реализация некоторых функций и поведение кэша).
- nb_epoch – определяет количество эпох обучения.
- verbose – определяет, выводить ли в консоль детали обучения или нет. Под деталями понимается информация о ходе обучения, времени и оценке качества за каждую итерацию.
- validation_data – определяет набор данных, по которому будет проводиться оценка качества модели.

Пример обучения модели приведен в листинге 91.

Листинг 91

```
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
verbose=1, validation_data=(X_test, Y_test))
```

Качество обученной модели проверяется кодом из листинга 92.

Листинг 92

```
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Напомним, что:

- Величина loss характеризует интегральную оценку всех среднеквадратических ошибок. Единицы этой величины совпадают с единицами Y. Чем меньше это значение, тем лучше. Однако для каждой задачи (каждого типа данных) конкретные величины будут разными, и оценивать эту величину нужно\можно, только зная характер данных и допустимые величины. Так, для оценки стоимости квартиры общая ошибка 300 000 на тренировочной выборке в несколько тысяч квартир может быть вполне приемлемой. А вот такая же величина loss для оценки уровня лейкоцитов в крови на тысячу пациентов просто неприемлема.
- Величина accuracy характеризует процент верных ответов из тестовой выборки. Чем ближе это значение к 100% (или 1.00), тем лучше.

Загрузка и сохранение сложных моделей

В одном из прошлых пунктов был рассмотрен вариант сохранения объектов посредством функций pickle.dump(obj, output) и pickle.load(f). Однако многие модели Keras имеют такую сложную структуру, что стандартный сериализатор объектов вызывает исключение из-за глубины ссылок. Поэтому для сохранения и загрузки сложных моделей нейронных сетей необходимо использовать следующий код (листинг 93).

Обратите внимание!

- *Настройки модели и веса сохраняются раздельно.*

Листинг 93

```
def SaveToJSon(model, fileName):
    model_json = model.to_json()
with open(fileName + ".net", "w") as json_file:
        json_file.write(model_json)
    model.save_weights(fileName + ".wgt")
print("model saved to disk")
def LoadFromJSon(fileName):
    json_file = open(fileName + ".net", 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    loaded_model.load_weights(fileName + ".wgt")
print("Loaded model from disk")
return loaded_model
```

Рекуррентные сети для прогнозирования временных рядов

Теперь рассмотрим пример работы с рекуррентными сетями средствами библиотеки Keras, а именно, их применение для решения задачи прогнозирования временных рядов. Пусть в файле «international-airline-passengers.csv» (скачать его можно отсюда: [41]) находятся данные о ежемесячном количестве пассажиров международного аэропорта в виде, представленном на рисунке 70. Применим рекуррентную сеть для прогнозирования количества пассажиро в будущем. Глубину прогноза возьмем равной одному месяцу, как и количество точек, по которому будем строить прогноз. Качество работы модели оценим по величине среднеквадратичной ошибки. Данная задача и код ее решения, показанный в листинге 94, взяты из [62].

Листинг 94

```
# -*- coding: utf-8 -*-
#Необходимые импорты
import numpy
import matplotlib.pyplot as plt
```

Продолжение листинга 94

```
from pandas import read_csv
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
# функция для создания выборки из исходного массива данных
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)
# фиксация состояния рандома для получения предсказуемого результата
numpy.random.seed(7)
# загрузка данных
dataframe = read_csv('international-airline-passengers.csv', usecols=[1], engine='python',
skipfooter=3)
dataset = dataframe.values
dataset = dataset.astype('float32')
# нормализация данных
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
# ручное разделение данных на обучающую и тестовую выборку
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
# Изменение исходного набора данных для того, чтобы в каждой строке X оказался
# временной ряд размером look_back, начиная от момента времени t0. A в Y
# временной ряд единичного размера от момента времени t0 + look_back.
# В данном примере, для look_back=4  X1 будет состоять из объектов:  Data[t0],
# Data[t1], Data[t2], Data[t4], а Y1 из объектов: Data[t5]. X2 начнется с элемента
# Data[t1] и т.д.
 trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
# изменение формы входных данных и приведение их к виду
# [количество объектов, размер временного ряда из объектов, количество признаков
каждого объекта]
```

Окончание листинга 94

```
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = numpy.reshape(testX, (testX.shape[0], testX.shape[1], 1))
# Создание графа вычислений (некоторой абстрактной модели)
model = Sequential()
#4 – это количество блоков LSTM, input_shape - это форма вектора входа
model.add(LSTM(4, input_shape=(look_back, 1)))
model.add(Dense(1))
#Компиляция модели
model.compile(loss='mean_squared_error', optimizer='adam')
#Обучение модели
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
# выполнение предсказания
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# так как прогнозное значение нормализовано, то необходимо его восстановление
# под восстановлением понимается приведение к исходному масштабу данных
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# вычисление ошибки прогноза
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
# подготовка данных для отрисовки
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] = testPredict
# построение графика
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Рисунок 70. Данные о пассажирах

На этом мы заканчиваем рассмотрение инструментальных средств для разработки приложений сферы машинного обучения на языке Python. Как вы могли заметить, в пособии не были рассмотрены реализации генетических алгоритмов и методов нечеткой логики. Это связано с тем, что их реализация на языке Pyhton не требует каких-либо специализированных библиотек, поэтому эту часть практического материала вам предлагается освоить самостоятельно в рамках выполнения задач по этим темам (см. раздел «Практические задания»). В случае затруднения вы можете обратиться к следующим ресурсам: реализация генетического алгоритма на Python – [63] и материалы учебного пособия [64], где рассматривается реализация генетических алгоритмов и некоторых операций нечеткой логики на языке Java.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ТЕСТОВЫЕ ЗАДАНИЯ

Тест «Общие сведения о машинном обучении»

1. Выберите верные утверждения
   a) Одна из задач машинного обучения – научиться делать прогнозы для признаков
   b) Объекты описываются с помощью признаков
   c) Одна из задач машинного обучения – научиться делать прогнозы для объектов
   d) Признаки описываются с помощью объектов
2. Какие из этих задач являются задачами классификации?
   a) Прогноз температуры на следующий день
   b) Разделение книг, хранящихся в электронной библиотеке, на научные и художественные
   c) Поиск групп похожих пользователей интернет-магазина
   d) Прогноз оценки студента по пятибалльной шкале на экзамене по машинному обучению в следующей сессии
3. Какие свойства данных препятствуют однозначному построению разделяющей поверхности?
   a) Ортогональность
   b) Мультиколлинеарность
   c) Противоречивость
   d) Категориальность
4. Какая способность людей и систем позволяет получать им новые знания по наблюдению отдельных прецедентов (примеров)?
   a) Корректировать ошибку
   b) Обобщать
   c) Запоминать
   d) Распознавать образы
5. Какая задача лучше всего подходит под следующее описание. Нахождение такой функции F, которая бы наилучшим образом отображала неизвестные ранее объекты X в конечное множество

целочисленных номеров (имен, меток), на основании обучающих пар (X, Y)?

a) Прогнозирование денежных затрат
b) Кластеризация клиентов
c) Классификация образов
d) Выявление особенностей в данных

6. Почему для обучения моделей используются такие методы, как Градиентный спуск?
   a) Потому что метод позволяет корректировать параметры модели постепенно
   b) Потому что аналитические решения не всегда дают корректное решение
   c) Потому что такой подход позволяет получать более точные решения (Глобальный экстремум в отличие от локального)
   d) Потому что при большой размерности входных данных подобные методы работают быстрее
7. Выберите верные утверждения
   a) Метод Байеса – это во многом классический подход к классификации, основанный на оценке частоты встреч объектов со схожими признаками
   b) Благодаря универсальности статистического подхода метод Байеса позволяет решать любые задачи без априорной информации
   c) Данный метод позволяет очень хорошо обобщать высокоуровневые признаки
   d) Закон, задающий распределение вероятностей, который используется в предсказательной модели, сильно влияет на способ обобщения

Проблема переобучения

1. Какие факторы влияют на переобучение?
2. Какие есть способы оценки переобучения?
3. Какие есть способы борьбы с переобучением?

Регрессия

1. Почему такая простая формула, как y=kx+b, позволяет делать прогнозы или классификацию?
2. В чем отличие линейной и логистической регрессий?
3. В чем отличие линейной от нелинейной регрессии?
4. В чем отличие линейной регрессии от полиномиальной?
5. Что позволяет делать LASSO?
6. В чем заключаются особенности Ridge регрессии?

Модели и методы нечеткой логики

1. Охарактеризуйте следующие понятия: нечеткие множества, операции нечеткой логики, нечеткие модели или нечеткие системы.
2. Сформулируйте понятие лингвистической неопределенности.
3. Напишите функцию лингвистической неопределенности некоторого объекта.
4. Дайте определение функции принадлежности.
5. К какому типу относятся нечеткие множества, если значения функции принадлежности нечеткого множества представлены точными числовыми значениями?
6. К какому типу относятся нечеткие множества, если значения функции принадлежности нечеткого множества моделируются другими нечеткими множествами?
7. Какие существуют способы задания функции принадлежности?
8. Приведите три примера функции принадлежности, задаваемых функциональным способом.
9. Напишите формулу компактной записи функции принадлежности.
10. Сформулируйте определение лингвистической переменной.
11. Опишите набор переменных, с помощью которого описывается лингвистической переменная.
12. Дайте определения следующих понятий: X – универсальное множество объектов, $\widetilde{X}$ – базовое терм-множество, G – синтакси-

ческие правила вывода (порождения) новых термов, Р – семантические правила.

13. Сформулируйте определение нечеткой логики.
14. Какие необходимые характеристики нечеткой логики ввел Л. Заде?
15. Что лежит в основе операций нечеткой логики?
16. Какие операции используются для моделирования основных логических связок И, ИЛИ над нечеткими множествами в нечеткой логике?
17. Сформулируйте определения триангулярной нормы и триангулярной конормы (t-норма и s-конорма) и докажите их двойственность. Приведите примеры.
18. Запишите композиционное правило Л. Заде.
19. Сформулируйте содержательное определение операции импликации.
20. Сформулируйте определение системы нечеткого логического вывода.
21. Какие объекты входят в систему нечеткого логического вывода?
22. Какие условия должны соблюдаться при построении правил на основе системы нечеткого логического вывода?
23. Перечислите пять способов реализации нечеткого логического вывода.
24. Подробно опишите реализацию нечеткого логического вывода с помощью алгоритма Мамдани. Нарисуйте схему процесса нечеткого вывода этого алгоритма.

Нечеткие временные ряды

1. Сформулируйте определение уровня временного ряда.
2. Сформулируйте определение нечеткого временного ряда.
3. Сформулируйте определение носителя нечеткой метки $\widetilde{x}_i$.
4. Приведите примеры прикладных задач обработки нечетких ВР.
5. Дайте определение нечеткому разбиению.

6. Опишите основные этапы алгоритма моделирования нечетких ВР в соответствии с нечеткой моделью Сонга.
7. В чем преимущество использования моделей нечетких ВР?
8. Что понимается под интеллектуальным анализом данных или Data Mining?
9. Какие задачи решаются на основе Data Mining?
10. Приведите виды темпорально-логических концептов ВР.
11. В чем заключается метод аппроксимации временного ряда на основе F-преобразования?

Нечеткая регрессия

1. Какие существуют подходы к построению моделей нечеткой линейной регрессии?
2. Какие существуют критерии для определения нечетких коэффициентов модели?
3. Какие вы знаете варианты методов на основе классификации «вход – выход»?

Генетические алгоритмы

1. Поясните происхождение термина «генетические алгоритмы».
2. Опишите сферу применения генетических алгоритмов.
3. Дайте определение гену в контексте генетических алгоритмов.
4. Дайте определение хромосоме в контексте генетических алгоритмов.
5. Дайте определение популяции в контексте генетических алгоритмов.
6. Дайте определение степени приспособленности в контексте генетических алгоритмов.
7. Дайте определение кроссовера в контексте генетических алгоритмов.
8. Дайте определение мутации в контексте генетических алгоритмов.
9. Перечислите методы отбора хромосом для кроссовера.
10. Расскажите о типовой схеме генетического алгоритма.

Нечеткая кластеризация

1. Дайте определение классификации.
2. Дайте определение кластеризации.
3. Поясните разницу между классификацией и кластеризацией.
4. Определите область применение кластеризации.
5. Определите цель алгоритма FCM-кластеризации.
6. Определите перечень входных данных для алгоритма FCM-кластеризации.
7. Перечислите шаги алгоритма FCM-кластеризации.
8. Почему кластеризация, проводимая с помощью алгоритма FCM, является нечеткой?
9. Поясните назначение параметра «степень нечеткости» в алгоритме FCM.
10. Что является выходными данными алгоритма FCM-кластеризации?

Искусственные нейронные сети и глубинное обучение

1. Какие достоинства и недостатки есть у ИНС по сравнению с Регрессией и Решающими Деревьями?
2. Сеть какого типа лучше использовать для прогнозирования?
3. Сеть какого типа можно использовать в условиях постоянного изменения данных, когда точной выборки еще не существует?

Тест «Искусственные нейронные сети»

1. Выберите верные утверждения:
   a) ИНС проще подобрать под любую нелинейную задачу. Все, что нужно сделать, это увеличивать число слоев пропорционально числу признаков
   b) ИНС позволяют обрабатывать более высокоуровневые признаки за счет нелинейной функции активации и последовательным слоям

c) По сравнению с Регрессией ИНС практически не подвержены Переобучению при любом количестве нейронов
d) С точки зрения математического аппарата ИНС – это комбинация полиномиальной регрессии высокого порядка и формулы Байеса
e) ИНС может аппроксимировать любую нелинейную непрерывную функцию, но это еще не гарантирует 100% сходимости на произвольных данных
f) ИНС в отличие от регрессии может хорошо обрабатывать высокую степень мультиколлинеартности и противоречивости в данных

2. Сеть какого типа лучше использовать для прогнозирования временных рядов?

a) Сверточную
b) ART MAP
c) Импульсную
d) MLP
e) Рекуррентную
f) Когнитрон

3. Сеть какого типа лучше использовать для обработки трехмерных сцен?

a) MLP
b) Рекуррентную
c) ART MAP
d) Сверточную
e) Когнитрон
f) Импульсную

4. Сеть какого типа лучше использовать для решения задачи классификации клиентов по одиночному вектору клиентских характеристик (с учетом того, что этот вектор содержит большое количество категориальных признаков)?

a) Автокодировщик
b) MLP

c) Когнитрон
d) ART MAP
e) Сверточную
f) Рекуррентную
g) Импульсную

5. Сеть какого типа можно использовать в условиях постоянного изменения данных, когда точной выборки еще не существует и сеть приходится постоянно дообучивать на новых классах?

a) MLP
b) Сверточную
c) Когнитрон
d) Рекуррентную
e) ART MAP
f) Автокодировщик
g) Импульсную

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Работа с файлом данных «Титаника»

Приведенные ниже задания основаны на данных 'titanic.csv', где содержатся сведения о пассажирах «Титаника».

Задачи:

1. Какое количество мужчин и женщин плыло на корабле?
2. Какой части пассажиров удалось выжить? Посчитайте долю выживших пассажиров. Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.
3. Какую долю пассажиры первого класса составляли среди всех пассажиров? Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.
4. Какого возраста были пассажиры? Посчитайте среднее и медиану возраста пассажиров. В качестве ответа приведите два числа через пробел.
5. Коррелируют ли число братьев/сестер/супругов с числом родителей/детей? Посчитайте корреляцию Пирсона между признаками SibSp и Parch.
6. Какое самое популярное женское имя на корабле? Извлеките из полного имени пассажира (колонка Name) его личное имя (First Name). Это задание – типичный пример того, с чем сталкивается специалист по анализу данных. Данные очень разнородные и шумные, но из них требуется извлечь необходимую информацию. Попробуйте вручную разобрать несколько значений столбца Name и выработать правило для извлечения имен, а также разделения их на женские и мужские.

Работа по отбору признаков

Доработайте код отбора признаков, использовав все модели из списка:

1. Линейная регрессия (LinearRegression)
2. Гребневая регрессия (Ridge)
3. Лассо (Lasso)
4. Случайное Лассо (RandomizedLasso)
5. Рекурсивное сокращение признаков (Recursive Feature Elimination – RFE)
6. Сокращение признаков Случайными деревьями (Random Forest Regressor)
7. Линейная корреляция (f_regression)

Отобразите получившиеся значения\оценки каждого признака каждым методом\моделью и среднюю оценку. Проведите анализ получившихся результатов. Какие 4 признака оказались самыми важными по среднему значению? (Названия\индексы признаков и будут ответом на задание).Какие выводы можно сделать по результатам отдельных методов?

Многослойный персептрон

1. Доработайте код задачи классификации (листинг 58). Попробуйте настроить различные параметры сети, такие как количество нейронов в первом или втором слое, вид функции активации (activation), алгоритм градиентного спуска (solver), количество максимальных итераций при обучении (max_iter), порог точности при обучении (tol). Варьируя эти параметры, попробуйте найти наилучший вариант (возможные значения переменных можно найти в документации Scikit-learn).

2. Насколько лучше или хуже работает MLPClassifier по сравнению с персептроном? Без нормализации данных и с нормализацией данных? Предположите, почему именно так ведут себя модели.

3. Проведите серию экспериментов с различными значениями random_state. Как случайная инициализация весов влияет на Персептрон и многослойную сеть?

4. Дополнительно постройте график распределения исходных данных.

Реализация алгоритма обратного распространения ошибки

1. Модифицируйте сеть из листинга 59. Добавьте еще один слой и восстановите работоспособность алгоритма. Критерием его работоспособности может быть стремительно уменьшающаяся ошибка.

2. Дополнительно обучите старую и новую модели на следующих данных:

```
X = [[0, 0, 1], [0.3, 1, 0],
    [1, 0.3, 0],  [0.6, 0.2, 1],
    [0.6, 0.2, 1]]
```

Проанализируйте исходные данные. Какая зависимость скрыта в них? Объясните, что произошло с качеством классификации. Почему? С чем может быть связано подобное в реальных задачах и как на это можно повлиять (если это вообще возможно)?

Регуляризация и сеть прямого распространения

1. Вернитесь к задаче из пункта «Влияние регуляризации на многослойную сеть прямого распространения». Проанализируйте результат кода при количестве нейронов, равном 100, в скрытом слое MLP. Напомним, что регуляризация – это механизм снижения больших весов модели при повышении ее сложности. Этот механизм помогает избежать переобучения. Параметр регуляризации alpha регулирует размер штрафов, которые накладываются на веса. Какой коэффициент регуляризации оптимален для данной задачи?

2. Задача сравнения персептрона, многослойной сети и регрессий. В пункте «Влияние регуляризации на многослойную сеть прямого распространения» была описана общая методика по проведению

серии экспериментов и выводу результатов (понятно, что это можно сделать не только в виде графиков, но и выводом информации в консоль или записи более детальной информации в файл\таблицу для дальнейшей обработки). Соответственно каркас этой программы можно повторно использовать во многих других экспериментах не только по анализу влияния регуляризации.

По аналогии с заданием выше сгенерируйте 3 задачи классификации со следующими параметрами: make_moons (noise=0.3, random_state=rs), make_circles (noise=0.2, factor=0.5, random_state=rs), X, y = make_classification (n_samples=500, n_features=2, n_redundant=0, n_informative=2, random_state=rs, n_clusters_per_class=1) и сравните следующие модели:

- Линейную регрессию
- Полиномиальную регрессию (например, со степенью 3)
- Гребневую полиномиальную регрессию (например, со степенью 4, alpha = 1.0)
- Персептрон
- Многослойный персептрон с десятью нейронами в скрытом слое (alpha = 0.01)
- Многослойный персептрон со ста нейронами в скрытом слое (alpha = 0.01)

Создайте отдельный файл и замените только код моделей. Постройте графики и отобразите качество моделей. В результате у вас должно получиться полотно графиков, похожее на рисунок 71 (причем в данном случае представлена только первая задача классификации на Moondataset).

Проанализируйте полученные результаты, рассчитав их при параметре random_state=25 для следующих функций\объектов:

- Perceptron
- MLPClassifier
- Ridge
- make_classification

- rng = np.random.RandomState(25)
- make_moons
- make_circles
- train_test_split

Рисунок 71. Пример работы программы по сравнению нескольких моделей на задачах moon, circle, linear

Сравнение эффективности моделей из библиотеки Keras

Вы научитесь:

- Проводить самостоятельный анализ результатов экспериментов;
- Проводить серию экспериментов;
- Интерпретировать результаты моделей;
- Подбирать соответствующие данным инструменты и параметры.

Необходимые import для выполнения работы представлены в листинге 95.

Листинг 95

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils
import matplotlib.pyplot as plt
from keras.models import model_from_json
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
```

Инструкции по выполнению представлены в листинге 96 (первые 3 шага), а также ниже. Прежде чем перейти к шагу 3, сравните качество полученных трех моделей на данных MNIST. Объясните полученные результаты. Есть ли превосходство сверточной сети перед обычной или перед полиномиальной регрессией?

Листинг 96

```
#Шаг 1. Загрузите данные MNIST
(X_train, y_train), (X_test, y_test) = mnist.load_data()
#Шаг 2. Создайте модель многослойного персептрона (на основе Keras)и сверточной
#сети (на основе Keras) и полиномиальной регрессии из Scikit-Learn.
#Для задания обычного линейного слоя в качестве первого слоя сети потребуется
#указать форму входного вектора (размер). Это можно сделать следующим образом:
Dense(128, input_dim=input_size)
#Шаг 3. Далее попробуйте эти три типа модели уже на других данных.
#Для этого загрузите и подготовьте данные с фотографиями людей с помощью
#библиотеки scikit-learn (изучите код самостоятельно):
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
# introspect the images arrays to find the shapes (for plotting)
n_samples, h, w = lfw_people.images.shape
X = lfw_people.images
# the label to predict is the id of the person
y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
print("n_classes: %d" % n_classes)
# split into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

Шаг 4. Отобразите данные, чтобы удостовериться в их корректной подготовке для дальнейшего обучения.

Шаг 5. Обучите модель многослойного персептрона, полиномиальной регрессии и сверточной сети. Попробуйте разное число нейронов и слоев в сетях (10, 20, 50, 100, 200, 500 нейронов) и (1-3 слоя). Большее число нейронов и слоев не нужно.

Замечание 1: Конечно же, не следует обучать модели сперва на MNIST, а потом эти же модели (конкретные экземпляры) на фотографиях. Это совсем разные задачи, и при малых количествах слоев и таких выборках вряд ли произойдет хорошее обобщение. Имеется в виду, что те же три типа моделей теперь надо сравнить на фотографиях.

Замечание 2: Если на CPU обучение идет слишком долго, то возьмите не всю обучающую выборку, а лишь половину или одну треть.

В заключение сравните качество обученных моделей на данных фотографий. Объясните полученные результаты и ответьте а вопросы:

- Почему получилось именно так?
- На что способны и не способны используемые модели?
- Чем отличаются данные из первой части задания от данных из второй части задания?

Работа с библиотекой OpenCV

Докажите гипотезу, сформулированную по завершению предыдущего задания. Что нужно сделать с данными MNIST, чтобы работающая ранее модель стала давать на них плохие результаты? Для этого вам может понадобиться библиотека OpenCV для Python. Которая достаточно легко устанавливается. Смотрите раздел «**Installing OpenCV from prebuilt binaries»** по следующей ссылке: [65].

Согласно Википедии [1]: «OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно

использоваться в академических и коммерческих целях – распространяется в условиях лицензии BSD.

Проект OpenCV возник в 1999 году в рамках исследования Intel по высокопроизводительным приложениям по обработке 3D сцен. Это крайне мощная библиотека и стандарт де-факто для обработки изображений. Сферы применения OpenCV:

Инструмент выделения 2D и 3D признаков;

Трехмерная навигация;

Системы распознавания лиц;

Системы распознавания жестов;

Человеко-компьютерные интерфейсы;

Мобильные роботы;

Распознавание эмоций;

Идентификация объектов;

Сегментация и распознавание;

Стереозрение;

Восстановление поверхности;

Отслеживание движения;

Дополненная реальность.

Кроме того, для поддержки некоторых вышеуказанных областей, OpenCV включает в себя библиотеки статистической обработки данных и машинного обучения:

Решающие деревья;

Алгоритм максимального правдоподобия;

Алгоритм k-ближайших соседей;

Наивный Байесовский классификатор;

Искусственные нейронные сети;

Случайные леса;

Метод опорных векторов.

В библиотеке OpenCV есть методы для многих видов деформаций изображений. Для базовой проверки способностей ИНС на двумерных изображениях вполне хватит следующих:

Повороты;

Линейные сдвиги;

Масштабирование».

Ниже (в листинге 97) приведен код, который позволяет выполнить преобразование исходного изображения images[i]. Каждый метод преобразования cv2 возвращает новое изображение. Для поворотных и линейных сдвигов используются специальные матрицы деформации. Код для их инициализации также приведен ниже. Недостающие переменные – это коэффициенты и константы. Определите их самостоятельно.

Листинг 97

```
import cv2
# setrotation
rot_Matrix = cv2.getRotationMatrix2D((cols / 2, rows/ 2), ang, 1)
cv2.warpAffine(images[i], rot_Matrix, (cols, rows))
# set scaling
cv2.resize(images[i], dsize=(rows, cols), fx=scale, fy=scale,
interpolation=cv2.INTER_CUBIC)
# set translation
trans_Matrix = np.float32([[1, 0, tx], [0, 1, ty]])
cv2.warpAffine(images[i], trans_Matrix, (cols, rows))
```

После преобразования части изображений проведите соответствующие эксперименты с разными моделями. Измерьте качество получившихся моделей. Вероятно, придется проделать много экспериментов с различными параметрами, чтобы добиться такой ситуации, когда результаты будут объяснять теорию.

Нечеткая логика

1. На языке Python разработайте скрипт, позволяющий задать нечеткое множество с треугольной функцией принадлежности и отобразить его название, параметры, а также степень принадлежности вводимого пользователем объекта.

2. На языке Python разработайте скрипт, позволяющий задать нечеткое множество с трапециевидной функцией принадлежности и отобразить его параметры, а также степень принадлежности вводимого пользователем объекта.

3. На языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

4. На языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

5. На языке Python разработайте скрипт, позволяющий выполнить операцию объединения заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – объединение данных нечетких множеств.

6. На языке Python разработайте скрипт, позволяющий выполнить операцию объединения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – объединение данных нечетких множеств.

7. На языке Python разработайте скрипт, позволяющий выполнить операцию дополнения, заданного пользователем нечеткого множества с треугольной функцией принадлежности. Входными данными будут параметры функции принадлежности и четкие объекты множества. Выходными – дополнение нечеткого множества.

8. На языке Python разработайте скрипт, позволяющий выполнить операцию дополнения, заданного пользователем нечеткого множества с трапециевидной функцией принадлежности. Входными данными будут параметры функции принадлежности и четкие объекты множества. Выходными – дополнение нечеткого множества.

9. На языке Python разработайте скрипт, позволяющий выполнить операцию импликации заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – результат импликации данных нечетких множеств. Причем результат вывести через лингвистические переменные. Импликацию моделировать минимумом.

10. На языке Python разработайте скрипт, позволяющий выполнить операцию импликации заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – результат импликации данных нечетких множеств. Импликацию моделировать минимумом.

Генетические алгоритмы

1. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N наименований продуктов, для каждого из которых известно m характеристик. Необходимо получить самый дешевый рацион из k наименований, удовлетворяющий заданным медицинским нормам для каждой из m характеристик.

2. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано n пунктов производства продуктов и k городов, которые в них нуждаются. Каждый город может потребить x продуктов, а каждый пункт произвести y продуктов. Необходимо получить оптимальный маршрут, так, чтобы все города получили нужный им объем продуктов без сильного его превышения, а транспортные расходы были минимальными.

3. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N наименований продуктов, для каждого из которых известно m характеристик. Необходимо получить самый лучший по характеристикам рацион из k наименований, удовлетворяющий заданным ценовым рамкам. Лучшим считается рацион с минимальным отклонением от нормы.

4. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано n пунктов производства продуктов и k городов, которые в них нуждаются. Каждый город может потребить x продуктов, а каждый пункт произвести y продуктов. Необходимо получить оптимальный маршрут, так, чтобы все города получили нужный им объем продуктов с минимальным его превышением, а транспортные расходы укладывались в определенные рамки.

5. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N полей и k культур для посева. Для каждого поля известна характеристика урожайности каждой из k культур, а для каждой культуры – его закупочная стоимость. Необходимо получить самый лучший урожай за наименьшую стоимость.

Нечеткая кластеризация объектов

1. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере заработной платы n людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

2. На языке Python разработайте скрипт, кластеризующий загруженные данные о возрасте n людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

3. На языке Python разработайте скрипт, кластеризующий загруженные данные о стоимости n автомобилей на определенные им

кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

4. На языке Python разработайте скрипт, кластеризующий загруженные данные о росте *n* людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

5. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере затрат на производство *n* продуктов на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

6. На языке Python разработайте скрипт, кластеризующий загруженные данные о длительности перелетов до *n* пунктов на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

7. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере затрат на связь среди *n* отделов на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

8. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере площадей *n* квартир на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

9. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере урожая *n* сельскохозяйственных культур на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

10. На языке Python разработайте скрипт, кластеризующий загру-женные данные о весе *n* людей на определенные им кластеры,

обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

Анализ временных рядов

1. *Лингвистические шкалы.* Разработать скрипт на Python, позволяющий задавать с использованием функций принадлежности лингвистическую шкалу для решения задачи по варианту (таблица 14). Количество оценок в шкале и параметры функций принадлежности должны задаваться по умолчанию, если в заданном пользователем файле не указано иное. Осуществить графическое отображение функций принадлежности меток шкалы, выделив каждую отдельным цветом. Предусмотреть загрузку файла с данными и оценку их по построенной шкале.

Таблица 14. Варианты для задачи «Лингвистические шкалы»

Вариант	**Функция принадлежности**	**Назначение шкалы**
1.	Треугольная	Оценка эффективности продаж
2.	Трапециевидная	Оценка эффективности операций с валютой
3.	Треугольная	Оценка прибыли
4.	Треугольная	Оценка затрат на оплату труда
5.	Трапециевидная	Оценка рентабельности
6.	Треугольная	Оценка объемов продаж
7.	Трапециевидная	Оценка объемов закупок
8.	Треугольная	Оценка загруженности сервера и сети
9.	Треугольная	Оценка уровня зарплаты
10.	Трапециевидная	Оценка убытков
11.	Треугольная	Оценка эффективности продаж
12.	Трапециевидная	Оценка эффективности операций с валютой
13.	Треугольная	Оценка прибыли

Окончание таблицы 14

Вариант	Функция принадлежности	Назначение шкалы
14.	Треугольная	Оценка затрат на оплату труда
15.	Трапециевидная	Оценка рентабельности
16.	Треугольная	Оценка объемов продаж
17.	Трапециевидная	Оценка объемов закупок
18.	Треугольная	Оценка загруженности сервера и сети
19.	Треугольная	Оценка уровня зарплаты
20.	Трапециевидная	Оценка убытков
21.	Треугольная	Оценка эффективности продаж
22.	Трапециевидная	Оценка эффективности операций с валютой
23.	Треугольная	Оценка прибыли
24.	Треугольная	Оценка затрат на оплату труда
25.	Трапециевидная	Оценка рентабельности
26.	Треугольная	Оценка объемов продаж
27.	Трапециевидная	Оценка объемов закупок
28.	Треугольная	Оценка загруженности сервера и сети
29.	Треугольная	Оценка уровня зарплаты
30.	Трапециевидная	Оценка убытков

2. *Нечеткие временные ряды и ряды нечетких тенденций*. Реализовать скрипт на языке Python, позволяющий загружать четкий временной ряд величины из задачи по лингвистическим шкалам (задача «Лингвистические шкалы»). С помощью построенной в предыдущей работе шкалы (задача «Лингвистические шкалы») построить из четкого временного ряда временной ряд нечетких значений и нечетких тенденций. Осуществить графическое отображение нечетких меток шкалы и нечетких тенденций, выделив каждую отдельным цветом.

3. *Кластеризация и лингвистические шкалы.* Язык реализации Python. Построить шкалу из задачи по лингвистическим шкалам (работа «Лингвистические шкалы») с помощью алгоритма fcm-клас-

теризации. Количество кластеров задается по умолчанию, но может редактироваться пользователем посредствам загрузки из файла. Каждый кластер – метка на шкале с присвоенным ему лингвистическим значением либо по умолчанию, либо пользователем. Осуществить графическое отображение меток шкалы, выделив каждую отдельным цветом.

3. *Прогнозирование значения временного ряда с помощью нейронной сети.* Реализовать прогноз следующего значения четкого временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» с помощью нейронной сети, но НЕ рекуррентного типа. Графически отобразить реальный ряд и прогнозное значение, а также проиллюстрировать результат тестового прогноза.

4. *Прогнозирование значения временного ряда с помощью модифицированного метода Сонга.* Реализовать прогноз следующего значения временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» с помощью метода, описанного в пункте «Пример моделирования временного ряда в нечетком подходе». Графически отобразить реальный ряд и прогнозное значение, а также проиллюстрировать результат тестового прогноза.

5. *F-преобразование.* Для временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» выделить тренд методом F-преобразования.

Работа с рекуррентными сетями

1. Модифицируйте код из листинга 91: поэкспериментируйте с разной структурой сети, разным количеством точек, по которым строится прогноз, и с разным количеством LSTM-блоков. Оцените качество работы сети в разных экспериментах. Интерпретируйте результаты.

2. Решите задачу «Прогнозирование значения временного ряда с помощью нейронной сети» из блока заданий «Анализ временных рядов» с помощью рекуррентной сети. Сравните и интерпретируйте результаты.

3. Объясните работу кода из листинга 98.

Листинг 98

```
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.utils import np_utils

numpy.random.seed(7)
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))
seq_length = 1
dataX = []
dataY = []
for i in range(0, len(alphabet) - seq_length, 1):
  seq_in = alphabet[i:i + seq_length]
  seq_out = alphabet[i + seq_length]
  dataX.append([char_to_int[char] for char in seq_in])
  dataY.append(char_to_int[seq_out])
  print seq_in, '->', seq_out
X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
X = X / float(len(alphabet))
y = np_utils.to_categorical(dataY)

model = Sequential()
model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, y, nb_epoch=500, batch_size=1, verbose=2)
scores = model.evaluate(X, y, verbose=0)
print("Model Accuracy: %.2f%%" % (scores[1]*100))
for pattern in dataX:
  x = numpy.reshape(pattern, (1, len(pattern), 1))
 x = x / float(len(alphabet))
  prediction = model.predict(x, verbose=0)
  index = numpy.argmax(prediction)
  result = int_to_char[index]
  seq_in = [int_to_char[value] for value in pattern]
  print seq_in, "->", result
```

4. Проведите два эксперимента, модифицируя код из листинга 92 следующим образом:

1. Изменив значение seq_length, сделав его равным 3 и изменив форму входных данных следующим образом:

X = numpy.reshape(dataX, (len(dataX), 1, seq_length))

2. Изменив значение seq_length, сделав его равным 3 и изменив форму входных данных следующим образом:

X = numpy.reshape(dataX, (len(dataX), 1, seq_length))

Интерпретируйте полученные результаты.

Заключение

Мы рассмотрели модели и алгоритмы, используемые в сфере машинного обучения для анализа данных, а также способы их реализации на языке Python с использованием библиотек Keras, Theano и других.

В настоящее время машинное обучение и область анализа данных являются весьма перспективными направлениями в ИТ. Задачей данной книги было познакомить вас с азами этого направления, рассмотрев наиболее важные модели, алгоритмы и их реализацию. Однако, чтобы стать специалистом в сфере анализа данных, недостаточно просто знать модели и инструменты. Необходимо понимать границы их применений и преимущества одного перед другим.

Данное учебное пособие лишь приоткрывает дверцу в мир машинного обучения, освещая небольшую его часть. Если же после работы с книгой вы поняли, что анализ данных – крайне интересная вещь, то вам непременно нужно двигаться дальше. Причем не только в изучении моделей, методов, алгоритмов и инструментов их реализации (что вы можете сделать, изучив приведенные в библиографическом списке источники), но и в совершенствовании своего мышления. Ведь пока искусственный интеллект еще не изобретен, всю семантическую нагрузку задач машинного обучения берет на себя разработчик-человек. Поэтому он непременно должен обладать острым аналитическим умом.

Перефразируя Шерлока Холмса, напомним, что «развитый мозг – это то, что всегда будет в цене». В связи с этим, закончим пособие поже-ланием: оттачивайте свой разум, покоряйте новые вершины, не останавливайтесь на достигнутом!

ГЛОССАРИЙ

DataMining – интеллектуальный анализ данных.

TimeSeries DataMining – интеллектуальный анализ временных рядов.

Аномалия временного ряда – новый, не типичный паттерн временного ряда.

Апостериорная вероятность – назначенная событию вероятность при условии наличия знаний, поддерживающих его наступление и полученных опытным путем.

Априорная вероятность – назначенная событию вероятность, при условии отсутствия знаний, поддерживающих его наступление.

Временной ряд (ВР) – последовательность упорядоченных в равноотстоящие моменты времени пар (момент_времени, значение_характеристики).

Генетические алгоритмы – адаптивные методы поиска, использующиеся для решения задач функциональной оптимизации.

Градиент – вектор, указывающий направление наибольшего возрастания некоторой величины, значение которой меняется в скалярном поле.

Дефаззификация – получение оптимального четкого значения по агрегированному нечеткому понятию.

Дисперсия – мера разброса элементов выборки относительно ее математического ожидания.

Задача классификации – распределение некоторого множества объектов по заданному множеству групп (классов).

Задача кластеризации – разделение некоторого множества объектов на непересекающиеся группы (кластеры) таким образом, чтобы каждая группа состояла из схожих объектов, а объекты разных кластеров существенно отличались.

Задача регрессии – приближение неизвестной целевой зависимости на некотором множестве данных.

Задача таксономии – задача построения древообразной иерархической структуры, упорядочивающей исходные данные.

Закон распределения – функция, определяющая для выборки вероятность попадания в некоторый интервал или вероятность получения определенного значения.

Знание – совокупность утверждений о закономерностях и свойствах процессов и явлений, а также связывающих их правил логического вывода и правил использования их при принятии решений.

Индексирование объектов временного ряда – построение индексов для эффективного выполнения запросов к базам данных ВР.

Классификация объектов временного ряда – назначение ВР или их паттернам одного из заранее определенных классов.

Кластеризация объектов временного ряда – поиск группировок ВР или их паттернов.

Комбинированная модель прогнозирования – модель прогнозирования, состоящая из нескольких индивидуальных (частных) моделей, называемых базовым набором моделей.

Концепт временной продолжительности – присутствие определенного паттерна или признака ВР на определенном интервале времени.

Концепт нечеткости ВР – нечеткость выраженности темпоральных событий и отношений.

Концепт одновременности ВР – совпадение во времени темпоральных событий (паттернов различных ВР).

Концепт очередности ВР – порядок следования паттернов ВР во времени.

Кроссовер – операция в генетическом алгоритме, позволяющая хромосомам обмениваться между собой своими частями.

Лингвистическая переменная – это переменная, значениями которой являются слова или высказывания естественного или искусственного языка.

Математическое ожидание – среднее значение вероятностных элементов выборки.

Медиана – число, характеризующее выборку по среднему из ее значений.

Метод градиентного спуска – нахождение локального экстремума (минимума или максимума) функции путем движения вдоль градиента в направлении наискорейшего спуска, задаваемого антиградиентом.

Мутация – операция в генетическом алгоритме, произвольным образом изменяющая хромосому.

Недообучение – ситуация, когда алгоритм при обучении с учителем не дает удовлетворительно малой средней ошибки на обучающем множестве.

Нечеткая логика – логика, оперирующая нечеткими высказываниями и рассуждениями на базе частичной истинности.

Нечеткий временной ряд (НВР) – упорядоченная в равноотстоящие моменты времени последовательность наблюдений над некоторым процессом, состояния которого изменяются во времени, если значение состояния процесса в каждый момент времени может быть выражено с помощью нечеткой метки .

Нечеткое множество – совокупность объектов, принадлежащих ему с определенной степенью.

Нормализация данных – процесс масштабирования вектора каждого признака к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

Обобщающая способность – свойство модели отражать исходные данные в требуемые результаты (X –> Y) на всем множестве исходных данных (во всех сценариях, а не только на тренировочных примерах).

Обучение – способность алгоритма при решении задач некоторого класса выдавать на некотором опыте лучшие результаты в смысле заданной меры качества при предъявлении нового опыта.

Ошибка – численно выраженная разница между ответом модели и требуемым (реальным) значением.

Переобучение – свойство натренированного алгоритма на объектах тренировочной выборки давать существенно меньшую вероятность ошибки, чем на объектах тестовой.

Пространство признаков – это N-мерное пространство, где N – число измеряемых характеристик объектов, выделенное для конкретной задачи.

Регуляризация – добавление некоторой дополнительной информации к условию минимизации ошибки.

Резюмирование временного ряда – формирование краткого описания ВР, содержащего существенные черты с точки зрения решаемой задачи.

Сегментация временного ряда – разбиение временного ряда на значимые сегменты.

Система нечеткого логического вывода – модель, описывающая поведение систем на естественном (или близком к естественному) языке в виде приближенных рассуждений на основе композиционного правила вывода.

Среднеквадратическое отклонение – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания.

Стандартизация данных – процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

Степень принадлежности – значение, задаваемое функцией принадлежности и находящееся в интервале от 0 до 1.

Фаззификация – процесс установки соответствия между четким значением x и нечетким f через функцию принадлежности.

Функция принадлежности – функция, отображающая базовое значение x и нечеткое f в интервал от 0 до 1.

Хромосома (в генетическом алгоритме) – битовая строка, описывающая решение.

Частотный анализ временного ряда – поиск часто проявляющихся паттернов ВР.

Энтропия – мера неупорядоченности системы.

Предметный указатель

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа». URL: http://ru.wikipedia.org/wiki/, (режим доступа – свободный), (дата обращения: 28.03.2017).

2. [Электронный ресурс]: Материалы открытого курса по машинному обучению от компании ODS. URL: https://habrahabr.ru/company/ods/blog/325654/, (режим доступа – свободный), (дата обращения: 28.03.2017).

3. Воронина, В. В. Разработка приложений для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина, В. С. Мошкин. – Ульяновск : УлГТУ, 2015. – 162 с.

4. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Переобучение, (режим доступа – свободный), (дата обращения: 28.03.2017).

5. [Электронный ресурс]: Статья по регуляризации. URL: https://ru.wikipedia.org/wiki/Регуляризация_(математика), (режим доступа – свободный), (дата обращения: 28.03.2017).

6. [Электронный ресурс]: Материалы сайта MSDN. URL: https://msdn.microsoft.com/ru-ru/magazine/dn904675.aspx, (режим доступа – свободный), (дата обращения: 28.03.2017).

7. [Электронный ресурс]: Материалы сайта machinelearning. URL:http://www.machinelearning.ru/wiki/index.php?title=Нормальное_распределение, (режим доступа – свободный), (дата обращения: 28.03.2017).

8. Паклин, Н. Б. Глава 9, // Бизнес-аналитика: от данных к знаниям : учебное пособие / Н. Б. Паклин, В. И. Орешков. – 2-е изд.. – СПб. : Питер, 2013. – С. 444-459.

9. [Электронный ресурс]: Ю. Лаходюк, Энтропия и деревья принятия решений. URL: https://habrahabr.ru/post/171759/, (режим доступа – свободный), (дата обращения: 28.03.2017).

10. [Электронный ресурс]: Статья по деревьям решений. URL: http://ru.wikipedia.org/wiki/Дерево_принятия_решений, (режим доступа – свободный), (дата обращения: 28.03.2017).

11. Клячкин, В. Н. Статистические методы анализа данных / В. Н. Клячкин, Ю. Е. Кувайскова, В. А. Алексеева. – М. : Финансы и статистика, 2016. – 240 с.

12. Клячкин, В. Н. Статистические методы в управлении качеством: компьютерные технологии / В. Н. Клячкин. – М. : Финансы и статистика, ИНФРА-М, 2009. – 304 с.

13. Валеев, С.Г. Регрессионное моделирование при обработке наблюдений / С. Г. Валеев. – М. : Наука, 1991. – 272 с.

14. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа»: Статья под названием «Робастные методы». URL: http://ru.wikipedia.org/wiki/Робастность, (режим доступа – свободный), (дата обращения: 08.07.2017).

15. Zadeh, A. Lotfi. Fuzzy Sets / Lotfi A. Zadeh // Information and Control. – 1965.

16. Заде, Л. А. Основы нового подхода к анализу сложных систем и процессов принятия решений / Л. А. Заде // Математика сегодня. – М. : Знание, 1974. – С. 5-49.

17. Штовба, С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. – М. : Горячая линия – Телеком, 2007. – 288 с.

18. Ярушкина, Н. Г. Основы теории нечетких и гибридных систем : учебное пособие / Н. Г. Ярушкина. – М. : Финансы и статистика, 2004. – 320 с.

19. Ярушкина, Н. Г. Интеллектуальный анализ временных рядов : учебное пособие / Н. Г. Ярушкина, Т. В Афанасьева., И. Г. Перфильева. – М. : ИД «ФОРУМ» : ИНФРА-М, 2012. – 160 с. – (Высшее образование).

20. Song, Q. Fuzzy time series and its models / Q. Song, B. Chissom // Fuzzy Sets and Systems. – №54 (1993) – P. 269-277.

21. [Электронный ресурс] Дегтярев, К. Ю. Применение специализированных компьютерных программ и методов, основанных на нечетких временных рядах для краткосрочного прогнозирования USB/RUB котировок / К. Ю. Дегтярев. URL: http://www.exponenta.ru/educat/news/degtyarev/paper.pdf, (режим доступа – свободный), (дата обращения: 08.07.2017).

22. Batyrshin, I. Perception Based Time Series Data Mining for Decision Making / I. Batyrshin // IFSA’07 Fuzzy Logic, Soft Computing and Computational Intelligence.

23. [Электронный ресурс]: Материалы IRAFM-2015. URL: http://irafm.osu.cz/cif2015/main.php?c=Static&page=results, (режим доступа – свободный), (дата обращения: 08.07.2017).

24. Новак, В. Математические принципы нечеткой логики / В. Новак, И. Перфильева, И. Мочкорж; пер. с англ.; под ред. А. Н. Аверкина. – М. : ФИЗМАТЛИТ, 2006.

25. Афанасьева Т. В. Модель ACL-шкалы для генерации лингвистических оценок в принятии решений / Т. В. Афанасьева // Вопросы современной науки и практики. Университет им. В. И. Вернадского. Т. 2. Серия «Технические науки». – 2008. – №4 (14). – С. 91-97.

26. [Электронный ресурс]: Саймон Хайкин - Нейронные сети. Полный курс. URL: http://neuralnetworksanddeeplearning.com, (режим доступа – свободный), (дата обращения: 08.07.2017).

27. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Модель_Мак-

Каллока_Питтса, (режим доступа – свободный), (дата обращения: 08.07.2017).

28. [Электронный ресурс]: Статья Логика мышления. Часть 3. Персептрон, сверточные сети. URL: https://geektimes.ru/post/214317/, (режим доступа – свободный), (дата обращения: 08.07.2017).

29. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Самоорганизующаяся_карта_Кохонена, (режим доступа – свободный), (дата обращения: 08.07.2017).

30. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа»: Python. URL: https://ru.wikipedia.org/wiki/Python, (режим доступа – свободный), (дата обращения: 08.07.2017).

31. [Электронный ресурс]: Материалы по синтаксису Python. URL: https://habrahabr.ru/post/31180/, (режим доступа – свободный), (дата обращения: 08.07.2017).

32. [Электронный ресурс]: Официальный сайт разработчиков Anaconda. URL: https://www.continuum.io/Downloads, (режим доступа – свободный), (дата обращения: 08.07.2017).

33. [Электронный ресурс]: Официальный сайт PyCharm. URL: https://www.jetbrains.com/pycharm/download/#section=windows, (режим доступа – свободный), (дата обращения: 08.07.2017).

34. [Электронный ресурс]: Официальный сайт SciPy. URL: https://scipy.org/ (режим доступа – свободный), (дата обращения: 08.07.2017).

35. [Электронный ресурс]: Покоряем Python – уроки для начинающих. Функции lambda. URL: https://pythlife.blogspot.ru/2012/11/lambda.html (режим доступа – свободный), (дата обращения: 08.07.2017).

36. [Электронный ресурс]: Документация по DataFrame. URL: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Data

Frame.html (режим доступа – свободный), (дата обращения: 08.07.2017).

37. [Электронный ресурс]: Уроки по Pandas. URL: https://bitbucket.org/hrojas/learn-pandas (режим доступа – свободный), (дата обращения: 08.07.2017).

38. [Электронный ресурс]: Документация по Scikit-learn. URL: http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-normalization (режим доступа – свободный), (дата обращения: 08.07.2017).

39. [Электронный ресурс]: Сервис Kaggle. Массив данных о пассажирах Титаника URL: https://www.kaggle.com/prkukunoor/TitanicDataset (режим доступа – свободный), (дата обращения: 08.07.2017).

40. [Электронный ресурс]: Документация по Scikit-learn:Ridge. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge (режим доступа – свободный), (дата обращения: 08.07.2017).

41. [Электронный ресурс]: Ссылка для скачивания упоминаемых в книге файлов с данными. URL: https://drive.google.com/drive/folders/0B5yyS8oSQ0FDelpKRXg3c3lKVlU (режим доступа – свободный), (дата обращения: 20.07.2017).

42. [Электронный ресурс]: Документация по Scikit-learn:TFIDFVectorizer. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer (режим доступа – свободный), (дата обращения: 08.07.2017).

43. [Электронный ресурс]: Документация по Scikit-learn: DictVectorizer. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html (режим доступа – свободный), (дата обращения: 08.07.2017).

44. [Электронный ресурс]: Документация по Scikit-learn: LogisticRegression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (режим доступа – свободный), (дата обращения: 08.07.2017).

45. [Электронный ресурс]: Документация по Scikit-learn: LinearRegression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (режим доступа – свободный), (дата обращения: 08.07.2017).

46. [Электронный ресурс]: Документация по Scikit-learn:Lasso. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (режим доступа – свободный), (дата обращения: 08.07.2017).

47. [Электронный ресурс]: Документация по Scikit-learn: MinMaxScaler. URL: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html(режим доступа – свободный), (дата обращения: 08.07.2017).

48. [Электронный ресурс]: Функции map и zip и lambda. Python. URL: http://ninjaside.info/blog/ru/funkcii-map-i-zip-i-lambda-python/ (режим доступа – свободный), (дата обращения: 08.07.2017).

49. [Электронный ресурс]: Документация по Scikit-learn: RandomizedLasso. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RandomizedLasso.html (режим доступа – свободный), (дата обращения: 08.07.2017).

50. [Электронный ресурс]: Документация по Scikit-learn:RFE. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html(режим доступа – свободный), (дата обращения: 08.07.2017).

51. [Электронный ресурс]: Документация по Scikit-learn: RandomForestRegressor. URL: http://scikit-learn.org/stable/modules/

generated/sklearn.ensemble.RandomForestRegressor.html (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

52. [Электронный ресурс]: Документация по Scikit-learn: f_regression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html(режим доступа – свобод-ный), (дата обращения: 08.07.2017).

53. [Электронный ресурс]: Документация по Scikit-learn: Pipeline. URL: http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html(режим доступа – свободный), (дата обращения: 08.07.2017).

54. [Электронный ресурс]: Документация по Scikit-learn: Cross-validation. URL: http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation(режим доступа – свободный), (дата обращения: 08.07.2017).

55. [Электронный ресурс]: Документация по Scikit-learn: Cross_val_score. URL: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html, (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

56. [Электронный ресурс]: Документация по Scikit-learn: Perceptron. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html, (режим доступа – свободный), (дата обращения: 08.07.2017).

57. [Электронный ресурс]: Документация по Scikit-learn: MLPClassifier. URL: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

58. [Электронный ресурс]: Официальный сайт NVIDIA. Проверка видеокарты. URL: https://developer.nvidia.com/cuda-gpus (режим доступа – свободный), (дата обращения: 08.07.2017).

59. [Электронный ресурс]: Официальный сайт NVIDIA. Скачивание CUDA. URL: https://developer.nvidia.com/cuda-downloads (режим доступа – свободный), (дата обращения: 08.07.2017).

60. [Электронный ресурс]: Официальный сайт NVIDIA. Скачивание CUDANN. URL: https://developer.nvidia.com/rdp/cudnn-download (режим доступа – свободный), (дата обращения: 08.07.2017).

61. [Электронный ресурс]: Документация по билиотеке Keras. URL: https://keras.io/layers/core/, (режим доступа – свободный), (дата обращения: 08.07.2017).

62. [Электронный ресурс]: Материалы сайта machinelearningmastery. URL: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/, (режим доступа – свободный), (дата обращения: 08.07.2017).

63. [Электронный ресурс]: Статья по реализации генетического алгоритма на Python. URL: http://easydan.com/arts/2016/genetic-optimization/, (режим доступа – свободный), (дата обращения: 08.07.2017).

64. Воронина, В. В. Разработка веб-сервисов для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина. – Ульяновск : УлГТУ, 2016. – 166 с.

65. [Электронный ресурс]: Документация «Installing OpenCV from prebuilt binaries». URL: http://docs.opencv.org/3.2.0/d5/de5/tutorial_py_setup_in_windows.html, (режим доступа – свободный), (дата обращения: 08.07.2017).

Учебное электронное издание

ВОРОНИНА Валерия Вадимовна
МИХЕЕВ Александр Вячеславович
ЯРУШКИНА Надежда Глебовна
СВЯТОВ Кирилл Валерьевич

ТЕОРИЯ И ПРАКТИКА МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Редактор Н. А. Евдокимова

ЛР № 020640 от 22.10.97.

ЭИ № 1004. Объем данных 3,6 Мб.

Печатное издание
Подписано в печать 08.11.2017.
Усл. печ. л. 16.97. Тираж 100 экз. Заказ 932.

Ульяновский государственный технический университет,
432027, г. Ульяновск, ул. Сев. Венец, д. 32.

ИПК «Венец», УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.
Тел.: (8422) 778-113
E-mail: venec@ulstu.ru
venec.ulstu.ru

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.03 Прикладные задачи анализа данных

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Прикладные задачи анализа данных

Профиль подготовки

Искусственный интеллект в автоматизации проектирования

Квалификация выпускника

Магистр

Формы обучения

очная

г. Ульяновск, 2021

ЛЕКЦИИ

Лекция 1. Введение в бизнес-аналитику. Предварительный анализ данных

Такая разная аналитика

Бизнес-аналитика. Бизнес-аналитика — процесс анализа данных, позволяющий руководителям, менеджерам и другим заинтересованным сторонам принимать обоснованные бизнес-решения.

- Бизнес-аналитик
- Продуктовый аналитик
- Маркетинговый аналитик
- Системный аналитик
- UX-аналитик
- Веб-аналитик
- **Аналитик данных**
    - BI-аналитик
    - Аналитик Big Data (Data Scientist)

Данные собирают все — от магазинов и ресторанов до компаний-монополистов и приложений с миллионной аудиторией. Аналитик данных помогает сделать так, чтобы собранная информация приносила пользу бизнесу.

Аналитик данных (или дата-аналитик) — это специалист, который собирает, обрабатывает, изучает и интерпретирует данные. Его работа помогает принимать решения в бизнесе, управлении и науке.

Хороший аналитик данных — не просто математик с навыками программиста. Он понимает бизнес-процессы и хорошо знает продукт. Такой специалист разбирается, на чем зарабатывает конкретный бизнес. В результате его работы компания может получать больше прибыли и делать своих пользователей счастливее. Сильный аналитик данных прежде, чем взяться за работу всегда спрашивает руководителя о том, какую задачу хочет решить бизнес.

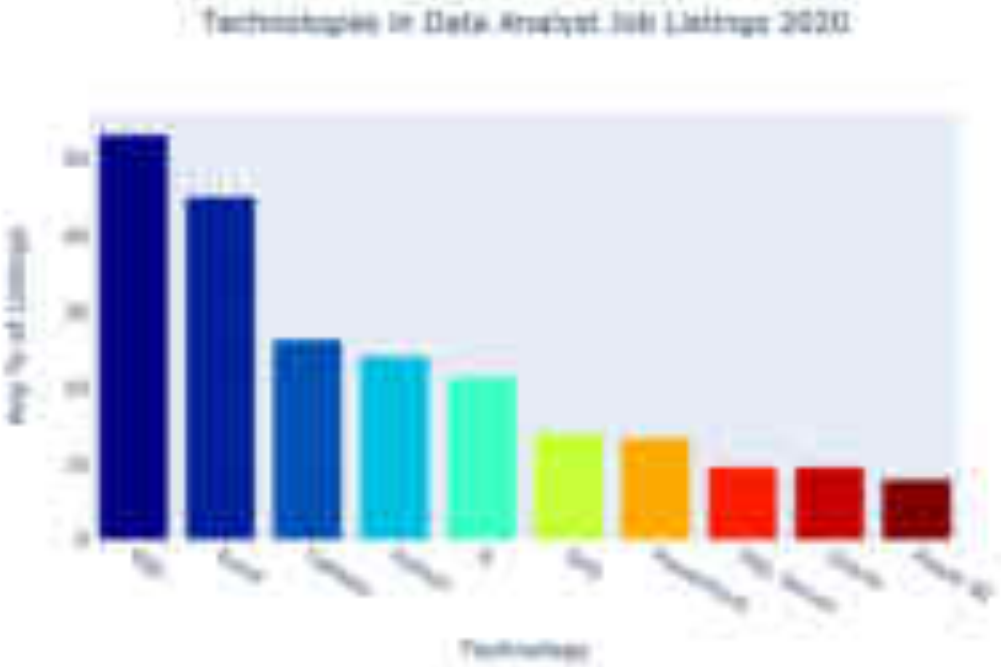
Аналитики данных берут данные, предоставленные инженерами данных, анализируют их и дают рекомендации. Они создают визуализации для отображения своих результатов в информационных панелях и презентациях.

В отличие от исследователей данных, аналитики данных обычно не создают прогностические модели, основанные на алгоритмах машинного обучения.

Знания и навыки

Стадии работы и инструменты

10 технологий из списков вакансий аналитика данных по состоянию на январь 2020 года



Аналитик данных, или Data Analyst, становится одной из самых востребованных ИТ-специальностей. По данным исследования рынка аналитиков, спрос на профессионалов значительно превышает предложение.

Сегодня создана целая наука, посвященная обработке данных, которая так и называется data science - наука о данных. Однако некоторые эксперты полагают, что это определение ошибочно, потому что data science – не "наука о данных", как написано в русскоязычной Википедии. Данные не являются предметом этой науки, поэтому называть data science синонимом предложенной Петером Науром науки datalogy ошибочно. Термин data science на русский язык, возможно, стоило бы переводить как "наука работы с данными" или "**научные методы работы с данными**". Задача, решаемая теми, кто занимается data science, состоит в извлечении знаний с использованием методов, объединенных под общим названием data mining, в объединении статистики и других методов анализа данных с целью понимания того, что содержат в себе данные.

Четвертая парадигма науки

Обладатель премии Тьюринга Джим Грей и астроном и футуролог Алекс Шалаи разделили научное прошлое человечества на три периода использования данных и дополнили его современным четвертым.

1. **А**нтичные времена — описание наблюдаемых феноменов и логические выводы, сделанные на основе наблюдений.
2. **X**VII век — создание теорий с использованием для доказательства их истинности аналитических моделей.
3. **X**X век — использование методов численного моделирования, ставшее возможным благодаря появлению компьютеров.
4. **X**XI век — **использование методов, основанных на анализе данных**; применение для работы с огромными объемами данных статистических и других методов извлечения полезной информации.

Анализ данных

Анализ данных — область математики и информатики, занимающаяся построением и исследованием наиболее общих математических методов и вычислительных алгоритмов извлечения знаний из экспериментальных (в широком смысле) данных; процесс исследования, фильтрации, преобразования и моделирования данных с целью извлечения полезной информации и принятия решений. **Анализ данных имеет множество аспектов и подходов, охватывает разные методы в различных областях науки и деятельности.**

Business Intelligence и бизнес-аналитика

Business intelligence (*BI*) — обозначение компьютерных методов и инструментов для организаций, обеспечивающих перевод транзакционной деловой информации в человекочитаемую форму, а также средства для массовой работы с такой обработанной информацией.

Цель BI — интерпретировать большое количество данных, заостряя внимание лишь на ключевых факторах эффективности, моделируя исход различных вариантов действий, отслеживая результаты принятия решений.

Традиционно BI-решения используются для статических отчетов о текущем или прошлом состоянии бизнеса. Они отвечают на такие вопросы, как: «Какая динамика объема продаж в прошедшем квартале? За счет чего произошел рост или падение продаж? Какой тип продукции произвели больше всего за месяц?». Это так называемый дескриптивный или описательный анализ. Кроме того, BI системы работают со структурированными данными, извлеченными из хранилищ данных, и представляют результат анализа в виде интерактивных информационных панелей — дашбордов или отчетов.

- Термины BI и «бизнес-аналитика» зачастую используются как синонимы, но между ними есть разница. Бизнес-аналитика (в узком понимании), в отличие от BI, имеет дело с уже очищенными, подготовленными для анализа данными, использует статистические и количественные инструменты для оценки текущей ситуации и прогнозирования, поэтому ее все чаще называют «углубленная аналитика».

- Business Intelligence, BI вначале занимается очисткой, консолидацией данных, преобразованием их в удобный для анализа формат, следующие задачи — интерпретировать большое количество данных, заостряя внимание лишь на ключевых факторах, влияющих на эффективность, моделировать исход различных вариантов действий, отслеживать результаты принятия решений. **Основное назначение BI — это именно принятие решений для бизнеса.**

- Основу методов data mining составляют всевозможные методы классификации, моделирования и прогнозирования, основанные на применении деревьев решений, искусственных нейронных сетей, генетических алгоритмов, эволюционного программирования, ассоциативной памяти, нечёткой логики. К методам data mining нередко относят статистические методы (дескриптивный анализ, **корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ, компонентный анализ, дискриминантный анализ, анализ временных рядов**, анализ выживаемости, анализ связей). Такие методы, однако, предполагают некоторые априорные представления об анализируемых данных, что несколько расходится с целями data mining (обнаружение ранее неизвестных нетривиальных и практически полезных знаний).

BI-системы развиваются по четырем основным направлениям:

1. **Хранение данных**. Данные в хранилище BI-системы (data warehouse, DW) структурируются специальным образом для более эффективного анализа и обработки запросов (в отличие от обычных баз данных, где информация организована таким образом, чтобы оптимизировать время обработки текущих транзакций).

2. **Интеграция данных**. Для формирования и поддержания хранилищ данных используются ETL-средства -–— инструменты, обеспечивающие извлечение данных (extract), их преобразование (transform), т.е. приведение к необходимому формату, и загрузку (load) данных в хранилище или в другую базу.
3. **Анализ данных.** Для всестороннего анализа данных используются OLAP-инструменты (on-line analytical processing). Они позволяют рассматривать различные срезы данных, выявлять тренды и зависимости (по регионам, продуктам, клиентам и т.п.).
4. **Представление данных.** Для представления данных используются различные графические средства — отчеты, графики, диаграммы. Общепринятым средством визуализации данных являются информационные панели (dashboards), на которых результаты отображаются в виде индикаторов и шкал, позволяющих контролировать текущие значения выбранных показателей, сравнивать их с минимально/максимально допустимыми и таким образом выявлять потенциальные угрозы для бизнеса.

Интеллектуальный анализ данных

Data mining (рус. добыча данных, интеллектуальный анализ данных, глубинный анализ данных) — собирательное название, используемое для обозначения совокупности методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.
Английское словосочетание «*data mining*» пока не имеет устоявшегося перевода на русский язык. Более полным и точным является словосочетание «*обнаружение знаний в базах данных*».

Постановка задачи

Первоначально задача ставится следующим образом:

- имеется достаточно крупная база данных;
- предполагается, что в базе данных находятся некие «скрытые знания».

Необходимо разработать методы обнаружения знаний, скрытых в больших объёмах исходных «сырых» данных. В текущих условиях глобальной конкуренции именно найденные закономерности (знания) могут быть источником дополнительного конкурентного преимущества.

Что означает «скрытые знания»? Это должны быть обязательно знания:

- ранее неизвестные — то есть такие знания, которые должны быть новыми (а не подтверждающими какие-то ранее полученные сведения);
- нетривиальные — то есть такие, которые нельзя просто так увидеть (при непосредственном визуальном анализе данных или при вычислении простых статистических характеристик);

- практически полезные — то есть такие знания, которые представляют ценность для исследователя или потребителя;
- доступные для интерпретации — то есть такие знания, которые легко представить в наглядной для пользователя форме и легко объяснить в терминах предметной области.

Data mining и искусственный интеллект

Знания, добываемые методами data mining, принято представлять в виде закономерностей (паттернов). В качестве таких выступают:

- *ассоциативные правила;*
- *деревья решений;*
- *кластеры;*
- *математические функции.*

Алгоритмы поиска таких закономерностей находятся на пересечении областей: Искусственный интеллект, Математическая статистика, Математическое программирование, Визуализация, OLAP.

Прикладная статистика

Прикладная статистика — наука о методах обработки статистических данных. Методы прикладной статистики активно применяются в технических исследованиях, экономике, менеджменте, социологии, медицине, геологии, истории и т. д.

Прикладная статистика нацелена на решение реальных задач. Поэтому в ней возникают новые постановки математических задач анализа статистических данных, развиваются и обосновываются новые методы. Обоснование часто проводится математическими методами, то есть путём доказательства теорем.

Машинное обучение

Глубинная суть машинного обучения — предсказание: предсказание наших желаний, результатов наших действий, путей достижения целей, изменений мира.

Машинное обучение принимает много разных форм и скрывается под разными именами: **распознавание паттернов, статистическое моделирование, добыча данных, выявление знаний, предсказательная аналитика, наука о данных, адаптивные и самоорганизующиеся системы** и так далее. Все они находят свое применение и имеют разные ассоциации.
Единственным способом заставить компьютер что-то делать — от сложения двух чисел до управления самолетом — было составление некоего алгоритма, скрупулезно объясняющего машине, что именно от нее требуется. Однако алгоритмы машинного обучения — совсем другое дело: они угадывают все сами, делая выводы

на основе данных, и чем больше данных, тем лучше у них получается. Это значит, что **компьютеры не надо программировать: они программируют себя сами**. Машинное обучение — технология, которая строит саму себя. Это новое явление в нашем мире.
Машинное обучение иногда путают с искусственным интеллектом. С формальной точки зрения это действительно подраздел науки об искусственном интеллекте, однако он очень разросся и оказался настолько успешным, что затмил гордого родителя. Цель искусственного интеллекта — научить компьютеры делать то, что люди пока делают лучше, а умение учиться — наверное, самый важный из этих навыков, без которого компьютерам никогда не угнаться за человеком.

Что такое машинное обучение?

Машинное обучение — это способ обработки и анализа данных, который позволяет компьютерам использовать существующие данные **для прогнозирования поведения, результатов и тенденций в будущем**.

С помощью машинного обучения компьютеры учатся сами, вам не приходится специально их программировать.

Прогнозы на основе машинного обучения помогают оптимизировать работу устройств и приложений. На основании ваших предыдущих покупок через Интернет машинное обучение позволяет рекомендовать товары, которые могут вам понравиться. Когда вы расплачиваетесь кредитной картой, машинное обучение сравнивает эту операцию с другими операциями в базе данных и помогает обнаружить мошенничество. Когда робот-пылесос убирается в комнате, машинное обучение помогает ему решить, что пора заканчивать.

Наука о данных

Наука о данных (**data science**; иногда даталогия — datalogy) — раздел информатики, изучающий проблемы анализа, обработки и представления данных в цифровой форме. Объединяет методы по обработке данных в условиях больших объёмов и высокого уровня параллелизма, статистические методы, методы интеллектуального анализа данных и приложения искусственного интеллекта для работы с данными, а также методы проектирования и разработки баз данных.

- Основная практическая цель профессиональной деятельности в науке о данных — **обнаружение закономерностей в данных, извлечение знаний из данных в обобщённой форме**.
- Наука о данных — это концепция объединения статистики, анализа данных, машинного обучения и связанных с ними методов" для понимания и анализа реальных явлений.
- Она использует методы и теории, взятые из многих областей в контексте математики, статистики, информатики и компьютерных наук

- В настоящее время термин data science часто используется взаимозаменяемо с более ранними концепциями, такими как business analytics (бизнес-аналитика), business intelligence (интеллектуальный анализ даннных), predictive modeling (прогнозное моделирование) и statistics (статистика). Во многих случаях более ранние подходы и решения теперь просто переименовываются в "науку о данных", чтобы стать более привлекательными.

С начала 2010-х годов считается одной из самых привлекательных, высокооплачиваемых и перспективных профессий!

В сравнении с классической статистикой, на методах которой во многом основывается и наука о данных, в ней подразумевается исследование сверхбольших разнородных массивов цифровой информации и неразрывная связь с информационными технологиями, обеспечивающими их обработку.

В сравнении с деятельностью в области проектирования и работы с базами данных, где предполагается предварительное проектирование модели данных, в науке о данных предполагается опора на аппарат **математической статистики, искусственного интеллекта, машинного обучения**, зачастую без предварительной загрузки данных в модели.

В сравнении с профессией аналитика, основная цель деятельности которого в описании явлений на основе накопленных данных относительно простыми пользовательскими средствами (вроде электронных таблиц или средств класса Business Intelligence), профиль специалиста по науке о данных в меньшей степени требует концентрации на содержании предметных областей, но требует более глубоких знаний в математической статистике, машинном обучении, программировании, и в целом более высокого образовательного уровня.

Основные отличия data science от business intelligence

В обоих случаях решающую роль играют специалисты. Главное различие между двумя специальностями заключается в том, что эксперт в области BI способен предоставить объективную картину от прошлого до текущего момента, в то время как data scientist должен понимать, как и что нужно делать

Непосредственно в курс по науке о данных входят следующие дисциплины:

- машинное обучение;
- системы управления базами данных;
- инженерия программного обеспечения;
- анализ данных (*intelligent data*) и вероятностный вывод (*probabilistic inference*), в описании дисциплины даются ссылки на байесовский вывод и алгоритмические методы моделирования, классификации и дискриминантного анализа данных на его основе;
- вероятностные модели и продвинутая статистика.

В чем разница между наукой о данных, искусственным интеллектом и машинным обучением?

Наука о данных — это широкая область знаний, которая включает в себя предварительную обработку, анализ и визуализацию структурированных и неструктурированных данных. Полученные из данных выводы затем применяются в широком спектре областей применения.

Искусственный интеллект означает обучение машины подражать человеческому поведению. Цели исследования ИИ включают представление знаний, планирование, обучение, рассуждения, обработку естественного языка, восприятие и способность манипулировать объектами.

Машинное обучение — это подмножество ИИ, которое фокусируется на том, как использовать данные и алгоритмы, чтобы имитировать способ обучения людей. Чем больше данных (также называемых обучающими данными) получает модель машинного обучения, тем точнее она делает прогнозы, не будучи явно запрограммированной на это.

ОСНОВНЫЕ ЗАДАЧИ АНАЛИЗА ДАННЫХ

Основная цель любого анализа данных – поиск и обнаружение закономерностей в объеме данных. В бизнес-анализе эта цель становится еще более широкой. Любому руководителю важно не только выявить закономерности, но и найти их причину. Знание причины позволит в будущем влиять на бизнес и дает возможность прогнозировать результаты того или иного действия.

Углубленная аналитика выходит за рамки ответа на вопросы о том, что произошло, когда это произошло и каковы последствия. Она пытается определить причины произошедшего и решить, что можно предпринять в будущем.

Углубленная аналитика подразумевает ряд мероприятий, включая использование сложных запросов SQL, прогностическое моделирование, интеллектуальный анализ данных, прогнозирование, оптимизацию и др.

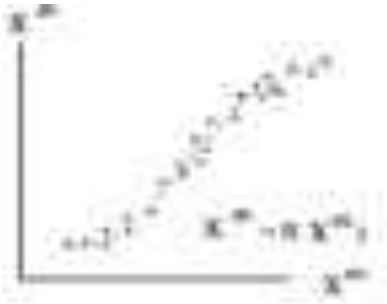
Углубленная аналитика идет дальше, чем базовая, включая в себя процедуры от сложных запросов SQL и прогнозирования до интеллектуального анализа данных и прогностического моделирования.

Углубленная аналитика представляет собой важную часть общей аналитической стратегии организации. Она может помочь вывести организацию на новый уровень. Углубленная аналитика включает в себя очень сложные запросы SQL или манипулирование данными наряду с моделированием, прогнозированием, интеллектуальным анализом данных и другими подобными методами.

1. Элементарные статистические выводы, проверка статистических гипотез

Предположим, что Вы изменили систему оплаты труда или перешли на выпуск новой продукции или использовали новую технологию и т.д. Вам кажется, что это дало положительный эффект, но действительно ли оно так? Может быть это естественная случайность, а завтра Вы можете получить прямо противоположный, но столь же случайный эффект? Для решения этой задачи надо сформировать два набора чисел, каждый из которых содержит значения интересующего Вас показателя эффективности до и после нововведения. Тогда статистические критерии сравнения 2-х выборок покажут Вам, случайны или неслучайны различия этих двух рядов чисел.

2. Исследование зависимостей одного признака от остальных



Формальная постановка: найти функцию зависимости, приближенно описывающую изменение целевого признака при изменении других признаков.

Примеры:

а) прогноз погоды по набору примет – «дым стелется, ласточки летают низко, росы нет, – значит, погода испортится».

б) прогнозирование будущего поведения некоторого временного ряда: изменение курса доллара, цен и спроса на продукцию или сырье и т.п.

Математический аппарат: корреляционный и регрессионный анализ.

3. Снижение размерности

Формальная постановка: устранить дублирующие друг друга признаки или найти (построить) новые признаки (меньшее число), описывающие данные. При этом определяющие признаки (факторы) могут находиться среди статистически обследованных характеристик, а могут быть латентными или скрытыми, т.е. непосредственно статистически не наблюдаемыми, но восстанавливаемыми по исходным данным (т.е. матрицам X или A).

Примеры: а) гениальный пример практической реализации этого принципа дает нам периодическая система элементов Менделеева: в этом случае роль идеально информативного единственного фактора играет, как известно, заряд атомного ядра.

б) нахождение системы признаков «размер - рост- полнота», описывает фигуру человека и определяющий тип размера при изготовлении готовой одежды.

На рисунке изображено двухмерное пространство описания. Структура данных в нем такова, что вместо двух исходных признаков $X^{(1)}$ и $X^{(2)}$ вполне можно обойтись одним $Y^{(1)}$. Если признаков много, то одной проекцией уже не обойдешься, и приходится сопоставлять результаты рассматривания структуры данных с различных точек наблюдения.

Математический аппарат: метод главных компонент, факторный анализ, многомерное шкалирование.

5. Классификация объектов (без учителя, когда исследователь не располагает обучающими выборками)

Формальная постановка: обнаружить в пространстве описания компактные скопления точек-объектов.

Примеры:

а) создание Карлом Линнеем классификации растений и животных по классам, отрядам, родам, видам.

б) исследование зависимости интенсивной миграции населения $X^{(P)}$ (профессиональной или территориальной) от ряда социально-экономических и географических факторов $x^{(1)}$, $x^{(2)}$, … , $x^{(p-1)}$ таких, как заработок, обеспеченность жильем, детские учреждения и т.д. Естественно предположить, что для различных однородных групп людей те же факторы влияют на $x^{(p)}$ в разной степени, а иногда в противоположных направлениях.

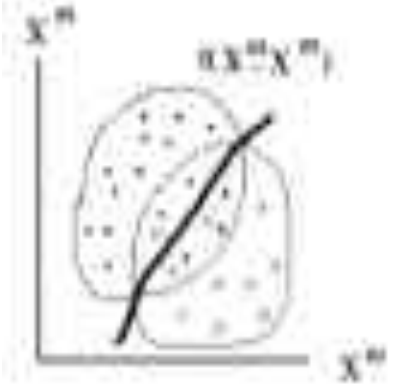
$X_i = (X^{(1)}_i, X^{(2)}_i, \ldots, X^{(p)}_i)$, $i=1,2,\ldots,n$

на однородные классы и решать далее поставленную задачу отдельно для каждого такого класса.

Здесь классификация выступает как необходимый предварительный этап статистической обработки многомерных данных.

Математический аппарат: кластерный анализ, методы ав-томатической классификации, методы расщепления смесей вероятностных распределений.

6. **Распознавание образов (с учителем, когда исследователь располагает обучающими выборками)**



Формальная постановка: найти в пространстве описания поверхность, разделяющую группу точек, соответствующих различным образам и описать ее как функцию исходных признаков; или найти, к какой группе точек (образу) относятся заданные точки-объекты.

Пример: этот пример показывает, как удачно построенный рисунок помог не просто получить информацию об объектах, а дал толчок совершенно новому направлению исследований. С давних пор астрономы знали о различной светимости звезд, т.е. о различной их «истинной яркости». В конце XIX в. были открыты также различные спектральные классы звезд, т.е. различный цвет их излучения (от красного до голубого).

Диаграмма Герцшпрунта-Рессела

До 1913 г. эти характеристики существовали отдельно в представлении ученых, но вот независимо друг от друга датский астроном Герцшпрунт и американец Ресселл сопоставили их между собой и построили двумерную проекцию объектов-звезд на плоскость признаков спектр-светимость. Визуализация здесь помо-гает понять, различаются ли вообще образы, оценить форму разделяющей поверхности и получить другую важную информацию.

Математический аппарат: Дискриминантный анализ. Сегодня для этого используют нейронные сети.

Основные этапы решения задачи анализа данных

На ранних стадиях развития науки анализировались, прежде всего очевидные, тесно связанные между собой причины и следствия – вспомним легенду о яблоке и Ньютоне. Сегодня исследователь любой ПО пытается найти методами АД причины и следствия очень «далекого диапазона».

Пример: имеется ли связь между жизненным успехом и именем, которое человек получил при рождении? Оказывается, люди с забавными или странными именами в 4 раза больше остальных предрасположены к психическим заболеваниям.

Этап	Содержание
Этап 1. Постановка задачи **(исходный анализ исследуемой системы)** **Этап 1А. составление детального плана сбора исходной статистической информации**	**1.1. Определение цели исследования** **1.2. Определение состава данных** **1.3. Сбор данных** **1.4. Выбор средств анализа данных** **1.5. Формализация данных**
Этап 2. Ввод данных в обработку	**2.1. Ввод данных в память ЭВМ** **2.2. Работа с архивом данных** **2.3. Формирование задания обработки**
Этап 3. Качественный анализ	**3.1. Первичная статистическая обработка данных** **3.2. Визуализация данных** **3.3. Анализ структуры данных**
Этап 4. Количественное описание данных	**4.1. Выбор модели данных** **4.2. Выполнение обработки**
Этап 5. Интерпретация результата	**5.1. Анализ результатов** **5.2. Принятие решения**

Этап 1. Постановка задачи

Одна из типичных ошибок исследователей состоит в том, что сначала собираются данные, а затем начинают формулироваться задачи их обработки. В этом случае **цель исследования подменяется** той или иной узкой целью обработки уже собранных конкретных данных. Тогда как заранее собранные данные могут отражать совсем другие характеристики явления, нежели те, которые важны для поставленной цели.

Состав данных — это, прежде всего состав признаков, которые характеризуют объект. Каждый реальный объект имеет бесконечное число различных свойств, отражающих его различные стороны. Выделить наиболее важные признаки – задача специалиста ПО. Необходимо также решать, как представить в ТЭД значения признаков – цифрами, числами, символами.

Определив состав признаков, можно приступить к **сбору данных**. На этой стадии основной вопрос – проблема выбора объектов, т.е. сколько нужно данных.

Следующая стадия – **выбор пакета программ или системы АД**. Решение зависит от следующих факторов: объема данных, числа объектов и признаков, типов признаков, квалификации исследователя.

Пятая стадия 1-го этапа – **формализация собранных данных**. На этой стадии ТЭД необходимо придать такой вид, какой требует от входных данных выбранная пользователем автоматизированная система АД.

Этап 2. Ввод данных в обработку

Собранные данные **вводят в ЭВМ**. Введенные данные попадают в архив. **Работа с архивом** – одна из наиболее трудоемких операций. Прежде всего, необходима проверка входных данных и исправление ошибок. На этой стадии можно выполнить некоторые простейшие подсчеты. Например, выделить объекты, имеющие одинаковые описания по одному или более признакам и т.п.

Последняя стадия второго этапа – **формирование задания обработки** очень ответственна, так как неграмотный выбор метода обработки может привести к одной из ошибок:

- выбор метода, не соответствующего сути задачи;
- выбор метода, неприменимого к имеющимся данным;
- выбор метода, дающего худший результат по сравнению с другими доступными методами.

Опыт показывает, что эти ошибки приносят наибольшее число неправильных или слабых конечных результатов.

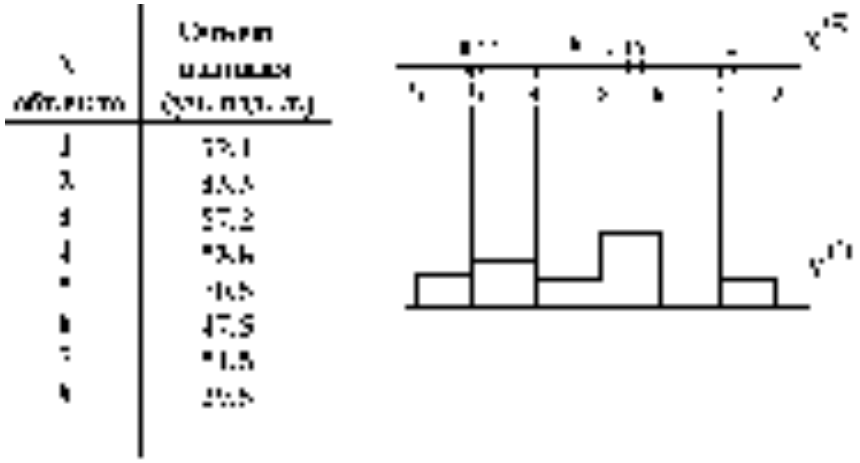
Этап 3. Качественный анализ

Решить задачу на качественном уровне – это, значит, представить собранные данные в **визуальной форме**, для того чтобы увидеть их пригодность для проверки выдвинутых гипотез или достижения поставленной цели. Почему именно увидеть? Зрительный анализатор человека – канал, по которому мозг получает наибольший объем внешней информации. В прикладной статике этот этап статического анализа принято называть разведочным (РАД – разведочный анализ данных). Цель такого анализа – представить наблюдаемые данные в возможно более компактной и простой форме, позволяющей выявить имеющиеся в них закономерности и связи.

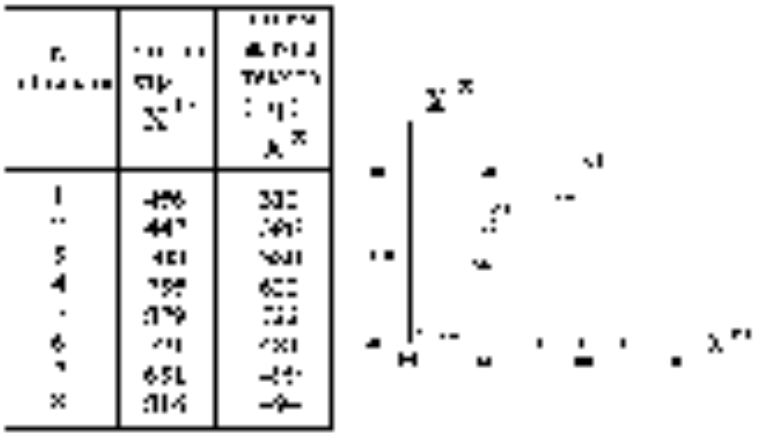
Рассмотрим наиболее важные «слова» этого языка визуальных образов.

Если взять из ТЭД один столбец чисел, то его можно изобразить в виде оси, на которой точками отмечено положение каждого объекта, а около каждой точки может стоять номер соответствующего объекта.

Это изображение называют проекцией данных на признак. Этот же признак можно изобразить иначе, если, разбив всю область его значений на некоторое количество интервалов, построить в каждом интервале столбец, высота которого пропорциональна числу попавших в интервал объектов. Такое изображение называют гистограммой объектов по признаку.



Добавим к первому признаку еще один. Тогда каждый объект можно изобразить точкой, расположенной уже в двумерном пространстве, то есть на плоскости. Такая проекция на плоскость двух признаков называется диаграммой рассеяния (ДР) объектов. Добавив 3-й, 4-й … признаки получим представление объектов в пространстве 3-х, 4-х и т.д. измерений. Таким образом каждому объекту ТЭД можно сопоставить точку в гипотетическом многомерном пространстве. Тогда подбор таких точек-объектов будет называться структурой данных в пространстве описания.



В качестве отображенных на ДР единиц могут выступать объекты, переменные(признаки), категории переменных (если переменные не количественные). Их называют отображаемыми единицами (ОЕ). Графически же элементы, с

помощью которых ОЕ изображаются на ДР, называются выразительными элементами (ВЭ).

Гипотез, объясняющих явление, может быть множество, следовательно, должен иметься аппарат, помогающий осуществить их проверку. В АД таким аппаратом является вычислительный эксперимент с данными, то есть применение к данным определенного метода машинной обработки, выбранного в соответствии с гипотезой исследователя.

На этом этапе основные гипотезы касаются **структуры данных** – именно ее необходимо исследовать. Поэтому задача исследователя состоит в построении проекций данных на различные пары признаков (на какие - следует определить исходя из выдвинутой гипотезы и смысла задачи); исследование отдельных признаков; поиск дублирующих друг друга или избыточных признаков и т.д.

Этап 4. Количественное описание данных

На этом этапе обычно ведется поиск параметров моделей, найденных на предыдущем этапе. Вычислительный эксперимент дает возможность испытывать различные варианты моделей, т.е. искать различные способы информативного описания данных, а сопоставительный анализ помогает отбирать лучшие варианты (не только на формальном, но и на содержательном уровне. Пуанкаре – формальные результаты теории относительности были получены за 20 лет до Эйнштейна).

Этап 5. Интерпретация результата

Все получаемые на ЭВМ результаты специалист по анализу данных (АД) может интерпретировать, на выходя за рамки понятий АД, в терминах информативных признаков, группировок объектов и т.д. пользователь же переводит полученный результат на привычный язык предметной области (ПО).

Таким образом и в предыдущем этапе выделяют два уровня – формальный и содержательный, причем определяющим является второй.

Однако обработка данных должна заканчиваться не результатом, а рекомендациями по его применению или принятию решения относительно дальнейших действий.

Здесь так же, как и при интерпретации результатов существуют два уровня – в рамках АД и ПО.

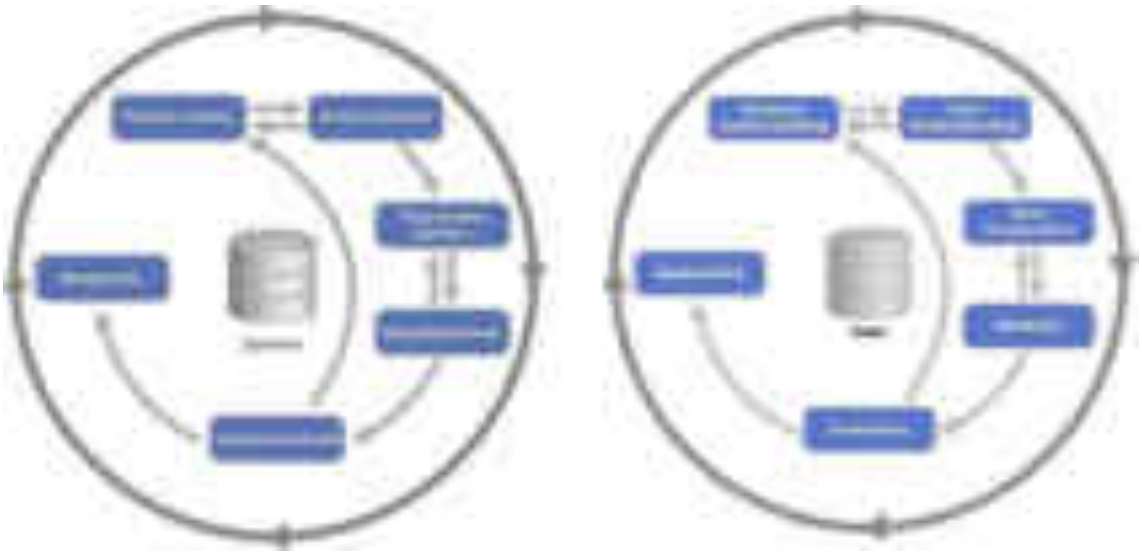
В рамках АД

1) Может быть принято решение прекращения дальнейшей обработки, так как поставленные цели достигнуты.
2) Продолжение обработки данных с использованием других методов, возможно с коррекцией данных.

CRISP-DM

CRISP-DM (Cross-Industry Standard Process for Data Mining) — межотраслевой стандартный процесс исследования данных. Это проверенная в промышленности и наиболее распространённая методология, первая версия которой была представлена в Брюсселе в марте 1999 года, а пошаговая инструкция опубликована в 2000 году.

CRISP-DM описывает жизненный цикл исследования данных, состоящий из 6 фаз, от постановки задачи с точки зрения бизнеса до внедрения технического решения.



Фазы CRISP-DM

Последовательность между фазами определена не строго, переходы могут повторяться от итерации к итерации. Все фазы CRISP-DM делятся на задачи, по итогам каждой должен быть достигнут конкретный результат.

Понимание бизнеса

Понимание бизнеса (Business Understanding) – определение целей проекта и требований со стороны бизнеса. Затем эти знания конвертируются в постановку задачи интеллектуального анализа данных и предварительный план достижения целей проекта. Задачи фазы Business Understanding:

- *Определить бизнес-цели*
- *Оценить ситуацию*
- *Определить цели анализа данных*
- *Составить план проекта*

Начальное изучение данных

Начальное изучение данных (Data Understanding) – сбор данных и знакомство с информацией, выявление проблем с качеством данных (ошибки или пропуски). Необходимо понять, какие сведения имеются, попробовать отыскать интересные

наборы данных или сформировать гипотезы о наличии в них скрытых закономерностей. Задачи фазы Data Understanding:

- *Собрать исходные данные*
- *Описать данные*
- *Исследовать данные*
- *Проверить качество данных*

Подготовка данных

Подготовка данных (Data Preparation) – получение итогового набора данных, которые будут использоваться при моделировании, из исходных разнородных и разноформатных данных. Задачи фазы Data Preparation могут выполняться много раз без какого-то заранее определенного порядка:

- *Отобрать данные (таблицы, записи и атрибуты)*
- *Очистить данные, в т.ч. выполнить их конвертацию и подготовку к моделированию*
- *Сделать производные данные*
- *Объединить данные*
- *Привести данные в нужный формат*

Моделирование

Моделирование (Modeling) – в этой фазе к данным применяются разнообразные методики моделирования, строятся модели и их параметры настраиваются на оптимальные значения. Обычно для решения любой задачи анализа данных существует несколько различных подходов. Некоторые подходы накладывают особые требования на представление данных. Таким образом часто бывает нужен возврат на шаг назад к фазе подготовки данных. Задачи фазы Modeling:

- *Выбрать методику моделирования*
- *Сделать тесты для модели*
- *Построить модель*
- *Оценить модель*

Оценка

Оценка (Evaluation) – анализ количественных характеристик качества модели, подтверждение или опровержение того, что, благодаря построенной модели все бизнес-цели достигнуты. Основной целью этапа является поиск важных бизнес-задач, которым не было уделено должного внимания. Задачи фазы Evaluation:

- *Оценить результаты*
- *Сделать ревью процесса*
- *Определить следующие шаги*

Внедрение

Внедрение (Deployment) – в зависимости от требований фаза развертывания может быть простой (составление финального отчета) или сложной, например, автоматизация процесса анализа данных для решения бизнес-задач. Обычно

развертывание — это внедрение полученных моделей в прикладную сферу. Задачи фазы Deployment:

- *Запланировать развертывание*
- *Запланировать поддержку и мониторинг развернутого решения*
- *Сделать финальный отчет*
- *Сделать ревью проекта*

Первичная обработка данных

- Методы предварительной обработки данных – очистка, нормализация, выделение признаков, преобразование, уплотнение.
- Описательные характеристики и отображение данных.
- Классификация статистических данных по различным критериям

Предварительная обработка данных является важным шагом в процессе интеллектуального анализа данных. Фраза «мусор на входе — мусор на выходе» применима, в частности, и для проектов интеллектуального анализа данных и машинного обучения. Здесь имеется в виду то, что даже самый изощренный анализ не принесет пользы, если за основу взяты сомнительные данные.

Методы сбора данных часто плохо контролируются. Это приводит к появлению недопустимых значений (к примеру: доход, равный −100), комбинаций данных, которые невозможны (к примеру: «мужской пол при наличии беременности»), отсутствию значений и прочее. В результате анализа данных, которые не защищены от такого рода проблем, можно прийти к неверным выводам. Качество данных является первостепенной задачей при проведении анализа.

Предварительная подготовка данных включает в себя:

- *очистку*
- *отбор экземпляров*
- *нормализацию*
- *преобразование данных*
- *выделение признаков*
- *отбор признаков*

Методы

Очистка данных используется для обнаружения, исправления или удаления ошибочных записей в наборе данных;

Нормализация данных используется для стандартизации диапазона значений независимых переменных или признаков данных (например, сведение к интервалам [0, 1] или [-1, +1]);

Преобразование данных используется для приведения данных в формат, который ожидает аудитория;

Выделение признаков используется для преобразования входных данных в набор признаков, которые они хорошо представляют;

Уплотнение данных используется для преобразования числовых данных в исправленный, упорядоченный и упрощённый вид. Это помогает уменьшить количество и/или размерность данных.

Описательные характеристики и отображение данных

Описательная статистика занимается получением, систематизацией, обработкой и изучением тех или иных данных с использованием различных числовых характеристик.

Задача описательной статистики, как следует из названия, — дать хорошее описание данных. Она не для предсказаний, выводов или преобразований — только внешняя форма данных, измеренная в показателях.

Основные статистические показатели

Основные статистические показатели можно разделить на две группы:

1. меры среднего уровня
2. меры рассеяния.

Меры среднего уровня

- Среднее значение
- Стандартная ошибка
- Стандартное отклонение
- Эксцесс
- Асимметрия
- Интервал
- Минимум
- Максимум
- Счёт
- Медиана
- Мода
- Квантиль
- Математическое ожидание
- Доверительный интервал

Меры рассеяния

Меры рассеяния показывают, насколько хорошо данные значения представляют данную совокупность.

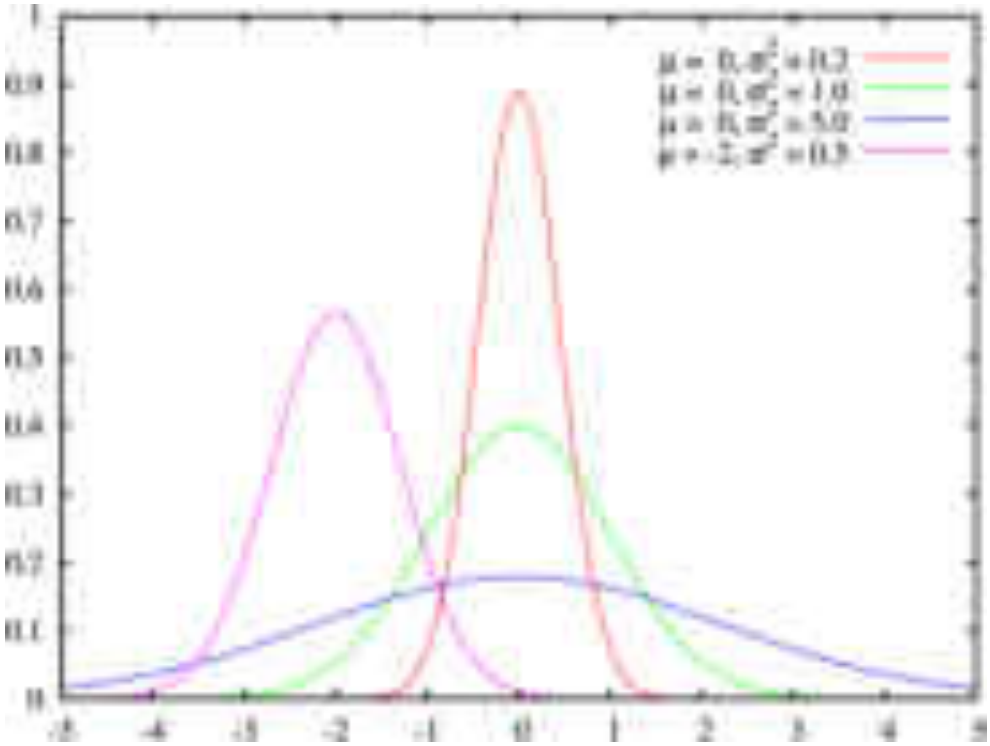
- Дисперсия случайной величины
- Среднеквадратическое отклонение
- Размах вариации
- Интерквантильный размах
- Среднее абсолютное отклонение

Распределения

Внешняя форма данных, выраженная в мерах описательной статистики, даёт нам информацию об их характере. Это как в жизни: по фигуре, походке и одежде человека обычно можно догадаться о его поле, возрасте и даже профессии. В случае числовых данных мы говорим о распределении.

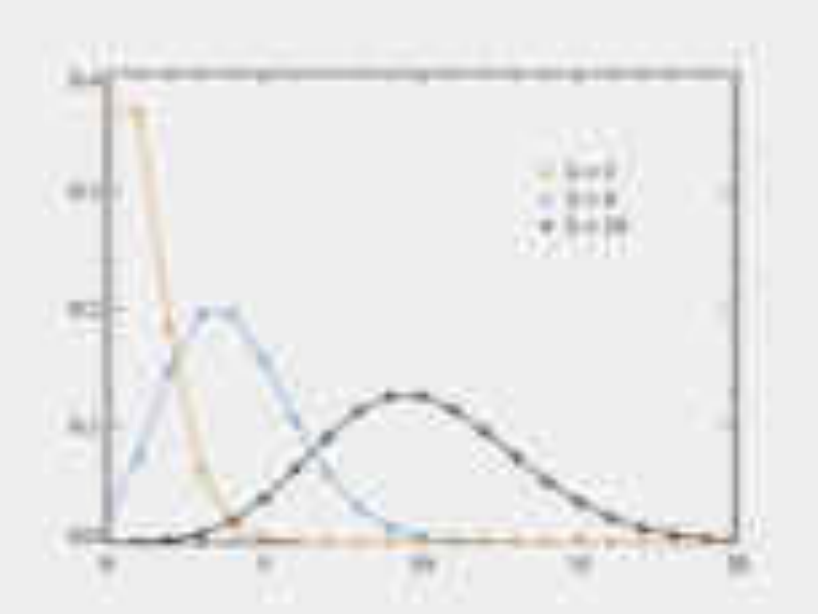
Нормальное распределение

Нормальное распределение описывает процессы, где результат является суммой многих случайных величин, каждая из которых слабо зависит от другой и вносит сравнительно небольшой вклад.

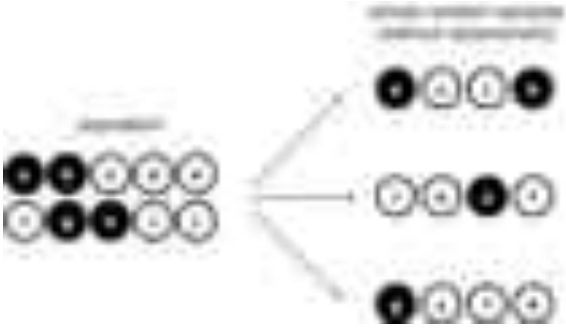


Распределение Пуассона

Распределение Пуассона тоже часто встречается в работе дата-сайентистов и аналитиков: это число событий за какой-то промежуток времени — при условии, что события независимы друг от друга и имеют некоторый порог интенсивности.



Семплирование



Семплирование — это группа статистических методов и приёмов, отвечающих за релевантность выборки. С помощью семплирования мы формируем нашу выборку так, чтобы она наилучшим образом отражала свойства генеральной совокупности.

Визуализация данных

Визуализация данных — это представление данных в виде, который обеспечивает наиболее эффективную работу человека по их изучению. Визуализация данных находит широкое применение в научных и статистических исследованиях (в частности, в прогнозировании, интеллектуальном анализе данных, бизнес-анализе), в педагогическом дизайне для обучения и тестирования, в новостных сводках и аналитических обзорах. Визуализация данных связана с визуализацией информации, инфографикой, визуализацией научных данных, разведочным анализом данных и статистической графикой.

По цели представления данных визуализация делится на презентационную (англ. «presentation», «explanation») и исследовательскую (англ. «exploration»). Презентационная визуализация предназначена для представления данных некоторой

аудитории (например, в рамках научной работы, доклада или аналитического обзора в новостях). Исследовательская визуализация предназначена для анализа и обработки набора данных, например, с целью обнаружения закономерностей в них.

Визуализация как этап анализа данных

Подсистема визуализации данных является важной составной частью качественных систем интеллектуального анализа данных, особенно ориентированных на обработку больших объёмов информации. В системах бизнес-аналитики визуализация может использоваться на всех этапах процесса обработки данных:

- Визуализация исходных данных. Этот этап полезен для оценки степени соответствия ожиданиям и пригодности данных к анализу, выдвижения гипотез о закономерностях и необходимых процедурах первичной обработки.
- Визуализация выборки, загруженной в систему обработки.
- Визуализация результатов первичной обработки.
- Визуализация промежуточных результатов.
- Визуализация окончательных результатов.

Классификация переменных

Переменные статистического анализа можно классифицировать:

1. По роли в исследовании
    - результирующие (выходные, отклик, прогнозируемые, эндогенные)
    - объясняющие (входные, предсказывающие, предикторные, экзогенные)
    - скрытые (латентные, остаточные)
2. По вероятностной природе
    - детерминированные
    - случайные
3. По шкале измерений
    - количественные
    - порядковые (качественные, ординальные)
    - номинальные (классификационные

В основу классификации по шкале измерений положен объем допустимых операций над числами.

Количественные признаки измеряются в некоторой шкале (длина, вес, скорость) и выражаются целыми или дробными числами.

Порядковые признаки можно естественным образом упорядочить по их значениям, они имеют градации, которые выражаются чаще всего словами, образующими упорядоченный ряд (плохо, удовлетворительно, хорошо, отлично) или целыми числами. Однако с этими значениями в отличие от количественных нельзя производить обычных математических действий: суммировать, умножать, делить и т.п.

Номинальные признаки отражают деление объектов на различные классы (пол, возраст, профессия), но не несут никакой информации об упорядоченности классов или об их количественных соотношениях. Чаще всего признаки задаются словами, но возможно использование также специальных симво-лов и чисел. Но эти символы и числа являются лишь метками, про которые можно сказать только, что они либо одинаковы, либо различны.

Лекция 2. Исследование зависимостей. Корреляционный анализ

ВВЕДЕНИЕ В МНОГОМЕРНЫЙ СТОХАСТИЧЕСКИЙ АНАЛИЗ

Довольно часто невозможно дать оценку какому-либо явлению или состоянию по одному признаку. Например:

- оценить способности выпускника, по одной оценке, необходимо знать все или большинство оценок аттестата или диплома;
- точность выстрела можно оценить только по двум координатам.

При этом исследовать признаки необходимо не изолированно друг от друга, а с учетом характера и структуры их взаимосвязи. Что понимается под многомерной структурой? Речь идет о множестве объектов:

$$\{O_1, O_2, \ldots, O_n\},$$

каждый из которых может характеризоваться многомерным вектором наблюдений. Результаты их статистического обследования представляются, как правило, в одной из двух форм:

1. Таблицы (матрицы) «объект-признак» (или «объект-свойство»).
2. Таблица (матрица) попарных сравнений

Таблица «объект-признак»

Примеры признаков: вес, длина, цвет, профессия, цена, пол, наличие и отсутствие симптома и т.д. Примеры объектов: люди, изделия, место рождения и т.д. Таблицы такого вида (табл.1) принято называть таблицей экспериментальных данных (ТЭД). Это название следует трактовать более широко, говоря не только об экспериментальных данных, а о данных научного исследования вообще.

Таблица 1.

Объект	Признаки (свойства, параметры, показатели)					

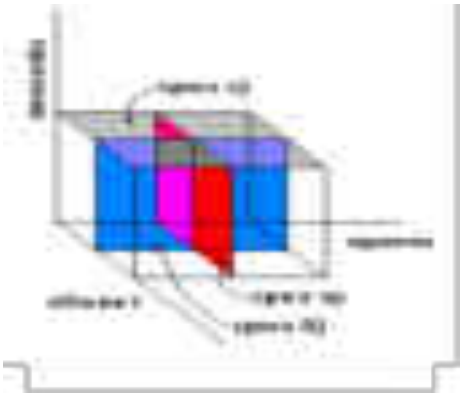
Если исследователь вводит в анализ еще одну составляющую – время, то мы переходим к кубу данных, то есть к трехмерному пространству (см. рис.1): «признак - объект – время» (p-o-t).

Фиксируя одну из трех координат, мы получим плоские двухмерные срезы куба данных:

а) при фиксации времени – матрицу объект-признак.

б) при фиксации объекта – матрицу признак-время, это временной срез данных.

в) при фиксации признака – матрицу объект-время, например, фиксируется цена на акции. Тогда объекты – виды акций, строки соответствуют временным интервалам (дни, недели и т.д.).



Представление информации в виде куба является грубым упрощением. В частности, можно предположить, что исследователя, помимо p, o и t интересуют условия получения информации. Условий может быть много, и исследователь может ставить своей задачей изучение изменений в пространстве p-o-t в зависимости от изменения условий.

Рис. 1. Куб данных

Таким образом может быть получен четырехмерный куб данных. Добавьте еще одну координату – способ измерения информации, и вы получите пятимерный куб и т.д.

Введем следующие обозначения:

X – случайная величина;

x_i – i-ая реализация случайной величины;

X = $(X^{(1)},X^{(2)},\ldots,X^{(p)})$ – случайный p-мерный вектор (многомерная случайная величина, многомерный признак);

$\mathbf{x}_i = (x^{(1)}_i,x^{(2)}_i,\ldots,x^{(p)}_i)$– i-ая реализация случайного вектора;

$\mathbf{X}^{p\times n}$ – матрица;

$\mathbf{X}^{1\times n}$ – одна строка таблицы данных.

Пусть **X**=$(X^{(1)},X^{(2)},\ldots,X^{(p)})$ – случайный p-мерный вектор, где координаты $X^{(1)},\ldots, X^{(p)}$ есть случайные величины (цена, размер, вес, пол и т.п.). Тогда реализация вектора **X** (уже не случайная величина) $x_i = (x_i^{(1)}, x_i^{(2)},\ldots, x_i^{(p)})$ и есть i-ая строка ТЭД. После n реализаций вектора **X** получим n неслучайных p-мерных векторов $\mathbf{X}^{p\times n}=(\mathbf{x_1},\mathbf{x_2},\ldots,\mathbf{x_n})^T$, которые называют многомерной выборкой и которые образуют двумерную матрицу данных. Таким образом, матрица данных – это одна выборка случайного многомерного вектора **X**=$(X^{(1)}, X^{(2)},\ldots,X^{(p)})$ с объемом выборки, равным n.

Таблица (матрица) попарных сравнений

$$A = \begin{vmatrix} a_{11} & a_{12} & \\ a_{21} & \ldots & \\ & & a_{nn} \end{vmatrix}$$

где a_{ij} – результат сопоставления объектов O_i и O_j (мера сходства или различия, мера их связи, геометрические расстояния, отношения предпочтения и т. д.).

Априорно можно предположить, что существует небольшое (по сравнению с “p”) число определяющих факторов, с помощью которых могут быть точно описаны как наблюдаемые характеристики анализируемых объектов (т.е. все элементы $x_i^{(k)}$ и a_{ij} матриц **X** и **A**) и характер связей между ними, так и искомая классификация самих объектов.

ИССЛЕДОВАНИЕ ЗАВИСИМОСТЕЙ

Функционирование многих систем можно задать описанием зависимостей между входными и выходными переменными. В сложной системе учесть влияние на результат всех переменных практически невозможно.

Поэтому выходные переменные всегда будут подвержены случайному неконтролируемому разбросу, обусловленному действию неучтенных факторов.

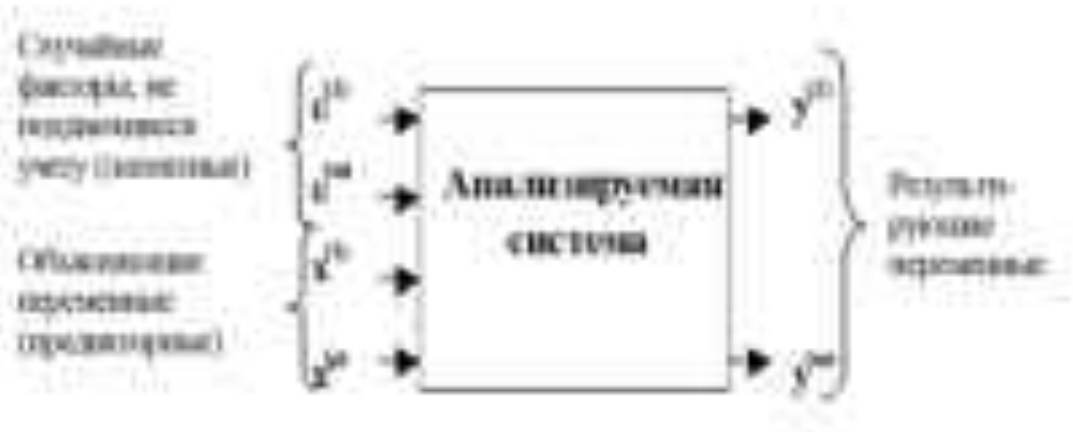


Рис.2 Модель системы

Представим исследуемую систему (рис.2) в виде «черного ящика», на вход которого воздействуют два вектора: предиктор $X=(x^{(1)},\ldots,x^{(p)})$ и случайный вектор $E=(\varepsilon^{(1)},\ldots,\varepsilon^{(m)})$. Результат представим вектором $Y=(y^{(1)},\ldots,y^{(m)})$. Тогда вместо модели

$$\boldsymbol{Y}= f(X^{(1)},X^{(2)},\ldots,X^{(k)}),$$

учитывающей действие всех переменных и где k слишком велико для практических целей, рассматривается зависимость с настолько малым числом объясняющих переменных **X**, что **Y** можно записать

$$\boldsymbol{Y=f(X, E).}$$

В этой модели суммарный эффект от воздействия остальных факторов отражает векторная переменная **E**. Для скалярной переменной Y можно записать

$$Y=f(\boldsymbol{X},\varepsilon).$$

В этих условиях задача исследователя состоит в том, чтобы по результатам N измерений исследуемых переменных **X** построить такую векторную функцию f**(X)**, которая позволила бы наилучшим образом восстанавливать значения Y по заданным значениям **X**.

Какова конечная цель статистического исследования зависимостей? С этого вопроса должно начинаться любое статистическое исследование. Можно выделить три основных типа прикладных целей:

1. Выявление причинных связей между **Y** и **X**, проникновение в «физический механизм» изучаемых связей.

2. Прогноз неизвестных значений индивидуальных или средних результирующих показателей по заданным значениям **X** соответствующих переменных.

3. Установление самого факта наличия (или отсутствия) статистической связи между **Y** и **X**.

КОРРЕЛЯЦИОННЫЙ АНАЛИЗ

Корреляционным анализом называется анализ структуры и тесноты статистической связи между исследуемыми переменными. Корреляционный анализ (КА) включает в себя следующие этапы:

1. Обоснованный выбор характеристики, измеряющей степень тесноты статистической связи.
2. Оценка числового значения выбранной характеристики.
3. Проверка гипотезы о том, что полученное числовое значение характеристики действительно свидетельствует о наличии статистической связи.
4. Исследование структуры связей между компонентами исследуемого многомерного признака.

Представим все виды характеристик связи классификационной схемой, представленной табл.3.

Зависимости между переменными могут носить как причинный, так и статистический характер (рис.8). Причинная связь – это связь, при которой осуществление одного события является достаточным условием для осуществления другого события.

Рис. 8. Обозначение статистической связи

Статистическая или корреляционная связь устанавливает лишь сам факт взаимосвязи, но ничего не говорит о направлении причинно-следственной связи. Известно, что имеется сильная положительная корреляционная взаимосвязь между общим числом больных и количеством больниц. Такую корреляционную связь можно объяснить наличием общего фактора – численности населения, влияющего на обе переменные. Таким образом, даже из тесной зависимости случайных величин не всегда следует их причинная взаимообусловленность.

Таблица 2

			Ковариация

Характеристики статистической связи	количественных переменных	Анализ парных связей	Коэффициент корреляции
			Корреляционное отношение
		Анализ множественных связей	Коэффициент множественной корреляции
	порядковых переменных	Анализ парных связей	Ранговый коэффициент корреляции Спирмена
			Ранговый коэффициент корреляции Кендалла
		Анализ множественных связей	Коэффициент конкордации Кендалла
	номинальных переменных	Анализ множественных связей	Таблица сопряжённости

КАК ПОЯВЛЯЮТСЯ СТАТИСТИЧЕСКИЕ ВЗАИМОСВЯЗИ

Рассмотрим на простом примере влияние на прогноз возмущающих неконтролируемых воздействий. Пусть система определена линейным уравнением Y=0,5X. Это функциональная зависимость (рис.3а).

Что произойдет, если в систему войдут несколько более сложные причинные отношения

$$X \xrightarrow{0,5} Y \xleftarrow{0,25} E.$$

Добавление новой переменной ведет к уменьшению степени чисто линейного соответствия между X и Y. Точки на диаграмме уже не ложатся строго на прямую и больше нельзя сделать точного прогноза значений Y для нового случая (рис.3б).

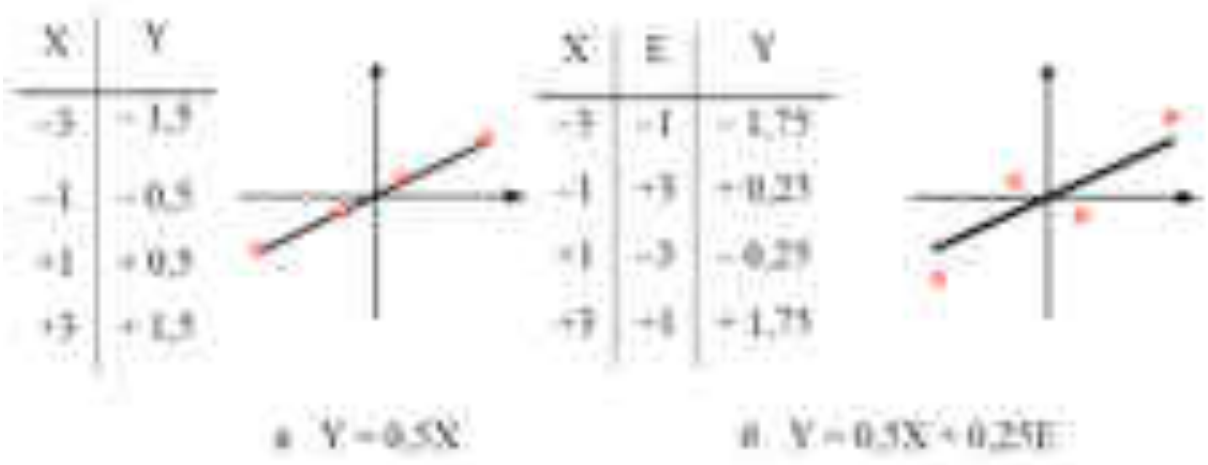


Рис. 3. Влияние возмущений на прогноз

Это показывает, каким способом возмущения искажают прогноз. В этом случае у Y имеется дополнительный разброс, производимый E, и поэтому X объясняет меньшую долю общего разброса Y. Можно сказать, что во втором случае Y имеет большую дисперсию, чем в первом.

Таким образом, наличие множества возмущающих неконтролируемых воздействий ведет к отклонению от функциональных зависимостей и появлению статистических связей.

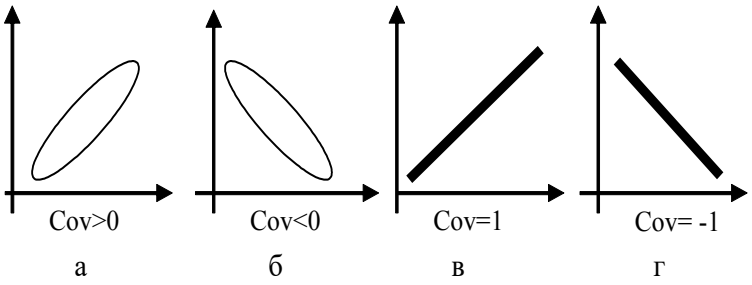
АНАЛИЗ ТЕСНОТЫ СВЯЗИ МЕЖДУ КОЛИЧЕСТВЕННЫМИ ПЕРЕМЕННЫМИ

Диаграммы рассеивания

О наличии или отсутствии статистической связи можно судить по виду диаграммы рассеивания. Представленные на диаграмме данные часто могут быть вписаны в геометрическую фигуру, имеющую форму эллипса (рис 10а и 10б). В этом случае можно предположить наличие линейной статистической связи. Ориентация эллипса отражает положительную (с ростом X растет и Y) или отрицательную связь. В общем случае, чем уже эллипс, тем выше степень тесноты связи. На рис. 11а отображена положительная корреляционная связь, характеризующаяся правым поворотом эллипса, рис.10б отображает отрицательную корреляционную связь.

Сужение или расширение эллипса ведет соответственно к двум крайним случаям:

Рис. 10. Наличие ковариационной связи



1. Линия – функциональная линейная зависимость (рис. 10в и 10г) или отсутствие зависимости (рис. 11а).

2. Круг – отсутствие зависимости (рис. 11б).

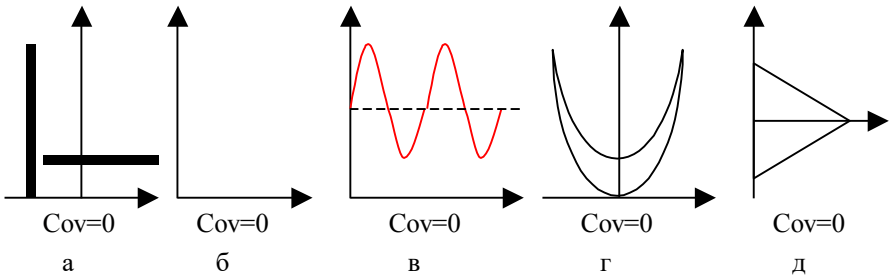


Рис. 11. Отсутствие ковариационной

На рис 11в отображен случай нелинейной функциональной зависимости с нулевой ковариацией. На рис 11г отображен случай нелинейной статистической зависимости с нулевой ковариацией.

Ковариация

Диаграммы рассеивания полезны для установления факта существования соответствия между двумя переменными и его формы. Однако для точного исследования необходимо оценить степень этого соответствия. Наиболее подходящая мера для этого – ковариация. Ковариация есть математическое ожидание произведения двух центрированных случайных величин:

$$Cov(X,Y) = K_{XY} = E(\dot{X}\cdot\dot{Y}) = E((X - EX)(Y - EY)). \quad (1)$$

Если X=Y, имеем:

$$Cov(X,X) = K_{XX} = E(\dot{X}\cdot\dot{X}) = E(X - EX)^2 = DX, \quad (2)$$

то есть ковариация случайной величины равна ее дисперсии.

Выборочная оценка ковариации обычно вычисляется по формуле:

$$\hat{C}ov(X,Y) = \hat{K}_{xy} = \frac{1}{N-1}\sum_{j=1}^{N}(x_j - \bar{x})(y_j - \bar{y}). \quad (3)$$

Дадим геометрическую интерпретацию ковариации. Предположим, что мы имеем два ряда наблюдений, представленных на диаграмме рассеяния (рис.12).

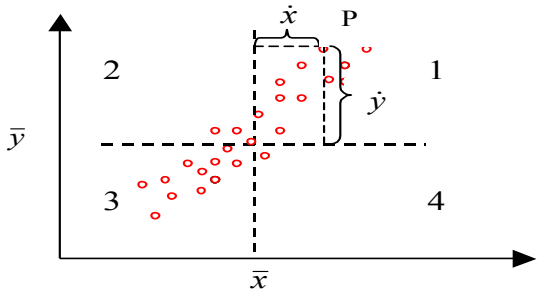


Рис. 5 Интерпретация ковариации

Рис. 12. Интерпретация ковариации

Разобьем диаграмму на четыре квадранта с помощью перпендикуляров к осям, проведенных из точки с координатами $\bar{X}$ и $\bar{Y}$. Тем самым для любой точки P с координатами (x_i , y_i) будут определены отклонения $\dot{x}_i = x_i - \bar{x}_i$ и $\dot{y}_i = y_i - \bar{y}_i$.

Такая процедура носит название центрирования (рис.13). В результате этого происходит перенос всех точек диаграммы рассеяния в новую систему координат $\dot{X}\dot{Y}$ с центром в начале координат.

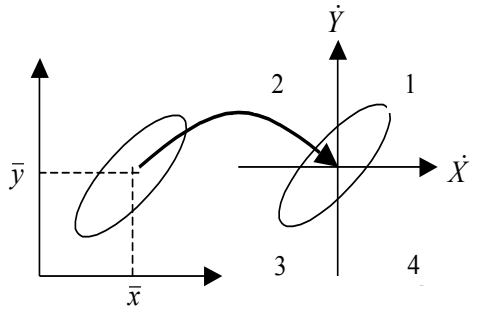


Рис. 13. Перенос координат – центрирование

Из рассмотрения диаграммы очевидным образом следует, что:

- для всех точек квадранта 1 произведение $\dot{x}_i\dot{y}_i$ положительно;
- для всех точек квадранта 2 произведение $\dot{x}_i\dot{y}_i$ отрицательно;
- для всех точек квадранта 3 произведение $\dot{x}_i\dot{y}_i$ положительно;
- для всех точек квадранта 4 произведение $\dot{x}_i\dot{y}_i$ отрицательно.

Следовательно, величина $\sum \dot{x}_i\dot{y}_i$ может служить мерой зависимости между переменными X и Y. Если зависимость положительна, так что большая часть точек лежит в 1 и 3-м квадрантах, то $\sum \dot{x}_i\dot{y}_i$ становится положительной. Если, наоборот, зависимость отрицательна, так что большая часть точек лежит во 2-м и 4-м

квадрантах, то $\sum \dot{x}_i \dot{y}_i$ становится отрицательной. Наконец, когда нет связи между X и Y, точки рассеиваются по всем четырем квадрантам и $\sum \dot{x}_i \dot{y}_i$ становится очень малой.

Теперь можно сделать выводы из анализа графиков рис. 10 и рис. 11:

1. Ковариационная связь будет положительной, если с ростом X возрастает и Y.

2. Ковариационная связь будет отрицательной, если с ростом X уменьшается Y.

3. Ковариационная связь будет отсутствовать, если экспериментальные данные ложатся на прямую, параллельную одной из координат.

4. Ковариация как мера статистической взаимосвязи пригодна для линейных связей.

5. Для функциональной линейной связи ковариация максимальна, для функциональной нелинейной связи может быть равной нулю (рис. 11в).

6. Нулевая ковариация не всегда означает отсутствие статистической связи. Для нелинейной статистической связи она может быть равной нулю (рис. 11г).

7. Ковариационная связь между случайными переменными – это частный случай их статистической связи вообще. Например, на рис. 11д ковариационная связь отсутствует, однако статистическая связь существует: с ростом X существенно уменьшается дисперсия – т. е. существует дисперсионная связь.

Коэффициент парной корреляции

Ковариация обладает двумя недостатками:

- ее численное значение может быть произвольно увеличено в результате добавления «отдаленных наблюдений»;
- оно может быть произвольно изменено путем выбора единиц измерения переменных X и Y.

Эти недостатки можно исправить, выразив отклонения в единицах стандартных отклонений, усреднение которых даст нам смешанный момент или коэффициент корреляции.

Пусть случайные векторы X и Y имеют двумерное нормальное распределение $N(\mu_1,\mu_2,\sigma_1,\sigma_2)$. Тогда простой (или смешанный) момент между X и Y есть коэффициент парной корреляции Пирсона

$$Cor(X,Y) = \rho = \frac{cov(X,Y)}{\sigma_x \sigma_y}. \quad (4)$$

Для случайной выборки получим оценку коэффициента корреляции

$$r = \hat{\rho} = \frac{cov(X,Y)}{s_x s_y} \quad (5)$$

Коэффициент корреляции является мерой тесноты только для линейной статистической связи между анализируемыми признаками. Только в случае совместной нормальной распределенности исследуемых случайных величин X и Y коэффициент корреляции r имеет четкий смысл как характеристика степени связи между ними. Если же априори допускается возможность отклонения от линейного вида зависимости, то можно построить примеры, когда, несмотря на r=0, исследуемые переменные оказываются связанными чисто функциональным соотношением. Поэтому о величинах, для которых r=0, обычно говорят, что они не коррелированы, и только после дополнительного анализа, можно сказать, следует ли отсюда их независимость. И, наоборот, из высокой степени коррелированности величин при сильных отклонениях распределения случайных величин X и Y от нормального не следует их столь же тесная взаимосвязь.

Как видно из рис.14, после добавления нового объекта коэффициент корреляции значительно возрос. Дело здесь в том, что последний объект является аномальным, резко выделяющимся, так что всю совокупность наблюдений уже нельзя считать выборкой из одной и той же нормальной генеральной совокупности.

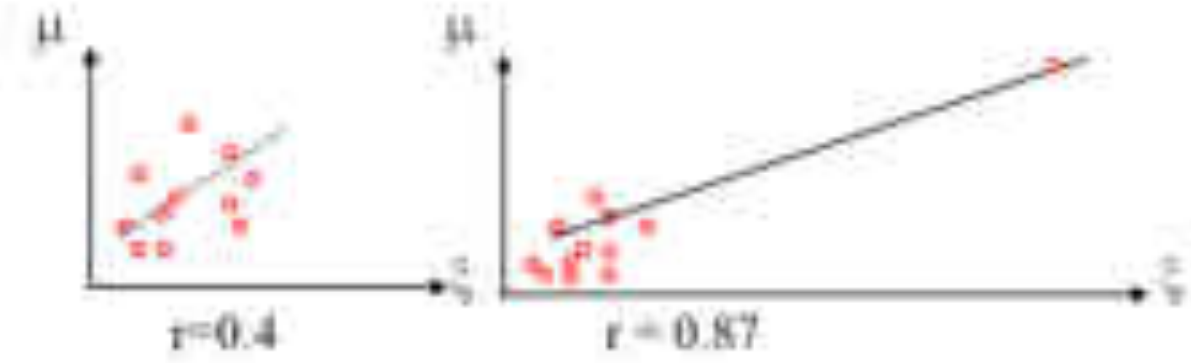
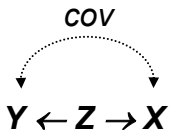


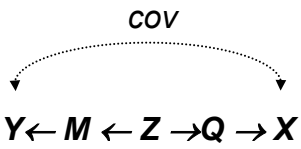
Рис. 14. Отклонение от нормального распределения

Но даже из тесной зависимости случайных величин не всегда следует их причинная взаимообусловленность. Рассмотрим примеры так называемой «ложной» корреляции. Например, переменные X и Y могут быть статистически коррелированы, если

- переменные зависят от входной переменной:

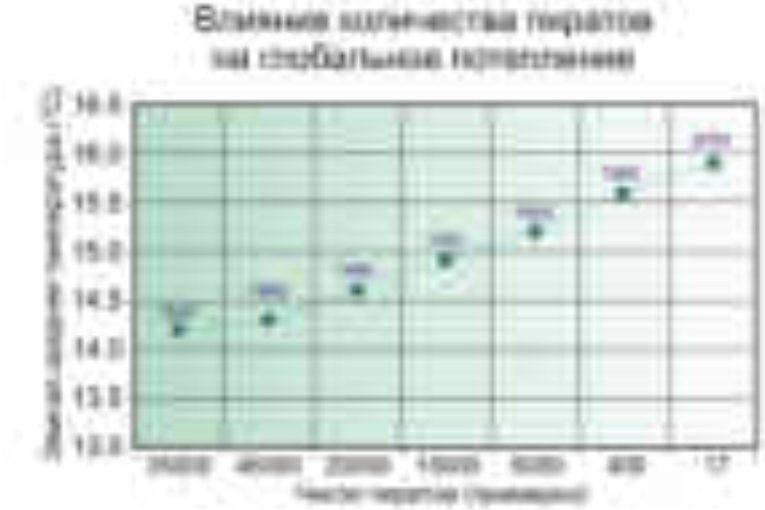


- существует координация между влияющими на них переменными



Это приводит к необходимости введения таких измерителей статистической связи, которые были бы очищены от опосредованного влияния других переменных, давали бы оценку степени тесноты связи между переменными X и Y при условии, что значения остальных переменных зафиксированы на постоянном уровне. Такой коэффициент корреляции называется частным, очищенным.

Например, можно наблюдать зависимость между ростом человека и длиной волос: чем выше человек, тем короче его волосы. Однако эта корреляция ложная, так как обе переменные «Рост» и «Длина волос» зависят от одной входной переменной «Пол»: женщины в среднем ниже мужчин и носят длинные волосы. Исключив влияние это переменной (зафиксировав ее значение), мы обнаружим отсутствие стохастической взаимосвязи.



Частные коэффициенты корреляции

Характеризуют тесноту связи между двумя случайными величинами при исключении влияния остальных случайных величин. Частный коэффициент корреляции величин X_1 и X_2, входящих в систему $\{X_1, X_2, X_3, \ldots X_n\}$ относительно величин X_3, X_4, …, X_n обозначается через $r_{12.34\ldots n}$. Его числовое значение находится из формулы

$$r_{12.34\ldots n} = -\frac{P_{12}}{\sqrt{P_{11}P_{22}}}, \qquad (6)$$

где P_{11}, P_{12}, ..., P_{22} – миноры детерминанта квадратной корреляционной матрицы (19).

Коэффициент множественный корреляции

Коэффициент корреляции Пирсона характеризует связь двух случайных величин. Коэффициент множественной корреляции используется для описания системы случайных величин $\{X^{(1)}, X^{(2)}, X^{(3)}, \ldots X^{(p)}\}$. Он служит характеристикой корреляции между величиной $X^{(1)}$ с одной стороны, и всей совокупностью величин $\{X^{(2)}, X^{(3)}, \ldots X^{(p)}\}$ с другой. Например, мы хотим выяснить, существует ли связь между ростом ($X^{(1)}$) с одной стороны и весом ($X^{(2)}$), средним ростом родителей($X^{(3)}$) с другой. Формально он определен для любой многомерной системы. Получим его из следующих соображений.

Пусть зависимость задана моделью регрессии

$$Y=F(\boldsymbol{X}) + E$$

где $F(\boldsymbol{X}) = \hat{Y}$ – регрессия; E – остаток, ошибка.

Определим остаточную дисперсию как средний квадрат расстояний между наблюдаемыми и ожидаемыми значениями

$$\sigma_{\varepsilon}^2 = \mathrm{E}(\mathrm{Y} - \hat{\mathrm{Y}})^2. \qquad (7)$$

Принимая во внимание, что ошибки прогноза не коррелированы с независимой переменной X, можно прийти к важному выводу:

$$DY = D[F(X)] + DE \qquad (8)$$

или

$$\sigma_Y^2 = \sigma_{\hat{Y}}^2 + \sigma_{\varepsilon}^2,$$

т.е. общая дисперсия предсказываемой переменной равна дисперсии предсказываемых значений плюс дисперсия остатков. Это означает, что полная вариация исследуемой зависимой переменной Y складывается из

контролируемой нами вариации функции регрессии и из не поддающейся нашему контролю вариации остаточной, случайной компоненты.

Какая доля общей дисперсии σ_y^2 обуславливается изменчивостью функции регрессии F(X)? Ответ на этот вопрос приводит к понятию множественной корреляции R_{XY}:

$$R_{xy}^2 = \frac{\sigma_{\hat{y}}^2}{\sigma_y^2} = \frac{\sigma_y^2 - \sigma_\varepsilon^2}{\sigma_y^2} = 1 - \frac{\sigma_\varepsilon^2}{\sigma_y^2}. \qquad (9)$$

Квадрат коэффициента множественной корреляции (КМК) часто называют коэффициентом детерминации. Если коэффициент детерминации определяет долю дисперсии, которая объясняется изменением F(X), то оставшаяся доля дисперсии 1 – R_{XY}^2 объясняется воздействием неконтролируемой, случайной остаточной компоненты (помехи).

Установим связь КМК с коэффициентом парной корреляции Пирсона. Для этого рассмотрим ситуацию, когда для заданного значения X=x получено множество значений Y. Их распределение, называемое условным распределением Y при данном X=x, есть нормальное одномерное распределение с условным средним

$$\mu_{Y|X} = \mu_Y + \frac{\sigma_{XY}}{\sigma_X^2}(X - \mu_X) \qquad (10)$$

и условной дисперсией

$$\sigma_{Y|X}^2 = \sigma_Y^2(1 - \rho^2). \qquad (11)$$

Легко понять, что дисперсия $\sigma_{Y|X}^2$, обусловленная влиянием помехи, есть не что иное, как остаточная дисперсия σ_ε^2. Простой расчет приводит к выводу, что в нормальной двумерной схеме множественный коэффициент корреляции совпадает с парным коэффициентом корреляции. Отметим, что приведенная выше формула (11) может быть использована для расчета остаточной дисперсии и при множественной регрессии. В этом случае

$$\sigma_\varepsilon^2 = \sigma_Y^2(1 - R_{XY}^2). \qquad (12)$$

Пример. Что означает фраза «коэффициент корреляции между ростом (X) и весом (Y), оцененный по выборке из 100 человек, равен 0,8»? Спроектируем данные на плоскость признаков РОСТ и ВЕС. Поскольку это совершенно

разные физические единицы, оба признака предварительно пронормируем. У нормированных признаков среднее станет нулевым, а стандартное отклонение – единичным. Если бы взаимосвязь признаков была точно линейной, то все объекты лежали бы точно на прямой Y =aX. Однако на самом деле каждому значению роста соответствует не только одно значение веса, а некоторое распределение весов людей, которое имеет ненулевую дисперсию $\sigma^2_{Y|X} = \sigma^2_{\varepsilon}$. Коэффициент корреляции дает возможность узнать, какую долю составляет стандартное отклонение этого распределения от стандартного отклонения признака «вес»:

$$\sigma^2_{Y|X} = \sigma^2_Y(1-\rho^2) = 1\cdot(1 - 0.8^2) = 0.36; \qquad \sigma_{Y|X} = 0.6.$$

Другая интерпретация значения коэффициент корреляции в данном примере – объяснение различий в значениях признака «вес». Из-за наличия не функциональной, а статистической зависимости дисперсия веса объясняется дисперсией роста только на $r^2 \times 100\% = 64\%$.

Лекция 3. Регрессионный анализ

РЕГРЕССИОННЫЙ АНАЛИЗ

Поставим задачу регрессионного анализа в общем виде: по результатам наблюдений количественных переменных

$$(x_i^{(1)}, \ldots, x_i^{(p)}, y_j^{(j)}); \quad i=1, \ldots, N; \quad j=1,\ldots,m$$

в условиях действия случайных факторов **E** найти и оценить статистическую связь между **X** и **Y**. Решение этой задачи и составляет содержание регрессионного анализа (РА).

Связь между **X** и **Y** представляется с помощью математической модели, которая называется уравнением регрессии. Как уже отмечалось выше, эффект, вызываемый многими из переменных, настолько слаб, что даже при наличии достоверных данных его статистическая оценка трудна и ненадежна. Поэтому влияние этого эффекта учитывается переменной ε:

$$y=F(\boldsymbol{X})+\varepsilon = \hat{Y} + \varepsilon, \qquad (1)$$

где y – скалярная результирующая переменная (m=1),

$F(\boldsymbol{X}) = \hat{Y}$ – функция регрессии.

Значок ^ будем применять для обозначения предсказанного наиболее вероятного значения случайной величины.

Регрессией в прямом смысле называется функция вида

$$y = b_0 + \sum_{i=1} b_i x_i + \sum_{i<j} b_{ij} x_i x_j + \sum_{i=1} b_{ij} x_i^2 + ... + \varepsilon \ , \qquad (2)$$

где b_0, b_i, b_{ij}- коэффициенты регрессии;

ε - невязка (ошибка, отклонение), обусловленная недостаточной пригодностью модели и ошибкой эксперимента. Эти причины обычно являются смешанными.

Регрессия (2) называется множественной. Наибольший практический интерес представляет множественная *линейная* регрессия, для которой b_{ij} =0.В том случае, если исследуется влияние одной переменной на результат эксперимента, то выражение (2) упрощается к виду

$$y=b_0+b_1x+\varepsilon. \qquad (3)$$

Выражение (3) представляет собой линейную одномерную регрессию. Кроме того, часто используется нелинейное представление регрессии (логарифмическое, экспоненциальное, гиперболическое и другое задаваемое исследователем). В этом случае переменная X подвергается соответствующему нелинейному преобразованию.

В понятие регрессии может быть вложено два смысла.

1. Регрессия в первом смысле: среднее по столбцам (устаревший взгляд).
2. Подбор аналитического выражения для линии регрессии.

Рассмотрим простой пример, иллюстрирующий смысл регрессии. Пусть изучается связь между весом $X^{(1)}$ и ростом $X^{(2)}$ мужчин одного возраста в данном регионе. Возможны три постановки задачи этого исследования:

1. Оценить силу связи, т.е. вычислить r ($x^{(1)}$,$x^{(2)}$) .
2. Узнать, каков наиболее вероятный рост людей данного веса, т.е. найти регрессию $X^{(2)}$ от $X^{(1)}$.
3. Узнать, каков наиболее вероятный вес людей данного роста, т.е. найти регрессию $X^{(1)}$ от $X^{(2)}$.

Получение регрессии включает три этапа:

1. Задается вид уравнения регрессии, включающего, кроме X и Y, несколько неизвестных параметров, которые находятся по опытным данным.
2. Определяются неизвестные коэффициенты, входящие в уравнение регрессии.
3. Проводится статистический анализ регрессии.

Рассмотрим выполнение этих этапов на примере простой линейной регрессии.

ВЫБОР МОДЕЛИ РЕГРЕССИИ

Первый и одновременно один из главных вопросов, который должен решать пользователь, приступая к описанию результирующей переменной Y методом регрессионного анализа – выбор модели (вида функции) регрессии, т.к. собственно регрессионный анализ оценивает лишь параметры этой модели. Основная

информация для выбора модели должна быть получена на этапе качественного решения, и заключаться в отборе независимых переменных, наиболее сильно влияющих на зависимую результирующую переменную, а также в проверке линейности взаимосвязи зависимой и независимой переменных.

Обычно уравнение регрессии задается (постулируется) в виде полинома. Наиболее распространенными и удобными регрессионными моделями можно считать следующие:

- Модель первого порядка по X, линейная относительно коэффициентов:

$$\hat{Y} = \beta_0 + \beta_1 X . \quad (4)$$

- Модель второго порядка по X, линейная относительно коэффициентов:

$$\hat{Y} = \beta_0 + \beta_1 X + \beta_2 X^2 . \quad (5)$$

На практике могут использоваться любые другие модели, например:

$$\hat{Y} = \alpha e^{\beta X}; \quad \hat{Y} = \alpha X^{\beta}; \quad \hat{Y} = \alpha + \beta * 1/X , \quad (6)$$

в том числе и нелинейные относительно коэффициентов, например:

$$\hat{Y} = \beta_0(x) + \beta_1(x) X. \quad (7)$$

Аппарат линейного регрессионного анализа наиболее разработан, линейные зависимости наиболее просты и понятны. Поэтому стремятся использовать преобразования как X, так и Y (а возможно и обеих этих величин), приводящих изучаемую закономерность именно к линейному виду. Современные статистические программы позволяют получить корреляции и диаграммы рассеяния для разнообразных комбинаций преобразований X и Y, например (X, logY), (logX,Y), (logY, logX) , ($\sqrt{X}$, logY), и т.д. Преобразование, для которого получается наибольшее по абсолютной величине значение r_{xy}, будет тем преобразованием, которому соответствует наиболее сильная линейная зависимость.

Проще всего такие преобразования выполняются, если полученная линейная зависимость содержит не более двух постоянных коэффициентов.

Отметим, что неудачно выбранная модель может вызвать лишние затраты времени, хотя и не приводит к принципиальным ошибкам в результате, поскольку адекватность модели проверяется на последнем этапе и плохая модель не используется. Поэтому на этапе выбора вида регрессии надо использовать всю имеющуюся информацию. Как правило, выбор выполняется на основе опыта, интуиции, исходя из содержания задачи, т.е. выбор лежит вне статистических соображений. Например, в регрессионной зависимости *Вес коровы=f(параметры коровы)* была использована формула расчета объема цилиндра, что удачно «легло» на экспериментальные данные.

Предположим, что имеется выборка парных наблюдений $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$. Существует два способа получения такой выборки:

1. При каждом фиксированном значении аргумента $X=x_i$ (i=1,…,k) производится несколько наблюдений результирующей переменной Y: $y_{i1},y_{i2},\ldots,y_{ini}$. При таком подходе только Y является случайной величиной.
2. Из генеральной совокупности случайным образом выбираются N объектов и у каждого из них измеряются переменные X и Y. Здесь случайными являются обе величины X и Y. Преимущество этого метода получения выборки заключается в том, что мы можем сделать статистические выводы относительно коэффициента корреляции между X и Y, в то время как при первом методе этого сделать нельзя.

Независимо от способа получения выборки для определения существования и степени линейной зависимости необходимо выполнить два предварительных шага:

- Анализ диаграммы рассеяния. Допустимо ли предположение о линейной зависимости? Есть ли необходимость в преобразованиях зависимости к линейному виду?
- Вычисление выборочного коэффициента корреляции.

Если предполагается линейная зависимость между X и Y, то теоретическая модель задается уравнением

$$y_i=\beta_0+\beta_1 x_i+\varepsilon_i, \quad i=1,\ldots,n \tag{8}$$

где y_i, x_i – значения Y и X в i-м опыте;

β_0,β_1 – неизвестные параметры, “истинные” коэффициенты регрессии, соответствующие модели

$$Y_{ист}=\beta_0+\beta_1 X \tag{9}$$

ε_i – ошибка, объясняющая отличие наблюдаемого значения от “истинного” и обусловленная воздействием на Y случайных факторов.

Для линейной модели принят ряд гипотез относительно свойств случайного возмущения ε. Эти гипотезы относятся к его среднему, дисперсии и ковариации:

$E[\varepsilon_i]=0$	i=1,…,n	– среднее равно нулю;
$D[\varepsilon_i]=\sigma^2_\varepsilon$	1,…,n	– дисперсия постоянна и однородна;
$Cov(\varepsilon_i,\varepsilon_j)=0$	$(i\neq j)$	– ошибки не коррелированны.

Если гипотеза о линейной зависимости справедлива и предположения об ошибке ε удовлетворяются, то графическая модель выглядит так:

Для каждого значения $X=x_i$ имеется распределение Y (не обязательно нормальное) со средним значением $\beta_0+\beta_1 x_i$ и дисперсией $\sigma=2\varepsilon$, i=1,…,n.

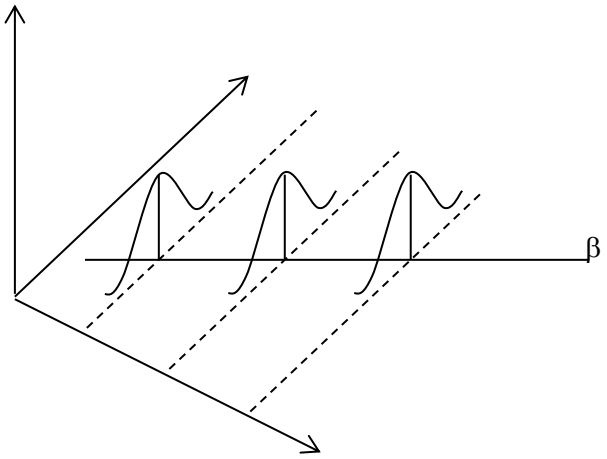


Рис 1. Модель линейной регрессии

ОЦЕНКА ПАРАМЕТРОВ МОДЕЛИ

После подбора типа зависимости по форме кривой оценивают число «k» коэффициентов, которые нужно вычислить. Для линейной модели k=2. Для оценки параметров модели можно использовать два метода:

1. ***Метод избранных точек***. На кривой зависимости y=f(x) выбирают k точек по числу определяемых коэффициентов. Для сложных кривых эти точки желательно выбрать как на гладких частях кривых, так и в областях экстремума, на перегибах и т. д. Таким образом получают систему k уравнений с k неизвестными, решая которую вычисляют значения коэффициентов.

2. ***Метод наименьших квадратов (МНК).*** Решение той же задачи по МНК приводит к наиболее точным результатам:

- число пар точек (x_i,y_i) может быть выбрано гораздо больше, чем число коэффициентов
- МНК позволяет вычислить не только значения коэффициентов, но и оценить их значимость, а также адекватность уравнения исходной кривой.

Найдем оценку неизвестных значений β_0 и β_1 методом наименьших квадратов.

Очевидно, что нам никогда не удастся найти β_0 и β_1, так как для этого потребуется провести бесконечное число опытов. В наших силах лишь оценить эти коэффициенты. Наилучшие оценки $\beta_0=b_0$ и $\beta_1=b_1$ получаются минимизацией соответственно по β_0 и β_1 суммы квадратов отклонений

$$R_{SS} = \sum_{i=1}^{N} (y_i - y_{iucm})^2 \to \min \qquad (10)$$

R_{SS} есть мера ошибки, возникающей при аппроксимации выборки прямой. Из (10) с учетом (8) и (9) следует другая запись условия МНК

$$R_{SS} = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2 = \sum_{i=1}^{N} \varepsilon_i^2 \to \min \ , \tag{11}$$

позволяющая найти β_0 и β_1.

Продифференцировав (11) по β_0 и β_1 и приравняв полученные частные производные к нулю, имеем:

$$\begin{aligned} \frac{\partial R_{SS}}{\partial \beta_0} &= 2\sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)(-1) = 0 \\ \frac{\partial R_{SS}}{\partial \beta_1} &= 2\sum_{i=1}^{N} (-x_i)(y_i - \beta_0 - \beta_1 x_i) = 0 \end{aligned} \tag{12}$$

Уравнения (12) получили название нормальных, их число в системе всегда равно числу неизвестных параметров модели. Решив систему уравнений и принимая во внимание, что N конечно, найдем оценки коэффициентов регрессионной модели:

$$b_0 = \bar{y} - b_1 \bar{x} \tag{13}$$

$$b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} \tag{14}$$

Эти оценки не смещены и имеют минимальную дисперсию среди всех несмещенных оценок β_0 и β_1, линейно зависящих от наблюдений y_1, y_2,…,y_N. Оценкой уравнения регрессии (или прямой наименьших квадратов) будет

$$\hat{y} = b_0 + b_1 x \tag{15}$$

Так что оценка значения Y при $X=x_i$ есть $\hat{y}_i = b_0 + b_1 x_i$. Разница между наблюдаемым Y и оцененным значением $\hat{Y}$ при $X=x_i$ называется отклонением или остатком

$$d_i = y_i - \hat{y}_i \tag{16}$$

доставляет минимум сумм квадратов отклонений

$$\hat{R}_{ss} = \sum_{i=1}^{N} d_i^2 \tag{17}$$

Соотношение между теоретической регрессионной прямой ($N=\infty$) и прямой наименьших квадратов можно увидеть на рис. 2 и 3.

Следует различать такие понятия как

- истинная функция регрессии, отражающая действительную, истинную взаимосвязь;
- теоретическая аппроксимирующая функция регрессии, характеризующая результат, к которому мы бы неограниченно приближались при $N=\infty$;

- выборочная аппроксимирующая функция регрессии (прямая наименьших квадратов).

К сожалению, для практики статистических исследований достаточна типична ситуация, когда исследователь «не угадывает» класс допустимых решений, т.е. истинная функция регрессии не принадлежит к выбранному классу. Другими словами, как бы мы не увеличивали объем выборки, мы не сможем добиться сходимости выборочной аппроксимирующей функции регрессии к истинной функции регрессии.

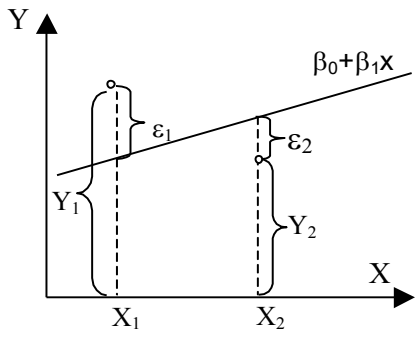


Рис.2.Теоретическая регрессия

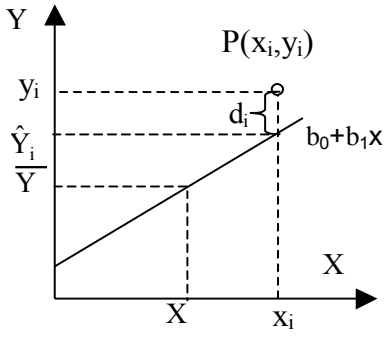


Рис.3. Прямая наименьших квадратов

АНАЛИЗ РЕГРЕССИИ

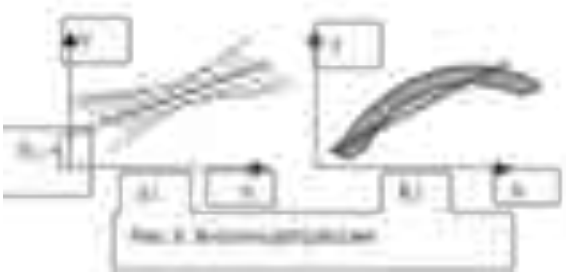
На этом этапе мы должны ответить на следующие три вопроса:

1. Все ли коэффициенты $\{b_i\}$ полученной регрессии значимо отличны от нуля, т.е. не является ли их отличие от нуля фиктивным, обусловленным естественным разбросом случайной величины Y , по которой они вычислены. Например, надо ли учитывать b_0 на рис. 4а или следует считать, что

   $b_0 = 0$.

2. Удачно ли выбрана исходная модель, то есть адекватно ли отражает полученные регрессии экспериментальный материал, положенный в ее основу. Например, не имеем ли мы дело с ситуацией, изображенной на рис. 4б, когда отличие $\hat{Y}$ от Y отчетливо заметно на фоне естественного разброса Y.

3. Какова точность полученной регрессии, то есть каков доверительный интервал у каждого из предсказанного ею значения $\hat{Y}$ (заштрихованная область на рис.4а).



Проверка значимости коэффициентов модели

На этом этапе проверяются гипотезы о значимом отличии от нуля каждого из коэффициентов регрессионной модели.

Для линейной модели

$$Y=\beta_0+\beta_1X_1+ \beta_2X_2+\ldots+ \beta_pX_p \qquad (18)$$

по очереди проверяется серия гипотез

1) H_0: $\beta_1=0$; H_1: $\beta_1\neq0$;

2) H_0: $\beta_2=0$; H_1: $\beta_2\neq0$;

.....

n) H_0: $\beta_p=0$; H_1: $\beta_p\neq0$.

Гипотеза H_0: $\beta_k=0$ для k=1, …, p может рассматриваться как гипотеза о том, что переменная X_k не улучшает предсказание Y по сравнению с предсказанием, получаемым с помощью регрессии Y по (p-1) остальным переменным.

Проверку значимости можно осуществлять двумя равноценными способами:

- проверкой по t-критерию Стьюдента;
- построением доверительного интервала.

Прежде всего необходимо найти дисперсию коэффициента регрессии Db_i. Например, для простой линейной модели после несложных преобразований выражений (13,14) можно получить:

$$Db_1 = s_{b1}^2 = \frac{DY}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad , \qquad (19)$$

$$Db_0 = s_{b0}^2 = s_{b1}^2 \frac{\sum_{i=1}^{N} x_i^{\,2}}{N} \quad , \qquad (20)$$

Воспользуемся критерием Стьюдента для проверки значимости

$$t = \frac{b-0}{s_b}$$

и сравним с критическим значением T-распределения на уровне значимости α при числе степеней свободы df =N-2, с которым определяется дисперсия s_Y^2, а следовательно и s_b^2 .В программах регрессионного анализа величины s_{b0} и s_{b1} называются *стандартными ошибками коэффициентов* регрессии. Для каждого β_i стандартная ошибка коэффициента s_{bi} есть оценка стандартного отклонения b_i от β_i . Если

- $|t| < t_{1-\alpha/2,(df)}$, то отличие от нуля случайно и H_0 не отвергается;

- $|t| < t_{1-\alpha/2,(df)}$, то отличие от нуля следует признать значимым и отвергнуть гипотезу H_0.

К аналогичным результатам можно прийти, вычислив доверительный интервал для истинных коэффициентов β

$$b - t_{1-\alpha/2,(df)}s_b < \beta < b + t_{1-\alpha/2,(df)}s_b \qquad (21)$$

и определив, входит ли в них нулевое значение.

Адекватность модели

Под адекватностью модели простой линейной регрессии подразумевается, что никакая другая модель не дает значимого улучшения в предсказании Y. Например, мы хотим проверить, значимо ли улучшится предсказание Y с помощью полиномиальной регрессии $y_i = \beta_o + \beta_1 x + \ldots + \beta_m x_m + \varepsilon_i$ для m >=2. Нулевой гипотезой в этом случае будет:

$$H_o: \beta_2 = \ldots = \beta_m = 0. \qquad (22)$$

Ниже (рис.5, а,б) приведены два рисунка с одинаковым расположением экспериментальных точек, с одинаковым разбросом относительно линии регрессии, но с различным средним разбросом в точках, иначе говорят с различной дисперсией воспроизводимости.

Разброс в точках показан отрезками прямых, составляющими доверительный интервал ±2S. Модель считается адекватной только в первом случае.

В данном случае разброс в точках такого же порядка, что и разброс относительно линии регрессии. Поэтому можно предположить, что модель пригодна. Во втором случае опыты «слишком» точны, т.е. указывают на более сложную нелинейную модель, в которой точность ее предсказания была бы сравнима с точностью эксперимента.

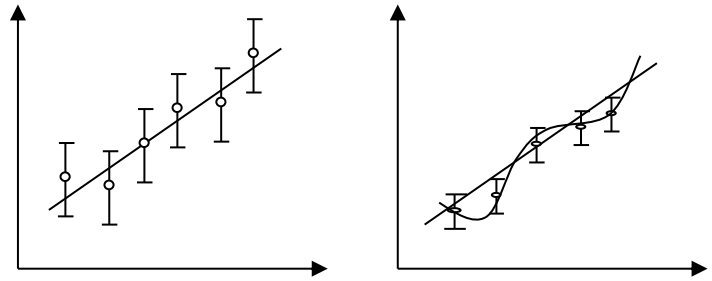


Рис.5. Проверка адекватности

Разброс опытных значений Y относительно линии регрессии Y можно оценить по остаточной сумме квадратов R_{SS}. Неудобство использования R_{SS} состоит в том, что R_{SS} зависит от числа коэффициентов в уравнении. Введите столько коэффициентов, сколько вы провели независимых опытов, и получите остаточную сумму, равную нулю. Поэтому предпочитают относить на один «свободный» опыт. Остаток суммы квадратов, деленный на число степеней свободы, называется остаточной дисперсией, или дисперсией адекватности.

$$S^2_{АД} = \frac{R_{SS}}{df_{АД}} \quad , \tag{23}$$

где df=N-p – число степеней свободы;

p=2 – количество коэффициентов линейной модели.

Предположим, что имеется k различных значений для X, например x_1, …., x_k. Для каждого из этих x_i имеется n_i наблюдений y_{i1}, y_{i2}, …, y_{ini} переменной Y, i=1, … k. Пусть $n_i>1$ и $\Sigma n_i=N$. Тогда модель линейной регрессии может быть записана в следующем виде:

$$Y_{ij}=\beta_0+\beta_1 x_i+\varepsilon_{ij}, \quad j=1, \ldots, n_i, \quad i=1, \ldots, k, \tag{24}$$

а разброс в i-ой точке находим по известной формуле для несмещенной дисперсии:

$$S_i^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (y_{ij} - \overline{y_i})^2 . \tag{25}$$

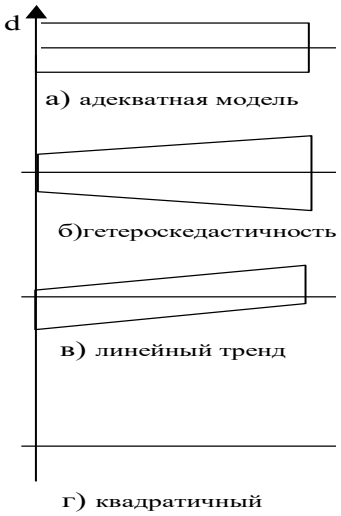


Рис.6. Примеры графиков остатков

Полученную группу выборочных дисперсий S_i^2 можно считать оценками для одной и той же генеральной дисперсии σ_Y^2.

Наилучшая оценка этой дисперсии имеет вид

$$S^2_{ВОС} = \frac{1}{к}\sum_{i=1}^{k} S_i^2 = \frac{1}{k}\sum_{i=1}^{k} \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (y_{ij} - \overline{y_i})^2 \tag{26}$$

и называется дисперсией воспроизводимости.

Если число опытов в каждой точке одинаково, т.е. $n_1=n_2=n_k=m$, то

$$S^2_{ВОС} = \frac{1}{к(m-1)} \sum_{i=1}^{k} \sum_{j=1}^{m} (y_{ij} - \overline{y_i})^2 \tag{27}$$

с числом степеней свободы $df_{вос}=k(m-1)$.

Для проверки гипотезы об однородности разброса в точках и разброса относительно линии регрессии

$$H_0: S^2_{АД}=S^2_{ВОС}; \qquad H_1: S^2_{АД} > S^2_{ВОС} \tag{28}$$

используется F – критерий:

$$F = \frac{S^2_{АД}}{S^2_{ВОС}} \tag{29}$$

Если $F<F^{\alpha}_{dfад,dfвос}$, то модель можно считать адекватной. Если гипотеза об адекватности отвергается, необходимо переходить к более сложной форме математического описания, либо, если это касается планирования активного эксперимента, проводить эксперимент с меньшим интервалом варьирования.

Исследование регрессионных остатков

Интересную статистическую информацию может дать анализ остатков

$$d_i = y_i - \hat{y}_i,\ i=1,\ldots N$$

Для проверки адекватности модели можно использовать график d_i в зависимости от x_i или $\hat{y}_i$. Рассмотрим типичные графики остатков:

1. Если остатки попадают в горизонтальную полосу с центром на оси абсцисс, модель можно рассматривать как адекватную (рис. 6а).
2. Если полоса расширяется, когда X или Y возрастают, это указывает на гетероскедатичность, т.е. на отсутствие постоянства дисперсии. В частности, стандартное отклонение может быть функцией $\beta_0+\beta_1 x$, что делает необходимым преобразование Y (рис. 6б).
3. Если остатки представляются линейным трендом, в модель необходимо ввести независимую дополнительную переменную (рис. 6в).
4. График вида, представленного на рис 6г, указывает, что в модель должен быть добавлен квадратичный член.

Оценка точности регрессии

Точность регрессии задается доверительным интервалом, внутри которого предсказанное среднее значение $\hat{Y}_{ист}$ располагается с вероятностью 1- α. Определим дисперсию среднего значения $D\hat{Y}$.

$$D\hat{Y} = \left[\frac{1}{N} + \frac{(x-\overline{x})^2}{\sum_{i=1}^{N}(x_i-\overline{x})^2}\right] DY \qquad (30)$$

Располагая $D\hat{Y}$ или ее оценкой $s2\hat{Y}$, определенной в соответствии с (30), найдем доверительный интервал

$$\hat{Y} - t_{1-\alpha/2,(df)} s_{\hat{Y}} < \hat{Y}_{ист} < \hat{Y} + t_{1-\alpha/2,(df)}\, s_{\hat{Y}} \qquad (31)$$

Из (31) следует, что наибольшая точность предсказаний соответствует центральной части диапазона изменения X; с удалением от среднего значения X точность предсказаний падает.

Лекция 4. Метод главных компонент

ВВЕДЕНИЕ

С данными бывают две проблемы: либо их слишком мало, либо их слишком много. Сегодня мы поговорим о второй проблеме.

Есть такая область исследования — определение авторства текстов. Допустим, у нас есть массив текстов, автор которых неизвестен. Может быть, эти тексты принадлежат одному и тому же человеку, может быть разным. Может быть, мы догадываемся о том, кто является автором, а может быть и нет. Если бы эти тексты были написаны ручкой на бумаге, мы могли бы ответить на какие-то вопросы об авторстве, сравнивая почерки. Но сейчас перед нами чаще «голый» текст, не содержащий таких естественных индивидуальных признаков, как почерк.

Тем не менее, разумно предположить, что эти признаки всё-таки есть. Одни люди пишут длинными предложениями, другие короткими. Одни используют много разных слов, другие ограничиваются небольшим запасом. Одни используют много глаголов, другие мало. У каждого автора есть свой стиль и этот стиль можно извлечь из текста. Как это сделать?

Для каждого текста можно вычислить огромное количество параметров. Среднюю длину предложения. Разброс длин предложений. Распределение слов. Процент существительных, прилагательных, глаголов. И ещё кучу всего. В общем, можно превратить каждый текст в длинный-длинный вектор. Взяв много текстов, можно записать их параметры в широкую табличку.

Но дальше возникает проблема: что, собственно говоря, делать с этим массивом данных? Как его обрабатывать? Как находить связи между разными параметрами? **Как их визуализировать?**

Если у нас есть один параметр, можно нарисовать для него гистограмму. Если параметра два, можно нарисовать диаграмму рассеяния (scatter plot), показывающую, как распределён каждый из них и как они связаны между собой. С большим количеством параметров так сделать нельзя: пространство, в котором можно было бы нарисовать соответствующие картинки, имеет высокую размерность и не помещается на двумерной бумаге или экране.

Что же делать? Есть разные подходы. Например, можно рассмотреть всевозможные пары параметров и для каждой нарисовать свою диаграмму рассеяния. Таким образом, однако, удастся визуализировать только попарные связи между параметрами, а этого зачастую недостаточно. К тому же этот метод приводит к появлению очень большого количества картинок, если число параметров велико.

Другой подход строится в предположении, что наш набор параметров избыточен и для описания наиболее важных свойств текстов достаточно всего нескольких чисел. Это кажется разумным: если среди наших многочисленных параметров есть такие, которые находятся в сильной связи друг с другом (а это вполне естественное предположение), то часть из них можно отбросить. Например, если один из параметров выражается через другие, то его можно просто выкинуть без потери информации.

Но как выяснить, какой именно набор параметров хорошо описывает наш набор данных, но при этом имеет небольшую избыточность? Иными словами, ***как уменьшить размерность пространства, в котором живут данные, потеряв при этом минимум информации***?

Способы решения этой задачи, называются **методами уменьшения размерности** (dimensionality reduction).

Сущность проблемы

Общее число признаков $X^{(1)}$, $X^{(2)}$, …, $X^{(p)}$, регистрируемых на каждом из множества обследуемых объектов очень велико – порядка ста и более. Тем не менее имеющиеся многомерные наблюдения следует подвергнуть статистической обработке, осмыслить либо внести в БД для того чтобы иметь возможность их использовать в нужный момент. Суть проблемы состоит в том, чтобы представить каждое из наблюдений в виде вектора Z некоторых вспомогательных показателей $Z^{(1)}$, $Z^{(2)}$, …, $Z^{(u)}$ с существенно меньшим чем числом компонент (u<<p)

От $X^{(1)}, X^{(2)}, \ldots, X^{(p)}$ к $Z^{(1)}, Z^{(2)}, \ldots, Z^{(u)}$

Зачем?

- Необходимость наглядного представления исходных данных (визуализация), что достигается их проецированием на специально подобранное трехмерное пространство (u=3), плоскость (u=2),или числовую прямую;
- Стремление к лаконизму исследуемых моделей, обусловленному необходимостью упрощения счета и интерпретации полученных статистических выводов;
- Необходимость существенного сжатия объемов хранимой статистической информации (без видимых потерь в ее информативности)

Предпосылки перехода

Существуют следующие основания говорить о возможности перехода от большого числа p исходных показателей анализируемой системы к существенно меньшему числу u наиболее информативных переменных

- Дублирование информации от сильно взаимосвязанных признаков
- Неинформативность признаков, мало меняющихся при переходе от одного объекта к другому

- Возможность агрегирования, то есть простого или взвешенного суммирования, по некоторым признакам

Методы снижения размерности

К основным методам снижения размерности относятся метод главных компонент и факторный анализ.

Другие способы уменьшения размерности данных — это

- метод независимых компонент,
- многомерное шкалирование, а также многочисленные нелинейные обобщения:
    - метод главных кривых и многообразий,
    - поиск наилучшей проекции (англ. Projection Pursuit),
    - нейросетевые методы «узкого горла»,
    - самоорганизующиеся карты Кохонена и др.

МЕТОД ГЛАВНЫХ КОМПОНЕНТ

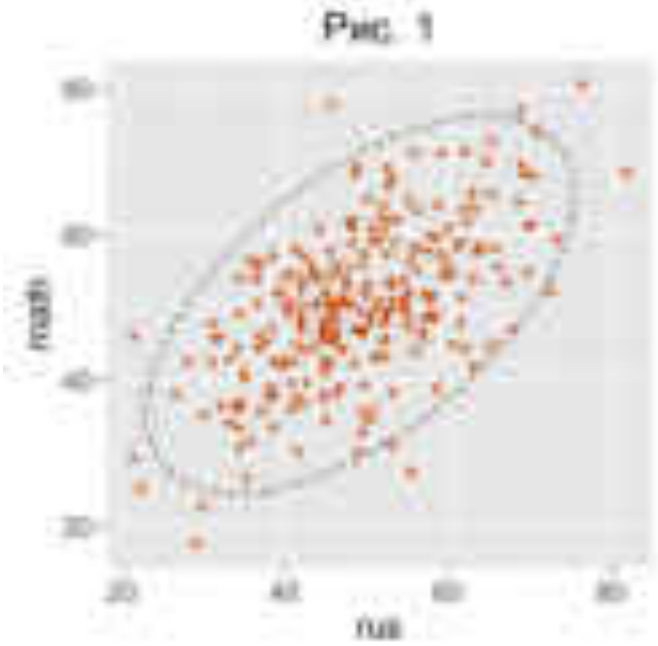
Метод главных компонент (principal component analysis, PCA) — один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации. Изобретён Карлом Пирсоном в 1901 году. Применяется во многих областях, в том числе, в эконометрике, биоинформатике, обработке изображений, для сжатия данных, в общественных науках.

Вычисление главных компонент может быть сведено к вычислению сингулярного разложения матрицы данных или к вычислению собственных векторов и собственных значений ковариационной матрицы исходных данных. Иногда метод главных компонент называют преобразованием Кархунена — Лоэва или преобразованием Хотеллинга.

Пример «Школьные оценки»

У нас есть табличка с результатами теста для школьников по двум предметам — например, по русскому языку и математике. Можно нарисовать вот такую картинку.

	rus	math
1	38.62	33.67
2	46.22	54.53
3	46.40	38.32
4	53.17	51.07
5	62.86	65.64



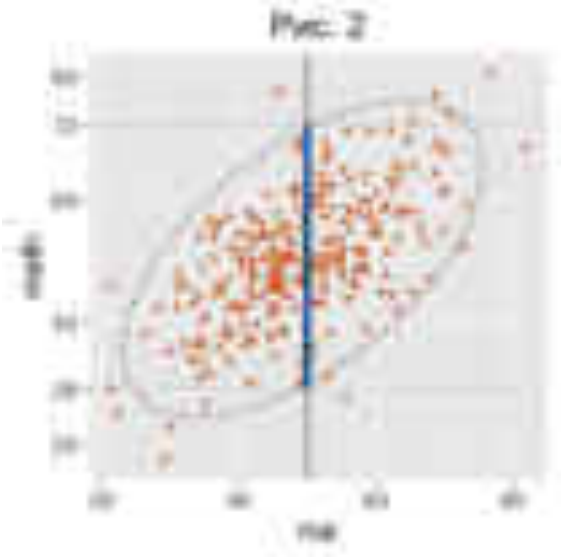
Мы видим, что оценки по этим двум предметам скоррелированы — среди школьников, получающих высокие баллы по математике, много тех, кто также получает высокие баллы по русскому языку, и наоборот (это согласуется с нашей интуицией). Есть и исключения, лежащие вне эллипса – *выбросы*. Для простоты изложения мы сейчас будем пренебрегать выбросами, а также считать, что если бы мы взяли побольше школьников, то соответствующие им точки практически заполнили бы весь эллипс.

Это наши исходные данные. Теперь предположим, что нам надо уменьшить размерность — вместо двух чисел на каждого школьника хранить только одно число. Например, мы выбираем, что записать в аттестат, и по закону это должно быть только одно число, а не два. И мы хотим в этом единственном числе закодировать как можно больше информации о школьнике. Как в этом случае поступить?

Можно просто отбросить одну из переменных и оставить другую. Например, можно записывать в аттестат только оценку по русскому языку, а оценку по математике игнорировать.

С одной стороны, это не такая уж и плохая стратегия. Как мы обсуждали выше, оценки скоррелированы, и человек, сдавший русский на высокий балл, скорее всего не имеет проблем и с математикой, и наоборот.

С другой стороны, какая-то информация всё-таки теряется. Представьте себе, что мы знаем о школьнике только тот факт, что он сдал русский на 50 баллов. Что это говорит о математических способностях школьника? Если не знать настоящих данных, а смотреть только на эллипс на картинке, то мы увидим, что соответствующая точка может находиться в любом месте синего отрезка — его оценка по математике в этом случае колеблется примерно между 29 и 72 — разброс очень большой.



Конечно, понятно, что, заменяя два числа одним, мы потеряем какую-то информацию, но хотелось бы всё-таки эти потери минимизировать.

Можем ли мы сделать что-то лучшее, чем сообщать только одну оценку из двух? Оказывается, можем.

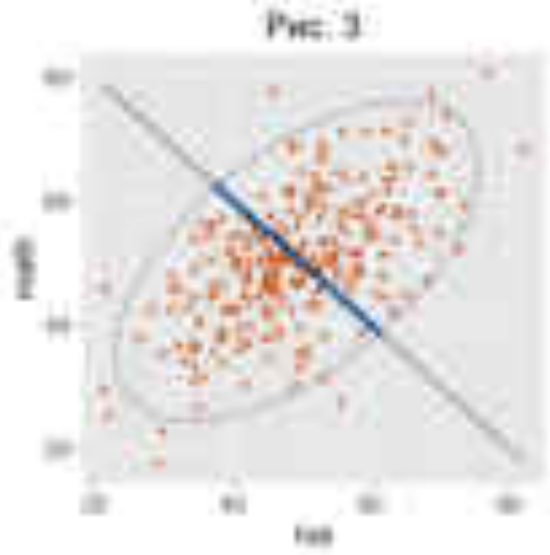
Мы можем сконструировать новое число из двух имеющихся!

Давайте рассмотрим самый простой вариант: впишем в аттестат сумму оценок за русский язык и математику. Иными словами, мы введём новую переменную — обозначим её через **PC1** (почему так — будет ясно позднее), которая связана с нашими старыми переменными таким образом:

PC1 = rus + math

Посмотрим, насколько этот метод лучше. Пусть мы знаем, что для некоторого школьника PC1=100. Что тогда можно сказать про его оценки?

Проведём на графике прямую, соответствующую условию PC1=100, то есть rus+math=100. Она пройдёт из левого верхнего угла в правый нижний и пересечёт наш эллипс по некоторому отрезку.



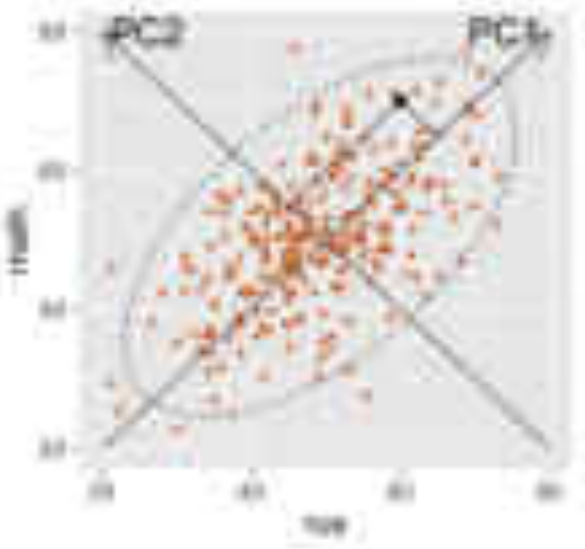
Как и в прошлый раз, наш школьник мог бы оказаться в любой точке этого отрезка: на этот раз мы не знаем наверняка ни оценку по математике (она колеблется где-то между 39 и 63), ни оценку по русскому языку (она между 37 и 61).

Тем не менее, если измерять степень нашего незнания длиной того самого отрезка, на котором может оказаться школьник, то мы видим, что она уменьшилась: новый отрезок короче старого, потому что сейчас мы пересекаем эллипс «поперек», а раньше пересекали «наискосок». Поэтому сообщать наше число PC1 лучше, чем сообщать только одну из оценок (если, конечно, мы не знаем заранее, что получателю этой информации какая-то из двух оценок важнее другой).

Метод главных компонент — это история про введение новой, более экономной системы координат, в которой описывать наши данные проще. Вот как эта система координат будет устроена в нашем примере с оценками.

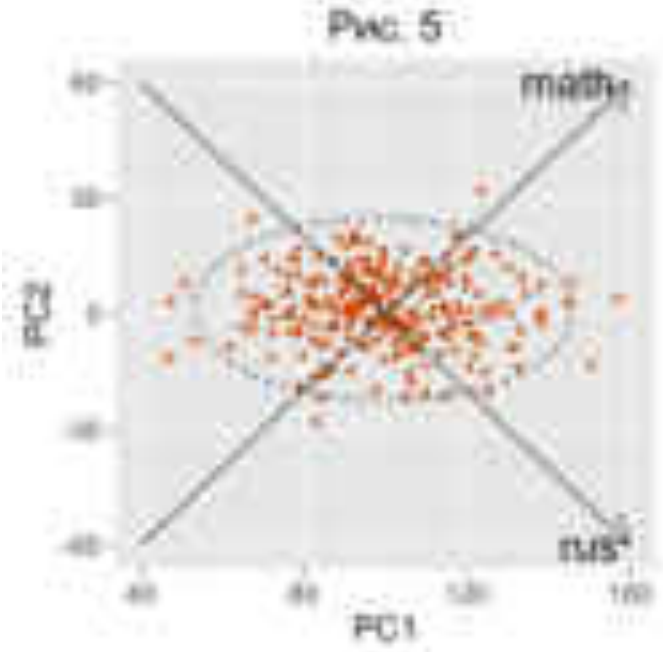
В качестве первой координаты точки мы возмём PC1, то есть сумму её старых координат, а в качестве второй координаты (обозначим её через PC2) возьмём разность её старых координат: PC2=math−rus

Рис. 4



Новая система координат

Рис. 5



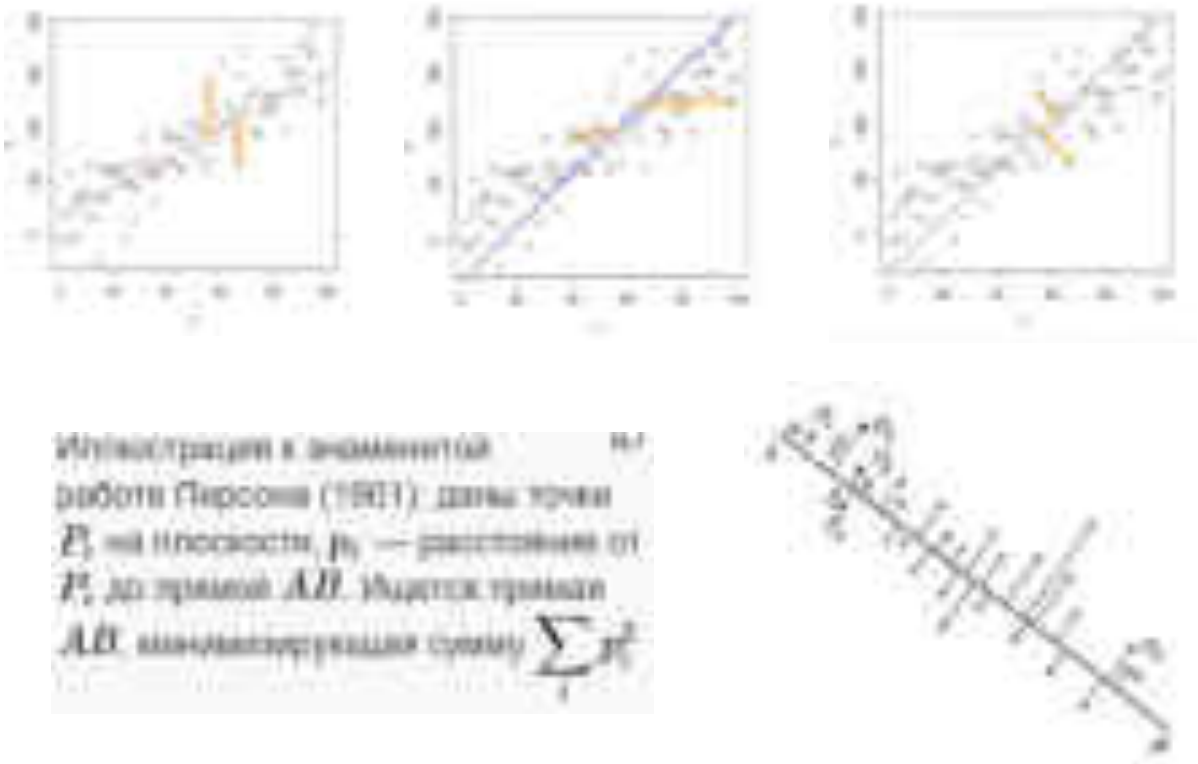
Аппроксимация данных линейными многообразиями

Метод главных компонент начинался с задачи наилучшей аппроксимации конечного множества точек прямыми и плоскостями (Пирсон, 1901). Дано конечное множество векторов $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$, для каждого $k = 0,1, \ldots, n-1$ среди всех k-мерных линейных многообразий в $\mathbb{R}^n$ найти такое множество линейных комбинаций $L_k \subset \mathbb{R}^n$, что сумма квадратов уклонений x_i от L_k минимальна:

$$\sum_{i=1}^{m} \operatorname{dist}^2(x_i, L_k) \to \min$$

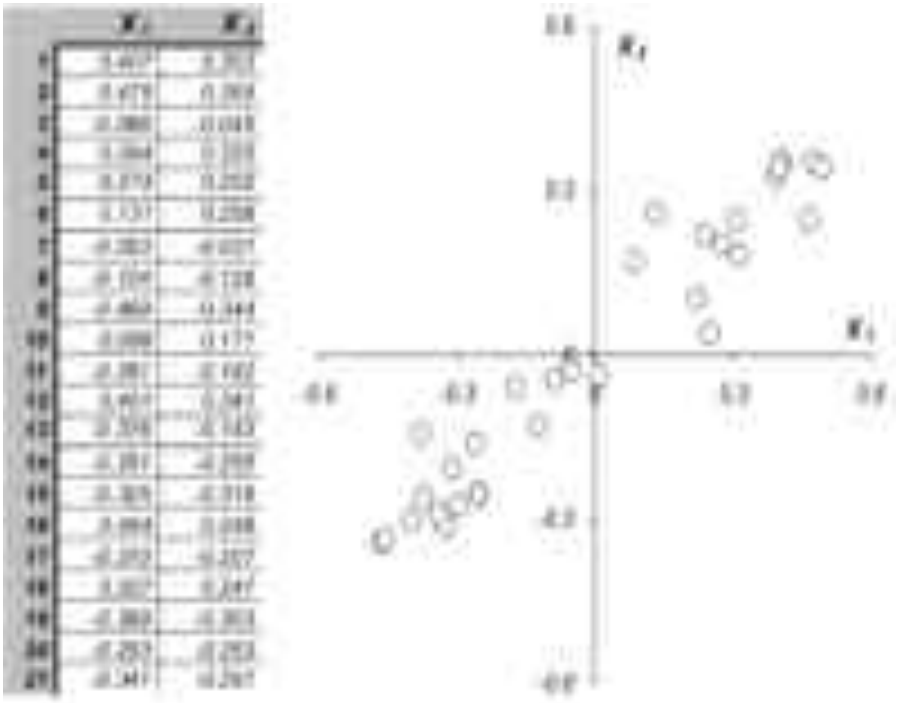
где $\operatorname{dist}(x_i, L_k)$ — евклидово расстояние от точки до линейного многообразия.

Три вида проекций



Интуитивный подход

Постараемся передать суть метода главных компонент, используя интуитивно–понятную геометрическую интерпретацию. Начнем с простейшего случая, когда имеются только две переменные x_1 и x_2. Такие данные легко изобразить на плоскости.



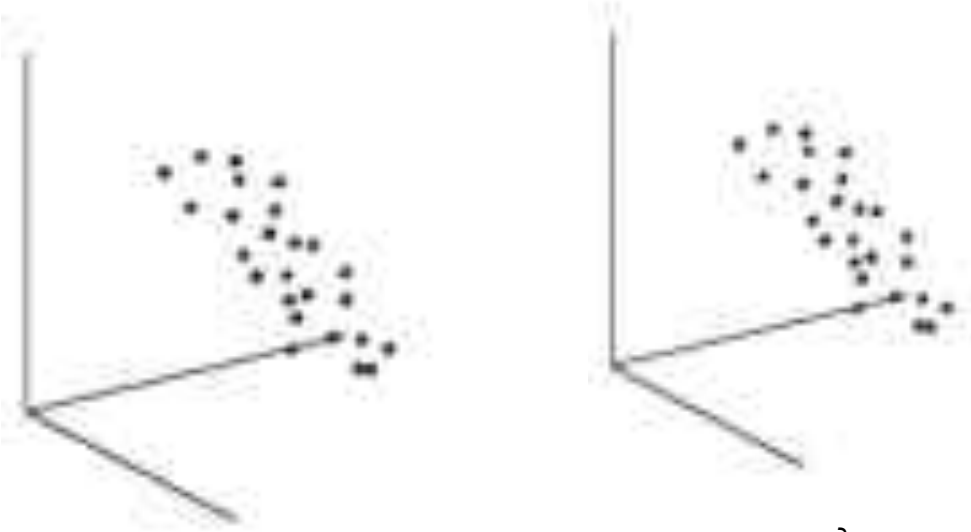
Каждой строке исходной таблицы (т.е. образцу) соответствует точка на плоскости с соответствующими координатами. Они обозначены **пустыми** кружками. Проведем через них прямую так, чтобы вдоль нее происходило максимальное изменение данных. На рисунке эта прямая выделена синим цветом; она называется **первой главной компонентой – PC1**. Затем спроецируем все исходные точки на эту ось. Получившиеся точки закрашены **красным** цветом. Теперь мы можем предположить, что на самом деле все наши экспериментальные точки и должны были лежать на этой новой оси. Просто какие–то неведомые силы отклонили их от правильного, идеального положения, а мы вернули их на место. Тогда все отклонения от новой оси можно считать шумом, т.е. ненужной нам информацией.

Правда, мы должны быть в этом уверены. Проверить шум ли это, или все еще важная часть данных, можно поступив с этими остатками так же, как мы поступили с исходными данными – найти в них ось максимальных изменений. Она называется **второй главной компонентой (PC2).** И так надо действовать, до тех пор, пока шум уже не станет действительно шумом, т.е. случайным хаотическим набором величин.

В общем, многомерном случае, процесс выделения главных компонент происходит так:

- Ищется центр облака данных, и туда переносится новое начало координат – это нулевая главная компонента (PC0)
- Выбирается направление максимального изменения данных – это первая главная компонента (PC1)

- Если данные описаны не полностью (шум велик), то выбирается еще одно направление (PC2) – перпендикулярное к первому, так чтобы описать оставшееся изменение в данных и т.д.



Метод главных компонент – как проекция на подпространство

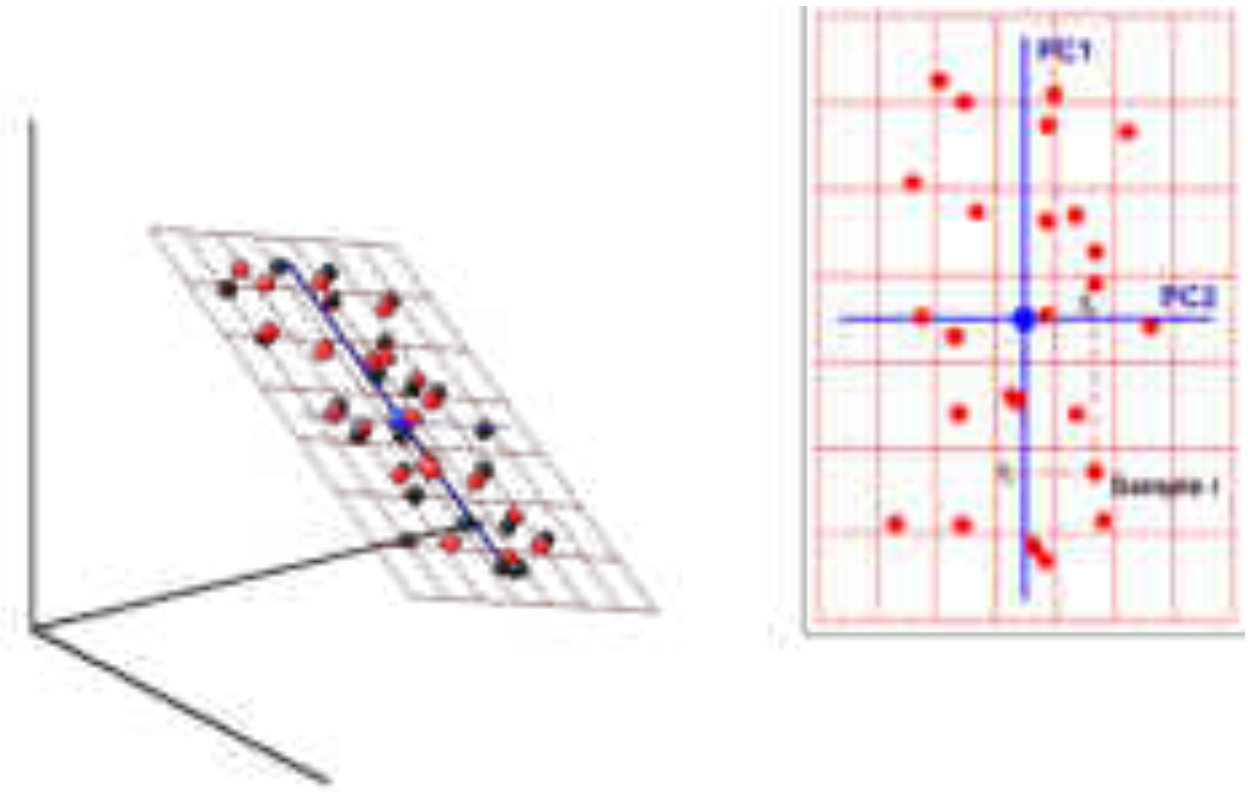
В результате мы переходим от большого количества переменных к новому представлению, размерность которого значительно меньше. Часто удается упростить данные на порядки: от 1000 переменных перейти всего к двум. При этом ничего не выбрасывается – все переменные учитываются. В то же время несущественная для сути дела часть данных отделяется, превращается в шум.

Найденные главные компоненты и дают нам искомые скрытые переменные, управляющие устройством данных.

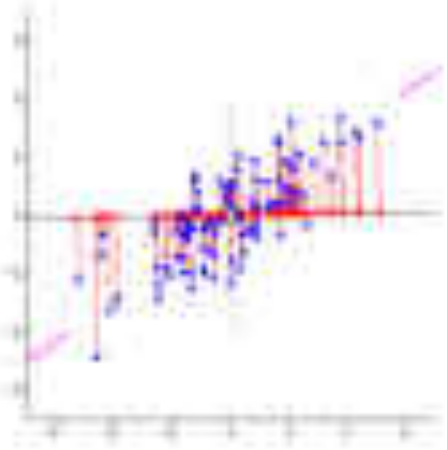
PCA найдет «лучшую» линию в соответствии с двумя разными критериями того, что является «лучшим».

Во-первых, изменение значений вдоль этой линии должно быть максимальным. Обратите внимание на то, как изменяется «разброс» (мы называем это «дисперсией») красных точек, когда линия вращается; видите ли вы, когда он достигает максимума?

Во-вторых, если мы восстановим исходные две характеристики (положение синей точки) от новой (положение красной точки), ошибка восстановления будет определяться длиной соединительной красной линии. Наблюдайте, как изменяется длина этих красных линий во время вращения линии; видите ли вы, когда общая длина достигает минимума?



Вот как выглядят эти проекции для разных линий (красные точки - это проекции синих точек):



Второй главный компонент рассчитывается таким же образом, при условии, что он не коррелирован (то есть перпендикулярен) первому главному компоненту и учитывает следующую по величине дисперсию. Это продолжается до тех пор, пока не будет вычислено p главных компонент, равное исходному количеству переменных.

С геометрической точки зрения, главные компоненты представляют собой Векторы данных, которые объясняют максимальное количество отклонений. Главные

компоненты – новые оси, которые обеспечивают лучший угол для оценки данных, чтобы различия между наблюдениями были лучше видны.

Поскольку существует столько главных компонент, сколько переменных в наборе, главные компоненты строятся таким образом, что первый из них учитывает наибольшую возможную дисперсию в наборе данных.

Постановка задачи

Линейное преобразование как переход к новой системе координат

Таблицу экспериментальных данных (ТЭД) геометрически можно интерпретировать как облако экспериментальных данных в **p**-размерном пространстве, где **p** – количество признаков, свойств объектов. Каждая точка такого пространства представляет собой один объект исследования.

- Из векторной алгебры хорошо известно, что точка (вектор) в заданной системе координат может быть представлена и в другой системе координат (в другом базисе), причем количество этих базисов бесконечно.
- Новая система координат $Y_1, Y_2, ..., Y_p$ может быть выражена через старые координаты $X_1, X_2, ..., X_p$, как их линейная комбинация.

Геометрия PCA

Каждый исходный образец x_i (строка в матрице X) можно представить как вектор в p–мерном пространстве с координатами

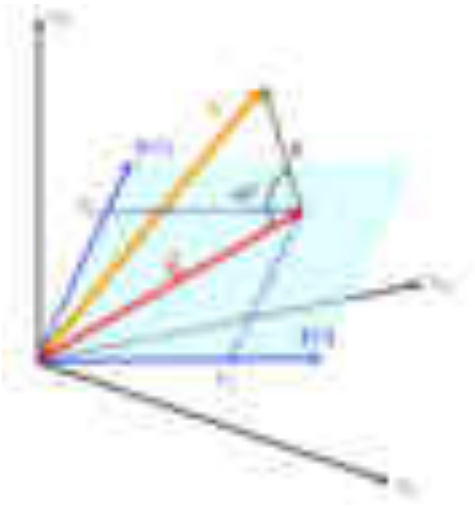
$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$

PCA проецирует его в вектор, лежащий в пространстве главных компонент,

$\mathbf{t}_i = (\mathbf{t}_{i1}, \mathbf{t}_{i2}, \ldots \mathbf{t}_{iA})$

размерностью *A*. В исходном пространстве этот же вектор $\mathbf{t}_i$ имеет координаты

$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$



Постановка задачи МГК

Дано:

1. Случайные переменные $x_1, \ldots, x_p$
2. Вектор средних $m=(m_{1, \ldots,} m_p)'$
3. Ковариационная матрица $\sum^{p \times p} = (\sigma ij)$

Задание

Найти такие линейные комбинации исходных переменных:

$$Y_1 = \sum_{j=1}^{p} a_{1j} x_j$$

$$\dots\dots\dots \qquad (1)$$

$$Y_p = \sum_{j=1}^{p} a_{pj} x_j$$

что

1. $\mathrm{cov}(Y_i, Y_j) = 0 \quad i, j = 1, 2, \dots, p, \quad i \neq j$ (2)
2. $D(Y_1) > D(Y_2) > \dots > D(Y_p)$ (3)
3. $\sum_{j=1}^{p} D(Y_j) = \sum_{j=1}^{p} s_j = D_0$ (4)

Из формул видно, что

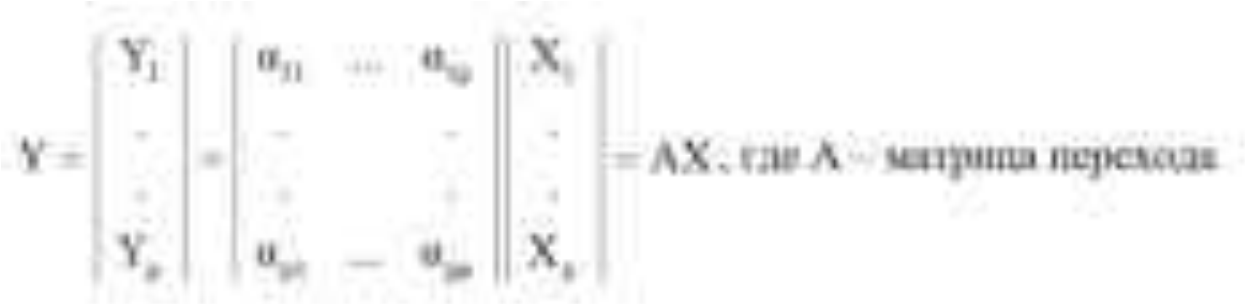
1. Переменные Y_1, …, Y_p не коррелированы (2) и упорядочены по возрастанию дисперсии (3).
2. Общая дисперсия D_0 после преобразования остается без изменений (4).

Ковариационная матрица представляет собой симметричную матрицу размера p × p (где p – количество измерений), где в качестве ячеек пребывают коэффициенты ковариации, связанные со всеми возможными парами исходных переменных.

Поскольку ковариация переменной с самой собой – это ее дисперсия, на главной диагонали (от верхней левой ячейки к нижней правой), у нас фактически есть дисперсии каждой исходной переменной. А поскольку ковариация коммутативна (в ячейке XY значение равно YX), элементы матрицы симметричны относительно главной диагонали.

Матричная запись

Уравнение (1) можно записать в матричной форме:



Вычисление ГК

Первой главной компонентой **Y_1** исследуемой системы показателей $X = (X_1, \ldots, X_p)$ называется такая нормировано-центрированная линейная комбинация этих показателей, которая среди всех других обладает наибольшей дисперсией. Из уравнения имеем

Тогда имеем

$$Y_1 = \alpha_{11} x_1 + \ldots + \alpha_{1p} x_p$$

Требуется найти такие **$a_{11}, \ldots, a_{1p}$**, чтобы величина

$$DY_1 = \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_{1i} \alpha_{1j} \sigma_{ij}$$

была максимальной при

$$\sum_{j=1}^{p} \alpha_{1j} = 1$$

Это условие обеспечивает единственность решения.

Решение

$$a_1 = (a_{11}, \ldots, a_{1p})'$$

называется собственным вектором и соответствует максимальному собственному значению матрицы . Это собственное значение равно дисперсии DY_1. Линейная комбинация (5) называется главной компонентой переменных $x_1, \ldots, x_p$. Она объясняет

$\frac{DY_1}{D_0}$ % общей дисперсии.

Аналогично можно показать, что

$$Y_k = \propto_k X$$

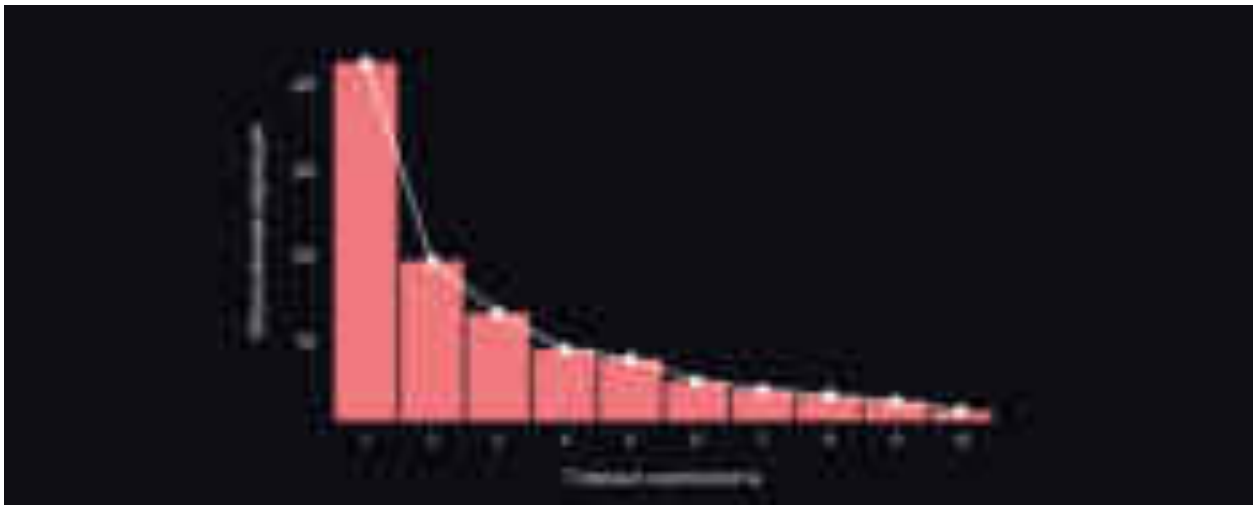
где $\propto_k$ – собственный вектор матрицы $\sum$, соответствующей k-тому по величине собственному значению λ_k этой матрицы.

Переменные $Y_1, \ldots, Y_k$ будут объяснять

$$100*(DY_1+\ldots+ DY_k)/D_0 \ \%$$

общей дисперсии.

Главная компонента – это новая переменная, смесь исходных. Эти комбинации выполняются таким образом, что новые переменные (то есть главные компоненты) не коррелированы, и большая часть информации в исходных переменных помещается в первых компонентах. Итак, идея состоит в том, что 10-мерный датасет дает нам 10 главных компонент, но PCA пытается поместить максимум возможной информации в первый, затем максимум оставшейся информации во второй и так далее, пока не появится что-то вроде того, что показано на графике ниже:



Такая организация информации в главных компонентах позволит нам уменьшить размерность без потери большого количества информации за счет отбрасывания компонент с низкой информативностью.

Здесь важно понимать, что главные компоненты менее интерпретируемы и не имеют никакого реального значения, поскольку они построены как линейные комбинации исходных переменных.

Матрицу ковариации для нашей выборки X можно представить в виде произведения X^TX. Из отношения Релея вытекает, что максимальная вариация нашего набора данных будет достигаться вдоль собственного вектора этой матрицы, соответствующего максимальному собственному значению. Таким образом главные

компоненты, на которые мы бы хотели спроецировать наши данные, являются просто собственными векторами соответствующих топ-k штук собственных значений этой матрицы.

Дальнейшие шаги просты — надо просто умножить нашу матрицу данных на эти компоненты и мы получим проекцию наших данных в ортогональном базисе этих компонент. Теперь если мы транспонируем нашу матрицу данных и матрицу векторов главных компонент, мы восстановим исходную выборку в том пространстве, из которого мы делали проекцию на компоненты. Если количество компонент было меньше размерности исходного пространства, мы потеряем часть информации при таком преобразовании.

Таким образом, соотношение для определения всех “p” ГК вектора X может быть представлено в виде

$$Y = AX ,$$

где $Y = (Y_1, \ldots, Y_p)^T$, $X = (X_1, \ldots, X_p)^T$, а матрица A состоит из строк $\propto_j=(\propto_{j1}, \ldots, \propto_{jp})$, j = 1, 2, …, p, являющихся собственными векторами матрицы $\sum$, соответствующим собственным числам λ_j. При этом сама матрица A по построению является ортогональной, т.е.

$$AA^T = A^TA = I$$

Коэффициент α_{ij} показывает, какой вклад вносит старый признак $X^{(j)}$ в i-тую главную компоненту. Поэтому матрица $A = \{\alpha_{ij}\}$ называется матрицей нагрузок на ГК.

Для визуального анализа данных часто используют проекции исходных век-торов на плоскость первых двух главных компонент. Обычно хорошо видна структура данных, выделяются компактные кластеры объектов и отдельно выделяющиеся вектора.

Когда матрица ковариации неизвестна, она оценивается вторичной ковари-ационной матрицей S. Для получения оценок ГК следует применить описанную выше процедуру к матрице S. В результате получаются оценки a_{ij} коэффициентов α_{ij}, i, j = 1, …,p.

Для получения ГК можно использовать вместо ковариационной матрицы корреляционную. Обычно, когда “p” переменных измеряются в различных единицах, не имеющих между собой ничего общего, линейные комбинации переменных бывает трудно интерпретировать. В этом случае переходят к z-оценкам.

$$z_i = (x_i - \mu)/\sigma_i \text{ или } z_i = (x_i - \bar{x}) / S_i \quad i = 1, \ldots, p$$

При этом общая дисперсия D0 равняется числу переменных.

Корреляция между переменной X_i и главной компонентой Y_i задается величиной

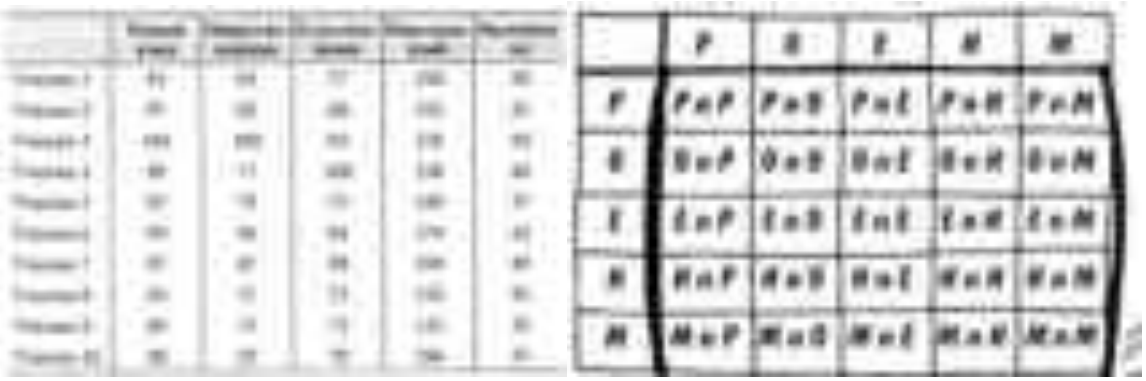
$$\text{corr}(X_i,Y_j) = \alpha_{ij} \frac{S_j}{S_i}$$

где S_i, S_j оценки стандартных отклонений X_i и Y_j. Следовательно, для сравнения вкладов переменных x_1, …, x_p в Y_j следует сравнить величины α_{ij}/si, i = 1, …, p.

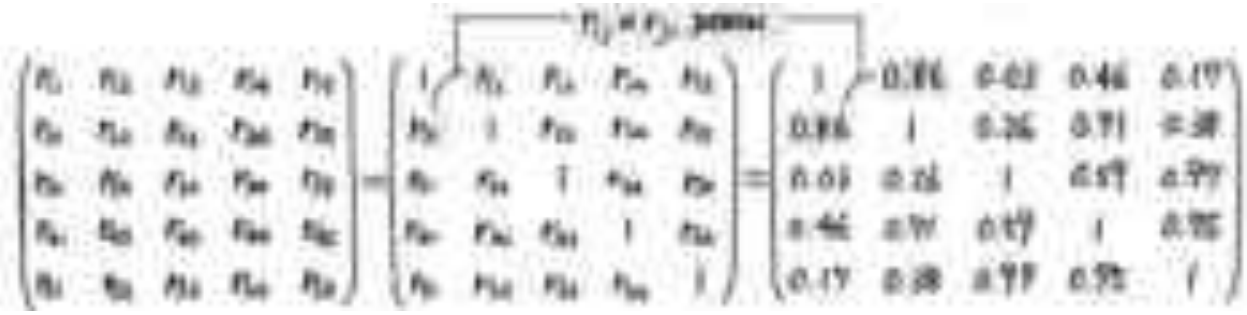
Когда известна корреляционная матрица, достаточно сравнить коэффициенты α_{ij}. В этом случае самый большой коэффициент показывает, какая переменная внесла наибольший вклад в j-тую ГК.

Корреляционная матрица

Исходная таблица данных. Ищем корреляцию предметов. В таблице хранятся набранные учениками баллы по дисциплинам.



Рассчитаем корреляционную матрицу

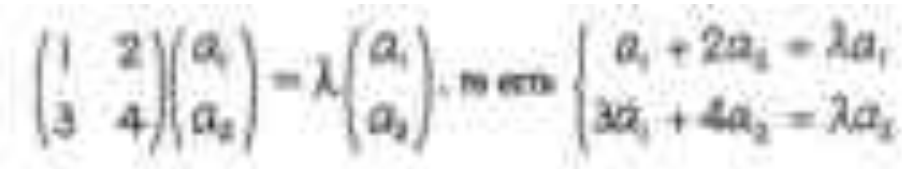


Обратите внимание:

1. Матрица симметрична. Почему?
2. По диагонали размещены единицы. Почему?

Собственные значения и собственные векторы матриц

У каждой матрицы есть собственный вектор и собственное число. Например, собственное значение для матрицы



это такое значение лямбда **l**, которое удовлетворяет вот этому уравнению:

$$R*V = lV$$

где R – матрица, для которой ищется решение;

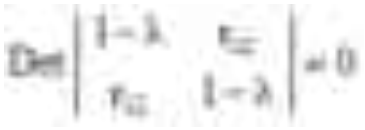
V – искомый собственный вектор.

l - собственное число.

Решение основано на простой форме в виде детерминанты матрицы

$$\mathbf{Det(R - lI) = 0}$$

Это дает для квадратной матрицы уравнение



которое может быть представлено в виде:

$$(1-l)^2 – r_{12}^2 = 0$$

Раскрывая скобки и группируя, получаем

$$\mathbf{l^2-2\,l + (1 - r_{12}^2) = 0}$$

Собственные числа могут быть получены при решении квадратного уравнения. Для двумерной корреляционной матрицы собственные числа имеют вид:

$l_1 = 1 + r_{12}$

$l_2 = 1 - r_{12}$

Если между двумя переменными имеется функциональная линейная зависимость, то одно собственное число будет 2, другое – 0. Для некоррелированных переменных оба собственных числа будут равны 1. Заметим, что

- сумма собственных чисел $l_1 + l_2 = (1 + r_{12}) +(1 - r_{12}) = 2$ равна числу переменных;
- а произведение $l_1l_2 = (1 - r_{12}^2)$ равна детерминанту корреляционной матрицы;

Эти свойства сохраняются для корреляционных матриц любой размерности, причем первое большее собственное число представляет величину дисперсии, соответствующую первой главной оси, а второе собственное число – величину дисперсии, соответствующее второй главной оси и т. д.

Так как при использовании корреляционной матрицы сумма собственных чисел равна **m** (число переменных), можем получить доли дисперсии, соответствующую данному направлению или компоненте:

$$\frac{\lambda_k}{m}$$

Набор собственных значений $\lambda_1,..., \lambda_N$ матрицы A называется спектром A.

Пример 1

$$\begin{pmatrix} -10 & 6 \\ -18 & 11 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} -10 \times 1 + 6 \times 2 \\ -18 \times 1 + 11 \times 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} -10 & 6 \\ -18 & 11 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -10 \times 2 + 6 \times 3 \\ -18 \times 2 + 11 \times 3 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \end{pmatrix} = -\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Следовательно,

- 2 и -1 являются собственными значениями данной матрицы;
- Собственным вектором для 2 является вектор $(1,2)^T$, а для -1 вектор $(2,3)^T$.

Пример 2

Матрица, имеющая P строк и P столбцов, как правило, имеет P пар собственных значений и собственных векторов.

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \times 1 + 0 \times 0 + 0 \times 0 \\ 0 \times 1 + 4 \times 0 + 0 \times 0 \\ 0 \times 1 + 0 \times 0 + 6 \times 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \times 0 + 0 \times 1 + 0 \times 0 \\ 0 \times 0 + 4 \times 1 + 0 \times 0 \\ 0 \times 0 + 0 \times 1 + 6 \times 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix} = 4 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \times 0 + 0 \times 0 + 0 \times 1 \\ 0 \times 0 + 4 \times 0 + 0 \times 1 \\ 0 \times 0 + 0 \times 0 + 6 \times 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} = 6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Пределы применимости и ограничения эффективности метода

Метод не всегда эффективно снижает размерность при заданных ограничениях на точность. **Прямые и плоскости не всегда обеспечивают хорошую аппроксимацию**. Например, данные могут с хорошей точностью следовать какой-нибудь кривой, а эта кривая может быть сложно расположена в пространстве данных. В этом случае метод главных компонент для приемлемой точности потребует нескольких компонент (вместо одной), или вообще не даст

снижения размерности при приемлемой точности.

Для работы с такими «кривыми» главными компонентами изобретен метод главных многообразий и различные версии нелинейного метода главных компонент. Больше неприятностей могут доставить данные сложной топологии. Для их аппроксимации также изобретены различные методы, например, самоорганизующиеся карты Кохонена, нейронный газ или топологические грамматики.

Применение МГК

1. **Визуализация данных**
2. **Сжатие данных.** Экономия памяти, храня только значения компонентов вместо значений всех пикселей для каждой фотографии. (как подобрать платье с помощью метода главных компонент)
3. **Обработка изображений**.
4. Индексация видео
5. Психодиагностика
6. Общественные науки

Лекция 5. Факторный анализ

При исследовании сложных объектов и систем часто невозможно измерить показатели, определяющие свойства этих объектов. Такие показатели называют факторами. Иногда нам неизвестны даже число и содержательный смысл этих факторов. Для измерений могут быть доступны иные величины, тем или иным способом зависящие от этих факторов. При этом, когда влияние неизвестного фактора проявляется в нескольких измеряемых признаках, эти признаки могут обнаруживать тесную связь между собой, поэтому общее число факторов может быть гораздо меньше, чем число измеряемых переменных-признаков, которое обычно выбирается исследователем в той или иной мере произвольно. Для обнаружения влияющих на измеряемые признаки факторов используются методы факторного анализа (ФА).

При анализе различных явлений и процессов часто необходимо учитывать большое число показателей, многие из которых связаны и дублируют друг друга. Нередко эти признаки лишь в косвенной мере отражают наиболее существенные, но не поддающиеся наблюдению и измерению внутренние явления.

Идея факторного анализа состоит в том, что из матрицы корреляции извлекается информация о взаимной зависимости признаков друг с другом. Если соответствующий коэффициент корреляции достаточно высок, то можно сказать, что эти признаки лишние и их можно заменить одним фактором. Всё это можно хорошо проиллюстрировать на графике линейной регрессии. **Если некоторый фактор**

сопоставить соответствующей линии регрессии, то можно говорить о снижении размерности. При этом новый фактор является линейной комбинацией прежних признаков.

Вообще, факторный анализ преследует две цели:

1. сокращение числа переменных (редукция данных) ;
2. классификацию переменных - определение структуры взаимосвязей между переменными.

Поэтому факторный анализ используется либо как метод сокращения данных, либо как метод классификации.

Существуют две модели факторного анализа:

1. метод главных компонент
2. анализ главных факторов.

Основное отличие этих методов в том, что в методе главных компонент мы допускаем, что должна учитываться вся дисперсия, в то время как в анализе главных факторов мы используем только общую дисперсию. В большинстве случаев эти два метода обычно дают очень близкие результаты. Тем не менее, метод главных компонент часто используется как метод снижения размерности, в то время как метод главных факторов часто предпочтителен, когда цель анализа — обнаружить структуру.

ФА можно применять только к количественным признакам! Использование порядковых, номинальных признаков для проведения ФА недопустимо. Это требование обусловлено тем, что входной информацией для ФА являются элементы ковариационной матрицы. Кроме того, представление переменных в виде линейной комбинации скрытых факторов и использование оценок факторов через линейные комбинации наблюдаемых переменных для порядковых переменных невозможно.

Сущность метода факторного анализа

Сущность метода факторного анализа – переход от описания некоторого множества объектов, заданного большим набором косвенных признаков, к описанию меньшим числом максимально информативных глубинных переменных. Такие переменные называются факторными и являются факторами исходных признаков. В ФА исходят из того, что признаки, входящие в исследуемый набор, не являются независимыми, напротив, они коррелированны.

Наличие корреляции между двумя или более признаками может трактоваться следующим образом: либо один из признаков определяет остальные, либо существует некий не включенный в набор скрытый параметр, оказывающий влияние на другие признаки. Такого рода скрытые параметры – о б щ и е факторы, а методы ФА предназначены для их выявления. Термин «общий» подчеркивает, что каждый фактор F_i имеет существенное значение для анализа всех переменных. В то же время

предполагается, что существуют и специфические или характерные факторы, каждый из которых влияет только на одну конкретную переменную.

Пример. Модель ФА можно интерпретировать в терминах интеллектуальных тестов.

p / n	Тесты				
	1	**2**	**3**	**…**	**p**
Экзаменуемый 1	оценка				
…					
Экзаменуемый n					

В качестве ненаблюдаемых общих факторов, от которых будут зависеть оценки по всем тестам, выступают такие факторы как характеристика общей одаренности, характеристики его математических, технических или гуманитарных способностей.

Постановка задачи

Пусть

X – p-мерный случайный вектор;

F – m-мерный вектор, компонентами которого являются непосредственно не наблюдаемые переменные (факторы), то есть

$$\mathbf{F}= (F_1 , \ldots, F_m);$$

U – вектор сумм ненаблюдаемых ошибок и специфических факторов. Согласно основному предположению факторного анализа, каждое конкретное измерение x_i может рассматриваться как сумма воздействий некоторого небольшого числа общих факторов $\{F_i\}$, взятых с определенными весами λ_{ij}, специфического фактора s_i, воздействующего только на данную переменную, и ошибки измерения e_i. Поскольку s_i и e_i в факторном анализе неразличимы, их обычно рассматривают как сумму $u_i=s_i+e_i$.

$\Lambda^{p*m}=\{\lambda_{ij}\}$ – матрица факторных нагрузок (i=1..p, j=1..m). Определим матрицу Λ порядка p*m как матрицу, элементами которой являются факторные веса, определяющие нагрузку i-ой переменной на j-ый фактор.

Примем следующие допущения:

1. Общие факторы не коррелированны и имеют единичные дисперсии.
2. Характерные факторы не коррелированны и $Du_i = \tau_i^2, \quad i = 1...p$, где τ_i^2 – специфическая дисперсия или специфичность i –ой исходной переменной.
3. Переменные f_i и u_i не коррелированны.
4. Зависимость признаков от общих факторов линейная.

Таким образом, каждый из признаков x_i, входящий в исследуемый набор, может быть представлен как функция небольшого числа общих факторов и специфического фактора:

$$x_i = f(F_1, F_2, ..., F_m, U),$$

или с учетом линейности модели:

$$x_1 = \sum_{j=1}^{m} \lambda_{1j} F_j + U_1, \quad \ldots \quad x_p = \sum_{j=1}^{m} \lambda_{pj} F_j + U_p.$$

Принимая во внимание допущения 1, 2, 3 получим:

$$Dx_i = \sum_{j=1}^{m} \lambda_{ij}^2 + \tau_i^2 + 2 \sum_{j<q=1}^{m} \lambda_{ij} \lambda_{iq} r_{F_i F_q} + 2\tau_i \sum_{j=1}^{m} \lambda_{ij} r_{F_j U_i} = \sum \lambda_{ij}^2 + \tau_i^2 = 1.$$

Первая сумма указывает долю дисперсии переменной x_i, приходящуюся на все общие факторы. Можно также записать:

$$Dx_i = h_i^2 + \tau_i^2$$

где h_i^2 – общность переменной x_i, τ_i^2 – характерность переменной x_i.

Техника факторного анализа направлена на ***оценку*** фактических нагрузок λ_{ij}, а также общих факторов с помощью значений исходных переменных. После того, как факторные нагрузки будут найдены, остается задача наилучшей интерпретации общих факторов. Для этого используется *метод вращения фактора*, который из-за субъективности является наиболее спорной частью факторного анализа.

Надо понимать, что на практике невозможно получить точную структуру факторной модели, можно только пытаться найти ***оценки*** параметров факторной структуры с использованием определенных статистических или практических критериев.

Необходимо отметить связь метода главных компонент и факторного анализа. В модели главных компонент вся дисперсия приписывается общим факторам, тогда как в факторном анализе дисперсия состоит из двух частей: дисперсии, обусловленной наличием общих факторов, и дисперсии, обусловленной специфичностью.

Порядок выполнения факторного анализа

На первом шаге процедуры факторного анализа происходит стандартизация заданных значений переменных (z-преобразование); затем при помощи стандартизированных значений рассчитывают корреляционные коэффициенты Пирсона между рассматриваемыми переменными.

Исходным элементом для дальнейших расчётов является корреляционная матрица. Для понимания отдельных шагов этих расчётов потребуются хорошие знания, прежде всего, в области операций над матрицами. Для построенной корреляционной матрицы определяются, так называемые, собственные значения и соответствующие им собственные векторы, для определения которых используются оценочные значения диагональных элементов матрицы (так называемые относительные дисперсии простых факторов).

Собственные значения сортируются в порядке убывания. После для чего обычно отбирается столько факторов, сколько имеется собственных значений, **превосходящих по величине единицу**. Собственные векторы, соответствующие этим собственным значениям, образуют факторы; элементы собственных векторов получили название **факторной нагрузки**. Их можно понимать как коэффициенты корреляции между соответствующими переменными и факторами. Для решения такой задачи определения факторов были разработаны многочисленные методы, наиболее часто употребляемым из которых является метод определения главных факторов (компонентов).

Описанные выше шаги расчёта ещё не дают однозначного решения задачи определения факторов. Основываясь на геометрическом представлении рассматриваемой задачи, поиск однозначного решения называют задачей вращения факторов. И здесь имеется большое количество методов, наиболее часто употребляемым из которых является ортогональное вращение по так называемому методу варимакса. **Факторные нагрузки повёрнутой матрицы могут рассматриваться как результат выполнения процедуры факторного анализа**. Кроме того, на основании значений этих нагрузок необходимо попытаться дать толкование отдельным факторам.

Если факторы найдены и истолкованы, то на последнем шаге факторного анализа, отдельным наблюдениям можно присвоить значения этих факторов, так называемые факторные значения. Таким образом, для каждого наблюдения значения большого количества переменных можно перевести в значения небольшого количества факторов.

Этапы факторного анализа

При решении задач факторного анализа выделяют этапы:

1. Формулировка проблемы
2. Подготовка исходной матрицы данных (ТЭД). Расчет соответствующей матрицы взаимосвязей признаков.
3. Выделение первоначальных ортогональных факторов (факторизация).
4. Вращение – преобразование факторов, облегчающее их интерпретацию.
5. Интерпретация данных.
6. Оценка значений факторов. Подсчет факторных значений по каждому фактору для каждого наблюдения.

В ходе выполнения ФА некоторые этапы можно опустить. Например, в качестве исходной матрицы данных можно использовать корреляционную матрицу или эквивалентную ей любую другую матрицу связей, подсчитанную в ходе какой-то другой вычислительной процедуры. Тогда работа сразу начинается с третьего этапа.

Кроме того, так как нет наблюдений, невозможно и произвести оценку значений факторов. Именно этот случай рассмотрен в одном из примеров ниже.

1. Формулировка проблемы

Формулировка проблемы включает несколько задач. Во-первых, четкое определение целей факторного анализа. Переменные, подвергаемые факторному анализу, задаются исходя из прошлых исследований, теоретических выкладок и по усмотрению исследователя. Важно, чтобы переменные измерялись в интервальной или относительной шкале. Выборка должна быть подходящего размера. Опыт подсказывает, что рекомендуется брать выборку, по крайней мере, в четыре или пять раз больше, чем число переменных. Часто при исследованиях размер выборки мал, и это отношение значительно меньше. В таких случаях следует осторожно интерпретировать результаты.

2. Вычисление матрицы взаимосвязей признаков

В основе анализа чаще всего лежит матрица корреляций между переменными. Ее анализ дает исследователям ценную информацию. Целесообразность выполнения факторного анализа определяется наличием корреляций между переменными. На практике так обычно и бывает. Если же корреляции между всеми переменными небольшие, то факторный анализ бесполезен. Следует также ожидать, что переменные, тесно взаимосвязанные между собой, должны также тесно коррелировать с одним и тем же фактором или факторами.

Для проверки **целесообразности использования факторной модели** анализа зависимости переменных существует несколько статистик. С помощью критерия сферичности Бартлетта проверяется нулевая гипотеза об отсутствии корреляций между переменными в генеральной совокупности: другими словами, рассматривается утверждение о том, что корреляционная матрица совокупности — это единичная матрица, в которой все диагональные элементы равны 1, а все остальные равны 0. Проверка с помощью критерия сферичности основана на преобразовании детерминанта корреляционной матрицы в статистику хи-квадрат. При большом значении статистики нулевую гипотезу отклоняют. Если же нулевую гипотезу не отклоняют, то целесообразность выполнения факторного анализа вызывает сомнения. Другая полезная статистика — критерий адекватности выборки Кайзера— Мейера—Олкина (КМО). Данный коэффициент сравнивает значения наблюдаемых коэффициентов корреляции со значениями частных коэффициентов корреляции. Небольшие значения КМО-статистики указывают на то, что корреляции между парами переменных нельзя объяснить другими переменными и что использование факторного анализа нецелесообразно.

Пример. В таблице ниже приведена корреляционная матрица между 8-ю признаками (размерами) человеческого тела. Как видно из анализа таблицы многие признаки имеют сильные корреляционные связи (красным цветом отмечены связи с r>0.7) .

Признаки		Коэффициенты корреляции между признаками							
		1	2	3	4	5	6	7	8
1	Рост	1	0,85	0,81	0,86	0,47	0,4	0,3	0,38
2	Размах рук		1	0,88	0,83	0,28	0,33	0,28	0,42
3	Длина предплечья			1	0,8	0,38	0,32	0,24	0,35
4	Длина ноги				1	0,44	0,33	0,33	0,37
5	Вес					1	0,76	0,73	0,33
6	Окружность бедер						1	0,58	0,58
7	Окружность груди							1	0,54
8	Ширина груди								1

3. Определение главных факторов (факторизация)

Первой задачей факторного анализа является определение по ковариационной (или корреляционной) матрице $S = \{s_{ij}\}$ (или $R = \{r_{ij}\}$) оценок $\hat{\lambda}_{ij}$ факторных нагрузок и оценок специфичности $\hat{\tau}_i^2$. Для определения искомых оценок существует ряд методов:

- метод главных факторов,
- центроидный метод,
- метод наибольшего правдоподобия.

Все эти методы строятся как методы решения задач на поиск экстремума некоторого критерия качества матрицы факторных нагрузок Λ при определенных ограничениях. Различие между методами определяется видом используемых критериев.

Чаще всего используется *метод главных факторов*. В качестве исходной информации задают:

1. Число общих факторов.
2. Вид матрицы, к которой следует применить факторный анализ.

Число общих факторов определяется целым числом **m** или постоянной константой **c**. В последнем случае **m** полагается равным числу собственных значений, превосходящих **c**. При определении числа **m** можно руководствоваться критериями:

1. ***Критерий Кайзера или критерий собственных чисел***. Этот критерий предложен Кайзером, и является, вероятно, наиболее широко используемым. Отбираются только факторы с собственными значениями равными или большими 1. Это означает, что если фактор не выделяет дисперсию, эквивалентную, по крайней мере, дисперсии одной переменной, то он опускается.

2. ***Критерий каменистой осыпи или критерий отсеивания***. Он является графическим методом, впервые предложенным психологом Кэттелом. Собственные значения возможно изобразить в виде простого графика. Кэттел предложил найти такое место на графике, где убывание собственных значений слева направо максимально замедляется. Предполагается, что справа от этой точки находится только «факториальная осыпь» — «осыпь» является геологическим термином, обозначающим обломки горных пород, скапливающиеся в нижней части скалистого склона. Однако этот критерий отличается высокой субъективностью и, в отличие от предыдущего критерия, статистически необоснован.

Недостатки обоих критериев заключаются в том, что первый иногда сохраняет слишком много факторов, в то время как второй, напротив, может сохранить слишком мало факторов; однако оба критерия вполне хороши при нормальных условиях, когда имеется относительно небольшое число факторов и много переменных. На практике возникает важный вопрос: когда полученное решение может быть содержательно интерпретировано. В этой связи предлагается использовать ещё несколько критериев.

3. ***Критерий значимости***. Он особенно эффективен, когда модель генеральной совокупности известна и отсутствуют второстепенные факторы. Но критерий непригоден для поиска изменений в модели и реализуем только в факторном анализе по методу наименьших квадратов или максимального правдоподобия.
4. ***Критерий доли воспроизводимой дисперсии***. Факторы ранжируются по доле детерминируемой дисперсии, когда процент дисперсии оказывается несущественным, выделение следует остановить. Желательно, чтобы выделенные факторы объясняли более 80 % разброса. Недостатки критерия: во-первых, субъективность выделения, во-вторых, специфика данных может быть такова, что все главные факторы не смогут совокупно объяснить желательного процента разброса. Поэтому главные факторы должны вместе объяснять не меньше 50,1 % дисперсии.
5. ***Критерий интерпретируемости и инвариантности***. Данный критерий сочетает статистическую точность с субъективными интересами. Согласно ему, главные факторы можно выделять до тех пор, пока будет возможна их ясная интерпретация. Она, в свою очередь, зависит от величины факторных нагрузок, то есть если в факторе есть хотя бы одна сильная нагрузка, он может быть интерпретирован. Возможен и обратный вариант — если сильные нагрузки имеются, однако интерпретация затруднительна, от этой компоненты предпочтительно отказаться.

Напомним, что

- Сумма собственных чисел равна числу переменных, то есть p.
- Общая дисперсия может быть вычислена как след ковариационной матрицы (то есть равна сумме ее диагональных элементов).

Тогда можно прийти к выводу, что для корреляционной матрицы общая дисперсия будет равна сумме собственных чисел, то есть p. Тогда, разделив собственное

число на число переменных, можем получить долю дисперсии, соответствующую данному фактору или компоненте:

$$Доля, соответствующая \cdot данной \cdot компоненте = \frac{Соответствующее \cdot собственное \cdot число}{p}$$

При проведении факторного анализа все расчеты носят последовательный характер. Процедура выполнения вычислительных операций схематично представлена на рис. 1.

При этом приводится подробное описание важнейших матриц. Вертикальные стрелки соответствуют основным проблемам, возникающим при проведении факторного анализа в тех местах схемы, куда указывают эти стрелки.

Любой метод факторного анализа начинается с Y матрицы исходных данных. По ней вычисляется корреляционная матрица R. По главной диагонали корреляционной матрицы затем проставляют оценки общностей и получают $R_h = (r^2_{ii})$. Это составляет проблему общности , которая состоит в установлении оценок общностей . Это самая первая проблема, которая возникает в ходе факторного анализа. Для ее решения в качестве общностей выбираем наибольшие значения элементов в каждом столбце матрицы R. Стрелка между R_h и A указывает на проблему факторов. Из R_h с помощью определенных способов извлекают факторы, получая в результате матрицу A. Столбцы матрицы A ортогональны и занимают

произвольную позицию в отношении переменных, определяемую методом выделения факторов. Матрица A воспроизводит R_h по равенству $R_h = A \cdot A'$

Выводы

1. На данном этапе определяется минимальное число факторов, адекватно воспроизводящих наблюдаемые корреляции, а также значения общностей каждой переменной.
2. Применение указанных методов приводит к набору ортогональных факторов, упорядоченных в порядке убывания их значимости.

Эти ограничения принимаются, чтобы обеспечить единственность решения. В результате этих ограничений

- Факторная сложность переменных, скорей всего будет больше единицы, независимо от вида истинной факторной структуры, то есть переменные будут иметь нагрузки более чем на один фактор (другими словами зависеть не от одного, а нескольких факторов);
- Все факторы, за исключением первого, являются биполярными, другими словами, некоторые переменные должны иметь положительную нагрузку на этот фактор, а некоторые – отрицательную.

Пример. Результаты факторного анализа

Собственные значения
Выделение: Главные компоненты

	Соб. зн.	% общей	Кумулятивн. собст. знач.	Кумулятивн. %
1	4,597349	57,46686	4,597349	57,46686
2	1,723141	21,53926	6,320489	79,00612

Фактор.нагрузки (без вращ.)
Выделение: Главные компоненты (Отмечены нагрузки >0,7)

	Фактор 1	Фактор 2
Рост	-0,873232	0,343122
Размах рук	-0,842998	0,444261
Длина предплечья	-0,828256	0,429134
Длина ноги	-0,855072	0,360417
Вес	-0,696270	-0,530817
Окружность бедер	-0,673535	-0,569179
Окружность груди	-0,615703	-0,616589
Ширина груди	-0,624085	-0,328739
Общ.дис.	**4,597349**	**1,723141**
Доля общ	**0,574669**	**0,215393**

Факторные нагрузки могут интерпретироваться как корреляции между соответствующими переменными и факторами – чем выше нагрузка по модулю, тем больше близость фактора к исходной переменной; таким образом, они представляют наиболее важную информацию для интерпретации полученных факторов. В сгенерированной таблице для облегчения трактовки выделены факторные нагрузки по абсолютной величине больше $0{,}7$.

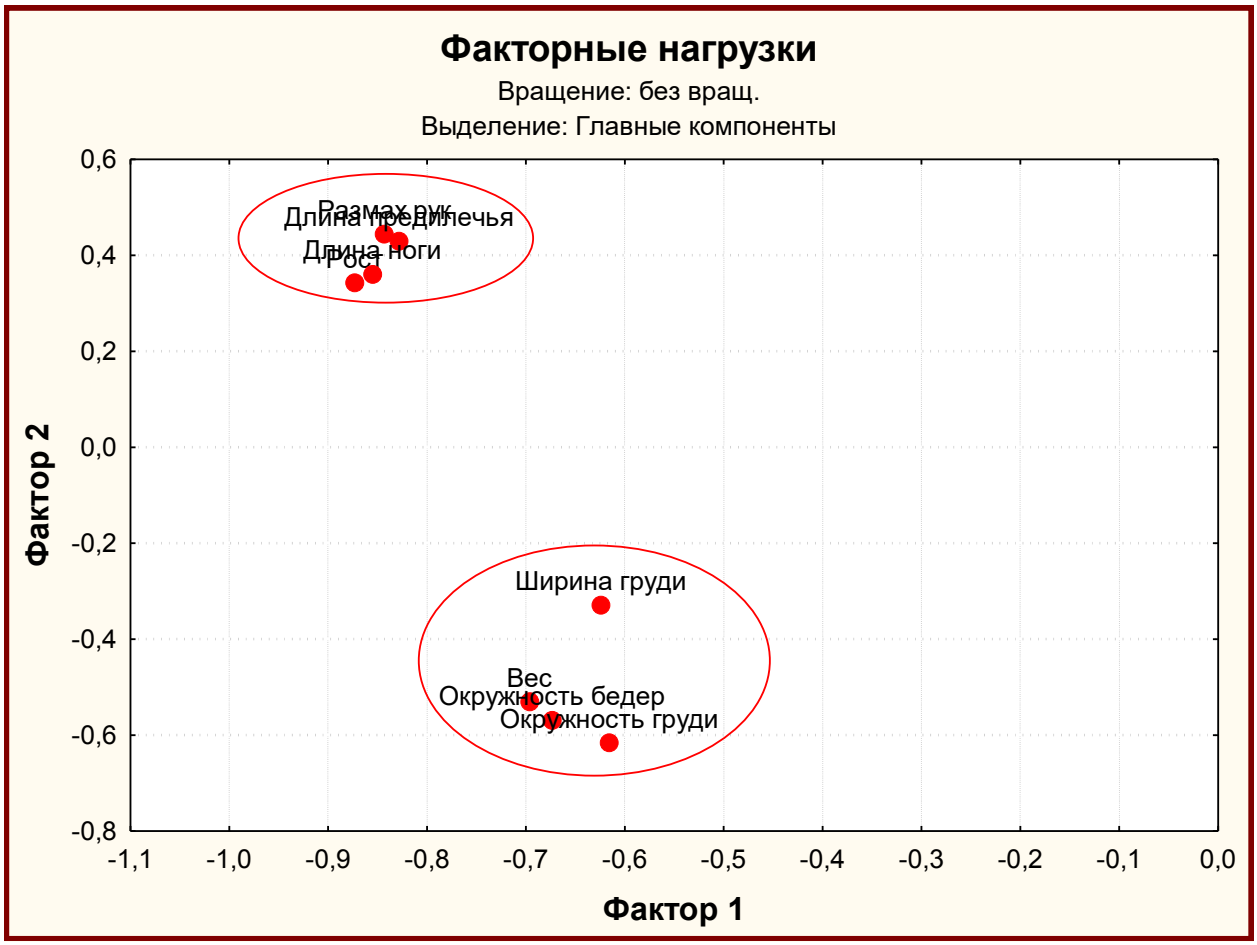


Рис.13.1.б Агрегирование некоторых признаков человеческого тела в результате факторного анализа

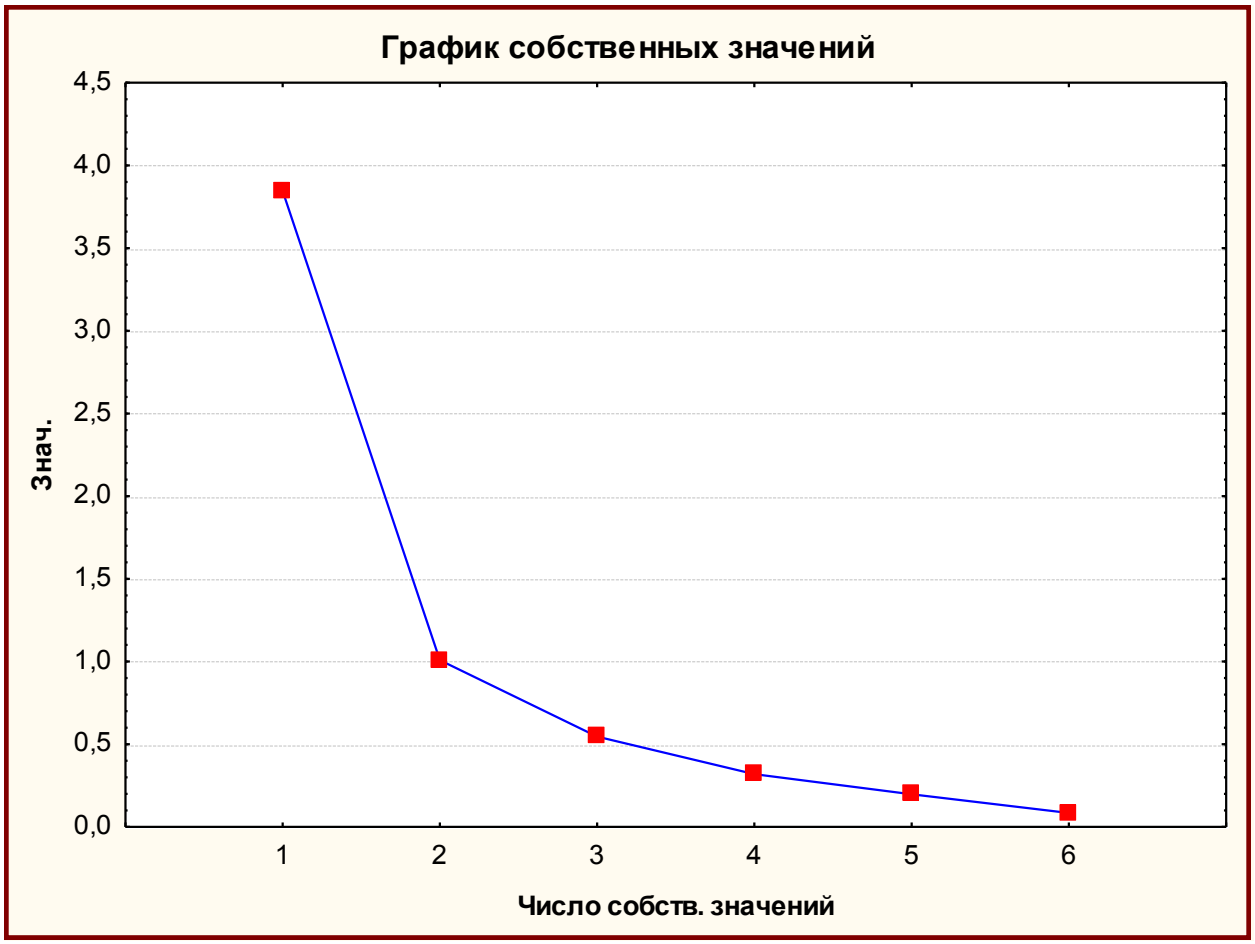
Результаты факторного анализа приведены на рисунке, из которого видно, что анализируемые в примере признаки человеческого тела образуют два хорошо выраженные агрегата: *рост, размах рук, длина предплечий и длина ног* образуют один агрегат, а *вес, окружность бедер, окружность груди и ширина груди* – второй

Общности
Выделение: Главные компоненты Вращение: без вращ.

	Из 1	Из 2	Множест.
Рост	0,762535	0,880267	0,840368
Размах рук	0,710645	0,908013	0,901323
Длина предплечья	0,686008	0,870164	0,833255

Длина ноги	0,731148	0,861048	0,800860
Вес	0,484792	0,766559	0,880394
Окружность бедер	0,453649	0,777614	0,775985
Окружность груди	0,379090	0,759272	0,748005
Ширина груди	0,389482	0,497552	0,618233

График каменистой осыпи



4. Вращение факторов

Следующим шагом факторного анализа является интерпретация каждого фактора. Для этого можно воспользоваться неоднозначностью определения факторов. Полученные факторы $F_1,...,F_m$ можно заменить их линейными комбинациями, которые взаимно не коррелированны и имеют единичные дисперсии. Таким образом, имеется бесконечное множество наборов факторов, удовлетворяющих данной модели.

Аналогично тому, как мы отображаем объект точкой в пространстве признаков, можно отобразить каждый признак точкой в пространстве факторов. При этом

проекции точки на факторы – есть величины факторных нагрузок. Представление всех переменных в пространстве общих факторов называется ***конфигурацией***.

Ниже на рисунках представлены соответственно случайная, простая ортогональная и косоугольная структуры расположения наблюдений в пространстве общих факторов F_1 и F_2..

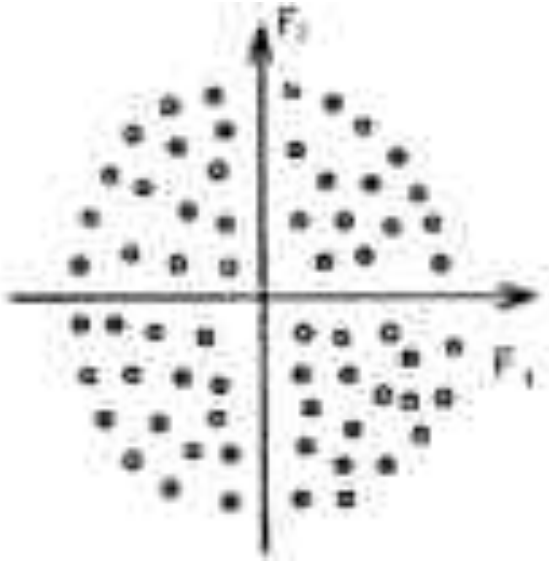
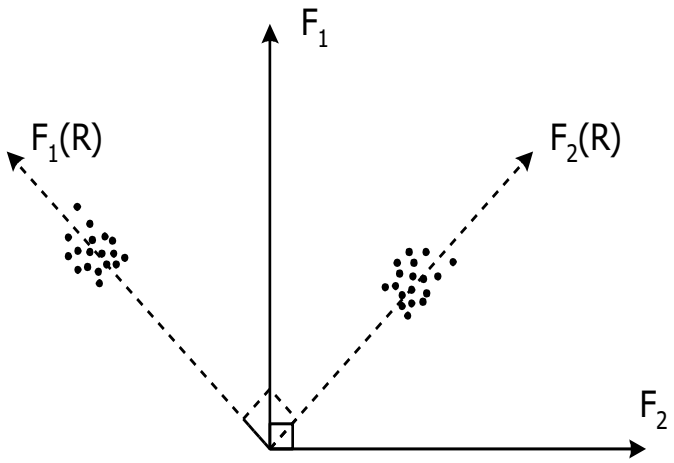


Рис. Случайная конфигурация векторов

Рис. Ортогональная простая структура



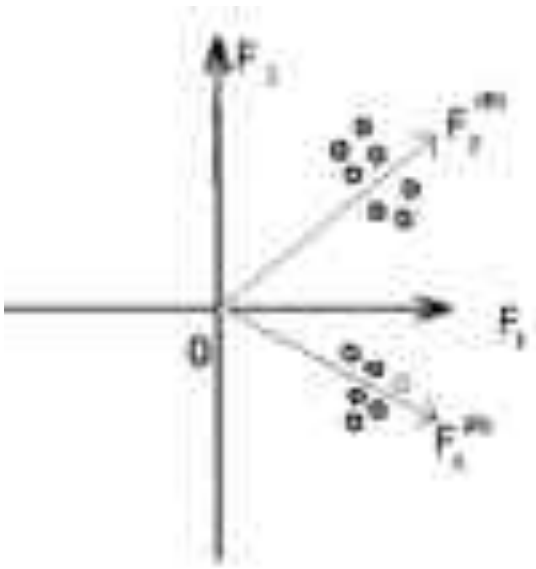


Рис. Косоугольная простая структура

Интерпретация факторов проводится с помощью процедуры вращения. При этом число факторов и значения общностей фиксируются. Будем вращать систему координат вокруг ее начала. При таком вращении остаются неизменными расстояния от точек до начала координат (длины векторов) и углы между векторами.

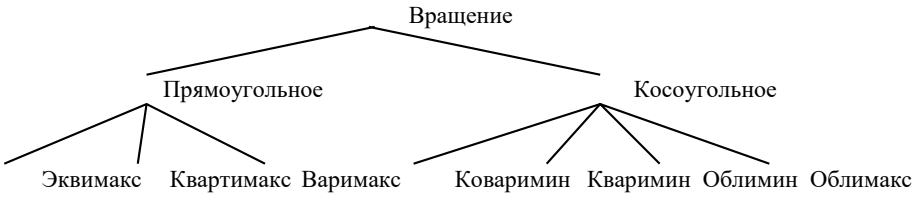
Целью всех вращений является получение наиболее простой факторной структуры. Хотя трудно определить минимальные требования к простой структуре, но если взять число факторов ***r*** и число переменных ***n***, то всегда можно сказать, какая структура наиболее проста. Факторная структура является простейшей, если когда все переменные имеют факторную сложность, равную 1, то есть когда каждая переменная имеет ненулевую нагрузку только на один общий фактор. При случайной конфигурации наблюдений невозможно получить такую структуру.

Существуют три различных подхода к проблеме вращения:

❶ Первый подход – графический. В этом случае исходные переменные рассматриваются как точки в факторном пространстве, координаты которых равны нагрузкам на факторы, а размерность определяется числом факторов. Вращение заключается в проведении новых осей, которые соответствуют некоторому критерию простой, легко интерпретируемой структуры. Если в пространстве факторов есть явные скопления (кластеры) точек (переменных), легко отделяемые друг от друга, простая структура получается в том случае, когда оси проведены через эти скопления. Но если такое разделение не очевидно или число факторов велико, графический метод неприменим.

❷ Второй подход связан с аналитическими методами. В этом случае выбирается некоторый объективный критерий, которым надо руководствоваться при выполнении вращения. В рамках этого подхода различают два вида вращения – ортогональное и косоугольное.

Существуют различные варианты ортогонального и косоугольного вращений:



После прямоугольного вращения модель может быть записана:

$$x_i = \sum_{j=1}^{m} c_{ij} F_j^{(R)} + U_i \ ,$$

где $i = 1,...,p$, c_{ij} – нагрузки постоянных факторов.

В результате ортогонального вращения факторов общность каждой переменной x_i остается без изменений:

$$c_{ij} = \sum_{k=1}^{m} \lambda_{ij} q_{kj}$$

Для облегчения интерпретации факторов постоянные q_{kj} выбираются так, чтобы результирующие нагрузки имели простую структуру. Структура фактических нагрузок считается простой, если большинство из C_{ij} не слишком сильно отличаются от 0.

При косоугольном вращении исходят из предположения, что важнее получить простую структуру факторных нагрузок, чем сохранить ортогональность факторов. Поэтому условие некоррелированности факторов ослабляется, и ищутся коррелированные факторы $F_1^{(R)},...,F_m^{(R)}$ с единичными дисперсиями, являющимися линейными комбинациями факторов $F_1,...,F_m$. Так как полученные факторы могут быть коррелированными, имеется более широкая область изменения $q_{k.j}$.

❸ Третий поход заключается в задание априорной целевой матрицы. Цель вращения – нахождение факторного отображения, наиболее близкого к некоторой заданной матрице.

Пример.

Факторные нагрузки (Варимакс нормал.) *Выделение: Главные компоненты (Отмечены нагрузки >,7)*		
	Фактор 1	**Фактор 2**
Рост	0,898597	0,269798
Размах рук	0,937320	0,171595

Длина предплечья	0,916382	0,174380
Длина ноги	0,895005	0,244978
Вес	0,219446	0,847586
Окружность бедер	0,177866	0,863700
Окружность груди	0,103101	0,865241
Ширина груди	0,287556	0,644099
Общ.дис.	**3,499945**	**2,820544**
Доля общ	**0,437493**	**0,352568**

Общности

Выделение: Главные компоненты Вращение: Варимакс нормал.

	Из 1	**Из 2**	**Множест.**
Рост	0,807476	0,880267	0,840368
Размах рук	0,878569	0,908014	0,901323
Длина предплечья	0,839755	0,870164	0,833255
Длина ноги	0,801034	0,861048	0,800860
Вес	0,048156	0,766559	0,880394
Окружность бедер	0,031636	0,777614	0,775985
Окружность груди	0,010630	0,759272	0,748005
Ширина груди	0,082689	0,497552	0,618233

Воспроизведенные корреляции

Выделение: Главные компоненты

	Рост	**Размах рук**	**Длина пред-плечья**	**Длина ноги**	**Вес**	**Окружность бедер**	**Окруж-ность груди**	**Ширина груди**
Рост	0,88	0,89	0,87	0,87	0,43	0,39	0,33	0,43
Размах рук	0,89	0,91	0,89	0,88	0,35	0,31	0,25	0,38
Длина пред-плечья	0,87	0,89	0,87	0,86	0,35	0,31	0,25	0,38
Длина ноги	0,87	0,88	0,86	0,86	0,40	0,37	0,30	0,42
Вес	0,43	0,35	0,35	0,40	0,77	0,77	0,76	0,61
Окружность бедер	0,39	0,31	0,31	0,37	0,77	0,78	0,77	0,61
Окружность груди	0,33	0,25	0,25	0,30	0,76	0,77	0,76	0,59
Ширина груди	0,43	0,38	0,38	0,42	0,61	0,61	0,59	0,50

Остаточные корреляции *Выделение: Главные компоненты (Отмеченные остатки > 0,10)*								
	Рост	**Размах рук**	**Длина пред-плечья**	**Длина ноги**	**Вес**	**Окружность бедер**	**Окружность груди**	**Ширина груди**
Рост	0,12	-0,04	-0,06	-0,01	0,04	0,01	-0,03	-0,05
Размах рук	-0,04	0,09	-0,01	-0,05	-0,07	0,02	0,03	0,04
Длина пред-плечья	-0,06	-0,01	0,13	-0,06	0,03	0,01	-0,01	-0,03
Длина ноги	-0,01	-0,05	-0,06	0,14	0,04	-0,04	0,03	-0,05
Вес	0,04	-0,07	0,03	0,04	0,23	-0,01	-0,03	-0,28
Окружность бедер	0,01	0,02	0,01	-0,04	-0,01	0,22	-0,19	-0,03
Окружность груди	-0,03	0,03	-0,01	0,03	-0,03	-0,19	0,24	-0,05
Ширина груди	-0,05	0,04	-0,03	-0,05	-0,28	-0,03	-0,05	0,50

5. Интерпретация факторов

Целью процедуры вращения является представление каждой исходной переменной x_i одним, или небольшим числом факторов. Нагрузки остальных факторов равны 0. Тогда задача интерпретации значительно облегчается, так как каждая нагрузка при использовании матрицы корреляции R, равна корреляции между исходной переменной и соответствующим фактором.

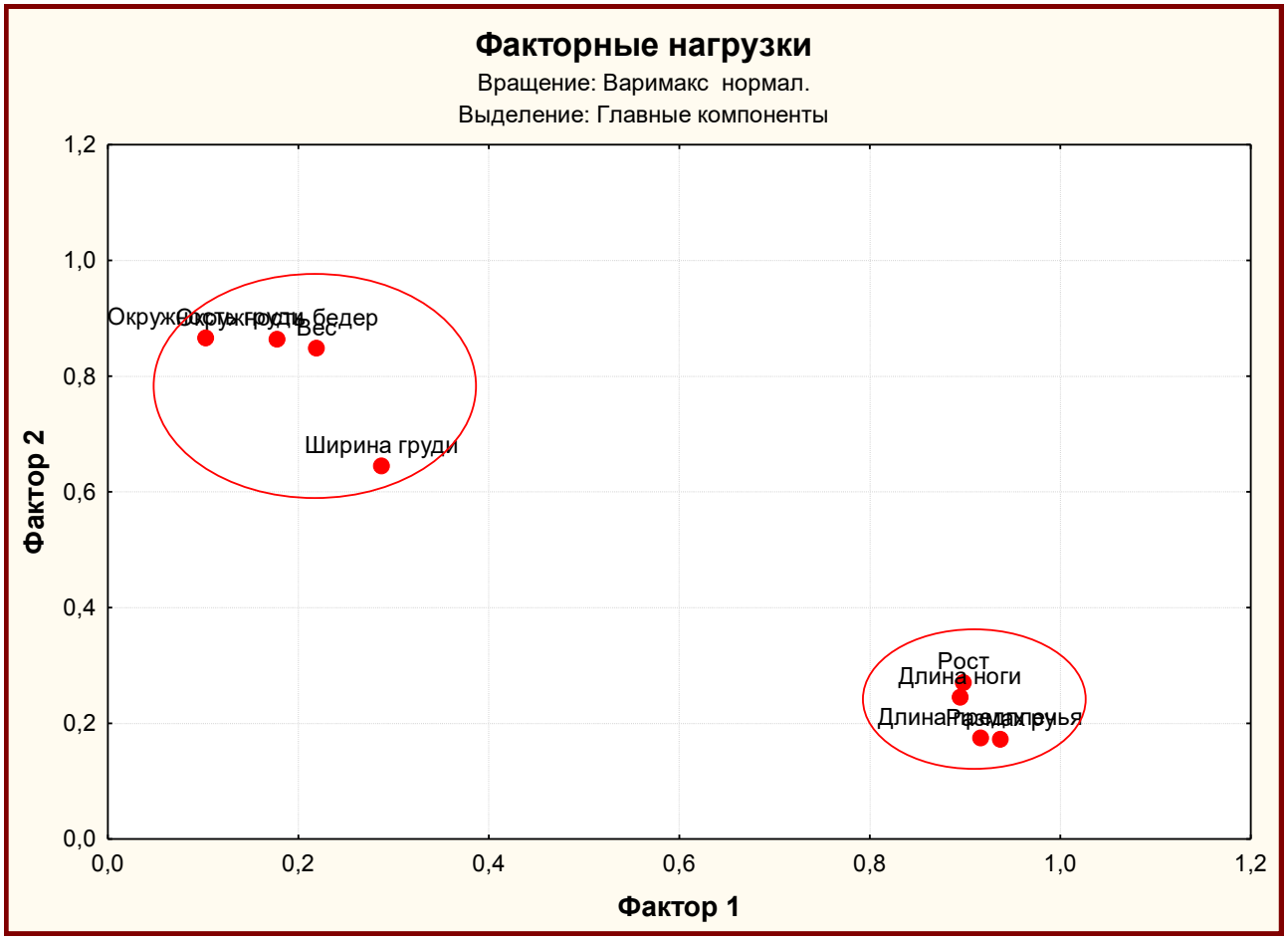
Как же интерпретировать факторы после вращения?

Для интерпретации факторов необходимо определить переменные, которые имеют высокие значения нагрузок по одному и тому же фактору. А затем этот фактор следует проанализировать с учетом этих переменных. Другое полезное средство интерпретации — графическое изображение переменных, координатами которых служат величины факторных нагрузок. Так, в конце оси расположены переменные, которые имеют большие нагрузки только в связи с этим фактором и, следовательно, характеризуют его. Переменные в начале координат имеют небольшие нагрузки в связи с обоими факторами. Переменные, расположенные вдали от осей, связаны с обоими факторами. Если фактор нельзя четко определить с точки зрения связи с исходными переменными, то его следует пометить как неопределяемый или генеральный (общий для всех переменных).

Если нет возможности провести вербальное объяснение факторов, то факторный анализ можно считать неудавшимся.

Пример. Исходя из анализа факторных нагрузок можно полагать, что значения перечисленных признаков человеческого тела в сущности являются проявлениями

двух других признаков (факторов) - ***стройности*** (F1) и ***полноты*** (F2), причем, если на содержательном уровне понятия стройности и полноты ясны любому из нас, то формализованных описаний этих свойств человеческого тела, тем более поддавшихся бы непосредственному инструментальному измерению до сих пор не существует.



6. Оценка значений общих факторов – факторное шкалирование

После интерпретации факторов необходимо вычислить их значения. Мало установить лишь сам факт существования небольшого числа скрыто существующих факторов, объясняющих природу взаимной коррелированности исходных признаков, и основную часть их дисперсии. Во многих случаях требуется определить значения факторов для данного вектора $X=(x^{(1)},\ldots,x^{(p)})$.

Под факторным шкалированием понимается процедура, позволяющая присвоить каждому объекту некоторые числовые оценки значений выделенных факторов, используя значения наблюдаемых переменных для этого объекта.

Для проведения подобной процедуры есть следующие основания:

1. после определения скрытой факторной структуры измеряемых данных для объектов исследователю может понадобиться представить каждый их этих объектов в терминах значений факторов, а не измеряемых переменных;
2. может появиться необходимость использования одного или более факторов в качестве переменных для дальнейшего анализа. ФА часто применяется в качестве средства для создания новых факторных переменных (шкал) для других исследований, чем для изучения самой скрытой структуры.

Главной целью факторного шкалирования является определение значений фактора (F) через наблюдаемые переменные (X_1, X_2, …). Общий фактор невозможно точно выразить посредством наблюдаемых переменных, поскольку каждая из них содержит также и характерную переменную, которую нельзя отделить от всей переменной. Можно получить лишь оценку значений общих факторов ($\hat{F}$) через наблюдаемые переменные. Поэтому шкалирование факторов всегда связано с некоторой неопределенностью.

Пример. Рассмотрим однофакторную модель с тремя переменными. Допустим, что все факторные нагрузки одинаковы (или что все коэффициенты корреляции равны). Этот пример показан на рисунке слева.

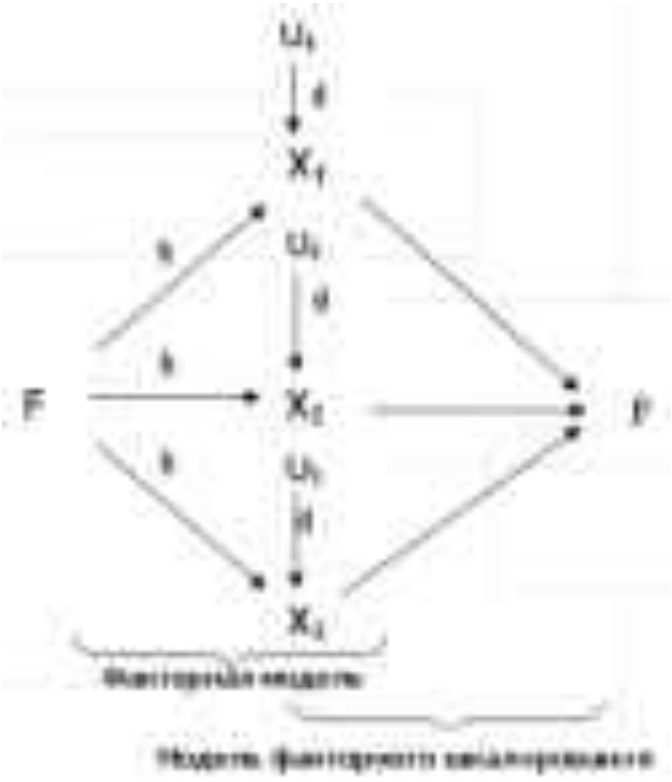


Рис. Графическая модель, иллюстрирующая зависимость между фактором и его оценкой

Для такой модели вычислить наблюдаемые коэффициенты корреляции между переменными можно с помощью перемножения факторных нагрузок. Так как все факторные нагрузки одинаковы, коэффициент корреляции будет равен квадрату факторной нагрузки:

$$r_{ij} = b_i b_j = b^2_i = b^2_j = h^2$$

В качестве оценки значения фактора берется линейная комбинация признаков X_1 , X_2 ,X_3. Так как каждая из этих переменных имеет одинаковую нагрузку от общих факторов, то естественно сложить их, беря соответствующие значения с одинаковым весом. Окончательное выражение будет иметь вид

$$\hat{F} = X_1 + X_2 + X_3,$$

а соответствующая диаграмма представлена в правой части рисунка.

Стандартного метода оценки значений факторов не существует. Как правило, для этой цели используется техника регрессионного анализа. Фактор представляет собой линейную комбинацию исходных переменных. Если рассматривать факторы как зависимые переменные, а исходные переменные x_i, i=1…p, считать независимыми, то можно записать следующие уравнения:

$$\hat{F}_j = \sum_{i=1}^{p} b_{ij} z_i \quad \text{j=1,…,m}$$

где $\hat{F}_j$ - оценка значения j-го параметра,

z_i – стандартизованная оценка i - ой переменной;

b_{ij} - оценки коэффициентов регрессии, иногда они называются коэффициентами значений факторов.

Веса или коэффициенты значения фактора, используемые для объединения нормированных переменных, получают из матрицы коэффициентов значения фактора. Большинство компьютерных программ позволяет вычислить значения факторов. Только в анализе главных компонент можно вычислить точные значения факторов. Более того, в анализе главных компонент эти значения не взаимосвязаны. В анализе общих факторов оценки значений факторов получают, но нет гарантии, что факторы не будут коррелировать между собой. Значения факторов можно использовать вместо исходных переменных в последующем многомерном анализе. Например, используя матрицу коэффициентов значения фактора в табл. 19.3, можно вычислить два значения фактора для каждого респондента. Если, нормированные значения переменной умножить на соответствующий коэффициент значения фактора, то получится значение данного фактора.

Определение подгонки модели

Последняя стадия факторного анализа заключается в определении соответствия модели факторного анализа исходным данным, т.е. степени ее подгонки. Основное допущение, лежащее в основе факторного анализа, состоит в том, что наблюдаемая корреляция между переменными может быть свойственна общим факторам. Следовательно, корреляции между переменными можно вывести или воспроизвести из определенных корреляций между переменными и факторами. Изучив разности между наблюдаемыми корреляциями (данными в исходной корреляционной матрице) и вычисленными корреляциями (определенными из матрицы факторных нагрузок), можно определить соответствие модели исходным данным. Эти разности называют остатками (residuals). Если много остатков с большими значениями, то факторная модель не обеспечивает хорошее соответствие данным и требует пересмотра.

Лекция 6. Классификация

Задача классификации — задача, в которой имеется множество *объектов* (ситуаций), разделённых, некоторым образом, на *классы*. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется *выборкой*. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный *классифицировать* произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование) класса, к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

В математической статистике задачи классификации называются также задачами дискриминантного анализа. В машинном обучении задача классификации решается, в частности, с помощью методов искусственных нейронных сетей при постановке эксперимента в виде обучения с учителем.

Существуют также другие способы постановки эксперимента —обучение без учителя, но они используются для решения другой задачи — кластеризации или таксономии. В этих задачах разделение объектов обучающей выборки на классы не задаётся, и требуется классифицировать объекты только на основе их сходства друг с другом. В некоторых прикладных областях, и даже в самой математической статистике, из-за близости задач часто не различают задачи кластеризации от задач классификации.

Некоторые алгоритмы для решения задач классификации комбинируют обучение с учителем с обучением без учителя, например, одна из версий нейронных сетей Кохонена — сети векторного квантования, обучаемые с учителем.

Типы входных данных

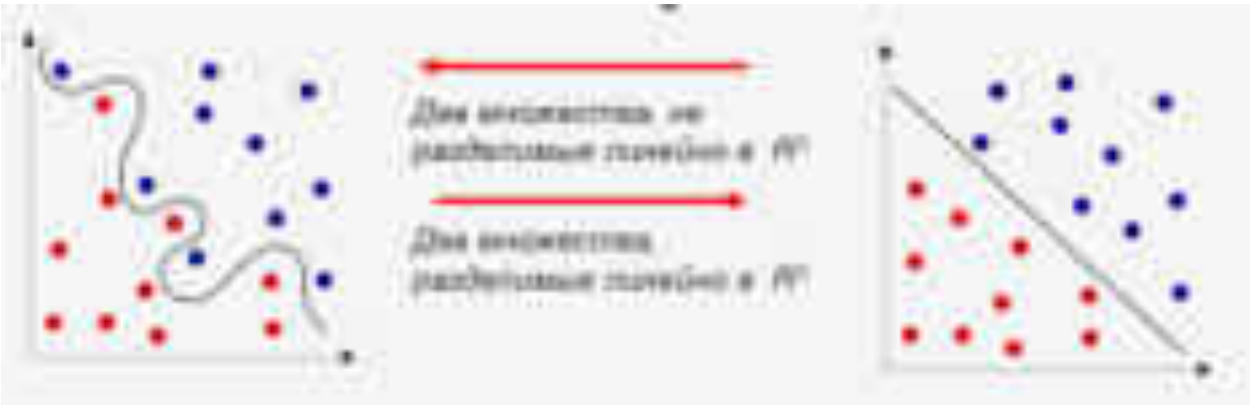
- **Признаковое описание** — наиболее распространённый случай. Каждый объект описывается набором своих характеристик, называемых *признаками*. Признаки могут быть числовыми или нечисловыми.
- **Матрица расстояний между объектами**. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки. С этим типом входных данных работают немногие методы, в частности, метод ближайших соседей, метод парзеновского окна, метод потенциальных функций.
- **Временной ряд** или **сигнал** представляет собой последовательность измерений во времени. Каждое измерение может представляться числом, вектором, а в общем случае — признаковым описанием исследуемого объекта в данный момент времени.
- **Изображение** или **видеоряд**.
- Встречаются и более сложные случаи, когда входные данные представляются в виде **графов**, текстов, результатов запросов к базе данных, и т. д. Как правило, они приводятся к первому или второму случаю путём предварительной обработки данных и извлечения признаков.
- Классификацию сигналов и изображений называют также распознаванием образов.

Типы классов

- **Двухклассовая классификация**. Наиболее простой в техническом отношении случай, который служит основой для решения более сложных задач.
- **Многоклассовая классификация**. Когда число классов достигает многих тысяч (например, при распознавании иероглифов или слитной речи), задача классификации становится существенно более трудной.
- **Непересекающиеся классы**.
- **Пересекающиеся классы**. Объект может относиться одновременно к нескольким классам.
- **Нечёткие классы**. Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

Линейная сепарабельность

Два множества точек в двумерном пространстве называются линейно сепарабельными (линейно разделимыми), если они могут быть полностью отделены единственной прямой. Для n-мерного пространства два набора точек линейно разделимы, если они могут быть отделены (n−1)-мерной гиперплоскостью.



В математических терминах: пусть X_0 и X_1 — два множества точек в n-мерном пространстве. Тогда X_0 и X_1 линейно разделимы, если существует n+1 действительных чисел $\omega_1, \omega_2, \ldots \omega_{n+1}$, таких, что каждая точка $x \in X_0$ удовлетворяет

$$\sum_{i=1}^{n} w_i x_i \geq w_{n+1}$$

и каждая точка $x \in X_1$ удовлетворяет

$$\sum_{i=1}^{n} w_i x_i < w_{n+1}$$

где x_i — i-й компонент x.

Линейный классификатор

Линейный классификатор — способ решения задач классификации, когда решение принимается на основании линейного оператора над входными данными. Класс задач, которые можно решать с помощью линейных классификаторов, обладают, соответственно, свойством линейной сепарабельности.

Пусть вектор $\boldsymbol{x}$ из действительных чисел представляет собой входные данные, а на выходе классификатора вычисляется показатель $\boldsymbol{y}$ по формуле:

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right)$$

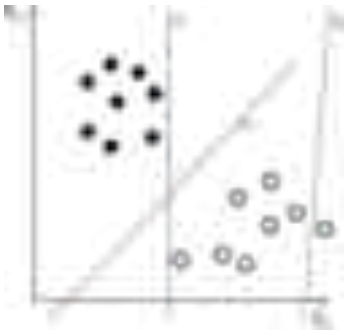
здесь $\boldsymbol{\omega}$ - действительный вектор весов, а $\boldsymbol{f}$ - функция преобразования скалярного произведения.

Иными словами, вектор весов $\boldsymbol{\omega}$ - ковариантный вектор или линейная форма отображения $\boldsymbol{x}$ в R. Значения весов вектора $\boldsymbol{\omega}$ определяются в ходе машинного обучения на подготовленных образцах. Функция $\boldsymbol{f}$ обычно простая пороговая функция,

отделяющая один класс объектов от другого. В более сложных случаях функция ***f*** имеет смысл вероятности того или иного решения.

Операцию линейной классификации для двух классов можно себе представить как отображение объектов в многомерном пространстве на гиперплоскость, в которой те объекты, которые попали по одну сторону разделяющей линии, относятся к первому классу ("да"), а объекты по другую сторону - ко второму классу ("нет")).

На картинке множества чёрных и белых шаров разделяются синей и красной линией. При этом красная линия проводит более точную классификацию, потому что **она максимально отстоит от обоих множеств**. Зелёная линия не является линейным классификатором, она не разделяет два множества.



Генеративная и дискриминативная модели

Существует два подхода к определению параметров ω для линейного классификатора - *генеративные* или *дискриминативные* модели.

Генеративная модель использует условное распределение P(x|class) Например:

- **Дискриминантный анализ** (LDA) — предполагает нормальное распределение по Гауссу.
- **Наивный байесовский классификатор** с Бернуллиевской моделью событий.

Дискриминативные модели стремятся улучшить качество выходных данных на наборе образцов для обучения. Например:

- **Логистическая регрессия** — стремление достичь максимального сходства через вектор ω из предположения, что наблюдаемый набор образцов генерировался в виде биномиальной модели от выходных данных.
- **Простой Перцептрон** — алгоритм коррекции всех ошибок на входном наборе образцов.
- **Метод опорных векторов** — алгоритм расширения разделительной зоны в гиперплоскости решений между образцами входных данных.

Дискриминантный анализ

Дискриминативные модели более точны, однако при неполной информации в данных легче использовать условное распределение.

Дискриминантный анализ — раздел вычислительной математики, представляющий набор методов статистического анализа для решения задач распознавания образов, который используется для принятия решения о том, какие переменные разделяют (то есть «дискриминируют») возникающие наборы данных (так называемые «группы»). В отличие от кластерного анализа в дискриминантном анализе группы известны априори.

С вычислительной точки зрения дискриминантный анализ очень похож на дисперсионный анализ (см. раздел Дисперсионный анализ). Рассмотрим следующий простой пример. Предположим, что вы измеряете рост в случайной выборке из 50 мужчин и 50 женщин. Женщины в среднем не так высоки, как мужчины, и эта разница должна найти отражение для каждой группы средних (для переменной Рост). Поэтому переменная Рост позволяет вам провести дискриминацию между мужчинами и женщинами лучше, чем, например, вероятность, выраженная следующими словами: "Если человек большой, то это, скорее всего, мужчина, а если маленький, то это вероятно женщина".

Основная идея дискриминантного анализа заключается в том, чтобы определить, отличаются ли совокупности по среднему какой-либо переменной (или линейной комбинации переменных), и затем использовать эту переменную, чтобы предсказать для новых членов их принадлежность к той или иной группе.

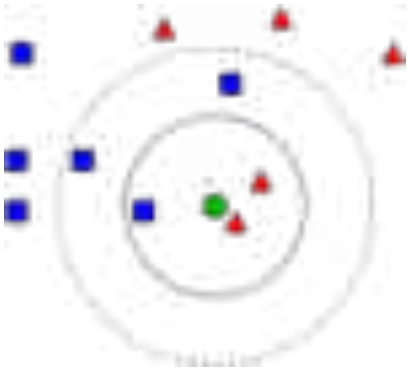
Метод k-ближайших соседей

Метод k-ближайших соседей (*k-nearest neighbors algorithm*, k-NN) — метрический алгоритм для автоматической классификации объектов или регрессии.

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны. В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Алгоритм может быть применим к выборкам с большим количеством атрибутов (многомерным). Для этого перед применением нужно определить функцию расстояния; классический вариант такой функции — евклидова метрика.

Пример классификации k-ближайших соседей. Тестовый образец (зелёный круг) должен быть классифицирован как синий квадрат (класс 1) или как красный треугольник (класс 2). Если k = 3, то он классифицируется как 2-й класс, потому что внутри меньшего круга 2 треугольника и только 1 квадрат. Если k = 5, то он будет классифицирован как 1-й класс (3 квадрата против 2 треугольников внутри большего круга)



Выделение значимых атрибутов

Некоторые значимые атрибуты могут быть важнее остальных, поэтому для каждого атрибута может быть задан в соответствие определённый вес (например, вычисленный с помощью тестовой выборки и оптимизации ошибки отклонения). Таким образом, каждому атрибуту k будет задан в соответствие вес z_k, так что значение атрибута будет попадать в диапазон [0; $z_k$max(k)] (для нормализованных значений по минимакс-методу). Например, если атрибуту присвоен вес 2,7, то его нормализованно-взвешенное значение будет лежать в диапазоне [0;2,7]

Взвешенный способ

При взвешенном способе во внимание принимается не только количество попавших в область определённых классов, но и их удалённость от нового значения.

Для каждого класса j определяется оценка близости:

$$q_j = \sum_{i=1}^{n_j} \frac{1}{d(x,a_i)^2}$$

где $d(x,a_i)$ — расстояние от нового значения x до объекта a_i.

У какого класса выше значение близости, тот класс и присваивается новому объекту.

С помощью метода можно вычислять значение одного из атрибутов классифицируемого объекта на основании дистанций от попавших в область объектов и соответствующих значений этого же атрибута у объектов:

$$x_k = \frac{\sum_{i=1}^{n} k_i d(x,a_i)^2}{\sum_{i=1}^{n} d(x,a_i)^2}$$

где

a_i — i-ый объект, попавший в область,

k_i — значение атрибута k у заданного объекта a_i ,

x — новый объект,

x_k — k-ый атрибут нового объекта.

Наивный байесовский классификатор

Наивный байесовский классификатор — простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше нейронных сетей во многих сложных жизненных ситуациях.

Достоинством наивного байесовского классификатора является малое количество данных, необходимых для обучения, оценки параметров и классификации.

Как работает наивный байесовский алгоритм?

Давайте рассмотрим пример. Ниже представлен обучающий набор данных, содержащий один признак «Погодные условия» (weather) и целевую переменную «Игра» (play), которая обозначает возможность проведения матча. На основе погодных условий мы должны определить, состоится ли матч. Чтобы сделать это, необходимо выполнить следующие шаги.

- Шаг 1. Преобразуем набор данных в частотную таблицу (frequency table).
- Шаг 2. Создадим таблицу правдоподобия (likelihood table), рассчитав соответствующие вероятности. Например, вероятность облачной погоды (overcast) составляет 0,29, а вероятность того, что матч состоится (yes) – 0,64.
- Шаг 3. С помощью теоремы Байеса рассчитаем апостериорную вероятность для каждого класса при данных погодных условиях. Класс с наибольшей апостериорной вероятностью будет результатом прогноза.

Задача. *Состоится ли матч при солнечной погоде (sunny)?*

Sunny – Солнечная погода. *Rainy* – Дождливая погода. *Overcast* – Облачная погода

Мы можем решить эту задачу с помощью описанного выше подхода.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P(Sunny)

Здесь мы имеем следующие значения:

- P(Sunny | Yes) = 3 / 9 = 0,33
- P(Sunny) = 5 / 14 = 0,36
- P(Yes) = 9 / 14 = 0,64

Теперь рассчитаем P(Yes | Sunny):

- P(Yes | Sunny) = 0,33 * 0,64 / 0,36 = 0,60

Значит, при солнечной погоде более вероятно, что матч состоится.

Аналогичным образом с помощью НБА можно прогнозировать несколько различных классов на основе множества признаков. Этот алгоритм в основном используется в области классификации текстов и при решении задач многоклассовой классификации.

Распознавания образов

Распознавание образов — это отнесение исходных данных к определённому классу с помощью выделения существенных признаков, характеризующих эти данные, из общей массы данных.

При постановке задач распознавания стараются пользоваться математическим языком, стремясь — в отличие от теории искусственных нейронных сетей, где основой является получение результата путём эксперимента, — заменить эксперимент логическими рассуждениями и математическими доказательствами.

Классическая постановка задачи распознавания образов:

- Дано множество объектов. Относительно них необходимо провести классификацию. Множество представлено подмножествами, которые называются классами.
- Заданы: информация о классах, описание всего множества и описание информации об объекте, принадлежность которого к определённому классу неизвестна.
- Требуется по имеющейся информации о классах и описании объекта установить — к какому классу относится этот объект.

Наиболее часто в задачах распознавания образов рассматриваются монохромные изображения, что дает возможность рассматривать изображение как функцию на плоскости. Если рассмотреть точечное множество на плоскости T, где функция f(x,y) выражает в каждой точке изображения его характеристику — яркость, прозрачность, оптическую плотность, то такая функция есть формальная запись изображения.

Множество же всех возможных функций f(x,y) на плоскости T есть модель множества всех изображений X. Вводя понятие сходства между образами можно поставить задачу распознавания. Конкретный вид такой постановки сильно зависит от последующих этапов при распознавании в соответствии с тем или иным подходом.

Методы распознавания графических образов

- Для оптического распознавания образов можно применить метод перебора вида объекта под различными углами, масштабами, смещениями и т. д. Для букв нужно перебирать шрифт, свойства шрифта и т. д.
- Второй подход — найти контур объекта и исследовать его свойства (связность, наличие углов и т. д.)
- Ещё один подход — использовать искусственные нейронные сети. Этот метод требует либо большого количества примеров задачи распознавания (с правильными ответами), либо специальной структуры нейронной сети, учитывающей специфику данной задачи.

Направления в распознавании образов

- Изучение способностей к распознаванию, которыми обладают живые существа, объяснение и моделирование их;
- Развитие теории и методов построения устройств, предназначенных для решения отдельных задач в прикладных целях.

Примеры задач распознавания образов

- Оптическое распознавание символов
- Распознавание штрих-кодов
- Распознавание автомобильных номеров
- Распознавание лиц
- Распознавание речи
- Распознавание изображений
- Распознавание локальных участков земной коры, в которых находятся месторождения полезных ископаемых
- Классификация документов

Деревья решений

Один из самых популярных методов классификации и регрессии –деревья решений. Деревья решений используются в повседневной жизни в самых разных областях человеческой деятельности, порой и очень далеких от машинного обучения. Деревом решений можно назвать наглядную инструкцию, что делать в какой ситуации. Приведем пример из области консультирования научных сотрудников института. Высшая Школа Экономики выпускает инфо-схемы, облегчающие жизнь своим сотрудникам. Вот фрагмент инструкции по публикации научной статьи на портале института.

В терминах машинного обучения можно сказать, что это элементарный классификатор, который определяет форму публикации на портале (книга, статья, глава книги, препринт, публикация в "НИУ ВШЭ и СМИ") по нескольким признакам: типу публикации (монография, брошюра, статья и т.д.), типу издания, где опубликована статья (научный журнал, сборник трудов и т.д.) и остальным.

Зачастую дерево решений служит обобщением опыта экспертов, средством передачи знаний будущим сотрудникам или моделью бизнес-процесса компании.

Например, до внедрения масштабируемых алгоритмов машинного обучения в банковской сфере задача кредитного скоринга решалась экспертами. Решение о выдаче кредита заемщику принималось на основе некоторых интуитивно (или по опыту) выведенных правил, которые можно представить в виде дерева решений.

В этом случае можно сказать, что решается задача бинарной классификации (целевой класс имеет два значения: "Выдать кредит" и "Отказать") по признакам "Возраст", "Наличие дома", "Доход" и "Образование".

Дерево решений как алгоритм машинного обучения – по сути то же самое: объединение логических правил вида "***Значение признака a меньше x И Значение признака b меньше y… => Класс 1***" в структуру данных "Дерево". Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Например, по схеме на рисунке выше можно объяснить заемщику, почему ему было отказано в кредите. Скажем, потому что у него нет дома и доход меньше 5000. Как мы увидим дальше, многие другие, хоть и более точные, модели не обладают этим свойством и могут рассматриваться скорее как "черный ящик", в который загрузили данные и получили ответ.

В связи с этой "понятностью" деревьев решений и их сходством с моделью принятия решений человеком (можно легко объяснять боссу свою модель), деревья решений получили огромную популярность, а один из представителей этой группы методов классификации, C4.5, рассматривается первым в списке 10 лучших алгоритмов интеллектуального анализа данных.

Деревья принятия решений являются удобным инструментом в тех случаях, когда требуется не просто классифицировать данные, но ещё и объяснить почему тот или иной объект отнесён к какому-либо классу.

Лекция 7. Кластеризация

Древняя китайская классификация животных

Животные подразделяются на: а)принадлежащих императору; б) набаль-замированных; в) дрессированных; г) молочных поросят; д) сирен; е) сказочных; ж) бродячих собак; з) включенных в данную классификацию; и) дрожащих, как сума-сшедшие к) неисчислимых; л) нарисованных самой лучшей китайской кисточкой;

м) других; н) тех, которые только что разбили цветочную вазу и о) тех, которые издалека напоминают мух

(Хорхе Луис Борхес, Другие исследования: 1937-1952).

Задача кластеризации сходна с задачей классификации, является ее логическим продолжением, но ее отличие в том, что классы изучаемого набора данных заранее не предопределены.

Синонимами термина "кластеризация" являются "автоматическая классификация", "обучение без учителя" и "таксономия".

Кластеризация предназначена для разбиения совокупности объектов на однородные группы (кластеры или классы). Если данные выборки представить как точки в признаковом пространстве, то задача кластеризации сводится к определению "сгущений точек".

Кластерный анализ объединяет различные процедуры, используемые для проведения кластеризации. В результате применения данных процедур исходная совокупность объектов разделяется на кластеры или группы (классы) сходных между собой объектов.

Список прикладных областей, где она применяется, широк: сегментация изображений, маркетинг, борьба с мошенничеством, прогнозирование, анализ текстов и многие другие. На современном этапе кластеризация часто выступает первым шагом при анализе данных. После выделения схожих групп применяются другие методы, для каждой группы строится отдельная модель. Аналитику часто легче выделить группы схожих объектов, изучить их особенности и построить для каждой группы отдельную модель, чем создавать одну общую модель на всех данных. Таким приемом постоянно пользуются в маркетинге, выделяя группы клиентов, покупателей, товаров и разрабатывая для каждой их них отдельную стратегию.

Следует отметить, что в результате применения различных методов кластерного анализа могут быть получены кластеры различной формы. Например, возможны кластеры "цепочного" типа, когда кластеры представлены длинными "цепочками", кластеры удлиненной формы и т.д., а некоторые методы могут создавать кластеры произвольной формы.

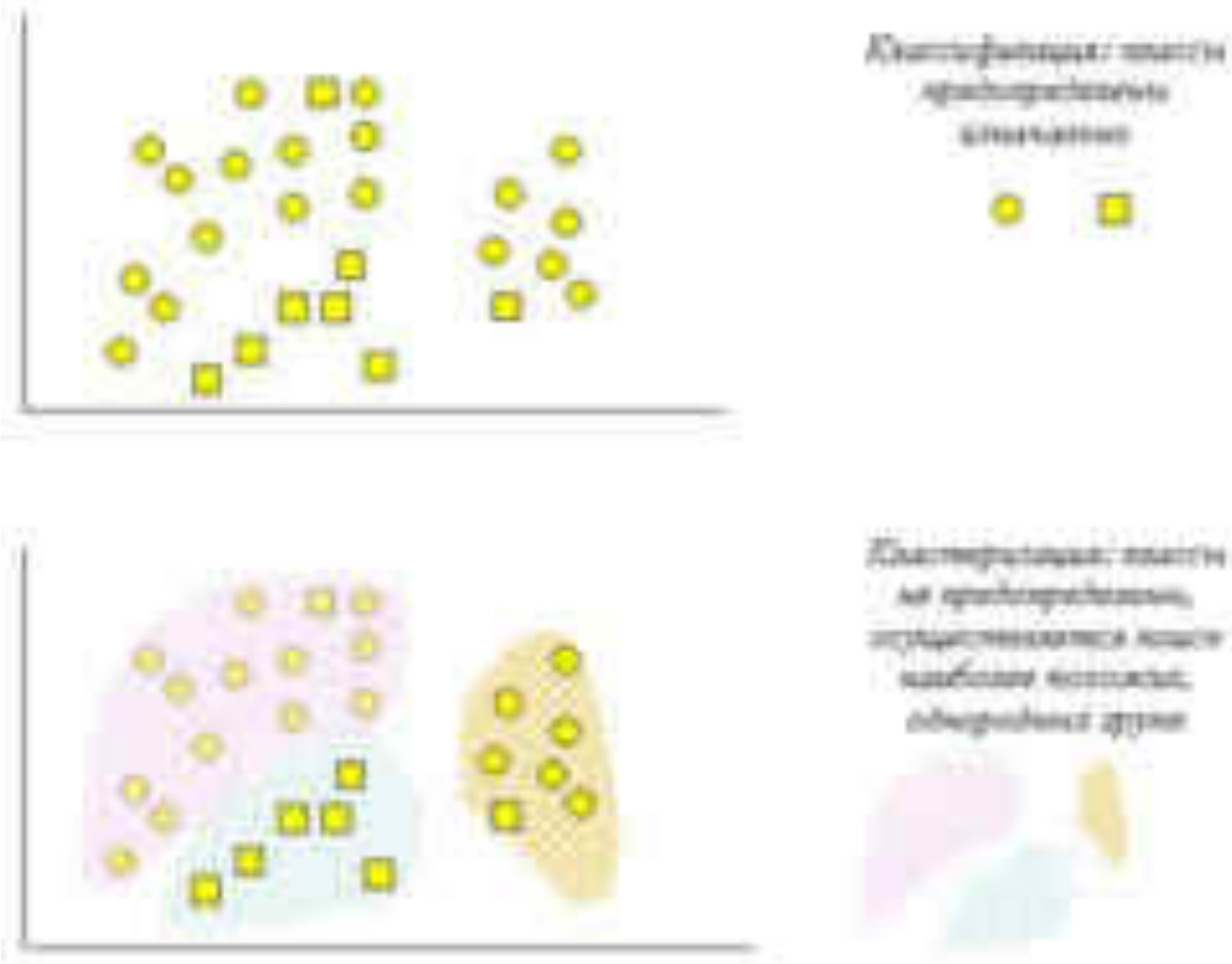


Рис. 7.1 Сравнение задач классификации и кластеризации

Кластеры могут быть непересекающимися, или эксклюзивными (non-overlapping, exclusive), и пересекающимися (overlapping). Схематическое изображение непересекающихся и пересекающихся кластеров дано на рис. 7.2.

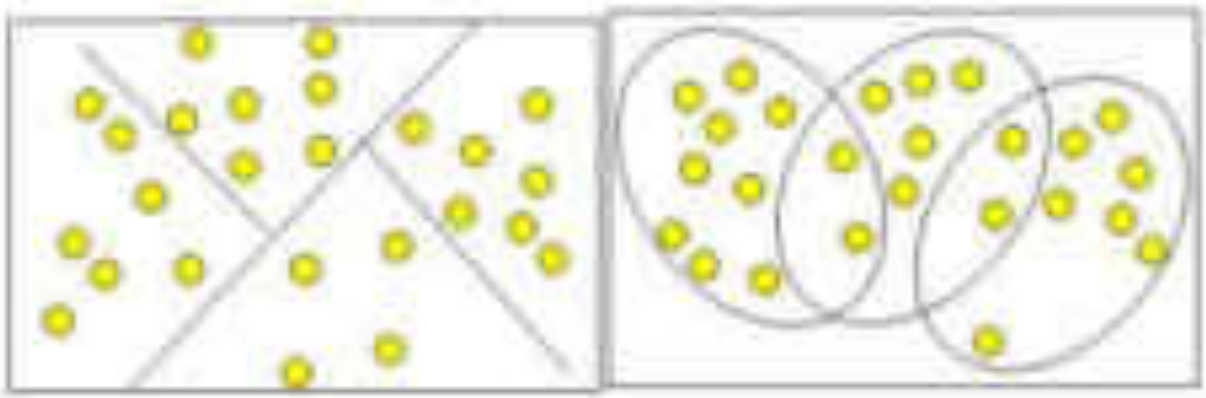


Рис. 7.2 Непересекающиеся и пересекающиеся кластеры

Говоря о кластерной совокупности объектов подразумеваем, что каждый из них задан соответствующей строкой матрицы X (TЭD), либо задана матрица попарных расстояний объектов. Тогда проблема кластеризации заключается в том, чтобы всю анализируемую совокупность объектов $O=\{O_1,O_2,\ldots,O_n\}$, статистически представленную в виде матриц **X** или **P**, разбить на сравнительно небольшое число (заранее известное или нет) однородных в определенном смысле, групп или классов.

Полученные в результате разбиения классы часто называют кластерами (таксонами, образами), а методы их нахождения – соответственно кластер-анализом, численной таксономией, распознаванием образов с самообучением.

Перед началом кластеризации каждый исследователь должен решить для себя, какую из двух задач он решает:

1. Обычная задача разбиения на гиперобласти группирования.
2. Определение естественного расслоения исходных наблюдений на четко выраженные кластеры. Каждый кластер не разбивается на столь же удаленные части.

Если первая задача всегда имеет решение, то во втором случае результат может быть и отрицательным. Например, множество исходных наблюдений образует один общий кластер.

Можно выделить следующие этапы кластерного анализа:

1. Получение выборки для кластеризации.
2. Определение множества признаков, по которым будут определяться объекты.
3. Вычисление значений меры сходства между объектами.
4. Применение методов кластерного анализа для создания групп.
5. Проверка достоверности результатов.

Меры сходства

То, что некоторые вещи обнаруживают между собой сходство или различие, является весьма важным моментом для кластеризации.

Выделяют четыре группы коэффициентов сходства:

- Коэффициенты корреляции
- Меры расстояния
- Коэффициенты ассоциативности
- Вероятные коэффициенты сходства.

Наиболее широкое распространение получили меры расстояния, которые мы и рассмотрим.

На практике меры расстояния точнее было бы назвать мерами несходства: если для большинства используемых коэффициентов большие значения соответствуют большему сходству, то для мер расстояния все наоборот.

Количественное оценивание сходства отталкивается от понятия метрики. При таком подходе объекты представляются точками пространства, расстояние между которыми и определяет их сходство или различие.

Выделим такие понятия как «расстояние между отдельными объектами» (мера близости объектов) и «расстояние между классами» (мера близости классов) и рассмотрим их отдельно.

Расстояние между отдельными объектами.

Выбор метрики (или меры близости) является узловым моментом исследования, от которого во многом зависит окончательный вариант разбиения объектов на классы.

В каждой конкретной задачи этот выбор должен производиться по-своему и опираться на физическую, статистическую природу вектора наблюдений X.

Рассмотрим некоторые примеры расстояний и мер близости.

- Расстояние Махаланобиса.

В общем случае зависимых компонент $x^{(1)} \ldots x^{(p)}$ вектора наблюдений X и их различной значимости в решении вопроса об отнесении объекта к тому или иному классу обычно пользуются взвешенным расстоянием Махалапобиса.

$$d_0(x_i, x_j) = \sqrt{(x_i - x_j)' \wedge \sum^{-1} \wedge (x_i - x_j)}$$

Здесь
$\sum$ - ковариационная матрица генеральной совокупности, из которой извлекается матрица наблюдений X;
Σ^{-1} – матрица, обратная ковариационной.
$\wedge$ - симметричная матрица весовых коэффициентов X_{mq} набора, чаще всего выбирается диагональной.

Когда корреляция между переменными равна нулю, расстояние Махаланобиса эквивалентно евклидову расстоянию.

Такое расстояние между объектами удобно тем, что оно инвариантно по отношению к линейным преобразованиям признакового пространства, т.е. любое линейное преобразование (растяжение, сдвиг, поворот и т. п.) оставляет расстояние Махаланобиса между объектами неизменным. Это расстояние применяют в том случае, если признаки x(1), x(2), ... , x(p) значительно коррелированны между собой, а также, если признаки имеют различную значимость для классификации.

- Обычное евклидово расстояние

$$d_E(x_i, x_j) = \sqrt{\sum_{k=1}^{p} \left(x_i^{(k)} - x_j^{(k)} \right)^2}$$

- Взвешенное евклидово расстояние. Формула для вычисления евклидова расстояния между двумя объектами в пространстве ***p*** переменных с учетом коэффициентов важности этих переменных преобразуется к виду:

$$d_{bE}(x_i, x_j) = \sqrt{\sum_{k=1}^{p} w_k \left(x_i^{(k)} - x_j^{(k)} \right)^2}$$

Обычно применяется в ситуациях, в которых так или иначе удается приписать каждой из компонент $X^{(k)}$ некоторый неотрицательный «вес» w_k, пропорциональный степени его важности.

a) Хеммингово расстояние

$$d_H(x_i, x_j) = \sum_{k=1}^{p} \left| x_i^{(k)} - x_j^{(k)} \right|$$

Это расстояние используется в основном как мера различия объектов, признаки которых измерены в шкалах наименований и порядка. В случае дихотомических признаков расстояние Хэмминга показывает число несовпадающих у объектов признаков.

Первоначально данная мера была введена для бинарных кодов, но она вполне применима и для сравнений любых упорядоченных наборов, которые состоят из элементов принимающих дискретные значения.

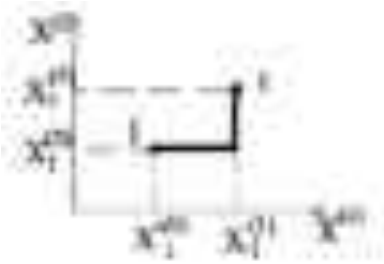
b) Расстояние Минковского

$$d_m(x_i, x_j) = \left(\sum_{k=1}^{p} \left| x_i^{(k)} - x_j^{(k)} \right| r \right) 1/r$$

при r=1 имеем манхэттенское расстояние, при r=2 – евклидово.

c) Манхэттенское расстояние или расстояние городских кварталов

$$d(x_i x_j) = \sum_{k=1}^{p} \left| x_i^{(k)} - x_j^{(k)} \right|$$



d) Расстояние Чебышева:

$$\text{distance}(\ x,\ c\ ) = \text{Maximum}\ |x_i - c_i|$$

Расстояние между классами

Для проведения кластеризации необходимо также ввести понятие меры близости двух групп объектов.

Пусть:

S_i – i-ая группа (класс, классификатор);

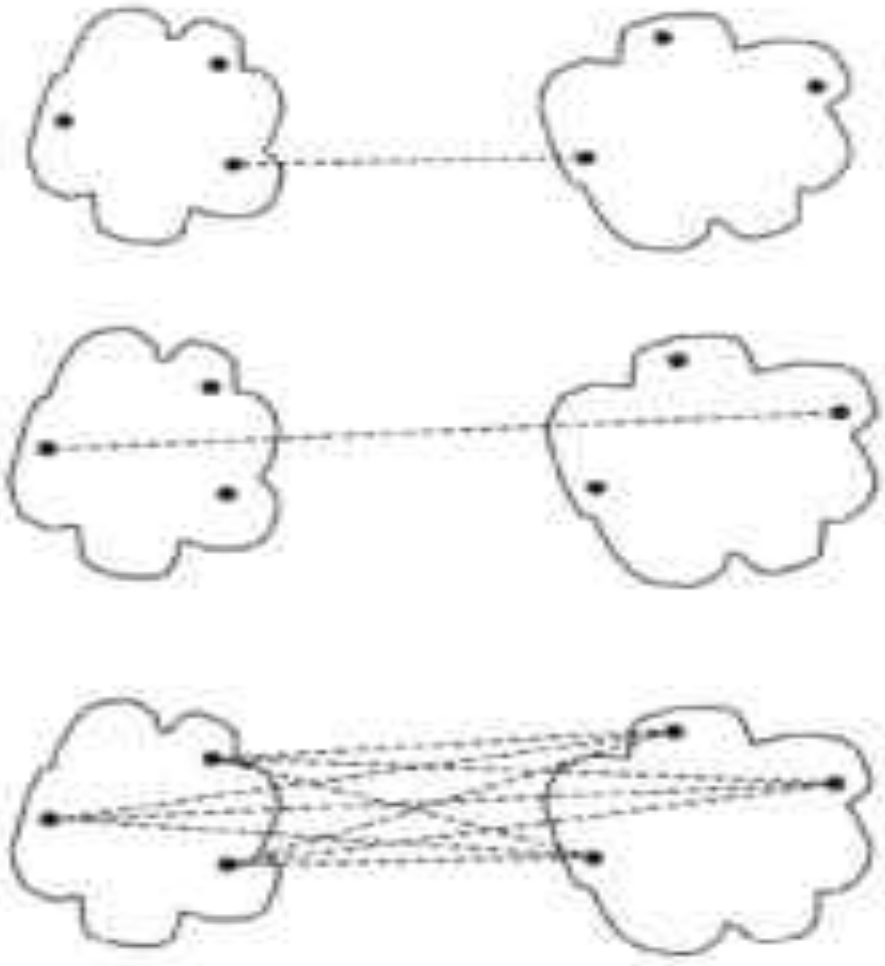
n_j – число объектов, образующих группу S_i;

$\overline{X}(i)$ – среднее арифметическое вектора наблюдений, т.е. «центр тяжести» i-й группы;

$\rho(S_l, S_m)$ – расстояние между группами.

а) Расстояние, измеряемое по принципу «ближнего соседа».

$$\rho_{\min}(S_l, S_m) = \min_{x_i \in S_l, x_j \in S_m} d\left(x_i, x_j\right)$$

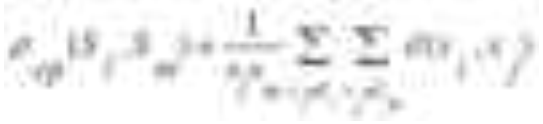


Расстояния между классами a),b) и c)

b) Расстояние, измеряемое по принципу «дальнего соседа».

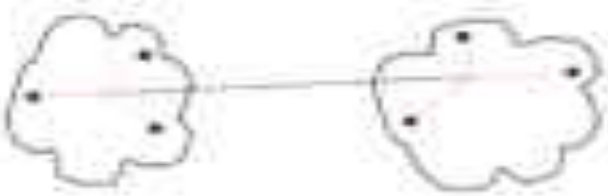
$$\rho_{\max}(S_l, S_m) = \max_{x_i \in S_l, x_j \in S_m} d\left(x_i, x_j\right)$$

c) Расстояние, измеряемое по принципу «средней связи».



d) Расстояние, измеряемое по центрам тяжести групп.

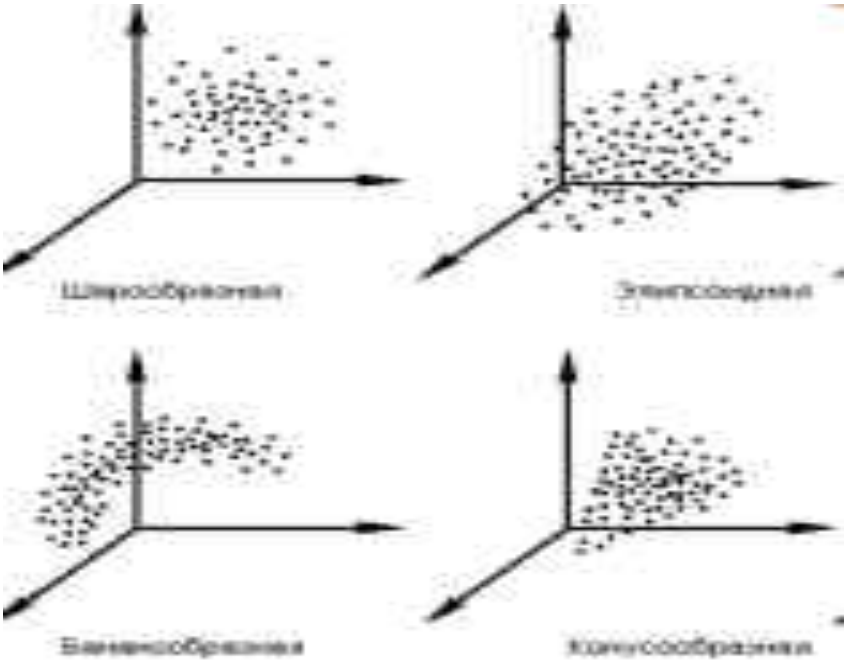
$$\rho(S_l, S_m) = d\left(\overline{X}(l), \overline{X}(m)\right)$$

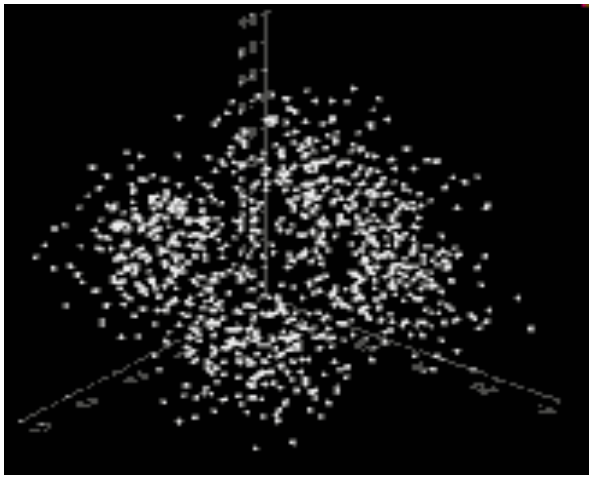


Определяется как арифметическое среднее всевозможных попарных расстояний между представителями рассматриваемых групп.

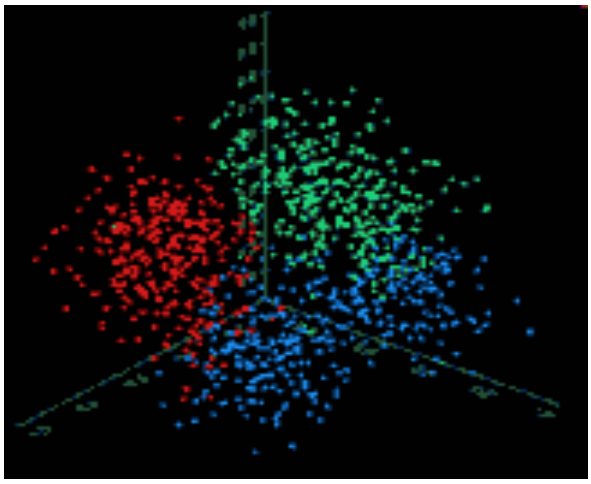
Кластер и кластеризация

Кластер — это множество объектов, близких между собой по некоторой мере сходства. В пространстве переменных кластеры представляют собой скопления точек (объектов) различной формы.





Так выглядят объекты до кластеризации



Объекты после кластеризации

Типы кластерных структур

Внутрикластерные расстояния, как правило, меньше межкластерных

ленточные кластеры

кластеры с центром

Обзор методов кластерного анализа

Следует отметить, что в результате применения различных методов кластерного анализа могут быть получены кластеры различной формы. Например, возможны кластеры "цепочного" типа, когда кластеры представлены длинными "цепочками", кластеры удлиненной формы и т.д., а некоторые методы могут создавать кластеры произвольной формы.

Различные методы могут стремиться создавать кластеры определенных размеров (например, малых или крупных) либо предполагать в наборе данных наличие кластеров различного размера.

Некоторые методы кластерного анализа особенно чувствительны к шумам или выбросам, другие - менее.

В результате применения различных методов кластеризации могут быть получены неодинаковые результаты, это нормально и является особенностью работы того или иного алгоритма.

Данные особенности следует учитывать при выборе метода кластеризации. На сегодняшний день разработано более сотни различных алгоритмов кластеризации.

Кластера – это непрерывные области некоторого пространства, с относительно высокой плотностью точек, отделенных от других таких же областей областями с относительно низкой плотностью точек.

Наиболее важными свойствами кластеров являются плотность, дисперсия, размеры, форма и отделимость. Отделимость характеризует степень перекрытия кластеров и насколько далеко они расположены в пространстве.

Разработанные кластерные методы образуют 6 основных семейств:

1. Иерархические

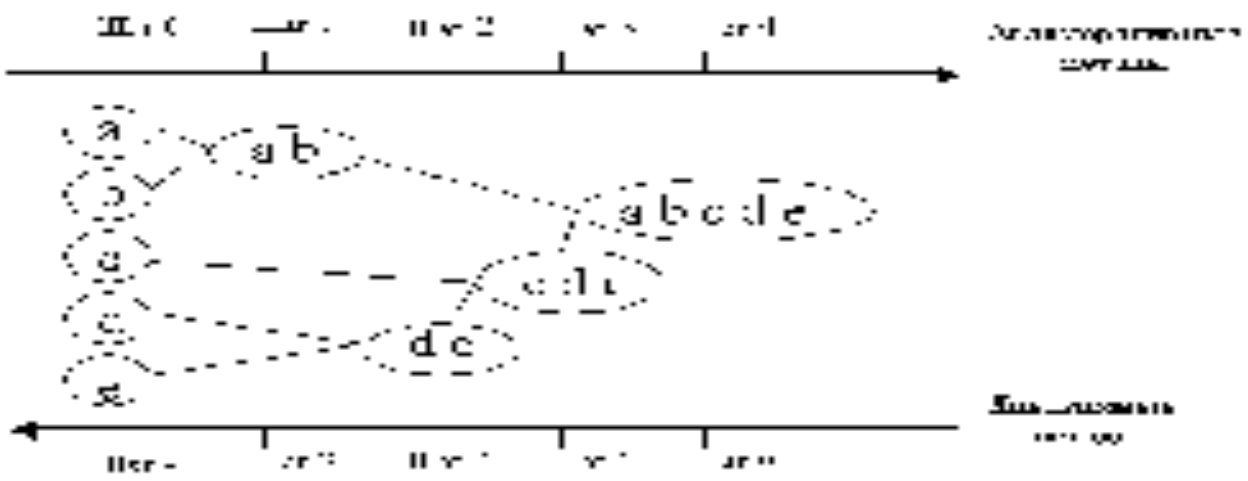
a. агломеративные;
b. дивизимные;

2. Итеративные методы группировки;
3. Методы поиска модальных значений плотности;
4. Факторные методы;
5. Методы сгущений;
6. Методы, использующие теорию графов.

К наиболее распространенным методам решения задачи кластеризации относят:

- метод k-средних (работает только с числовыми атрибутами),
- иерархический кластерный анализ (работает также с символьными атрибутами),
- метод SOM.

Сложностью кластеризации является необходимость ее оценки.

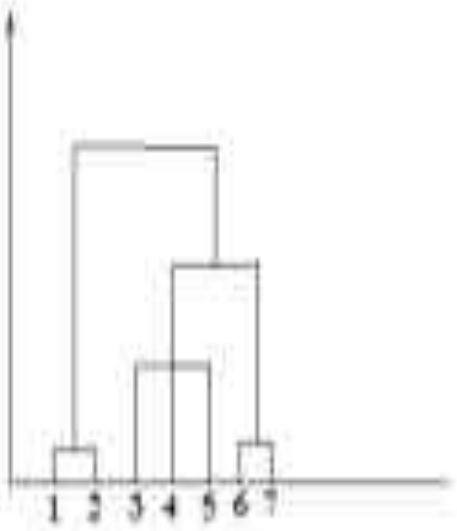


Дендрограмма агломеративных и дивизимных методов

Агломеративная иерархическая кластеризация

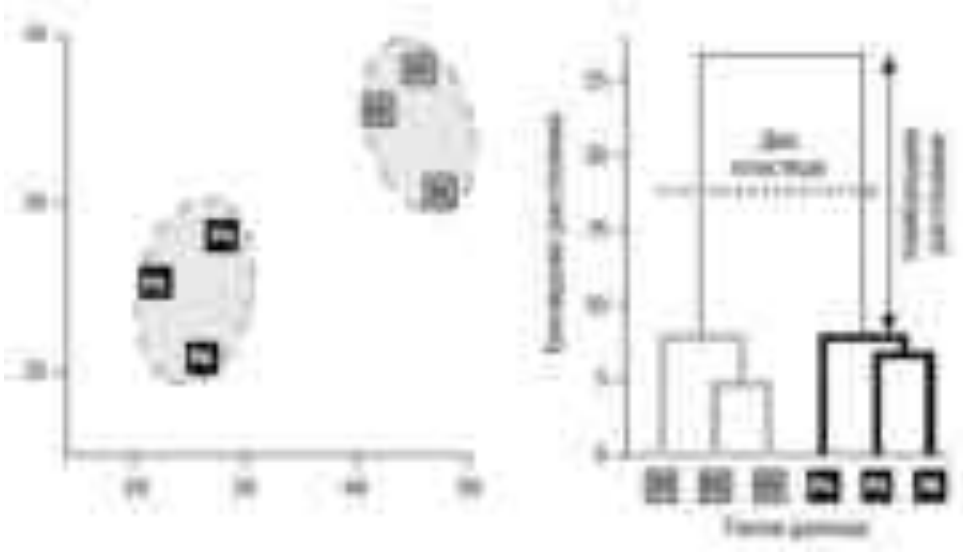
Рассмотрим более подробно метод иерархической классификации.

1. Иерархические алгомеративные методы просматривают матрицу сходства размерностью N×N (где N – число объектов) и последующую, объединяют схожие объекты. Именно поэтому они называются агломеративными или объединяющими.
2. Последовательность объединений кластеров можно представить в виде древовидной диаграммы, называемой дендрограммой.



Для полной кластеризации этим методом на основе матрицы сходства размерность N×N требуется ровно N-1 шагов. На 1-ом шаге объекты рассматриваются как самостоятельные кластеры. На последнем шаге все события (объекты) объединяются в одну большую группу самостоятельные кластеры. На последнем шаге все события (объекты) объединяются в одну большую группу.

3. Для понимания иерархических агломеративных методов не нужны обширные знания.



Несмотря на простоту методов, они обладают некоторыми недостатками:

- Вычисление и хранение большой матрицы сходств (например, для 500 объектов потребуется матрица из 125000 элементов).

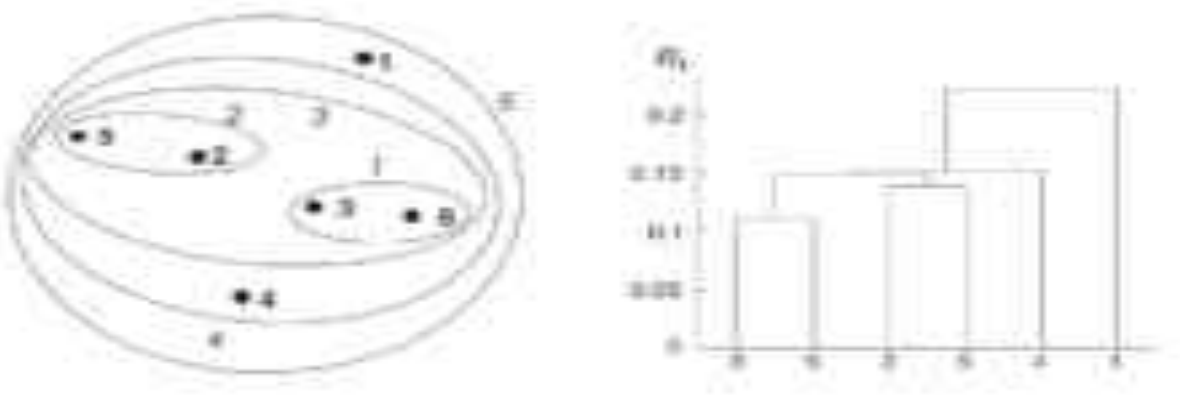
- Объекты распределяются по кластерам за один проход, поэтому плохое начальное разбиение множества данных не может быть изменено на последующих шагах процесса кластеризации.
- Все методы (за исключением метода одиночной связи) могут порождать разные решения в результате простого переупорядочивания объектов в матрице сходства и, кроме того, их результаты изменяются, если некоторые объекты исключить из рассмотрения. Устойчивость – важное свойство «естественной» группировки по сравнению с теми группами, которые исчезают, если некоторые объекты переупорядочены или исключены из анализа.

Иерархические алгомеративные методы различаются главным образом по правилам построения кластеров. Существует много различных правил группировки, каждое из которых порождает специфический иерархический метод.

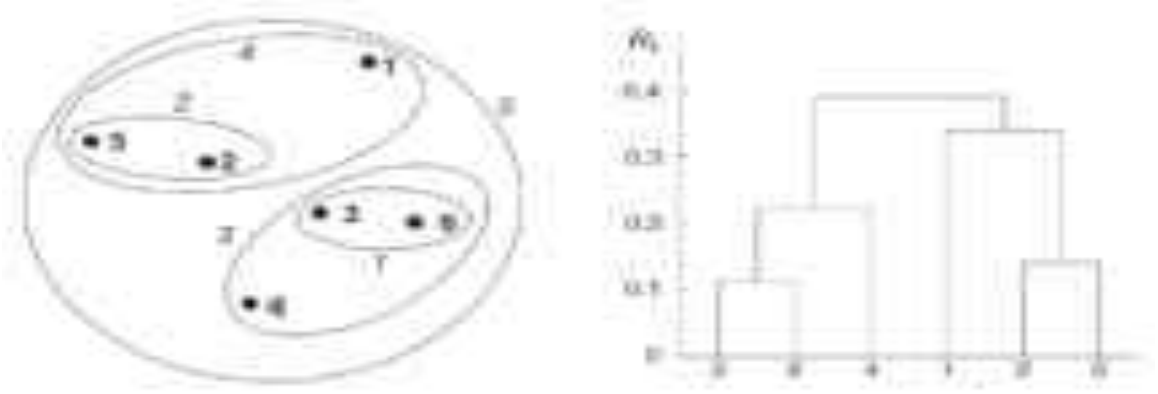
Рассмотрим наиболее распространенные:

1. **метод одиночной связи;**
2. **метод полной связи;**
3. **метод средней связи;**
4. **метод Уорда (Ward).**

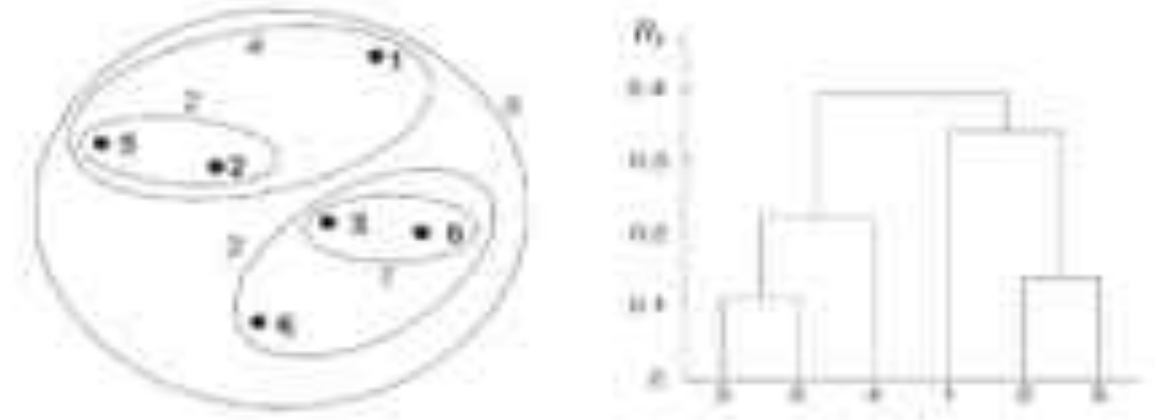
1. В ***методе одиночной связи*** на первом шаге объединяются два объекта, имеющие между собой максимальную меру сходства. На следующем шаге к ним присоединяется объект с максимальной мерой сходства с одним из объектов кластера, т.е. используется принцип «ближнего соседа». Таким образом процесс продолжается дальше. Итак, для включения объекта в кластеризацию требуется максимальное сходство с одним членом кластера. Отсюда и название метода одиночной связи. Нужна только одна связь, чтобы присоединить объект к кластеризации. Недостатки этого метода это образование слишком больших продолговатых кластеров.



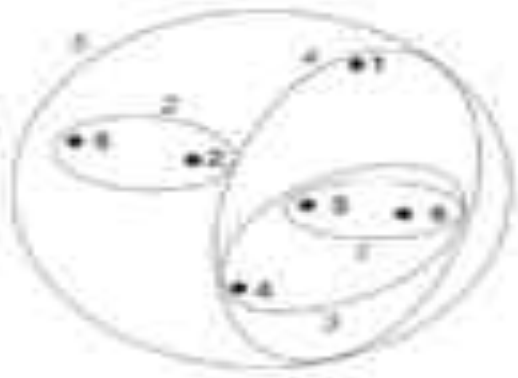
2. ***Метод наиболее удаленных соседей или полной связи*** позволяет устранить этот недостаток. В качестве правила построения кластеров используется принцип «дальней связи»: расстояние между объектом-кандидатом и самым дальним объектом кластера не должно превышать некоторого уровня.

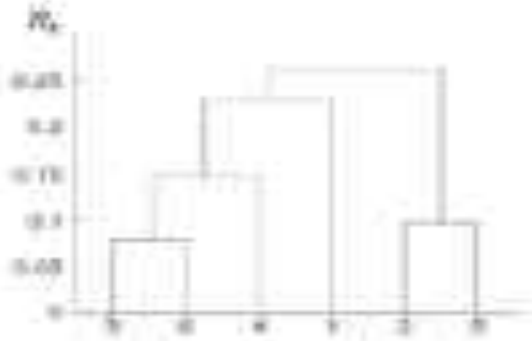


3. В ***методе средней связи*** вычисляется среднее сходство рассматриваемого объекта со всеми объектами в одном существующем кластере, а затем, если найденное среднее значение сходства достигает или превосходит некоторый заданный пороговый уровень сходства, объект присоединяется к этому кластеру. В других вариантах метода средней связи вычисляется сходство между центрами тяжести двух кластеров, подлежащих объединению.



4. ***Метод Уорда*** (Ward's method). В качестве расстояния между кластерами берется прирост суммы квадратов расстояний объектов до центров кластеров, получаемый в результате их объединения (Ward, 1963). В отличие от других методов кластерного анализа для оценки расстояний между кластерами, здесь используются методы дисперсионного анализа. На каждом шаге алгоритма объединяются такие два кластера, которые приводят к минимальному увеличению целевой функции, т.е. внутригрупповой суммы квадратов. Этот метод направлен на объединение близко расположенных кластеров и "стремится" создавать кластеры малого размера.





Математические характеристики кластера

Кластер имеет следующие математические характеристики:

1. Центр кластера — это среднее геометрическое место точек в пространстве переменных.

$$\bar{x}_{kj} = \frac{\sum_{i=1}^{n_k} w_i x_{ij}}{l_k}$$

2. Дисперсия кластера - это мера рассеяния точек в пространстве относительно центра кластера:

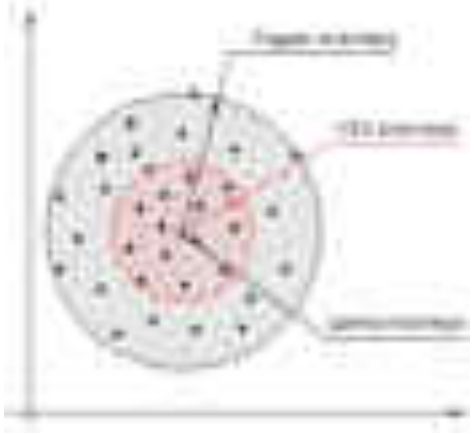
$$D_k = \frac{\sum_{i=1}^{n_k} w_i (x_{ij} - \bar{x}_{kj})^2}{l_k - 1}$$

3. Среднеквадратичное отклонение (СКО) объектов относительно центра кластера:

$$\sigma_k = \sqrt{D_k}$$

4. Радиус кластера - максимальное расстояние точек от центра кластера:

$$R_k = \max \sqrt{\sum_{i=1}^{n_k} w_i (x_{ij} - \bar{x}_{kj})^2}$$



Спорные объекты и перекрывающиеся кластеры

Спорный объект - это объект, который по мере сходства может быть отнесен к нескольким кластерам.

Размер кластера может быть определен либо по радиусу кластера, либо по среднеквадратичному отклонению объектов для этого кластера. Объект относится к кластеру, если расстояние от объекта до центра кластера меньше радиуса кластера. Если это условие выполняется для двух и более кластеров, объект является спорным. Неоднозначность данной задачи может быть устранена экспертом или аналитиком.

Методы кластерного анализа. Итеративные методы.

При большом количестве наблюдений иерархические методы кластерного анализа не пригодны. В таких случаях используют неиерархические методы, основанные на разделении, которые представляют собой итеративные методы дробления исходной совокупности. В процессе деления новые кластеры формируются до тех пор, пока не будет выполнено правило остановки.

Такая неиерархическая кластеризация состоит в разделении набора данных на определенное количество отдельных кластеров. Существует два подхода. Первый заключается в определении границ кластеров как наиболее плотных участков в многомерном пространстве исходных данных, т.е. определение кластера там, где имеется большое "сгущение точек". Второй подход заключается в минимизации меры различия объектов.

Неиерархические алгоритмы основаны на оптимизации некоторой целевой функции, определяющей оптимальное в определенном смысле разбиение множества объектов на кластеры. В этой группе популярны алгоритмы семейства k-средних (k-means, fuzzy c-means, Густафсон-Кесселя), которые в качестве целевой функции используют сумму квадратов взвешенных отклонений координат объектов от центров искомых кластеров. Кластеры ищутся сферической либо эллипсоидной формы.

Алгоритм k-средних (k-means)

Наиболее распространен среди неиерархических методов алгоритм k-средних, также называемый **быстрым кластерным анализом**. В отличие от иерархических методов, которые не требуют предварительных предположений относительно числа кластеров, для возможности использования этого метода необходимо иметь гипотезу о наиболее вероятном количестве кластеров.

Алгоритм k-средних строит k кластеров, расположенных на возможно больших расстояниях друг от друга. Основной тип задач, которые решает алгоритм k-средних, - наличие предположений (гипотез) относительно числа кластеров, при этом они должны быть различны настолько, насколько это возможно. Выбор числа k может базироваться на результатах предшествующих исследований, теоретических соображениях или интуиции.

Общая идея алгоритма: заданное фиксированное число k кластеров наблюдения сопоставляются кластерам так, что средние в кластере (для всех переменных) максимально возможно отличаются друг от друга.

Описание алгоритма

1. ***Первоначальное распределение объектов по кластерам.***

- Произвольно выбирается k объектов, как исходных центров кластеров. На первом шаге эти точки считаются "центрами" кластеров. Каждому кластеру соответствует один центр.
- Выбор начальных центроидов может осуществляться следующим образом:
    - выбор k-наблюдений для максимизации начального расстояния;
    - случайный выбор k-наблюдений;
    - выбор первых k-наблюдений.
- Каждый объект назначается ближайшему центроиду.

2. ***Итеративный процесс.***

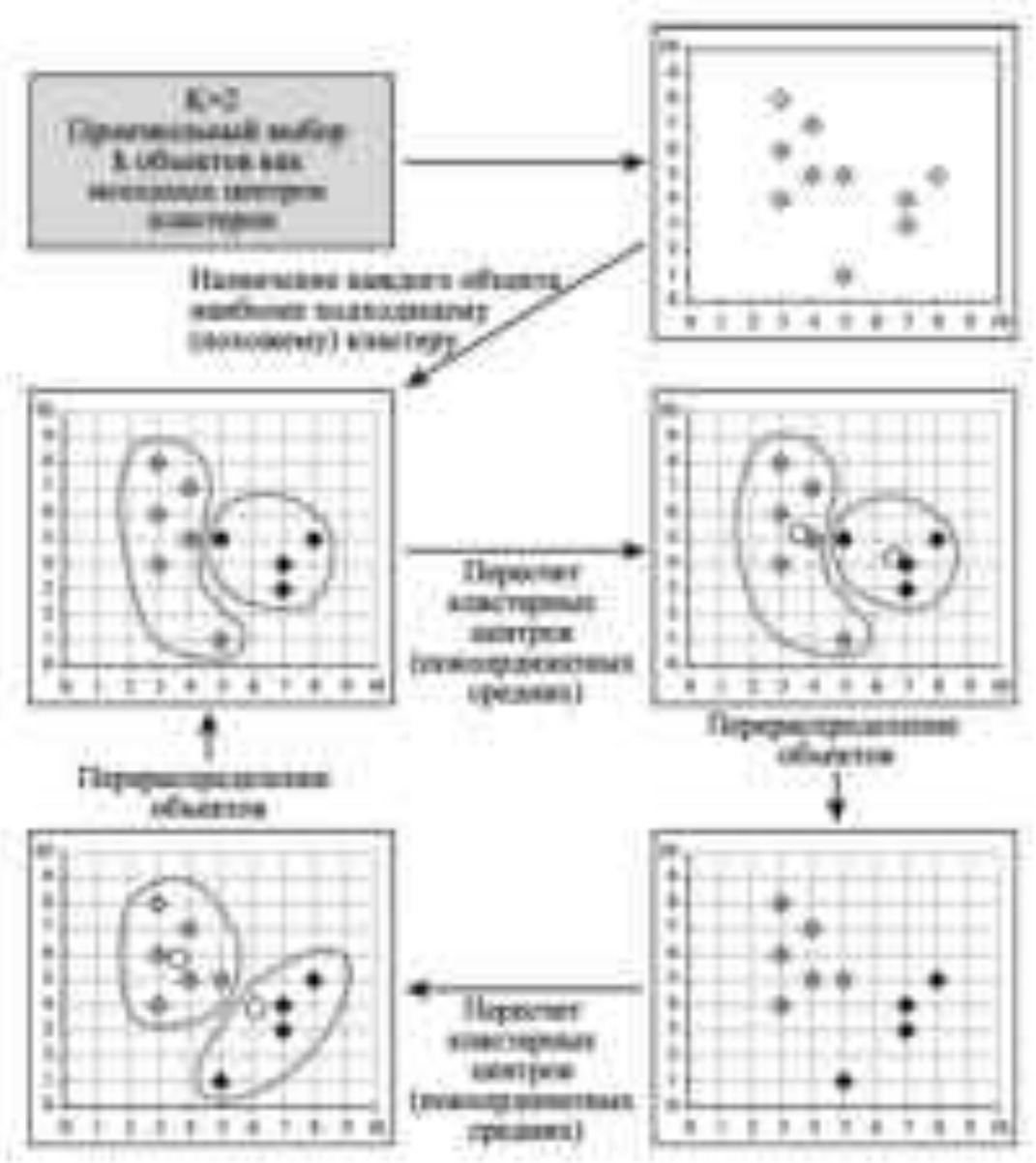
Вычисляются центры кластеров, которыми затем и далее считаются покоординатные средние кластеров. Объекты опять перераспределяются.

Процесс вычисления центров и перераспределения объектов продолжается до тех пор, пока не выполнено одно из условий:

- кластерные центры стабилизировались, т.е. все наблюдения принадлежат кластеру, которому принадлежали до текущей итерации;
- число итераций равно максимальному числу итераций.

На рисунке приведен пример работы алгоритма k-средних для k, равного двум.

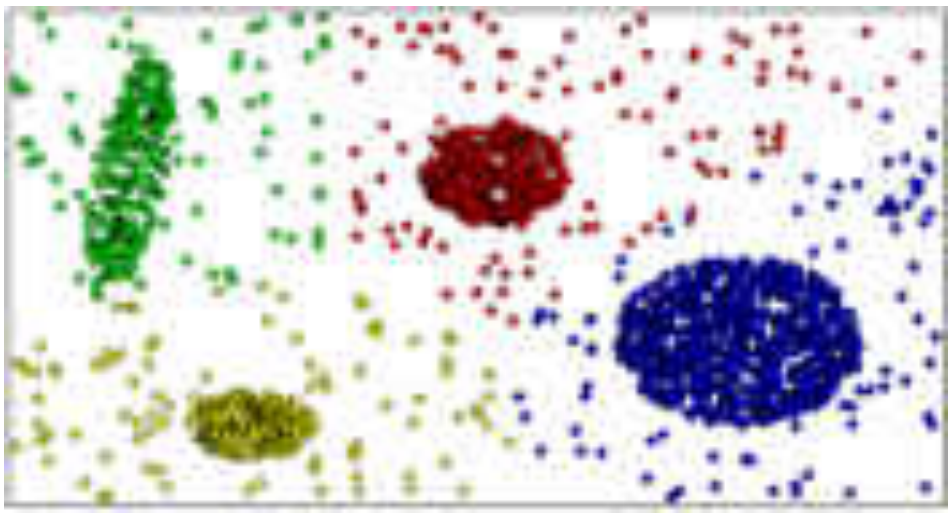
Выбор числа кластеров является сложным вопросом. Если нет предположений относительно этого числа, рекомендуют создать 2 кластера, затем 3, 4, 5 и т. д., сравнивая полученные результаты.



Проверка качества кластеризации методом k-средних

После получений результатов кластерного анализа методом k-средних следует проверить правильность кластеризации (т. е. оценить, насколько кластеры отличаются друг от друга).

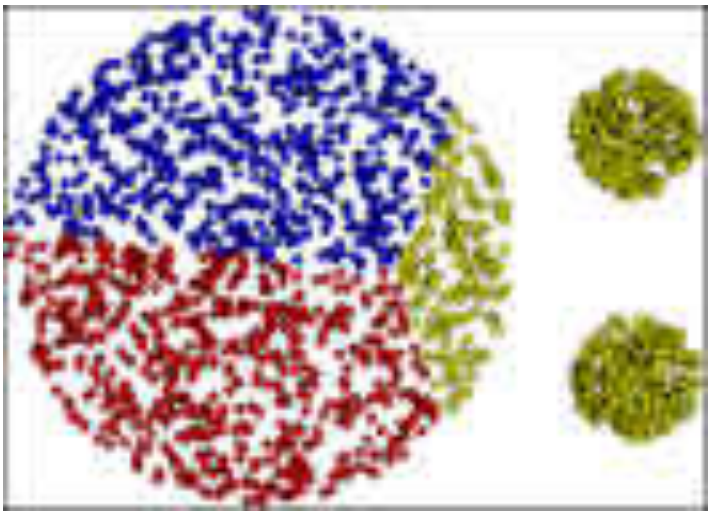
Для этого рассчитываются средние значения для каждого кластера. При хорошей кластеризации должны быть получены сильно отличающиеся средние для всех измерений или хотя бы большей их части.



Результат кластеризации алгоритмом k-means
(http://www.basegroup.ru/library/analysis/clusterization/datamining/)

Алгоритм оптимизации целевой функции в неиерархических алгоритмах, основанных на расстояниях, носит итеративный характер, и на каждой итерации требуется рассчитывать матрицу расстояний между объектами. При большом числе объектов это неэффективно и требует серьезных вычислительных ресурсов. Вычислительная сложность 1й итерации алгоритма k-means оценивается как O(kmn), где k,m,n – количество кластеров, атрибутов и объектов соответственно. Но итераций может быть очень много! Придется делать много проходов по набору данных.

Имеет массу недостатков в k-means сам подход с идеей поиска кластеров сферической или эллипсоидной формы. Подход хорошо работает, когда данные в пространстве образуют компактные сгустки, хорошо отличимые друг от друга. А если данные имеют вложенную форму, то ни один из алгоритмов семейства k-means никогда не справится с такой задачей. Также алгоритм плохо работает в случае, когда один кластер значительно больше остальных, и они находятся близко друг от друга – возникает эффект "расщепления" большого кластера.



Эффект расщепления большого кластера

Достоинства алгоритма k-средних:

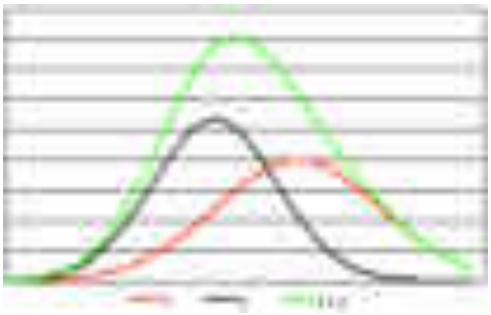
- простота использования;
- быстрота использования;
- понятность и прозрачность алгоритма.

Недостатки алгоритма k-средних:

- алгоритм слишком чувствителен к выбросам, которые могут искажать среднее. Возможным решением этой проблемы является использование модификации алгоритма - алгоритм k-медианы;
- алгоритм может медленно работать на больших базах данных. Возможным решением данной проблемы является использование выборки данных.

EM-алгоритм (Expectation-Maximization)

Среди неиерархических алгоритмов, не основанных на расстоянии, следует выделить EM-алгоритм (Expectation-Maximization). В нем вместо центров кластеров предполагается наличие функции плотности вероятности для каждого кластера с соответствующим значением математического ожидания и дисперсией. В смеси распределений (рис. 2) ведется поиск их параметров (средние и стандартные отклонения) по принципу максимума правдоподобия. Алгоритм EM и есть одна из реализаций такого поиска. Проблема заключается в том, что перед стартом алгоритма выдвигается гипотеза о виде распределений, которые оценить в общей совокупности данных сложно.



Распределения и их смесь

Еще одна проблема появляется тогда, когда атрибуты объекта смешанные – одна часть имеет числовой тип, а другая часть – категорийный. Например, пусть требуется вычислить расстояние между следующими объектами с атрибутами (Возраст, Пол, Образование):

(1) {23, муж, высшее}

(2) {25, жен, среднее}.

Первый атрибут является числовым, остальные – категорийными. Если мы захотим воспользоваться классическим иерархическим алгоритмом с какой-либо мерой сходства, нам придется каким-то образом произвести дискредитацию атрибута "Возраст". Например, так:

(1) {до 30 лет, муж, высшее}

(2) {до 30 лет, жен, среднее}.

При этом часть информации, мы, безусловно, потеряем. Если же мы будем определять расстояние в евклидовом пространстве, то возникнут вопросы с категорийными атрибутами. Понятно, что расстояние между "Пол муж" и "Пол жен" равно 0, т.к. значения этого признака находятся в шкале наименований. А атрибут "Образование" можно измерить как в шкале наименований, так и в шкале порядка, присвоив каждому значению определенные балл. Какой вариант выбрать? А что делать, если категорийные атрибуты важнее числовых? Решение этих проблем ложится на плечи аналитика. Кроме того, при использовании алгоритма k-средних и ему подобных возникают трудности с пониманием центров кластеров у категорийных атрибутов, априорным заданием количества кластеров.

Оценка качества кластеризации

Оценка качества кластеризации может быть проведена на основе следующих процедур:

- ручная проверка;
- установление контрольных точек и проверка на полученных кластерах;
- определение стабильности кластеризации путем добавления в модель новых переменных;
- создание и сравнение кластеров с использованием различных методов. Разные методы кластеризации могут создавать разные кластеры, и это является нормальным явлением. Однако создание схожих кластеров различными методами указывает на правильность кластеризации.

Лекция 8. Поиск ассоциативных правил

В последнее время неуклонно растет интерес к методам обнаружения знаний в базах данных. Объемы современных баз данных, которые весьма внушительны, вызвали устойчивый спрос на новые масштабируемые алгоритмы анализа данных. Одним из популярных методов обнаружения знаний стали алгоритмы поиска ассоциативных правил.

Обучение ассоциативным правилам (далее Associations rules learning — ARL) или поиск ассоциативных правил— это метод обучения машин на базе правил обнаружения интересующих нас связей между переменными в большой базе данных. Метод предлагается для установления сильных правил, обнаруженных в базе данных с помощью некоторых мер интересности.

Этот основанный на правилах подход генерирует также новые правила по мере анализа дополнительных данных. Конечной целью, исходя из достаточно большого набора данных, помочь машине имитировать выделение признаков человеческим и создать возможность нахождения абстрактных ассоциаций из новых неклассифицированных данных.

Ассоциативные алгоритмы относятся к обучению без учителя и определяют вероятность того, что несколько предметов окажутся вместе в определённом наборе данных. В основном они используются при анализе рыночной корзины.

Самый распространённый среди них **алгоритм добычи данных Apriori**. Обычно он применяется в транзакционных базах данных. С его помощью можно найти часто встречающиеся наборы элементов, а затем сформировать из них определённые ассоциативные правила.

Например, если человек покупает молоко и хлеб, то, скорее всего, он также возьмёт яйца. Такие прогнозы строятся исходя из предыдущих покупок разных посетителей. После чего создаются ассоциативные правила в соответствии с конкретным показателем доверенности, который определяется алгоритмом на основе того, как часто встречаются эти товары вместе.

Транзакционная или операционная база данных (Transaction database) представляет собой двумерную таблицу, которая состоит из номера транзакции (TID) и перечня покупок, приобретенных во время этой транзакции.

Исследование компании Teradata

В 1992 году группа по консалтингу в области ритейла компании Teradata провела исследование **1.2 миллиона транзакций в 25 магазинах** для ритейлера Osco Drug (Drug Store — формат разнокалиберных магазинов у дома). После анализа всех этих транзакций самым сильным правилом получилось «**Между 17:00 и 19:00 чаще всего пиво и подгузники покупают вместе**». Такое правило показалось руководству Osco Drug настолько контринтуитивным, что ставить подгузники на полках рядом с пивом они не стали. Хотя объяснение паре пиво-подгузники вполне себе нашлось: когда оба члена молодой семьи возвращались с работы домой (как раз

часам к 5 вечера), жены обычно отправляли мужей за подгузниками в ближайший магазин. И мужья, недолго думая, совмещали приятное с полезным — покупали подгузники по заданию жены и пиво для собственного вечернего времяпрепровождения. Существует много мнений, насколько история истинна.

Кроме примера выше об анализе рыночной корзины, ассоциативные правила используются ныне во многих других областях, включая

- Web mining,
- обнаружение вторжений,
- непрерывное производство и
- биоинформатику.

Ассоциативные правила

Обучение ассоциативным правилам обычно не учитывает порядок элементов внутри транзакции или по транзакциям.

Пусть имеется база данных, состоящая из покупательских транзакций. Каждая транзакция – это набор товаров, купленных покупателем за один визит. Такую транзакцию еще называют рыночной корзиной.

Задача поиска ассоциативных правил ставится следующим образом:

- Пусть дан набор $I = \{i_1, i_2, \ldots, i_n\}$ из n двоичных атрибутов, называемых объектами, то есть это множество (набор) товаров, называемых также элементами.
- Пусть дан набор $D = \{t_1, t_2, \ldots, t_m\}$ транзакций, называемый базой данных или коллекцией.

Каждая транзакция в D имеет уникальный ID (номер) транзакции и состоит из подмножества объектов из ***I***.

- Бинарная коллекция

  $$I = \{i_1, i_2, \ldots i_n\} = \{\text{хлеб, вода, молоко, масло, …}\}$$

- t_1 = [1,0,0,0,1,0,0,1,1,0,0, …] первая транзакция (*itemset*)
- t_2 = [1,1,0,0,1,0,1,1,1,0,0, …] вторая транзакция (*itemset*)
- t_3 = [0,0,1,0,1,1,0,0,0,1,0, …] …
- …
- t_m = [1,0,0,1,0,0,0,1,1,0,1, …]

Любое правило состоит из двух различных наборов объектов, известных также как ***наборы объектов X*** и ***Y***, где ***X*** называется *первым операндом* или *левой частью*, а ***Y*** — *вторым операндом* или *правой частью*.

Правило определяется как импликация вида:

$$X \Rightarrow Y, \text{ где } X,Y \subseteq I.$$

Примером правила для супермаркета может служить

{масло, хлеб} ⟹ {молоко},

что означает, что, если куплены масло и хлеб, покупатель также купит и молоко.

Каждая транзакция представляет собой бинарный вектор, где t[k]=1, если i_k элемент присутствует в транзакции, иначе t[k]=0.

Мы говорим, что транзакция T содержит некоторый набор элементов X из I, если $X \subseteq T$.

Ассоциативным правилом называется импликация $X \Rightarrow Y$, где $X \subset I$, $Y \subset I$ и $X \cap Y = \emptyset$.

Правило $X \Rightarrow Y$ имеет **поддержку** s (*support*), если s% транзакций из D, содержат $X \Rightarrow Y$, то есть удовлетворяют этому правилу, $supp(X \Rightarrow Y) = supp(X \cup Y)$.

Достоверность правила показывает какова вероятность того, что из X следует Y.

Правило $X \Rightarrow Y$ справедливо с достоверностью (*confidence*) **c**, если c% транзакций из D, содержащих X, также содержат Y, $conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$.

Покажем на конкретном примере:

- *75% транзакций, содержащих хлеб, также содержат молоко. 3% от общего числа всех транзакций содержат оба товара.*
- 75% – это **достоверность** (*confidence*) правила,
- 3% – это **поддержка** (*support*), или 'Хлеб' 'Молоко' с вероятностью 75%.

Другими словами, целью анализа является установление следующих зависимостей: если в транзакции встретился некоторый набор элементов **X**, то на основании этого можно сделать вывод о том, что другой набор элементов **Y** также должен появиться в этой транзакции. Установление таких зависимостей дает нам возможность находить очень простые и интуитивно понятные правила.

Алгоритмы поиска ассоциативных правил предназначены для нахождения всех правил X Y, причем поддержка и достоверность этих правил должны быть выше некоторых наперед определенных порогов, называемых соответственно **минимальной поддержкой** (*minsupport*) и **минимальной достоверностью** (*minconfidence*).

Задача нахождения ассоциативных правил разбивается на две подзадачи:

- Нахождение всех наборов элементов, которые удовлетворяют порогу *minsupport*. Такие наборы элементов называются часто встречающимися.
- Генерация правил из наборов элементов, найденных согласно п.1. с достоверностью, удовлетворяющей порогу *minconfidence*.

Значения для параметров *минимальная поддержка* и *минимальная достоверность* выбираются таким образом, чтобы ограничить количество найденных правил.

Если поддержка имеет большое значение, то алгоритмы будут находить правила, хорошо известные аналитикам или настолько очевидные, что нет никакого смысла проводить такой анализ. С другой стороны, низкое значение поддержки ведет к генерации огромного количества правил, что, конечно, требует существенных вычислительных ресурсов. **Тем не менее, большинство интересных правил находится именно при низком значении порога поддержки**. Хотя слишком низкое значение поддержки ведет к генерации статистически необоснованных правил.

Поиск ассоциативных правил совсем не тривиальная задача, как может показаться на первый взгляд. Одна из проблем – алгоритмическая сложность при нахождении часто встречающих наборов элементов, т.к. с ростом числа элементов в I (| I |) экспоненциально растет число потенциальных наборов элементов.

Пример из области супермаркета

Чтобы проиллюстрировать концепцию, используем маленький пример из области супермаркета. Множество объектов **I** — это молоко, хлеб, масло, пиво, памперсы, и в таблице ниже показана маленькая база данных, содержащая объекты, в которой значение 1 означает наличие объекта в соответствующей транзакции, а значение 0 означает отсутствие объекта в транзакции.

Пример базы данных с 5 транзакциями и 5 элементами

ID транзакции	молоко	хлеб	масло	пиво	памперсы
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

Таким образом, датасет представляет собой разреженную матрицу со значениями {1,0}. Это будет **бинарный датасет**. Существуют и другие виды записи – **вертикальный датасет** (показывает для каждого отдельного *item* вектор транзакций, где он присутствует) и **транзакционный датасет** (примерно как в кассовом чеке).
Данные преобразовали, как найти правила?
Существует целый ряд ключевых понятий в ARL, которые нам помогут эти правила вывести.

Полезные концепции

Чтобы выбрать вызывающее интерес правило из множества всех возможных правил, используются ограничения на различные меры значимости и содержательности. Наиболее известными ограничениями являются **минимальный порог поддержки и доверия.**

Пусть X будет набором объектов, $X \Rightarrow Y$ будет ассоциативным правилом, а T — набором транзакций данной базы данных.

Поддержка (Support)

Поддержка — это показатель, насколько часто набор объектов обнаруживается в базе данных.

Поддержка набора X по отношению к T определяется как отношение числа транзакций t в базе данных, содержащих набор X, к общему числу транзакций.

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

где X — itemset, содержащий в себе i-items, а |T| — количество транзакций.

В нашем примере набор данных X={пиво, памперсы} имеет поддержку 1/5 = 0,2, поскольку он обнаруживается в 20 % всех транзакций (1 из 5 транзакций). Аргумент функции supp() является множеством предусловий и потому становится более ограничивающим по мере расширения (в отличие от более охватывающего).

Доверие, достоверность (Confidence)

Доверие — это показатель, насколько часто правило оказывается верным. Значение доверия правила $X \Rightarrow Y$ по отношению к набору транзакций T является отношением числа транзакций, которые содержат как набор X, так и набор Y, к числу транзакций, содержащих набор X.

Доверие определяется как:

$$conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$$

Например, правило {масло, хлеб} $\Rightarrow$ {молоко} имеет доверие 0,2/0,2=1,0 в базе данных, что означает, что для 100% транзакций, содержащих масло и хлеб, правило верно (в 100 % случаев, когда покупается масло и хлеб, молоко покупается также).

Заметим, что supp(X∪Y) означает поддержку объектов в X и Y. Это несколько запутывает, поскольку мы обычно думаем в терминах вероятности событий, а не терминах набора объектов. Мы можем переписать supp(X∪Y) как вероятность P(EX ∩

EY), где EX и EY являются событиями, что транзакция содержит наборы X и Y соответственно.

Доверие можно понимать как оценку условной вероятности P(EY|EX), вероятности нахождения правой части правила в транзакциях при условии, что транзакции содержат левую часть правила.

Лифт (lift)

Лифт правила определяется как

$$\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X) \times \mathrm{supp}(Y)}$$

или отношение наблюдаемой поддержки к математическому ожиданию события, если бы X и Y были бы независимы. Например, правило {молоко, хлеб} => {масло} имеет лифт

$$\frac{0.2}{0.4 \times 0.4} = 1.25$$

Если правило имеет лифт 1, это означает, что событие в левой части независимо от события в правой части. Если два события независимы, никакого правила нельзя вытащить из этих двух событий.

Если лифт > 1, это позволяет нам знать степень, насколько события связаны друг с другом, и делает эти правила потенциально полезными для предсказания следствия в будущих наборах данных.

Если лифт < 1, это означает, что объекты заменяют друг друга. Это означает, что наличие одного объекта имеет отрицательный эффект на присутствие второго объекта, и наоборот.

Значение лифта принимает во внимание как доверие правила, так и общие данные.

Уверенность, убедительность (Conviction)

Уверенность правила определяется как

$$\mathrm{conv}(X \Rightarrow Y) = \frac{1 - \mathrm{supp}(Y)}{1 - \mathrm{conf}(X \Rightarrow Y)}$$

Например, правило {молоко, хлеб} => {масло} имеет уверенность

$$\frac{1-0,4}{1-0,5}=1,2$$

и может пониматься как отношение ожидаемой частоты, что X встречается без Y (говоря иначе, частота, что правило даёт неправильное предсказание), если бы X и Y были бы независимыми, и наблюдаемой частоты неверных предсказаний.

Процесс

От ассоциативных правил обычно требуется выполнение определённой пользователем минимальной поддержки и определённого пользователем минимального доверия. Генерация ассоциативного правила обычно разделяется на два шага:

1. Минимальный порог поддержки используется для поиска всех частых наборов объектов в базе данных.
2. Ограничение минимального доверия применяется к этим наборам для формирования правила.

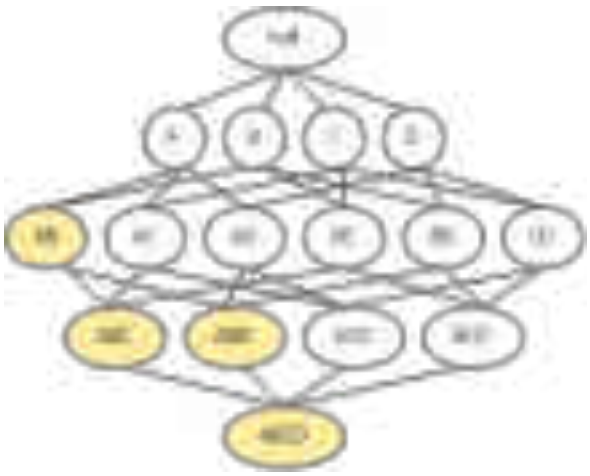
Второй шаг прост и ясен, а первый шаг требует большего внимания.

Свойство анти-монотонности

Выявление часто встречающихся наборов элементов – операция, требующая много вычислительных ресурсов и, соответственно, времени. Примитивный подход к решению данной задачи – простой перебор всех возможных наборов элементов. Это потребует O(2|I|) операций, где |I| – количество элементов. Поэтому используют одно из свойств поддержки, гласящее: поддержка любого набора элементов не может превышать минимальной поддержки любого из его подмножеств. Например, поддержка 3-элементного набора {Хлеб, Масло, Молоко} будет всегда меньше или равна поддержке 2-элементных наборов {Хлеб, Масло}, {Хлеб, Молоко}, {Масло, Молоко}.

Дело в том, что любая транзакция, содержащая {Хлеб, Масло, Молоко}, также должна содержать {Хлеб, Масло}, {Хлеб, Молоко}, {Масло, Молоко}, причем обратное не верно. Это свойство носит название **анти-монотонности** и служит для снижения размерности пространства поиска.

Свойству анти-монотонности можно дать и другую формулировку: с ростом размера набора элементов поддержка уменьшается, либо остается такой же. Из всего вышесказанного следует, что любой k-элементный набор будет часто встречающимся тогда и только тогда, когда все его (k-1)-элементные подмножества будут часто встречающимися. Все возможные наборы элементов из I можно представить в виде решетки, начинающейся с пустого множества, затем на 1 уровне 1-элементные наборы, на 2-м – 2-элементные и т.д. На k уровне представлены k-элементные наборы, связанные со всеми своими (k-1)-элементными подмножествами.



Рассмотрим рисунок иллюстрирующий набор элементов I – {A, B, C, D}. Предположим, что набор из элементов {A, B} имеет поддержку ниже заданного порога и, соответственно, не является часто встречающимся. Тогда, согласно свойству антимонотонности, все его супермножества также не являются часто встречающимися и отбрасываются. Вся эта ветвь, начиная с {A, B}, отмечена желтым фоном. Использование этой эвристики позволяет существенно сократить пространство поиска.

Алгоритмы

В процессе поиска ассоциативных правил может производиться обнаружение всех ассоциаций, поддержка и достоверность для которых превышают заданный минимум.

Простейший алгоритм поиска ассоциативных правил рассматривает все возможные комбинации условий и следствий, оценивает для них поддержку и достоверность, а затем исключает все ассоциации, которые не удовлетворяют заданным ограничениям. Число возможных ассоциаций с увеличением числа предметов растет экспоненциально. Поэтому в процессе генерации ассоциативных правил широко используются методики, позволяющие уменьшить количество ассоциаций, которое требуется проанализировать.

Было предложено много алгоритмов для генерации ассоциативных правил.

Несколько алгоритмов хорошо известны, это Apriori, Eclat и FP-Growth, но они делают только половину работы, поскольку они предназначены для отыскания часто встречающихся наборов объектов. Нужно сделать ещё один шаг после того, как часто встречающиеся наборы найдены в базе данных.

Алгоритм Apriori

В основе алгоритма Apriori лежит понятие частого набора. Под частотой понимается простое количество транзакций, в которых содержится данный предметный набор.

Частый предметный набор – предметный набор с поддержкой больше заданного порога либо равной ему. Этот порог называется **минимальной поддержкой.**

Работа данного алгоритма состоит из нескольких этапов, каждый из этапов состоит из следующих шагов:

1. *формирование кандидатов;*
2. *подсчет кандидатов.*

Формирование кандидатов (candidate generation) - этап, на котором алгоритм, сканируя базу данных, создает множество i-элементных кандидатов (i - номер этапа). На этом этапе поддержка кандидатов не рассчитывается.

Подсчет кандидатов (candidate counting) - этап, на котором вычисляется поддержка каждого i-элементного кандидата. Здесь же осуществляется отсечение кандидатов, поддержка которых меньше минимума, установленного пользователем (min_sup).

Оставшиеся i-элементные наборы называем **часто встречающимися**.

Чтобы сократить пространство поиска ассоциативных правил, алгоритм Apriori использует свойство антимонотонности. Свойство утверждает, что если предметный набор Z не является частым, то добавление некоторого нового предмета A к набору Z не делает его более частым. Данное полезное свойство позволяет значительно уменьшить пространство поиска ассоциативных правил.

На первом этапе алгоритма Apriori формируются частые однопредметные наборы – множество F_1.

Для поиска F_k, то есть k-предметных наборов, алгоритм Apriori сначала создает множество F_k кандидатов в k-предметные наборы путем связывания множества F_{k-1} с самим собой. Затем F_k сокращается с использованием свойства антимонотонности.

Предметные наборы множества F_k, которые остались после сокращения, формируют F_k.

После того, как все частые предметные наборы найдены, можно переходить к генерации на их основе ассоциативных правил. Для этого к каждому частому предметному набору s можно применить процедуру, состоящую из 2 шагов.

1. Генерируются все возможные поднаборы s
2. Если поднабор ss является непустым поднабором s, то рассматривается ассоциация R:ss→(s-ss), где s-ss представляет собой набор s без поднабора ss. R считается ассоциативным правилом, если удовлетворяет условию заданного минимума

поддержки и достоверности. Данная процедура повторяется для каждого подмножества ss из s.

Рассмотрим работу алгоритма Apriori на примере базы данных D. Минимальный уровень поддержки равен 3.

На первом этапе происходит формирование одноэлементных кандидатов. Далее алгоритм подсчитывает поддержку одноэлементных наборов. Наборы с уровнем поддержки меньше установленного, то есть 3, отсекаются. В нашем примере это наборы e и f, которые имеют поддержку, равную 1. Оставшиеся наборы товаров считаются часто встречающимися одноэлементными наборами товаров: это наборы a, b, c, d.

Далее происходит формирование двухэлементных кандидатов, подсчет их поддержки и отсечение наборов с уровнем поддержки, меньшим 3. Оставшиеся двухэлементные наборы товаров, считающиеся часто встречающимися двухэлементными наборами ab, ac, bd, принимают участие в дальнейшей работе алгоритма.

Если смотреть на работу алгоритма прямолинейно, на последнем этапе алгоритм формирует трехэлементные наборы товаров: abc, abd, bcd, acd, подсчитывает их поддержку и отсекает наборы с уровнем поддержки, меньшим 3. Набор товаров abc может быть назван часто встречающимся.

Однако алгоритм Apriori уменьшает количество кандидатов, отсекая априори тех, которые заведомо не могут стать часто встречающимися, на основе информации об отсеченных кандидатах на предыдущих этапах работы алгоритма.

Отсечение кандидатов происходит на основе предположения о том, что у часто встречающегося набора товаров все подмножества должны быть часто встречающимися.

Если в наборе находится подмножество, которое на предыдущем этапе было определено как нечасто встречающееся, этот кандидат уже не включается в формирование и подсчет кандидатов.

Так наборы товаров ad, bc, cd были отброшены как нечасто встречающиеся, алгоритм не рассматривал товаров abd, bcd, acd.

При рассмотрении этих наборов формирование трехэлементных кандидатов происходило бы по схеме, приведенной в верхнем пунктирном прямоугольнике. Поскольку алгоритм априори отбросил заведомо нечасто встречающиеся наборы, последний этап алгоритма сразу определил набор abc как единственный трехэлементный часто встречающийся набор (этап приведен в нижнем пунктирном прямоугольнике).

Алгоритм Apriori рассчитывает также поддержку наборов, которые не могут быть отсечены априори. Это так называемая негативная область (negative border), к ней принадлежат наборы-кандидаты, которые встречаются редко, их самих нельзя

отнести к часто встречающимся, но все подмножества данных наборов являются часто встречающимися.

Статистически обоснованные ассоциации

Одним из ограничений стандартного подхода к обнаружению ассоциаций является то, что при поиске в большом числе возможных ассоциаций набора объектов, которые могут быть ассоциированными, есть большой риск нахождения большого числа случайных ассоциаций. Это наборы объектов, которые оказываются вместе с неожиданной частотой в данных, но чисто случайно. Например, предположим, что мы рассматриваем набор из 10.000 объектов и ищем правило, содержащее два объекта в левой части и один объект в правой части. Имеется примерно 1.000.000.000.000 таких правил. Если мы применим статистический тест независимости с уровнем 0,05 это означает, что имеется только 5 % шанса принять правило при отсутствии ассоциации.

Если мы предполагаем, что нет никаких ассоциаций, мы должны, тем не менее, ожидать нахождения 50.000.000.000 правил. Статистически обоснованное обнаружение ассоциаций контролирует этот риск, в большинстве случаев сокращая риск нахождения любой случайной ассоциации для заданного пользователем уровня значимости.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Тематика лабораторных работ

Номер	Наименование лабораторной работы
1	Ввод и предварительный анализ данных
2	Регрессионный анализ
3	Факторный анализ
4	Кластерный анализ

Лабораторная работа 1. Ввод и предварительный анализ данных

Ввод данных

1. В специальной папке STATISTICA на рабочем столе выберите модуль *Работа с данными* и запустите его на выполнение.
2. Используя описание пакета, познакомьтесь с основными элементами интерфейса.
3. Используя команду **File|New Data…**, создайте таблицу следующей структуры:

Таблица 1

Примечание: Объектам (случаям) дать имена – фамилии студентов.

4. Введите исходные данные в созданную таблицу (см. табл.1).
5. В каталоге STATISTICA сохраните ее под именем AABB. STA, где AA – номер группы, BB – год поступления, например 2198.sta.

Первичная обработка данных

1. *Обработка пропусков.* Способ обработки пропущенных данных может корректироваться индивидуально для каждого вида анализа. Обычно он может быть установлен из стартовой панели конкретного статистического модуля. Пользователь имеет возможность

   - устранить данные из вычислений;

   – заменить их средним значением;

   – интерполировать данные.

   - Если в вашей таблице нет пропущенных данных, создайте их искусственно.
   - В модуле по работе с данными **Data managment** – *Управление данными* при помощи команды **Replace Missing Data by Means –** *Замена пропущенных значений на среднее* произведите обработку пропусков для переменной BEC.
   - При необходимости восстановите исходное состояние таблицы.

2. *Проверка значений данных*. Команда **Verify Data Values –** *Проверка значений данных* позволяет проверить введенные данные. Проверяется выполнение некоторых, предварительно заданных пользователем условий. Имеется возможность задания до 4 условий. При это можно установить между ними операции логического НЕ, И, ИЛИ. Заданные условия можно сохранить в специальном файле.

   - Установите условия проверки, например:
     Условие 1 – v5>=140 and v5<=200
     Условие 2 – v6>=40 and v6<=100
     Условие 3 – v8<=5 and v9<=5 and v10<=5 and v11<=5
   - В секции **Range** задайте номера проверяемых случаев, например от 1 до 25.
   - Проверьте введенные данные, исправьте обнаруженные ошибки.
   - Сохраните условия в специальном файле. При следующем вызове этот файл можно загрузить и использовать для проверки.

3. *Функциональные преобразования* количественных переменных используются для преобразования данных к форме, более соответствующих природе задачи, ее «физическому смыслу». Например, часто встречаются несимметричные

распределения с длинным “хвостом”, что затрудняет статистический анализ. Распределение приобретает симметричный вид, если перейти к логарифму переменной.

Для выполнения функциональных преобразований создадим дополнительную таблицу и выполним в ней заданные действия:

- Используйте команду **Create subset from data file –** *Создать подмножество данных из файла* для создания новой таблицы из существующей. Включите в эту новую таблицу с 5 по 7 переменные (5-7) и все (**All**) случаи. Необходимые переменные для новой таблицы можно выбрать и предварительно, то есть до выбора команды **Create subset from data file.**
- Перейдите к новой переменной ln(РОСТ). Формулы преобразования переменных задаются в диалоговом окне спецификаций переменных.
- Вернитесь к прежней переменной РОСТ, используя функцию Exp (=Exp(v1))

4. *Нормировка количественных признаков* используется для приведения их к стандартному виду, удобному для обработки. Обычно при расчете расстояния между объектами нормируют признаки, измеренные в разнородных физических единицах. В STATISTICA значения переменных изменяются на стандартизованные значения по следующей формуле:

Новое значение = (Старое значение – Среднее) / Стандартное отклонение

- Выполните стандартизацию переменных РОСТ и ВЕС по команде **Standardize Variables**.
- Вернитесь к первоначальной таблице.

5. *Сортировка данных.* Команда **Sort Cases – Сортировка случаев** выполняет иерархическую сортировку данных. На каждом уровне сортировки (их число ограничено 7 уровнями) должен быть задан ключ. В качестве ключа могут быть использованы либо переменные, либо случаи. Для различных ключей допускаются различные направления сортировки: по убыванию и возрастанию. Сортировка может проводиться как по текстовым, так и по числовым значениям переменной.

- Отсортируйте данные, например, по полу и внутри пола по убыванию роста.
- Вернитесь к прежней сортировке (по случаям, текстовое значение).

Работа с данными в системе STATISTICA

1. Добавьте еще одну переменную в конец таблицы. Имя новой переменной – СРБАЛЛ (средний балл по всем предметам), формат 6.2. В диалоговом окне

Спецификации переменной секции **Long name** задайте формулу расчета значений новой переменной и произведите расчет.

2. Переместите новую переменную на другую позицию – после переменной BEC.
3. Полученную таблицу добавьте в конец общей таблицы, содержащей обобщенную информацию по всем группам. Таблица ROSTVES.STA хранится в каталоге STATISTICA. Для выполнения указанной операции используйте команду **Merge two Data Files** – *Объединить два файла данных*.
4. Отработайте основные операции над случаями и переменными, приведенные в описании пакета.
5. Изучите операции с выделенным блоком значений.

Предварительный анализ данных. Визуализация данных

1. *Настройка графических компонент*. Перед началом работы с графиками желательно произвести предварительную настройку. Инструменты настройки доступны только при открытом графике, поэтому:
   - Откройте произвольный график, например, двумерную диаграмму рассеяния BEC–POCT. Для этого выделите произвольную ячейку в столбце POCT, на ПИ нажмите клавишу **Quick Stats Graphs** (на кнопке буква S) и затем последовательно – **Scatterplot by…| regular | BEC**.
   - Откройте панель настройки **Options|Global Defaults…**
   - Выберите шрифты для всех заголовков в секции TITLES.
   - Выберите шрифты для символов осей в секции AXES.
   - Сохраните установки в специальном файле, чтобы в следующий раз не устанавливать все заново.

Все компоненты графика могут быть настроены при помощи двух основных диалоговых окон – **General Layout –** *Общая разметка* и **Plot Layout –** *Размещение графика*. Из диалогового окна **General Layout** можно перейти в **Plot Layout** и наоборот.

Рекомендация. Иногда проще всего задать параметры настройки, дважды щелкнув на том или ином элементе графика, либо один раз нажать на правую кнопку мыши и выбрать необходимую команду из контекстного меню.

2. *Проекция данных на один признак – гистограмма.* Работа с гистограммой предполагает
   - оценку наличия группировки объектов. Если на гистограмме просматриваются 2-3 горба, то это явный признак наличия групп;

- оценку параметров теоретического распределения. Если гистограмма примерно симметрична, ее “хвосты” не слишком длинные (например, не больше пяти объектов на сто лежат вне интервала ±2s), то оценками параметров являются среднее значение выборки $\bar{x}$ и выборочное стандартное отклонение s. Значения $\bar{x}$ и s используются в качестве характеристик положения и разброса всех выборок независимо от того, из какого они сделаны распределения. При резко асимметричной гистограмме более удобной характеристикой центра является медиана, которая более устойчива к резким выбросам в данных, чем среднее;
- анализ ‘засоренности’, выявление ошибок в данных. Подозрительные объекты обычно находятся на краях диаграммы.

Выполните следующие действия:

- Постройте гистограммы переменных РОСТ и ВЕС. Найдите основные статистики (БСГ|Values | Stats of РОСТ). Проверьте на наличие групп.
- Постройте гистограммы переменных РОСТ и ВЕС отдельно для студентов и студенток.
- Проведите анализ полученных диаграмм.

3. *Проекция на плоскость двух признаков – двумерная диаграмма рассеяния*.
   - Постройте двумерные гистограммы вида «рост-вес», «вес-объем талии» (см. рис.1).
   - Проверьте диаграммы на ошибки и выбросы.
   - Исследуйте влияние размерности на вид диаграмм.

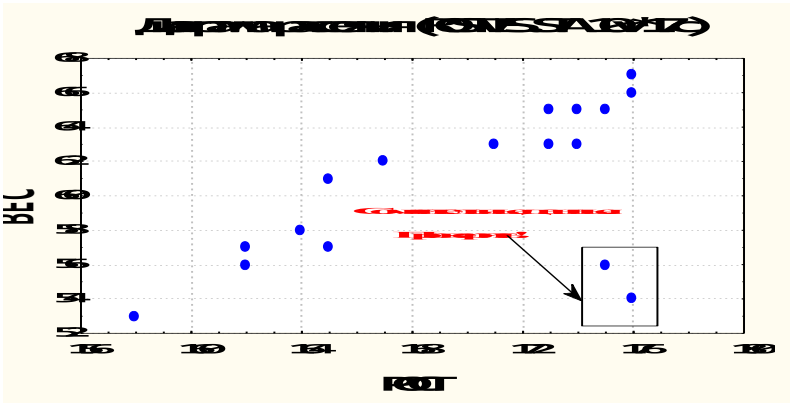


Рис. 1

4. *Проекция на плоскость трех признаков – объемная диаграмма рассеяния*
   - Постройте объемную гистограмму вида «рост-вес, объем талии».

- Изучите возможности вращения и прокрутки объектов. Проверьте работу в режиме анимационного расслоения.

Вероятностный калькулятор

Вероятностный калькулятор – это средство, позволяющее максимально быстро построить график наиболее употребляемых функций и их плотностей, вычислить процентные точки.

- Запустите модуль **Basic Statistics/Tables** из **Переключателя модулей.** Откройте окно **Probability calculator**.
- Изучите стандартные распределения (нормальное, Стьюдента, хи-квадрат, F-распределение, логарифмически-нормальное, биномиальное), задавая различные значения параметров распределения. Как меняются графики распределений при изменении среднего, стандартного отклонения, степени свободы?
- Научитесь рассчитывать вероятность по заданному значению и наоборот. Предполагая, что распределение роста в группах студентов подчиняется нормальному закону, рассчитайте вероятность, что случайно выбранный студент имеет рост выше 200 см.
- Исследуйте на графике нормального распределения правило 2 и 3 сигма. Задавая различные параметры нормального распределения и указывая в строке Z вероятностного калькулятора 2 и 3 стандартных отклонения, убедитесь, правила 2 и 3 сигм действительно справедливы.

Генерация случайных чисел в Statistica

📖 Исследуйте зависимость стандартной ошибки выборочного среднего и выборочной дисперсии от объема выборки, для чего

1. Создайте выборку объемом N из генеральной совокупности с нормальным распределением N(μ,σ) и произвольными параметрами.
2. По выборке рассчитайте выборочное среднее $\bar{x}$ и выборочное стандартное отклонение s.
3. Повторите действия 1 и 2 несколько k раз.
4. Постройте гистограмму для выборочного среднего $\bar{x}$ и стандартного отклонения s.
5. Рассчитайте стандартную ошибку выборочного среднего.
6. Сравните стандартную ошибку среднего со стандартным отклонением и сделайте выводы.

📖 Исследуйте влияние объема выборки на достоверность результатов статистического анализа, для чего

1. Сформируйте выборку объемом N из генеральной выборки, имеющей стандартизованное нормальное распределение N(μ=0,σ=1).
2. По каждому признаку найдите среднее значение координат выбранных объектов

$$m_i=(x^{(j)}_1+ x^{(j)}_2 + \ldots+x^{(j)}_N)/N, \quad j=1,p.$$

3. В качестве критерия оценки координат центра совокупности возьмем расстояние от полученной по выборке точки с координатами (m1,m2,…,mp) до истинного центра (M1,M2,…,MP), т.е. до начала координат

$$L = \sqrt{(M_1 - m_1)^2 + (M_2 - m_2)^2 + \ldots + (M_p - m_p)^2} = \sqrt{m_1^2 + m_2^2 + \ldots + m_p^2}$$

Величина L характеризует ошибку, которую мы допустим, если припишем всей совокупности координаты центра сделанной небольшой выборки.

Повторите указанные действия для различных значений N и P:

N ={10, 30, 100, 300,1000} P = {1,2,5,10,30,150}

На основании полученных данных постройте следующие зависимости:

- Значение ошибки L при оценивании положения центра генеральной совокупности, имеющей стандартизованное p-мерное нормальное распределение в зависимости от объема выборки N для p=10 и p=150.
- Зависимость ошибки L от числа признаков для N=10 и N=100.

Лабораторная работа 2. Регрессионный анализ

Теоретическое введение

Регрессионные методы позволяют выявить связи между переменными, причем особенно эффективно, если эти связи не совершенны или не имеют точного функционального описания между этими переменными. В анализе используются независимые переменные x и одна зависимая переменная y.

Регрессией в прямом смысле называется функция вида

$$y = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i<j}^{k} b_{ij} x_i x_j + \sum_{i=1}^{k} b_{ii} x_i^2 + \ldots + \varepsilon \qquad (1)$$

где b_0, b_{ij}, b_{ii},… - известные коэффициенты регрессии;

ε - невязка (ошибка, отклонение), обусловленная недостаточной пригодностью модели и ошибкой эксперимента. Эти причины обычно являются смешанными.

Регрессия (1) называется множественной.

В том случае, если исследуется влияние одной переменной на результат эксперимента, то выражение (1) упрощается к виду

$$y = b_0 + b_1 x_1 + \varepsilon \tag{2}$$

Выражение (2) представляет собой линейную регрессию.

Кроме того, часто используется нелинейное представление регрессии (логарифмическое, экспоненциальное, гиперболическое и другое задаваемое исследователем). В этом случае переменная x подвергается соответствующему нелинейному преобразованию.

Для нахождения коэффициентов уравнений (1) и (2) используется метод наименьших квадратов. Сущность метода заключается в том, чтобы минимизировать сумму квадратов отклонений

$$SS = \sum_{i=1}^{n} \left[y_i - y_i^0 \right]^2 \rightarrow \min \tag{3}$$

где y_i^0 - значение результата, вычисленное по уравнению (1) в точке x_i ;

y_i - экспериментальное значение результата в этой же точке.

Существует и другой смысл регрессии. В этом смысле под термином регрессии понимается метод обработки статистических данных, позволяющий подобрать некоторую сглаживающую функцию по упорядоченным данным. При этом полученное функциональное описание сглаженной кривой необязательно имеет физический смысл.

Регрессия может использоваться в разных целях:

- для получения функциональной кривой описывающей исследуемый процесс;
- для свертывания информации заданной в табличном виде в вид функциональной зависимости;
- для прогнозирования результата исследования в промежутках между дискретным описанием исследуемого процесса (задача интерполяции);

- для предсказания результата за границами исследования (задача прогнозирования).

Корректность выводов при регрессионном анализе обеспечивается при выполнении следующих условий:

1. Результаты наблюдений $y_1, y_2, \ldots, y_n$ являются независимыми, нормально распределенными случайными величинами.
2. Ошибки измерения независимых переменных $x_1, x_2, \ldots, x_n$ пренебрежимо малы по сравнению с ошибкой определения y.
3. Переменные $x_1, x_2, \ldots, x_n$ - линейно - независимые величины.

Выполнение первого условия осуществляется с использованием критериев Колмогорова - Смирнова и Шапиро-Вилкоксена в тех случаях, когда число параллельных наблюдений по каждому случаю (Case) не менее трех.

Второе условие требует, чтобы ошибки измерения факторов x_i были пренебрежимо малы по сравнению с ошибкой определения y_i .

Третье условие, налагающее ограничения на взаимную связь между значениями факторов, проявляется в процессе определения коэффициентов регрессии b. Может оказаться, что число уравнений (Case), из которых находятся b , будет меньше количества определяемых коэффициентов (количество переменных + 1).

Выполнение регрессионного анализа с использованием пакета STATISTICA включает в себя несколько этапов:

Предварительный анализ. Проводится с целью определения выполнения рассмотренных выше условий и предварительного анализа вида предполагаемой регрессии. В простых случаях этого исследования может быть достаточно для формирования выводов. Этот вид анализа выполняется при использовании блока «Описательные статистики» и при построении графиков **LinePlots (XY Trace).**

***Проведение* регрессионного анализа**. Проводится с использованием блока Line Regression. В результате анализа определяются коэффициенты уравнения регрессии при заданных условиях, уровень их значимости (p) и значения t- критерия для них, коэффициент корреляции R, значения F - критерия и уровень его значимости (p).

Анализ полученных результатов и формирование выводов. В том случае, если полученное уравнение регрессии адекватно описывает исследуемый процесс, что подтверждается высоким уровнем значимости F - критерия, делается вывод о соответствии уравнения регрессии исследуемому процессу. Если значение F- критерия не соответствует требуемому уровню значимости, гипотезу об адекватности уравнения регрессии исследуемому процессу отвергают. Анализ повторяют снова, используя различные методы преобразования переменных. Кроме того, для получения более обоснованных статистических выводов проводят анализ остатков. Это

позволяет выявить случаи (Case), которые описываются полученным уравнением регрессии, провести их анализ и сделать обоснованные выводы.

А. Примеры использования модуля регрессионного анализа в Statistica

Ниже рассмотрены два примера одномерной и многомерной линейной регрессии, иллюстрирующие работу модуля «Множественная регрессия».

Пример 1. Линейная регрессия

Формулировка задачи

В качестве примера рассмотрим реальные данные. Пусть даны оптовые цены на марочные вина в зависимости от года закладки вина (табл.3). Цены указаны в долларах за одну бутылку.

Таблица 3

ГОД	ЦЕНА	ГОД	ЦЕНА
1890	50,00	1941	10,00
1900	35,00	1944	5,99
1920	25,00	1948	8,98
1931	11,98	1950	6,98
1934	15,00	1952	4,99
1935	13,00	1955	5,98
1940	6,98	1960	4,98

Естественно предположить, что между возрастом вина и его стоимостью имеется некоторая зависимость. Интуитивно понятно, что чем больше выдержка, тем вино должно быть дороже. Данные показывают, что в общем эта тенденция выполняется, однако имеются и исключения. Итак, мы хотим приближенно выразить зависимость между этими двумя переменными. Зачем это нужно? Например, имея такую формулу, можно прогнозировать стоимость вин на следующем аукционе (в 1973 году) или, если вы планируете продавать вино, года закладки которого нет в таблице (например,1945), вы можете грамотно назначить на него цену.

Математическая постановка задачи

В нашем примере независимая переменная – год закладки (или выдержка, которая равна возрасту вина), а зависимая переменная – его цена на аукционе. Иногда используется и другая терминология. Зависимая переменная часто называется откликом, а независимые переменные – предикторами, контролируемыми

переменными или факторами. Эта терминология подчеркивает, что ряд переменных оказывает влияние на одну переменную – отклик.

В результате исследований невозможно найти точные значения параметров β_1, β_0, однако можно получить их оценки, которые мы будем обозначать через b_1, b_0 и которые являются приближениями к значениям неизвестных параметров. После того, как оценки получены, мы можем записать уравнение связи в виде

$$\hat{y} = b_1 x + b_0 \qquad (4)$$

Уравнение (4) позволяет нам найти оценку отклика при любом значении контролируемой переменной.

Запуск STATISTICA

Запустите Windows 95. Нажмите кнопку *Пуск* и в меню *Программы* выберите папку, которая содержит систему STATISTICA. В этой папке выберите ярлык программы STATISTICA и дважды щелкните на нем мышью.

Выбор статистического модуля

После запуска программы на экране появится **Переключатель модулей – Module Switcher**, при помощи которого можно выбрать необходимый для работы модуль. Выберите модуль **Множественная регрессия – Multiple Regression**. Для этого подведите указатель мыши к названию этого модуля и дважды щелкните левой кнопкой мыши.

После запуска модуля на экране откроется основное окно системы STATISTICA. При запуске системы в нее автоматически загружается последний файл, с которым вы работали в ней. Если вы запускаете STATISTICA в первый раз, то по умолчанию в ней открывается файл с исходными данными, который называется *Adstudy.sta.* Одновременно с этим появляется *Стартовая панель* модуля, содержащая основные операции, доступные в запущенном модуле, которая также позволяет определить различные параметры анализа. В *Стартовой панели* можно открыть необходимый файл данных для анализа, приписать веса переменным, отобрать (если это требуется) необходимое подмножество наблюдений и выбрать переменные для анализа (в нашем случае – зависимую и независимые переменные). Напомним, что в STATISTICA реализован принцип постоянной логической подсказки. Если вы не знаете, что нужно делать на следующем шаге обработки, то просто нажмите на клавишу ENTER. STATISTICA сама отправит вас к нужному диалоговому окну. Например, если вы не выбрали переменные для анализа, то откроется диалоговое окно выбора переменных, в котором вам будет предложено выбрать эти переменные; если вы не задали значения каких-либо параметров, то они будут заданы по умолчанию и т.д.

Создание электронной таблицы с исходными данными

Исходные данные в системе STATISTICA организованы в виде таблицы. Большинство реальных данных могут быть структурированы в табличную форму. Если читатель имеет опыт работы с электронными таблицами (например, MS Excel), то ему будет просто освоиться с электронными таблицами в STATISTICA. Электронная таблица в STATISTICA состоит из строк и столбцов. Столбцы таблицы называются **Variables** – *Переменные*, а строки **Cases** – *Наблюдения*. В качестве переменных выступают исследуемые величины, а случаи – это значения, которые принимают переменные и которые изменяются в процессе наблюдения. В нашем примере в качестве переменных могут естественно выступать *Год* закладки вина и его Цена на аукционе.

Ввод исходных данных и дополнительной информации

Переменные в электронной таблице могут принимать как текстовые, так и численные значения. Электронная таблица с данными из нашего примера приведена на рис. N. В первом столбце содержится переменная *Год* (год закладки), во втором – переменная *Цена* – цена бутылки (в долларах). Кроме значений переменных, таблица может содержать дополнительную информацию (название таблицы, комментарии об источнике данных и т.д.)

Перед непосредственным применением той или иной статистической процедуры часто возникает необходимость преобразования данных. Так, например, может потребоваться вычислить *Возраст* (выдержку) вина, которая определяется как разность года аукциона (1972) и года закладки вина.

Преобразование исходных данных

В электронных таблицах STATISTICA вы можете выполнить все необходимые преобразования. Например, перейдем от переменной *Год* и *Цена* к новым переменным *Возраст* и *Цена_Лог*, которые связаны с исходными данными при помощи формул

Возраст = 1972 – *Год, Цена_Лог* = ln(*Цена*).

После этого таблица будет содержать четыре переменные и примет вид, изображенный на рис 2.1.

Формулы преобразований переменных задаются в диалоговом окне спецификаций переменной. Для его вызова достаточно дважды щелкнуть мышью на имени переменной в электронной таблице с исходными данными.

Визуализация данных

Теперь имеет смысл отобразить данные на графике. *STATISTICA* включает в себя большое количество разнообразных категорий и типов графиков. Это всевозможные графики на плоскости и в пространстве, включая научные графики в различных системах координат, деловые графики и диаграммы, специализированные статистические графики (включая гистограммы, матричные, категоризированные

графики, диаграммы рассеяния и др.), пиктографики. Графические средства системы *STATISTICA* доступны в любом модуле и на любом шаге статистического анализа.

Рис 2.1 Преобразование переменных в таблице и диалоговое окно, в котором задаются все спецификации переменных

Для вызова графических возможностей системы можно воспользоваться меню **Graphics –** *Графика* и выбрать необходимый тип графика. В нашем примере мы воспользуемся двумерными диаграммами рассеяния. В диалоговом окне при помощи кнопки **Variables –** *Переменные* выберите необходимые переменные, которые вы хотите отобразить графически и необходимый тип графика. После нажатия на кнопку ОК график будет выведен в отдельном окне на рабочем пространстве системы.

После построения этих графиков на рабочем пространстве системы будет открыто три окна. В одном из них будет электронная таблица с исходными данными, а в двух других – графики. Пользователь имеет возможность сохранить необходимые ему графики в различных форматах или вывести их на принтер. Если нет необходимости сохранять графики, можно просто закрыть окно с ними.

Замечание

Преобразование логарифма стабилизирует дисперсию и часто применяется в статистике. Его можно интерпретировать и следующим образом: чем выше абсолютное значение переменной, тем выше и уровень случайных ошибок. При логарифмировании все ошибки становятся примерно одинаковыми. Поэтому мы будем искать линейную зависимость не между *Возрастом* и *Ценой*, а между *Возрастом* и логарифмированной ценой, получая при этом более устойчивые оценки параметров модели. Впоследствии, когда модель будет построена, можно перейти к исходным величинам. Итак, наша задача построить модель вида:

$$\text{ЦЕНА\_ЛОГ} = b_1 * \text{ВОЗРАСТ} + b_0, \quad (5)$$

где b_1 – неизвестный коэффициент; b_0 – свободный член (также неизвестен).

При этом модель для цены будет иметь вид:

$$ЦЕНА = \exp(b_1*ВОЗРАСТ+b_0), \quad (6)$$

Мы не только оценим неизвестные параметры, но исследуем значимость регрессии и адекватность построенной модели исходным данным.

Вызов стартовой панели модуля и определение анализа

Для начала статистического анализа вам необходимо вызвать *Стартовую панель* модуля. Это основное диалоговое окно модуля, в котором необходимо задать различные опции анализа. Если *Стартовая панель* модуля закрыта, то откройте ее. Для этого войдите в меню **Analysis –** *Анализ* и выберите команду **Startup Panel –** *Стартовая панель*.

Выбор переменных для анализа

Далее необходимо выбрать переменные для анализа. В нашем случае имеется одна зависимая переменая *Цена_Лог* и одна независимая переменная *Возраст*. Для их задания воспользуйтесь кнопкой **Variables –** *Переменные* из *Стартовой панели*.

В открывшемся окне **Select dependent and independent variable list –** *Выбор зависимой и списка независимых переменных* в качестве зависимой переменной выберите переменную с номером *4 – Цена_Лог*, а в качестве независимой переменной переменную с номером *3 – Возраст*. Нажмите кнопку **ОК** в правом верхнем углу. Вы вновь окажетесь в *Стартовой панели* модуля **Множественная регрессия**.

Задание дополнительных параметров анализа

Заметьте, что в *Стартовой панели* вы можете задать и дополнительные опции и параметры анализа. Например, вы можете выбрать определенное подмножество наблюдений для анализа, приписать вес переменным – эти опции относятся к исходным данным. Вы также можете задать и опции, которые относятся непосредственно к статистической процедуре: задать правило обработки пропущенных данных, выбрать метод анализа по умолчанию и др. Мы выбрали опцию - **Расчет с расширенной точностью** и выбор метода по умолчанию. Выбор таких опций не является необходимым.

Вывод результатов и их анализ

В стартовой панели нажмите на кнопку **ОК**. Система произведет вычисления и через секунду окно результатов появится на вашем экране.

Окно результатов анализа имеет следующую простую структуру: верхняя часть окна – информационная, нижняя содержит функциональные кнопки, позволяющие всесторонне просмотреть результаты анализа.

Информационная часть

Рассмотрим вначале информационную часть окна. В ней содержится краткая информация о проведенном анализе, а именно:

- *Dep. Var* – Имя зависимой переменной. В нашем случае – *Цена_Лог*.
- *No. of Cases* – Число наблюдений, по которым построена регрессия. В примере число равно 14.
- *Multiple R* – Коэффициент множественной корреляции (Эта статистика полезна в множественной регрессии, когда вы хотите описать зависимости между переменными).
- *R-square (R2) – RI* – Квадрат коэффициента множественной корреляции, обычно называемый коэффициентом детерминации. Коэффициент детерминации является одной из основных статистик в данном окне, он показывает долю общего разброса (относительно выборочного среднего зависимой переменной), которая объясняется построенной регрессией.
- *Adjusted R-square: adjusted RI* – Скорректированный коэффициент детерминации, определяемый как:

$$\textbf{Adjusted R-square} = 1 - (1 - R^2) * (n / (n-p)),$$

где n – число наблюдений в модели, p – число параметров модели (число независимых переменных плюс 1 из-за свободного члена).

- *Std. Error of estimate* – Стандартные ошибки оценки. Эта статистика является мерой рассеяния наблюдаемых значений относительно регрессионной прямой.
- *Intercept* – Оценка свободного члена регрессии. Значение коэффициента b_0 в уравнении регрессии.
- *Std. Error* – Стандартная ошибка оценки свободного члена. Стандартная ошибка коэффициента b_0 в уравнении регрессии.
- *t(df) and p-value* – Значение t-критерия и уровень p. T-критерий используется для проверки гипотезы о равенстве нулю свободного члена регрессии.
- *F* – значения F-критерия
- *df* – число степеней свободы F-критерия
- *p* – уровень значимости

В информационной части прежде всего необходимо смотреть на значение коэффициента детерминации. В нашем примере RI = 0.929. Это значит, что построенная регрессия объясняет 92,9% разброса значений *Цена_Лог* относительно среднего. Это хороший результат.

Далее вы смотрите на значение F-критерия и уровень его значимости p. F-критерий используется для проверки значимости регрессии. В данном случае для проверки

гипотезы, утверждающей, что между зависимой переменной *Цена_Лог* и независимой переменной *Возраст* нет линейной зависимости, т.е. $b_1 = 0$, против альтернативы b_1 не равен 0. В данном примере большое значение F-критерия =157.0486 и даваемый в окне уровень значимости p = 0.0000 показывают, что построенная регрессия высоко значима.

Функциональные кнопки

Нажмите далее на кнопку **Regression Summary –** *Краткие результаты регрессии*. Вы увидите электронную таблицу с результатами анализа.

В третьем столбце таблицы вы видите оценки неизвестных параметров модели:

b_0=1.142891;

b_1=0.034652.

Рассмотрим вторую часть информационного окна. В этой части система сама говорит нам о значимых регрессионных коэффициентах, высвечивая строку ЦЕНА_ЛОГ beta =0,964. В данном случае **beta** есть стандартизированный коэффициент b_1, т. е. коэффициент при независимой переменной ВОЗРАСТ. Перейдем в функциональную часть окна результатов.

Прежде всего нажмем кнопку *Итоговый результат регрессии* **– Regression summary.** На экране появится электронная таблица вывода – **spredsheet**, в которой представлены итоговые результаты оценивания регрессионной модели.

Это стандартная таблица вывода регрессионного анализа. В первом столбце таблицы даны значения коэффициентов **beta** – стандартизованные коэффициенты регрессионного уравнения, во втором – стандартные ошибки **beta,** в третьем – точечные оценки параметров модели:

Свободный член b_0 =1.143891.

Коэффициент b_1 **=**0.034652.

Далее, стандартные ошибки для b_0, b_1, значения статистик t-критерия и т.д.

Итак, искомая модель зависимости логарифма цены от возраста имеет вид:

$$\text{Цена\_Лог} = 0.034652*\text{Возраст}+1.143891 \qquad (7)$$

Требуемая регрессия построена. После перехода к исходным переменным модель примет вид:

$$\text{Цена} = \exp(0.034652*\text{Возраст}+1.143891) \qquad (8)$$

График приведен на рис.2.2.

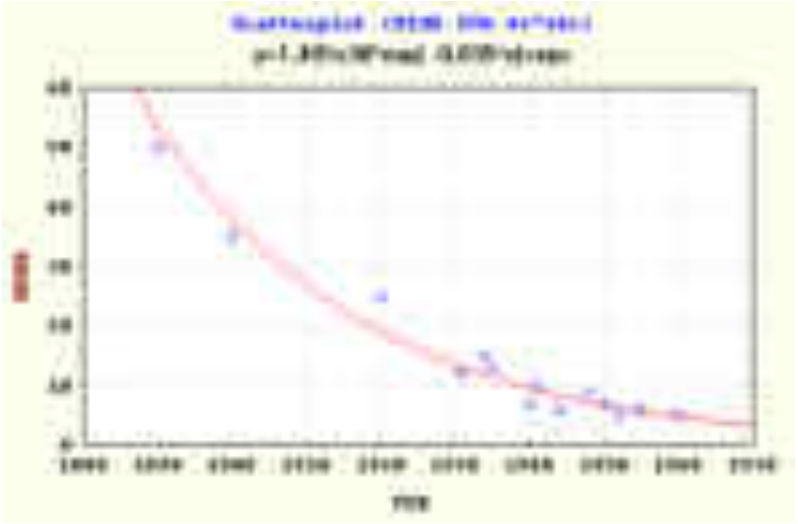


Рис 2.2. График зависимости цены от года закладки

Оценка адекватности модели. Исследование остатков.

После того как доказана адекватность модели, полученные результаты могут быть использованы по назначению. Анализ адекватности основывается на анализе остатков. Что такое остатки модели? Оценки представляют собой разности между наблюдаемыми значениями и модельными (предсказанными), т. е. значениями, подсчитанными по модели с оцененными параметрами (см. рис. 2.3.).

Рис 2.3. Значения остатков

Пересчитаем значение *Цена_Лог* исходя из построенной модели для различных значений независимой переменной *Возраст*. Эти значения и называются *Predicted values – Предсказанные значения* или модельные, т.е. значения, предсказанные с

помощью модели. *Очевидно,* эти значения будут отличаться от значений *Цена_Лог*, имеющихся в исходном файле vine.sta. Разность между исходными (наблюдаемыми) значениями зависимой переменной и предсказанными значениями называются остатками.

В модуле **Множественная регрессия** системы *STATISTICA* остатки исследуются в специальном окне **Анализ остатков**. Исследуя остатки, вы можете оценить степень адекватности модели. Для этого нажмите в окне **Результаты множественной регрессии** кнопку **Residual analysis – Анализ остатков**. Нажав данную кнопку, вы раскроете окно **Анализ остатков**. С помощью функциональных кнопок в данном окне можно всесторонне просмотреть остатки модели как в графическом виде, так и в электронных таблицах.

Вначале для оценки адекватности модели лучше всего использовать визуальные методы и затем, если потребуется, перейти к статистическим.

Для оценки адекватности модели рассмотрим график остатков, например, на нормальной вероятностной бумаге (**Normal plot of resids**) (рис 2.4).

Из графика остатков на нормальной вероятностной бумаге видно, что они достаточно хорошо ложатся на прямую, которая соответствует нормальному закону. Поэтому предположение о нормальном распределении ошибок выполнено.

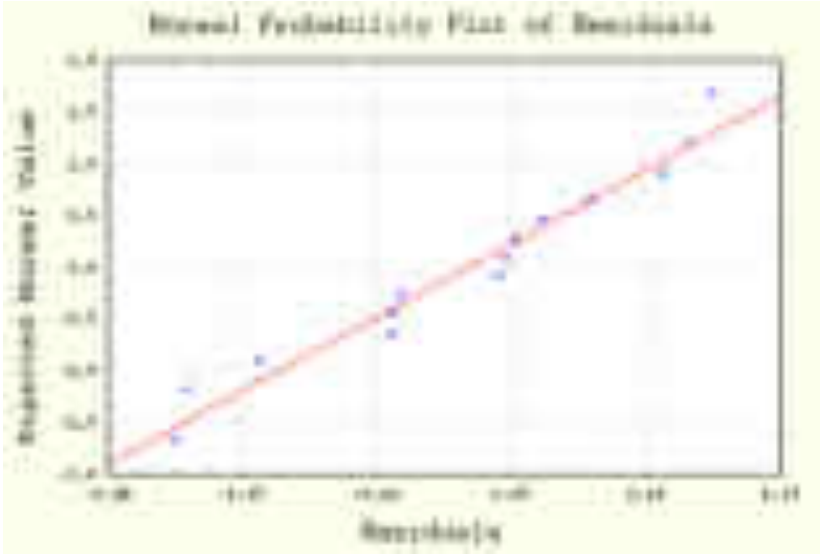


Рис 2.4. График остатков на нормальной вероятностной бумаге

Исследуем зависимость остатков от наблюдаемых значений (**Obs & residuals**), а также от предсказанных значений (**Pred & residuals**) (рис.2.5). Остатки хаотично разбросаны относительно прямой, в их поведении нет закономерностей. Нет оснований говорить, что остатки коррелированы между собой, нет также резко резко выделяющихся остатков. Отсюда можно заключить, что модель достаточно адекватно описывает данные.

Замечание. Следует заметить, что мы имеем очень небольшое число данных – всего 16. Поэтому мы используем графические методы оценки адекватности модели. В сложных задачах графические и статистические методы оценки адекватности должны использоваться совместно, естественно дополняя друг друга. Статистические процедуры анализа остатков собраны в левой части окна в группе *Statistics – Статистики*

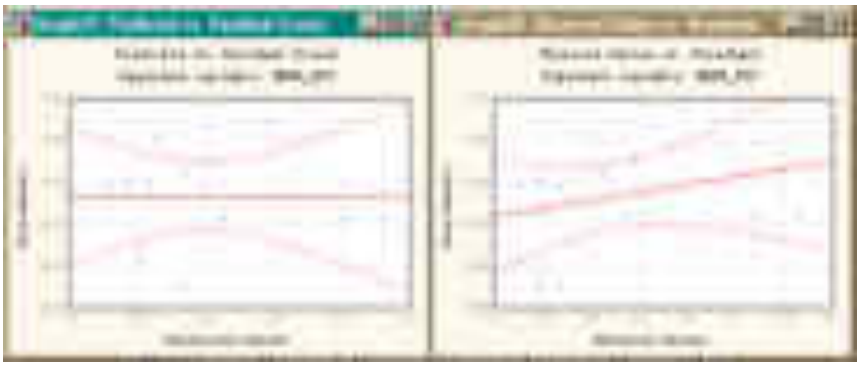


Рис 2.5 Зависимость остатков от предсказанных и наблюдаемых значений

Вывод результатов анализа в файл с отчетом

Возможен вывод результатов анализа не только в виде электронных таблиц и графиков на рабочее пространство системы. Можно также создать специальный файл с отчетом. Для него на рабочем пространстве системы откроется специальное окно, в которое можно «распечатать» любой документ, например таблицу или график.

Пример 2. Множественная регрессия

Рассмотрим теперь более сложный пример множественной регрессии. При этом в модель будет включена не одна, а несколько независимых переменных.

Описание примера

В таблице ниже приведены данные о капитальных затратах на строительство атомных электростанций с реактором водяного охлаждения.

Данные собраны для 32 различных станций США. Требуется: оценить зависимость между ценой станции и рядом параметров, приведенных в таблице, предсказать величину капитальных затрат на строительство новой станции, попробовать выделить наиболее значимые величины, влияющие на цену станции.

Данные имеют следующую структуру:

Таблица 4

№	C	D	T1	T2	S	PR	NE	CT	BW	N	PT
1	460.05	68.58	14	46	687	0	1	0	0	14	0

2	452.99	67.33	10	73	1065	0	0	1	0	1	0
3	443.22	67.33	10	85	1065	1	0	1	0	1	0
4	652.32	68.00	11	67	1065	0	1	1	0	12	0
5	642.23	68.00	11	78	1065	1	1	1	0	12	0
6	345.39	67.92	13	51	514	0	1	1	0	3	0
7	272.37	68.17	12	50	822	0	0	0	0	5	0
8	317.21	68.42	14	59	457	0	0	0	0	1	0
9	457.12	68.42	15	55	822	1	0	0	0	5	0
10	690.19	68.33	12	71	792	0	1	1	1	2	0
11	350.63	68.58	12	64	560	0	0	0	0	3	0
12	402.59	68.75	13	47	790	0	1	0	0	6	0
13	412.18	68.42	15	62	530	0	0	1	0	2	0
14	495.58	68.92	17	52	1050	0	0	0	0	7	0
15	394.36	68.92	13	65	850	0	0	0	1	16	0
16	423.32	68.42	11	67	778	0	0	0	0	3	0
17	712.27	69.50	18	60	845	0	1	0	0	17	0
18	289.66	68.42	15	76	530	1	0	1	0	2	0
19	881.24	69.17	15	67	1090	0	0	0	0	1	0
20	490.88	68.92	16	59	1050	1	0	0	0	8	0
21	567.79	68.75	11	70	913	0	0	1	1	15	0
22	665.99	70.92	22	57	828	1	1	0	0	20	0
23	621.45	69.67	16	59	786	0	0	1	0	18	0
24	608.80	70.08	19	58	821	1	0	0	0	3	0
25	473.63	70.42	19	44	538	0	0	1	0	19	0

26	697.14	71.08	20	57	1130	0	0	1	0	21	0
27	207.51	67.25	13	63	745	0	0	0	0	8	1
28	288.48	67.17	9	48	821	0	0	1	0	7	1
29	284.88	67.83	12	63	886	0	0	0	1	11	1
30	280.36	67.83	12	71	886	1	0	0	1	11	1
31	217.38	67.25	13	72	745	1	0	0	0	8	1
32	270.71	67.83	7	80	886	1	0	0	1	11	1

Здесь:

C – цена в млн долларах, приведенная к курсу 1976 г.

D – срок разрешения на строительство.

T1 – время между обращением за разрешением и получением разрешения на строительство.

T2 – время между получением оперативной лицензии и разрешением на строительство.

S – номинальная мощность электростанции, МВт.

PR – наличие в той же самой местности ранее построенной электростанции на PBO. Если значение равно 1, то имеется уже построенная станция.

NE – характеристика района, в котором строится станция.

CT – использование нагревательной башни. Если равно 1, то используется, если 0 – нет.

BW – использование силовой установки производства фирмы Babcock-Wilcox. Если значение равно 1, то используется установка этой фирмы, 0 – нет.

N – суммарное число электростанций, построенное архитектором-инженером станции.

PT – электростанция, строящаяся под частичным надзором. **PT=1**, если надзор есть, **PT=0**, если надзора нет.

Математическая постановка задачи

Для исследования данной задачи воспользуемся методами регрессионного анализа. В отличие от предыдущего, в этом примере имеется несколько независимых переменных, поэтому применяется метод множественной регрессии.

Воспользуемся векторными обозначениями. Обозначим через y – вектор наблюдений, состоящий из n элементов, через x – матрицу независимых переменных, размером m на n, где m – число независимых переменных, а n – число наблюдений.

В этих обозначениях задача может быть сформулирована следующим образом:

$$y = x\beta + \varepsilon , \qquad (9)$$

где ε есть независимые случайные ошибки со средним **0**, которые интерпретируются как ошибки наблюдений, а β - вектор неизвестных параметров, которые необходимо оценить. Оценки параметров β обозначим через **b**.

В данном примере зависимая переменная – цена станции, а независимые – D, T1, T2, S, PR, NE, CT, BW, N, PR (т.е. все остальные переменные, перечисленные в таблице). Зависимость между переменными предполагается линейной.

Создание электронной таблицы с исходными данными

Ввод исходных данных. Текстовые и численные значения

Переменные в электронной таблице могут принимать как текстовые, так и численные значения. Текстовые значения вводятся аналогично численным. Необходимо поместить указатель на ячейку в таблице, щелкнуть левой кнопкой мыши и ввести требуемое значение с клавиатуры. Для переменных, которые принимают текстовые значения, в STATISTICA используется так называемое соглашение «двойной записи», при котором каждому текстовому значению приписывается некоторый численный эквивалент. Для просмотра переменных, принимающих текстовые значения, переключитесь в режим просмотра текстовых значений при помощи кнопки ABC на панели инструментов электронной таблицы. Например, для повышения наглядности восприятия можно ввести для переменных PR и BW текстовые значения. Для переменной PR – ДА (1) будет обозначать наличие уже построенной в этой местности станции на PBO, а НЕТ (0) – ее отсутствие. Аналогично и для переменной BW введены текстовые значения ИСП и НЕИСП для обозначения использования установок фирмы BW или нет. Для просмотра этих значений нажмите на кнопку ABC123 **Менеджера текстовых значений.**

Преобразование исходных данных

В электронных таблицах STATISTICA вы можете выполнить все необходимые преобразования. Такая задача часто возникает в процессе обработки данных. В систему STATISTICA включено большое количество общих математических и специализированных статистических функций. Для некоторых из переменных мы применим, аналогично предыдущему примеру, преобразование логарифмирования. Формулы преобразования задаются в диалоговом окне спецификаций переменной, которое вызывается двойным щелчком на имени переменной в строке заголовка

электронной таблицы. Вам, возможно, потребуется вставить дополнительные строки и столбцы в таблицу. Воспользуйтесь для этого кнопками **Vars Cases** для вызова соответствующих команд по работе с переменными и наблюдениями.

При помощи кнопки вы можете просмотреть спецификации всех переменных в электронной таблице с исходными данными.

Поставим задачу построения линейной регрессии между зависимой переменной Log_C = Ln(C) и независимыми переменными D, PR, NE, CT, BW, PT, Log_N, Log_S, Log_T1, Log_T2.

Предварительный анализ и визуализация данных

Построим ряд специализированных статистических графиков для более полного исследования исходных данных. Для этого поместите указатель мыши на ту переменную в таблице, которую необходимо отобразить графически, щелкните правую кнопку мыши и из появившегося контекстного меню выберите необходимый тип графика. В этом случае в диалоговом окне определения графика при помощи кнопки Variables – Переменные выберите необходимые переменные, которые вы хотите отобразить графически, и необходимый тип графика. Постройте гистограмму для переменной T1 и график диапазона значений для C.

Вызов стартовой панели модуля и определение процедуры анализа

Для начала статистического анализа вам необходимо вызвать *Стартовую панель* модуля. Это основное диалоговое окно модуля, в котором необходимо задать различные опции анализа. Если *Стартовая панель* модуля закрыта, то откройте ее. Для этого зайдите в модуль **Analysis** – *Анализ* и выберите команду **Startup Panel** – *Стартовая панель*.

Выбор переменных для анализа

Далее необходимо выбрать переменные для анализа. В нашем примере имеется одна зависимая переменная Log_C и набор независимых переменных. Для их задания воспользуйтесь кнопкой **Variables** – *Переменные* из *Стартовой панели*.

В открывшемся окне **Select dependent and independent variable list – Выбор зависимой переменной и списка независимых переменных** выберите необходимые переменные. Для выбора переменной щелкните мышью на ее имени. Для выбора нескольких переменных удерживайте при этом клавишу CTRL. Нажмите кнопку **ОК** в правом верхнем углу. Вы вновь окажетесь в *Стартовой панели* модуля **Множественная регрессия**.

Задание дополнительных параметров анализа

В *Стартовой панели* можно задать и дополнительные опции и параметры анализа. Например, вы можете выбрать определенное подмножество наблюдений для анализа, приписать веса переменным – эти опции относятся к исходным данным. Вы также можете задать и опции, которые относятся непосредственно к

статистической процедуре: задать правило обработки пропущенных данных, выбрать метод анализа по умолчанию и др. Отменим выбор метода анализа по умолчанию (**Perform default analysis**). После нажатия на кнопку **ОК** появится диалоговое окно определения метода (рис 2.6.).

В прокручиваемом списке методов выберите один из пошаговых регрессионных методов, например, **Forward stepwise**, значения остальных параметров оставьте неизменными. Нажмите **ОК**.

Замечание

Метод пошаговой регрессии состоит в том, что на каждом шаге в модель включается, либо исключается какая-то независимая переменная. Таким образом, выделяется множество наиболее “значимых” переменных. Это позволяет сократить число переменных, которые описывают зависимость.

В данном случае выбран пошаговый метод **включения**. При использовании этого метода в регрессионное уравнение последовательно включаются независимые переменные, пока уравнение не станет удовлетворительно описывать входные данные. Включение переменных определяется при помощи **F**-критерия.

Рис 2.6. Окно результатов анализа. Отмечены переменные, которые были включены в модель

Рис 2.7 Окно с результатами анализа. Красным цветом выделены значимые коэффициенты регрессии

Вывод результатов и их анализ

В стартовой панели нажмите на кнопку **ОК**. Система произведет вычисления и на экране появится окно результатов (рис.2.6.)

Нажав на кнопку **ОК**, вы откроете основное окно анализа результатов (рис.2.7.).

Окно результатов анализа имеет следующую простую структуру: верхняя часть окна – информационная, нижняя содержит функциональные кнопки, позволяющие всесторонне просмотреть результаты анализа. Оно описано в предыдущем примере.

В информационной части вы прежде всего смотрите на значение коэффициента детерминации. В нашем примере RI = 0.857. Это значит, что построенная регрессия объясняет 85.7% разброса значений относительно среднего.

Далее вы смотрите на значение F-критерия и уровень его значимости p. F-критерий используется для проверки значимости регрессии.

Щелкните далее на кнопку **Regression summary – Краткие результаты регрессии**. Вы увидите следующую электронную таблицу с результатами анализа:

Рис 2.8 Краткие результаты регрессии

В третьем столбце расположены искомые коэффициенты. Итак, искомая регрессия имеет вид:

Log_C = -13.2603 + 0.2261*PT + 0.7234*Log_S + 0.2124*D + 0.249*NE + 0.1404*CT – 0.0876*Log_N

Качественно построенное уравнение можно интерпретировать следующим образом:

- Стоимость строительства растет с увеличением мощности станции (S), при использовании нагревательной башни и при строительстве в NE районе.
- Стоимость уменьшается с возрастанием опыта инженера-архитектора и при строительстве под частичным надзором.

Итак, на рассмотренных примерах мы проследили технологию обработки данных и стиль работы в *STATISTICA* и увидели, что даже несложные модели линейной регрессии позволяют в реальных задачах получать содержательные результаты.

Остатки

Ниже приведены графики остатков данной модели:

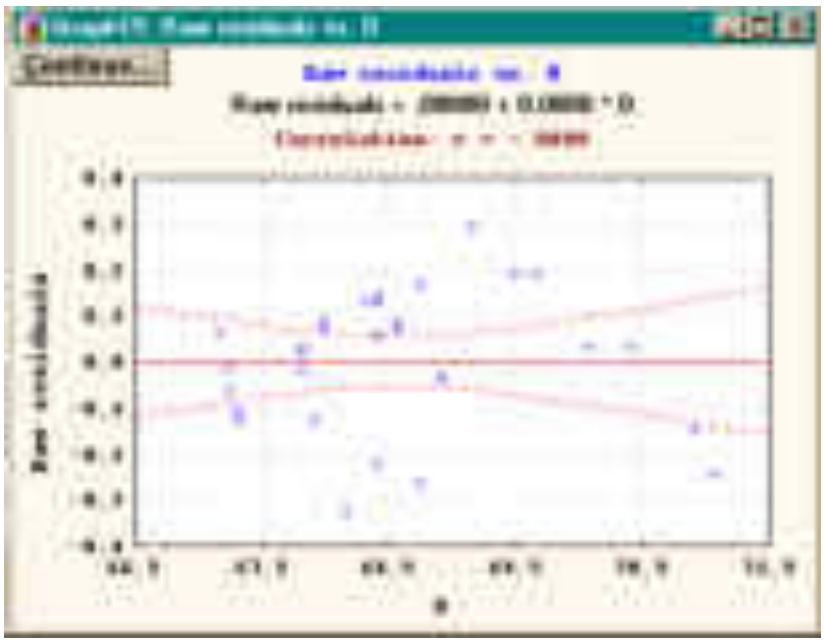


Рис. 2.9. Остатки как функция от D

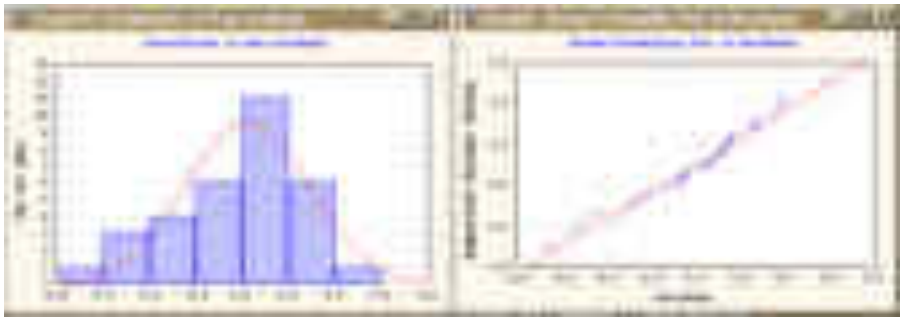


Рис 2.10 Гистограмма остатков

Рис 2.11 График остатков на НВБ

Б. Исследование статистической зависимости Вес-Рост

Постановка задачи

В первой лабораторной работе вами были собраны статистические данные измерения веса и роста студентов соответствующих групп. Результаты измерений должны быть сведены в одну общую таблицу. Найти уравнение регрессии, описывающей вес студента в зависимости от его роста. Статистические выводы обосновать.

Последовательность решения задачи

1. Подготовить таблицу.
2. Оценить корректность условий проведения регрессионного анализа
3. Провести предварительный анализ и определить предполагаемый вид функции. Выяснить, нужны ли дополнительные преобразования.

4. Провести регрессионный анализ. Результаты анализа позволят ответить на следующие вопросы:
    - Какова степень статистической связи между двумя переменными?
    - Адекватна ли модель исходным данным?
    - Значимы ли коэффициенты модели?
5. Исследовать остатки.

Лабораторная работа 3. Факторный анализ

Теоретическое введение

Цель факторного анализа - выделение из выборки величин, так называемых факторов, количество которых значительно меньше участвующих в анализе переменных. Эти факторы с достаточной точностью воспроизводят наблюдаемые корреляции между переменными и не доступны для непосредственного измерения. Не всегда эти факторы имеют практический смысл, так как в основу факторного анализа положены алгебраические методы преобразования корреляционных матриц. В психологических исследованиях методами факторного анализа объясняют взаимозависимость психологических явлений, например характер, способности, потребности и успеваемость учащихся.

Пусть R - корреляционная матрица, диагональные элементы этой матрицы равны 1. Для n=4, где n - количество переменных, корреляционная матрица имеет вид

На первом этапе факторного анализа корреляционная матрица R преобразуется в редуцированную матрицу R_h , главное отличие которой от исходной корреляционной матрицы заключается в том, что ее диагональные элементы не равны 1 и называются **оценками общности**. Преобразование корреляционной матрицы R в Rh составляет проблему общности.

На втором этапе факторного анализа путем алгебраического преобразования матрицы R_h осуществляется выделение m факторов, m<<n. Это преобразование происходит при минимальной потере информации. Если обозначить матрицу факторов как **A**, а через $\mathbf{A}^T$ ее транспонированную матрицу, в которой изменены столбцы на строки и строки на столбцы, то проблему **выделения факторов** можно представить в виде

$$\boldsymbol{R = A\,A^T}$$

или

В приведенном выражении показана матрица **A** с выделенным одним фактором. Аналогично представляются матрицы и с несколькими факторами. Значения a_1,a_2, . . . a_n - называются факторными нагрузками. Факторные нагрузки определяют в какой степени выделенный фактор описывается каждой переменной и имеют смысл коэффициента корреляции переменной и фактора.

Очень редко полученные на этом этапе результаты могут найти практическое применение в силу того, что нет четкой связи выделенного фактора с переменными. Такая связь считается удовлетворительной, если факторные нагрузки по абсолютному значению превышают 0.7-0.8. В связи с этим возникает третья проблема факторного анализа - **проблема вращения**.

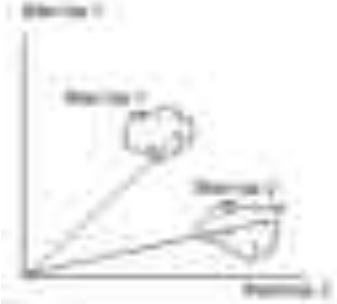


Рис.3.1.

После выделения факторов их взаимное положение в пространстве переменных можно представить геометрически в следующем виде (рис.3.1)

Рис.3.2.

После выполнения вращения положение факторов в пространстве изменяется. Вместе с этим изменяются и факторные нагрузки (рис.3.2).

Для вращения используются различные методы, в том числе метод главных компонент, варимаксный метод и другие. В основе этих методов положены различные способы выбора системы координат, приводящие к расположению факторов в ортогональных (взаимно перпендикулярных) плоскостях.

Наконец, последняя проблема применения факторного анализа заключается в оценке значений факторов и их содержательной интерпретации.

А. Пример использования модуля «Факторный анализ»

В файле factor.sta собраны результаты опросов 100 взрослых людей относительно степени их удовлетворенности жизнью. Это типичный пример данных, возникающих при социологических опросах. В файле даны значения следующих 10 переменных:

1. *Work 1* – удовлетворенность работой, – первая компонента,
2. *Work 2* – удовлетворенность работой, – вторая компонента,
3. *Work 3* – удовлетворенность работой, – третья компонента,
4. *Hobby 1* – удовлетворенность свободным временем, –первая компонента,
5. *Hobby 2* – удовлетворенность свободным временем, – вторая компонента,
6. *Home 1* – удовлетворенность домашней жизнью, – первая компонента,
7. *Home 2* – удовлетворенность домашней жизнью, – вторая компонента,
8. *Home 3* – удовлетворенность домашней жизнью, – третья компонента,
9. *Miscel 1* – общая удовлетворенность, – первая компонента,
10. *Miscel 2* – общая удовлетворенность, – вторая компонента.

Многомерность каждой переменной объясняется тем, что рассматриваются различные аспекты удовлетворенности, например, вы можете быть удовлетворены зарплатой, получаемой на работе, но не удовлетворены коллективом или тем, сколько времени тратите, чтобы до нее добраться, и т.д. Фактически здесь имеется 4 многомерных переменных.

Щелкните в этом окне на кнопку **Shrink** — *Сократить*. Далее щелкните на кнопку **Ok** — Да и вернитесь в стартовое окно модуля.

Файл открыт и переменные для анализа выбраны. Теперь можно начать анализ по выявлению главных факторов, влияющих на удовлетворенность человека жизнью.

Начало анализа

Щелкнув в стартовом окне модуля на кнопку Ok — Да, вы начнете анализ выбранных переменных.

STATISTICA обработает пропущенные значения тем способом, какой вы ей указали, вычислит корреляционную матрицу и предложит на выбор несколько методов факторного анализа.

Вычисление корреляционной матрицы, если она не задается сразу, — первый этап факторного анализа.

Итак, пусть вы имеете исходный данные, как в файле factor.sta.

После щелчка на кнопку **Ok** — *Да* перед вами появится окно **Define Method of Factor Extraction** — *Определить метод выделения факторов*. Данное окно имеет следующую структуру.

Первая, верхняя часть окна является информационной: здесь сообщается, что пропущенные значения обработаны методом **casewise**. Обработано 100 случаев и 100 случаев принято для дальнейших вычислений. Корреляционная матрица вычислена для 10 переменных.

Вторая, нижняя часть диалогового окна **Define Method of Factor Extraction** — *Определить метод выделения факторов* содержит 4 функциональные кнопки и опции выбора метода — левая часть окна, — а также поля, в которых проводятся установки для итеративного вычисления общностей.

Обратите внимание на поля в правой части окна:

- **Maximum no. of Factors** — *Максимальное число факторов*;
- **Minimum eigenvalue** — *Минимальное собственное значение*.

Эти поля определяют число факторов, которые будут выделены системой. Собственные значения меньше указанного в поле игнорируются.

Рассмотрим функциональные клавиши окна.

Из 4 функциональных кнопок — две стандартные: кнопка **Ok** — *Да* нажимается для запуска вычислительной процедуры, когда выбран метод и проведены установки в полях. Кнопка **Cancel** — *Отменить* прекращает работу в окне и возвращает в стартовое окно модуля.

Инициировав кнопку **Review corrs/means/SD** — *Просмотреть корреляции/средние/стандартные отклонения*, вы откроете окно *Просмотреть описательные статистики*.

С помощью кнопки **Perform multiple regression** — *Выполнить множественную регрессию* вы можете выполнить множественную регрессию, не выходя из модуля.

Группа опций, объединенных под заголовком **Extraction Method** — *Метод выделения*, позволяет выбрать метод обработки.

В зависимости от критерия оптимальности возможен анализ либо методом **Principal components** — *Методом главных компонент*, либо одним из методов, объединенных в группу **Principal factor analysis** — *Анализ главных (общих) факторов*.

Система предлагает следующие методы в группе **Principal factor analysis** — *Анализ главных факторов*:

Communalities=multiple R2** — *Общности равны квадрату коэффициента множественной корреляции*.

Iterated communalities (MINRES) — *Итеративных общностей* (минимальных остатков).

Centroid method — *Центроидный метод*.

Principal axis method — *Метод главных осей*.

В окне **Define Method of Factor Extraction** — *Определить метод выделения факторов* щелкните мышью по кнопке **Review corrs/means/SD** — *Просмотреть корреляции/средние /стандартные отклонения*.

Перед вами появится окно просмотра описательных статистик для анализируемых данных, где можно посмотреть средние, стандартные отклонения, корреляции, ковариации, построить различные графики.

Щелкните по кнопке **Correlations** — *Корреляции*.

Вы увидите на экране корреляционную матрицу выбранных ранее переменных.

Вы можете сохранить корреляционную матрицу, инициировав кнопку **Save correlations** — *Сохранить корреляции*. Далее в модуле **Факторный анализ** при дальнейших исследованиях этих данных можно сразу работать с корреляционной матрицей.

Методическое замечание: часто на главной диагонали корреляционной матрицы вместо единиц ставятся квадраты коэффициентов множественной корреляции или

другие оценки общностей. По определению квадрат коэффициента множественной корреляции случайной величины X со случайными величинами Y(1),…,Y(l) равен

$$\min E(X - (a(1)*X(1) + \ldots + a(l)*X(l))**2,$$

где минимум берется по всевозможным наборам значений a(1),…,a(l).

Щелкните на кнопку **Continue** — *Продолжить* в верхнем левом углу таблицы и вернитесь в окно **Define Method of Factor Extraction** — *Определить метод выделения факторов*.

Выберите опцию **Principal components** — *Главные компоненты* и щелкните на **Ok** — *Да*.

Система быстро произведет вычисления, и на экране появится окно **Factor Analysis Results** — *Результаты факторного анализа*.

Структура окна «Результаты факторного анализа»

В верхней части окна окно **Factor Analysis Results** — *Результаты факторного анализа* дается информационное сообщение:

Number of variables — *Число анализируемых переменных* 10,

Method — *Метод анализа: главные компоненты*,

log(10) determination of correlation matrix – *Десятичный логарифм детерминанта корреляционной матрицы*: -4.1096,

Number of factor extraction — *Число выделенных факторов*: 2,

Eigenvalues — *Собственные значения*: 6.11837, 1.80068.

В нижней части окна находятся функциональные кнопки, позволяющие всесторонне просмотреть результаты анализа численно и графически.

Обратите внимание на кнопку **Factor rotation** — *Вращение факторов*.

Щелкнув по данной кнопке, вы откроете окно **Factor Rotation** — *Вращение факторов*.

В окне **Factor Rotation** вы можете выбрать различные повороты осей.

Если пространство общих факторов найдено, то с помощью поворота системы координат в принципе можно получить бесчисленное множество решений.

Важно найти интерпретируемое решение!

Вы можете искать нужный поворот эмпирически, однако STATISTICA предлагает несколько полезных процедур. Попробуйте оценить их возможности.

Возможны следующие методы:

- **Varimax** — *варимакс*
- **Biquartimax** — *биквартимакс*

- **Quartimax** — *квартимакс*
- **Equamax** — *эквимакс*

Дополнительный термин в названии методов — normalized – *нормализованные* — указывает на то, что факторные нагрузки в процедуре нормализуются, т.е. делятся на корень квадратный из соответствующей общности. Термин **raw** — *исходные* показывает, что вращаемые нагрузки не нормализованы.

Например, щелкнув по кнопке **Varimax normalized** — *Варимакс нормализованный*, вы проведете вращение факторных нагрузок методом варимакс, тогда как опция **Varimax Raw** — *Варимакс* исходных означает применение метода *варимакс* к ненормализованным (исходным) факторам.

Щелкните на кнопку **Plot of loadings, 2D** — *Двумерный график нагрузок* и посмотрите результаты факторного анализа на графике (рис 3.3.).

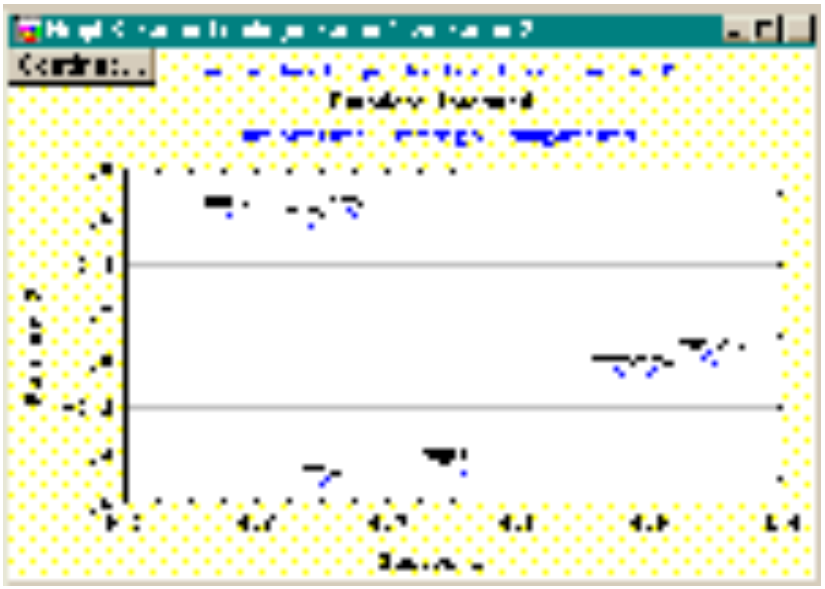


Рис. 3.3. Факторное решение для данных «Удовлетворенность жизнью»

Щелкнув по кнопке **Factor loadings** — Факторные нагрузки, вы сможете посмотреть нагрузки численно:

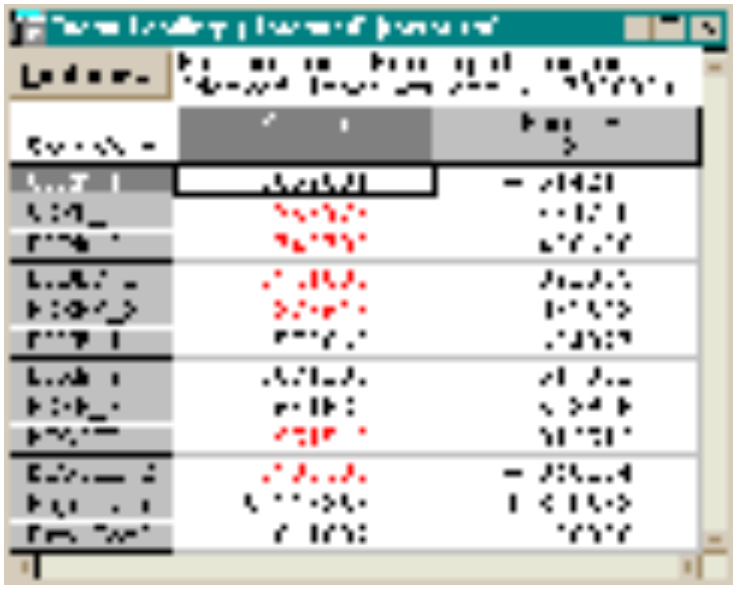


Рис.3.4. Таблица факторных нагрузок

Рассматривая решение на графике, вы видите, что его трудно проинтерпретировать. Не понятно, какой смысл придать двум выделенным факторам и как в этих терминах описывать удовлетворенность жизнью индивидуума.

В таких случаях целесообразно прибегнуть к повороту осей, надеясь получить решение, которое можно проинтерпретировать в предметной области.

Щелкните кнопку **Factor rotation** — *Вращение факторов*.

В появившемся окне Factor Rotation — Вращение факторов инициируйте кнопку **Varimax normalized** — *Варимакс нормализованный.*

Система произведет вращение факторов методом нормализованного варимакса, и окно **Factor Analysis Results** — *Результаты факторного анализа* снова появится на мониторе. Вновь инициируйте в этом окне кнопку **Plot of loadings, 2D** — *Двумерный график нагрузок*. Вы опять увидите двумерный график нагрузок:

Этот график отличается от предыдущего. Теперь найденное решение уже можно интерпретировать. Посмотрим еще нагрузки численно, инициировав кнопку **Factor loadings** — Факторные нагрузки.

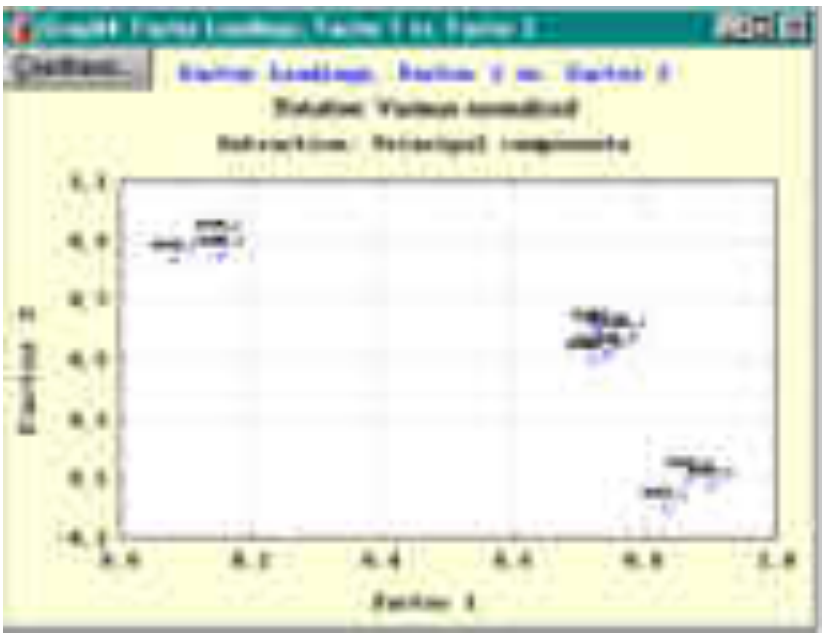


Рис. 3.5. Факторное решение после поворота осей

На этом графике выделено два общих фактора Factor 1, Factor 2: Factor 1 отвечает за удовлетворение, получаемое опрошенными на работе, удовлетворенность работой (значения переменных Work 1 — Work 3 максимально большие по этой оси и малы по другой). Относительно Factor 2 можно сказать, что он измеряет удовлетворенность домашней жизнью.

Глядя на эти результаты, вы можете сделать вывод, что общая удовлетворенность человека определяется лишь двумя факторами: удовлетворенностью работой и удовлетворенностью домом.

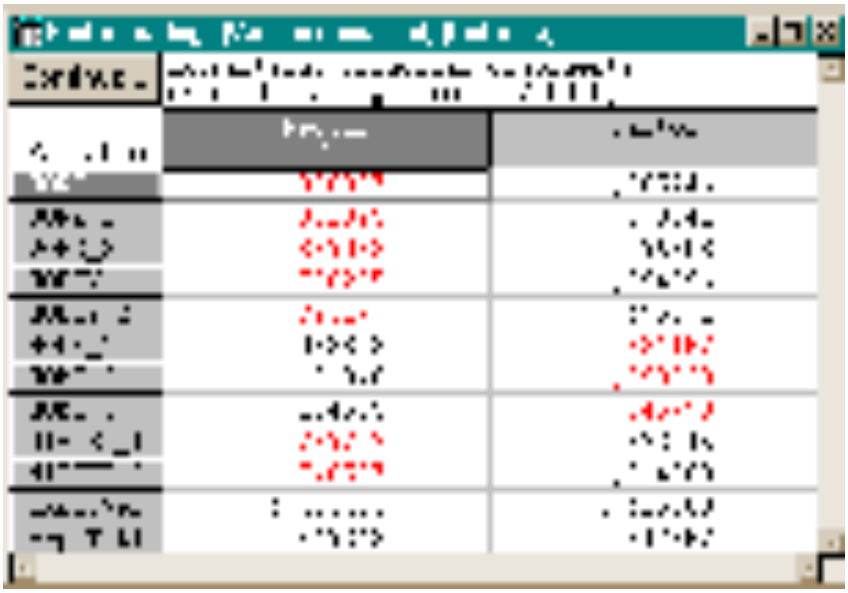


Рис.3.6. Таблица факторных нагрузок после применения метода главных компонент и вращения факторов

Методические замечания

1. Выбор числа факторов. Не существует общего решения о выборе числа факторов, однако существует несколько полезных процедур, которыми пользуются на практике.

Критерий Кайзера. Сначала мы можем оставить только факторы, соответствующие собственным значениям, большим 1. Это означает, что мы оставляем факторы, которые дают дисперсию не меньше, чем исходные переменные. Используя этот критерий, в нашем примере мы должны сохранить 2 фактора (главные компоненты).

Критерий каменистой осыпи. Графический метод, называемый критерием каменистой осыпи (the scree test). Мы можем показать собственные значения на простом линейном графике. Предлагается найти точку на графике, справа от которой собственные значения уменьшаются гладко. Справа от этой точки возможно найти только "факторную каменистую осыпь". Согласно этому критерию, мы должны сохранить 2 или 3 фактора в нашем примере.

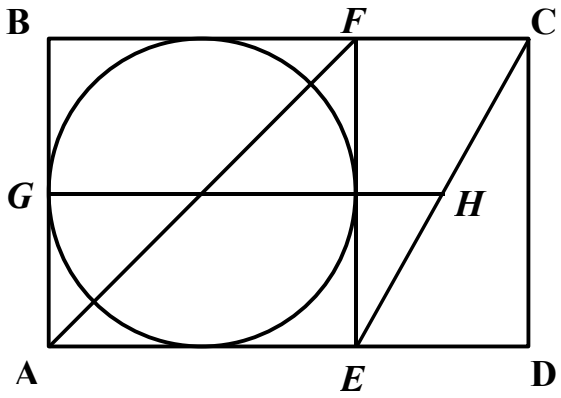
Первый метод (критерий Кайзера) иногда сохраняет слишком многие факторы, а вторая техника (критерий каменистой осыпи) — слишком мало.

2. Критерий согласия для необходимого числа факторов. Если оценки нагрузок строятся методом максимального правдоподобия, то имеется возможность использовать статистический критерий проверки гипотезы о числе факторов в модели. Критерий доступен по клавише **Goodness of Fit Test –** *Критерий согласия* в окне **Factor Analysis Results** — *Результаты факторного анализа. Э*та клавиша доступна лишь в тех случаях, когда решение найдено методом максимального правдоподобия.

Б. Задание для самостоятельного анализа

Описание задачи

Пусть объектом исследования будет обычный прямоугольник, описываемый восьмью признаками. Скрытыми факторами будут являться высота и длина прямоугольника.



Признаки несут следующую смысловую нагрузку:

x_1= периметр ABCD

x_2= длина окружности

x_3= периметр AFE

x_4= отрезок ED

x_5= периметр EFCD

x_6= диагональ AF

x_7= периметр ABFE

x_8= средняя линия трапеции ABCE

Таким образом, каждый из признаков, входящий в исследуемый набор, может быть представлен как функция двух общих факторов и специфического фактора:

1. $X_1=2F1+2F2+U_1$
2. $X_2=\pi F1+U_2$
3. $X_3=(2+\sqrt{2})F1+U_3$
4. $X_4=F1-F2+U_4$
5. $X_5=2F2+U_5$
6. $X_6=\sqrt{2}F1+U_6$
7. $X_7=4F1+U_7$
8. $X_8=0.5F1+0.5F2+U_8$

Априорно известно, что признаки распределены по нормальному закону. Требуется провести факторный анализ и определить скрытые определяющие факторы. Создайте таблицу данных из 8 переменных и 100 объектов, используя специальную функцию генерации случайных нормально распределенных чисел. Проведите факторный анализ.

Порядок проведения анализа

1. Расчет корреляционной матрицы
2. Выделение первоначальных ортогональных факторов
3. Вращение факторов
4. Оценка значений факторов

Со держание отчета:

1. Корреляционная матрица
2. Результаты факторного анализа
3. Таблица собственных значений
4. Таблица общностей
5. Остаточная матрица корреляций
6. Таблица факторных нагрузок до вращения
7. График факторных нагрузок до вращения
8. Таблица факторных нагрузок после вращения
9. График факторных нагрузок после вращения
10. Оценки общих факторов

Проведите аналогичный анализ, включив факторы «высота» и «длина» прямоугольника в список общих переменных.

Лабораторная работа 4. Кластерный анализ

Анализ данных, основанный на выделении кластеров. Под кластером понимают группу переменных в статистическом анализе с общими признаками. В кластерном анализе используются следующие типы классификации:

1. ***Исключающие (неисключающие).*** В исключающей классификации один элемент может появляться в только в одном подмножестве, а в неисключающих – в нескольких;

1. ***Внутренние (внешние).*** В первом случае классификация основывается на заданном наборе переменных. При внешней классификации используются другие переменные, которые объединяют несколько переменных ;

2. ***Иерархические (неирархические).*** В иерархической классификации группы выбираются таким образом, чтобы каждая была возможно более однородной. Неирархические методы классификации, как правило, не используются;

3. ***Агломеративные (дивизивные).*** Различия между этими классификациями состоит в направлении объединения подмножеств. В агломеративной классификации переменные объединяются в подмножества возрастающего объема. В дивизивной классификации переменные постепенно разделяется до тех пор, пока не будет получена заданная степень разделения;
4. ***Монотетические (политетические)*** . В монотетической классификации деление производится на основании одного признака, имеющего максимальную информативность, тогда как в политетической классификации все признаки учитываются в равной степени.

В пакете STATISTICA реализованы три метода кластерного анализа:

- ***Дерево кластеров (Tree Clustering).*** Метод основан на иерархической классификации. В психологических исследованиях этот метод применяется наиболее часто. В этом методе реализуется 7 различных правил объединения (по выбору):
    - Простое связывание переменных (Single Linkage);
    - Полное связывание (Complete Linkage);
    - Невзвешенное парно-групповое среднее (Unweighted Pair-group Average);
    - Взвешенное парно-групповое среднее (Weighted Pair-group Average);
    - и др. правила.

      По умолчанию используется простое связывание переменных.

В качестве меры сходства используется Евклидова метрика (Euclidean Distances), Квадрат Евклидовой метрики (Squared Euclidean Distances), Манхеттенская метрика (Manhatten Distances) и др.

2. ***Алгоритм К внутри групповых средних (K - means Clustering Results)*** . Метод основан на выявлении кластера путем определения минимума суммы квадратов расстояний всех точек , входящих в кластерную область, до центра кластера. Качество алгоритма зависит от выбора исходных центров кластеров и их числа.
3. ***Блочная кластеризация (Block Clustering).*** Метод основан на моделировании кластеров переменных и вариант (Case). Использует эвристический параметр Treshold.

Постановка задачи

Сгенерировать выборку случайных величин, образующих несколько кластеров, по следующим законам:

1. VAR1 = RND(7), где функция RND(7) представляет собой псевдослучайное число распределенное равномерно в интервале [0,7]. Эти числа получаются по специальному алгоритму, производящему некоторые простые арифметические действия над исходным числом. При каждом последующем обращении базовое число изменяется. Это приводит к появлению псевдослучайной последовательности чисел.

1. VAR2 = V1+ RND(3), где V1 - сокращенное название переменной VAR1, а RND(3) - случайное число, распределенное равномерно в интервале [0,3]. Это предполагает, что между переменной V1 и V2 существует статистическая связь, которая должна проявиться как кластерное образование.

2. VAR3 = RND(7). Хотя переменные VAR1 и VAR3 генерируются по одним и тем же законам и имеют одинаковые статистики (выборочное среднее, выборочную дисперсию и т.д.) тем не менее в каждой конкретной реализации (Case) они существенно различаются.

3. VAR4 = V3+RND(3)-RND(3). Переменная V4 также статистически связана с переменной V3, но по другому закону.

4. VAR5 = V1+RND(3). Эта переменная статистически зависима от переменной V1.

Таким образом, в результате кластерного анализа должно проявиться по крайней мере 2 кластерных образования, хотя по сгенерированной выборке их обнаружить невозможно.

Последовательность решения задачи

1. Запустить пакет STATISTICA.

1. В появившемся окне статистического анализа выбрать Clustering Method и нажать клавишу Switch To (Переключить).

2. В появившемся верхнем горизонтальном меню выбрать раздел File, а затем New Data.. В результате этих действий должна появиться электронная таблица с стандартным обозначением переменных VAR1, VAR2, . . . , VAR10 и 10 строками.

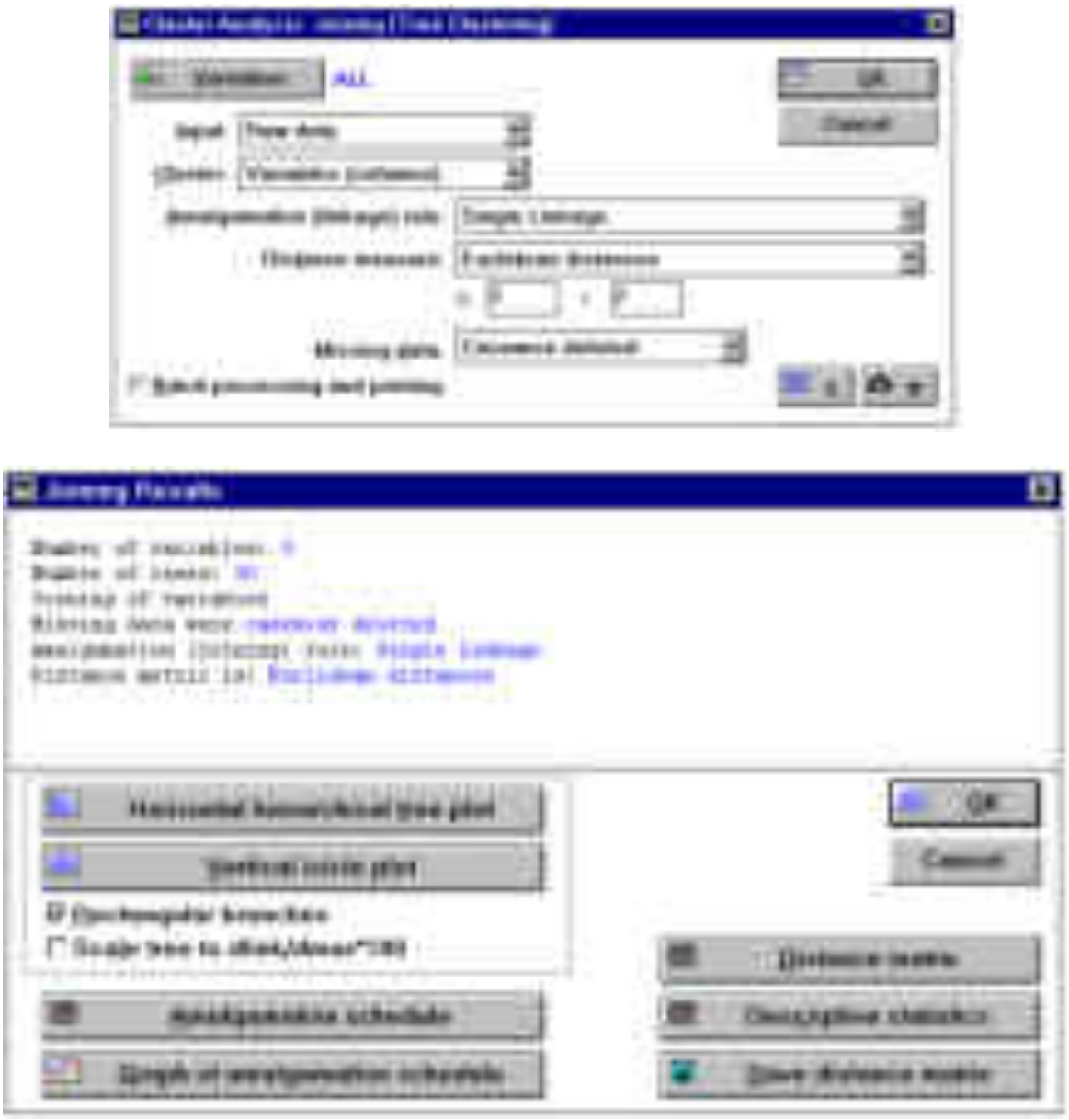
3. Преобразовать таблицу. С помощью мышки выделить переменные VAR6-VAR10 и удалить их выбрав в меню VAR функцию DELETE. Нажатием на кнопку Cases выбрать функцию ADD (Добавить строки таблицы). Затем в появившемся окне заполнить необходимые данные о количестве добавляемых строк и после какой строки это выполнить (90, 10).

1. Ввести формулы для моделирования значений переменных по исходному заданию. Для этого выделить с использованием мышки переменную и нажать комбинацию клавиш Ctrl+F2. В появившемся окне (рис. 33) установить поле Decimal равным 0, а в поле Long Name записать формулу для моделирования переменной по заданным исходным данным. Точно также повторить для других переменных. После нажатия клавиши Ok дать подтверждение на расчет значений переменной в электронной таблице.

1. Раскрыть панель для выбора метода анализа, выбрав в горизонтальном меню раздел Analysis (рис.34). Выбрать метод Tree Clustering и нажать Ok.

1. Ввести необходимые исходные данные для проведения анализа (рис.35). Переменные ALL (Все), способ выделения кластеров по указанным переменным - Variable(columns), метод выделения кластеров Single Linkage, мера сходства и различия признаков - Euclidian Distances. После ввода всех данных нажать клавишу Ok.

Результаты анализа можно увидеть через промежуточную форму (рис.36).

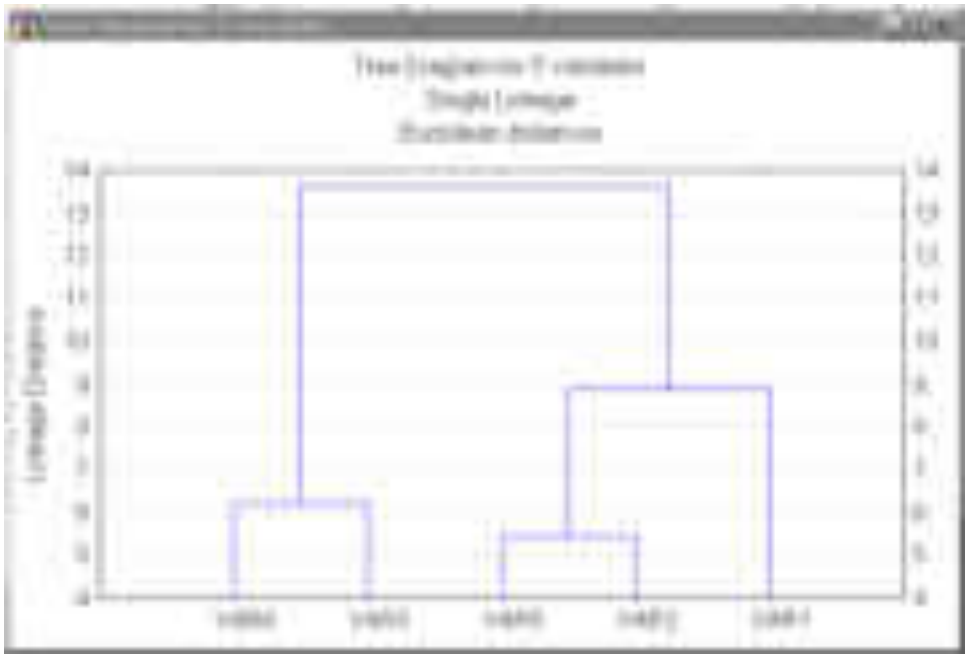


Кнопки Horizontal hierarchical tree plot и Vertical icicle plot позволяют наглядно представить дерево кластеров в иерархическом виде (рис.37). Переключатель Rectangular Branches меняет форму представления дерева кластеров с прямоугольной на треугольную. Переключатель Scale tree масштабирует изображения относительно максимальной длины связи. Следующие две кнопки Amalgamation (Образования) позволяют показать меры сходства в табличном и графическом видах.

Выводы

Результаты анализа дерева кластеров показывают, что заданная исходная модель данных практически оказалась выделенной в виде двух кластеров с

переменными VAR3 и VAR4 , а также VAR5, VAR2 и VAR1. Кроме того, выделено кластер VAR5 и VAR2, что связано с незначительными отклонениями в их математических моделях. Это показывает и описательная статистика.



Задание

1. Повторить расчеты, изменив закон формирования VAR5 = v1+RND(5). Что при этом измениться ? Сделать выводы.
2. Как изменяются результаты выполнения кластерного анализа, если изменить масштаб одной или двух переменных, умножив их значения на 10 ? Выводы обосновать.
3. Провести анализ с результатами данных таблицы 5 и сделать статистически обоснованные выводы.
4. Провести анализ с результатами данных таблицы 6 и сделать статистически обоснованные выводы.
5. Измените способ группирования данных в кластеры не по переменным (столбцам), а по строкам (Case). Как в этом случае интерпретируются результаты, например таблицы 5 ?
6. Попробуйте проверить какой-либо тест- опросник на предмет выделения заложенных в него шкал на вашей группе. Сделайте статистически обоснованные выводы. (Не выполнять в этом 2000 году)

Выберите какой—либо подходящий файл статистических данных из каталога с примерами, входящего в состав пакета Statistica, и проведите самостоятельный кластерный анализ.

САМОСТОЯТЕЛЬНАЯ РАБОТА ОБУЧАЮЩИХСЯ

Виды самостоятельной работы распределяются в течение семестра. Подготовка к промежуточной аттестации ведется в установленные календарным учебным графиком сроки.

Рекомендуемый перечень учебной литературы:

1. Мхитарян В.С. Статистика. В 2-х частях. Часть 1. Учебник и практикум для академического бакалавриата. Издательство: Юрайт, 2020
2. Дуброва. Анализ данных:Учебник. - Москва: Издательство Юрайт, 2018.
3. Анализ данных и процессов: учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009.
4. Соловьев В.И. Анализ данных в экономике: Теория вероятностей, прикладная статистика, обработка и анализ данных в Microsoft Excel. Учебник. издательство: КноРус, 2021
5. Миркин. Введение в анализ данных [Электронный ресурс]:Учебник и практикум. - Москва: Издательство Юрайт, 2019. - 174
6. Воронов В. И., Воронова Л. И., Усачев В. А.. Data Mining - технологии обработки больших данных [Электронный ресурс]:Учебное пособие. - Москва: Московский технический университет связи и информатики, 2018. - 47 с.
7. Федин Ф. О., Федин Ф. Ф.. Анализ данных. Часть 1. Подготовка данных к анализу [Электронный ресурс]:Учебное пособие. - Москва: Московский городской педагогический университет, 2012. - 204 с. – Режим доступа: http://www.iprbookshop.ru/26444.html
8. Нейтан Яу - Искусство визуализации в бизнесе. Как представить сложную информацию простыми способами
9. Джин Желязны - Говори на языке диаграмм. Пособие по визуальным коммуникациям
10. Методы бизнес-анализа / Пол Тернер, Джеймс Кадл, Дебра Пол
11. Ключевые показатели эффективности. 75 показателей, которые должен знать каждый менеджер / Бернард Марр.

Ниже приведены ресурсы информационно-телекоммуникационной сети «Интернет», полезные для самостоятельной работы.

1. Бизнес-аналитика. Введение https://sf.education/vvedenie-v-biznes-analitiku?admitad_uid=b045cb655ef2605764eff6e800a29960&utm_source=admitad&utm_campaign=442763&utm_medium=cpa
2. Анализ данных - Мхитарян В.С. Учебные материалы для студентов https://studme.org/93298/statistika/analiz_dannyh
3. Студия машинного обучения AZURE. https://azure.microsoft.com/ru- ru/services/machine-learning-studio/
4. Платформа машинного обучения ML.NET от компании Microsoft. https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet
5. Видео об обработке и анализе данных для начинающих https://docs.microsoft.com/ru-ru/dotnet/machine-learning/resources/basics
6. Школа прикладного бизнес-анализа https://babok-school.ru/blogs/dfd-diagram-practical-example/

ТЕСТОВЫЕ ЗАДАНИЯ

Тестовые задания по теме «Введение в бизнес-аналитику. Предварительный анализ данных»

1. Как называется совокупность подходов и методов для автоматизации анализа текстовых документов, включая задачи установления степени сходства текстов, категоризации документов, формирования аннотаций и пр.?
    1. Генетические алгоритмы (Genetic algorithms)
    2. Вычислительный интеллект (Computational intelligence)
    3. Анализ текстов (Text analysis)
    4. Кластер-анализ (Cluster analysis)
2. К классу описательных задач относятся:
    1. кластеризация и классификация;
    2. кластеризация и поиск ассоциативных правил;
    3. классификация и регрессия;
    4. классификация и поиск ассоциативных правил.
3. К классу предсказательных задач относятся:
    1. кластеризация и классификация;
    2. кластеризация и поиск ассоциативных правил;
    3. классификация и регрессия;
    4. классификация и поиск ассоциативных правил.
4. К классу задач supervised learning (обучение с учителем) относятся:
    1. кластеризация и классификация;
    2. кластеризация и поиск ассоциативных правил;
    3. классификация и регрессия;
    4. классификация и поиск ассоциативных правил.
5. К классу задач unsupervised learning (обучение без учителя) относятся:
    1. кластеризация и классификация;
    2. кластеризация и поиск ассоциативных правил;
    3. классификация и регрессия;
    4. классификация и поиск ассоциативных правил.
6. Задача классификации сводится к …
    1. нахождения частых зависимостей между объектами или событиями;
    2. определения класса объекта по его характеристикам;
    3. определение по известным характеристикам объекта значение некоторого его параметра;
    4. поиска независимых групп и их характеристик во всем множестве анализируемых данных.
7. Задача регрессии сводится к …
    1. нахождения частых зависимостей между объектами или событиями;
    2. определения класса объекта по его характеристиками;
    3. определение по известным характеристикам объекта значение некоторого его параметра;
    4. поиска независимых групп и их характеристик в всем множестве анализируемых данных.
8. Задача кластеризации заключается в …
    1. нахождения частых зависимостей между объектами или событиями;
    2. определения класса объекта по его характеристиками;

3. определение по известным характеристиками объекта значение некоторого его параметра;
4. поиска независимых групп и их характеристик во всем множестве анализируемых данных.

9. Целью поиска ассоциативных правил является …
   1. нахождения частых зависимостей между объектами или событиями;
   2. определения класса объекта по его характеристиками;
   3. определение по известным характеристиками объекта значение некоторого его параметра;
   4. поиска независимых групп и их характеристик во всем множестве анализируемых данных.
10. Модели классификации описывают …
    1. правила или набор правил, в соответствии с которыми можно отнести описание любого нового объекта к одному из классов;
    2. функции, которые позволяют прогнозировать изменения непрерывных числовых параметров;
    3. функциональные зависимости между зависимыми и независимыми показателями и переменными в понятной человеку форме;
    4. группы, на которые можно разделить объекты, данные о которых подвергаются анализа.

Тестовые задания по теме «Исследование зависимостей»

1. Регрессивные модели описывают …
   1. правила или набор правил, в соответствии с которыми можно отнести описание любого нового объекта к одному из классов;
   2. функции, которые позволяют прогнозировать изменения непрерывных числовых параметров;
   3. функциональные зависимости между зависимыми и независимыми показателями и переменными в понятной человеку форме;
   4. группы, на которые можно разделить объекты, данные о которых подвергаются анализа
2. Задача регрессии сводится к …
   1. нахождения частых зависимостей между объектами или событиями;
   2. определения класса объекта по его характеристикам;
   3. определение по известным характеристикам объекта значение некоторого его параметра;
   4. поиска независимых групп и их характеристик во всем множестве анализируемых данных.
3. Многомерный анализ – это:
   1. одновременный анализ по нескольким измерениям;
   2. одновременный анализ по нескольким параметрам;
   3. одновременный анализ по нескольким данным.
4. Коэффициент парной корреляции между процентом охвата населения прививками и заболеваемостью на 10 000 населения равен (-0,86). Можно сделать следующие выводы:
   1. связь между изучаемыми явлениями отсутствует, т.к. коэффициент корреляции отрицательный
   2. связь между изучаемыми явлениями обратная (отрицательная)
   3. связь между изучаемыми явлениями сильная и обратная
   4. связь между изучаемыми явлениями слабая и обратная

5. связь между изучаемыми явлениями средняя и обратная.

5. При каком значении коэффициента корреляции связь можно считать умеренной?
   1. r = 0,47;
   2. r = 0,71.
   3. r = 0,21;
6. Верно или нет следующее утверждение (и почему): если линия регрессии имеет большой наклон, то корреляция между переменными также должна быть большой?
   1. Да, потому что при большем наклоне увеличение одной переменной ведет к более сильному увеличению другой переменной
   2. Да, потому коэффициент корреляции пропорционален углу наклона линии регрессии.
   3. Нет, корреляция не зависит угла наклона.
7. Определить коэффициент корреляции двух переменных: агрессивности (x_a) и IQ у школьников (y_{iq}) по полученным данным тестирования

   1. 0,27
   2. -0,42
   3. -0,64
   4. 0,81
8. Знания 10 студентов проверены по двум тестам A и B. Оценки по стобалльной системе сведены в таблицу

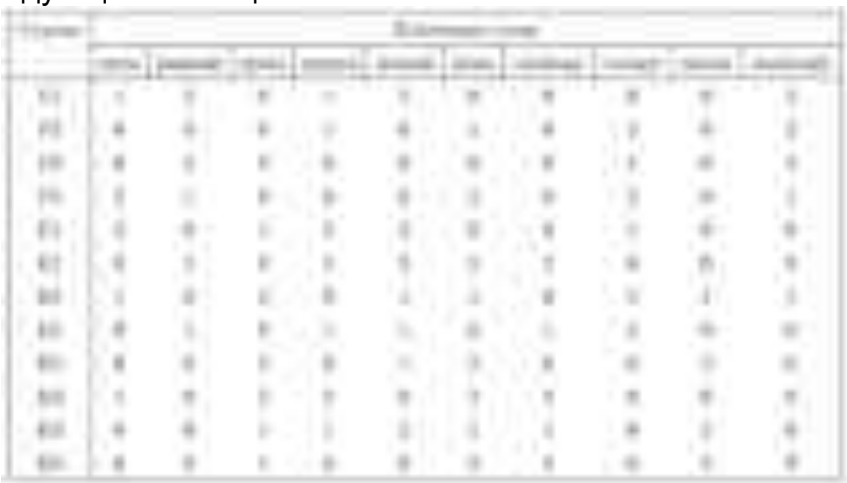
A	95	90	86	84	75	70	62	60	57	50
B	92	93	83	80	55	60	45	72	62	70

Найти выборочный коэффициент ранговой корреляции Спирмена между оценками по двум тестам.

   1. 0,27
   2. 0,42
   3. 0,64
   4. 0,81
9. По данным задачи №8 определить коэффициент ранговой корреляции Кендалла
   1. 0,28
   2. 0,47
   3. 0,63

4. 0,81

10. Если величина коэффициента корреляции Пирсона между переменными равна -0,75 то
    1. это очень слабая корреляция и в большинстве случаев мы не берем ее в расчет;
    2. это высокая корреляция и мы принимаем ее в расчет.
11. Что означает совсем низкое или нулевое значение коэффициента корреляции двух количественных признаков?
    1. наличие неизвестного вида связи
    2. наличие квадратичной зависимости
    3. отсутствие линейной связи
    4. наличие линейной связи
12. Термин регрессия в статистике понимают как: а) функцию связи, зависимости; б) направление развития явления вспять; в) функцию анализа случайных событий во времени; г) уравнение линии связи
    1. а, б
    2. в, г
    3. а, г
13. Для анализа связи между двумя номинальными признаками составляют следующую таблицу. Строки таблицы соответствуют категориям одного признака, а столбцы — категориям другого признака. Элемент на пересечении строки и столбца — количество объектов, обладающих соответствующими категориями и того и другого признаков. Такая таблица называется таблицей _____
14. Рассмотрим модель «мешок слов» в задаче выявления корреляции по данным следующей таблицы.



Объектами являются газетные статьи, разделенные на три категории в соответствии с темами «Феминизм», «Развлечения» и «Домохозяйство». Каждая статья характеризуется своим набором ключевых слов, представленных в соответствующей строке таблицы. Чтобы уменьшить эффект случайности отбора статей в таблицу данных, примем, что «мешок» содержит по одному появлению каждого ключевого слова, независимо от того, появилось ли оно в статьях данной категории или нет. Какова вероятность слова «играть» в категории Н?

1. 2/41
2. 7/41
3. 4/41

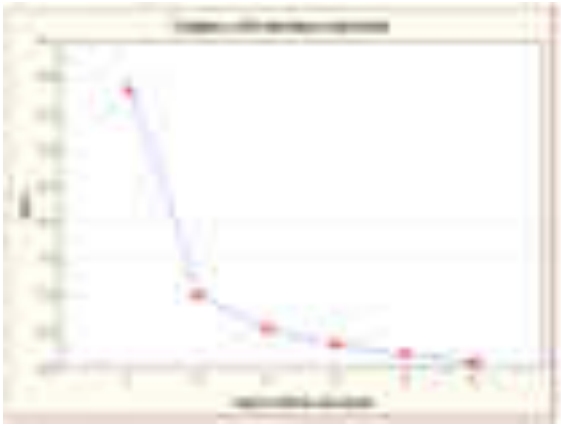
4. 3/41

Тестовые задания по теме «Снижение размерности признакового пространства»

1. Укажите действия, позволяющих уменьшить число факторов:
    1. Устранение дублирующей информации при наличии сильно коррелированных признаков
    2. Редукция слабоинформативных (маломеняющихся для различных объектов) признаков
    3. Агрегирование (объединение) нескольких признаков в один
2. Математический метод нахождения главных осей заключается в
    1. вычислении собственных чисел и собственных векторов ковариационной матрицы S.
    2. вычислении детерминанта ковариационной матрицы S
3. Даны четыре примера (наблюдения) в трехмерном пространстве признаков: A(1;4;10), B(2;5;6), C(1;3;8) и D(2;4;8). В результате применения метода главных компонент исходное пространство признаков свели к двумерному пространству признаков на плоскости. Найдите евклидово расстояние между примерами C и D в редуцированном пространстве с точностью до одного знака после запятой: _____
4. Какова идея метода главных компонент?
    1) поиск гиперплоскости заданной размерности, такой чтобы ошибка проектирования выборки на данную гиперплоскость была минимальной
    2) поиск проекции на гиперплоскость с сохранением большей части дисперсии в данных
    3) проекция данных на гиперплоскость с критической ошибкой проектирования
5. Укажите верное утверждение
    1. Метод главных компонент использует меньшее количество компонент, в отличие от метода независимых компонент
    2. Метод главных компонент добивается ортогональности между полученными компонентами, а метод независимых компонент - не ортогональности
    3. Метод независимых компонент работает с коррелированными данными, в отличие от метода главных компонент
    4. Метод главных компонент применяется в основном для задач, где необходимо разделять сигналы, а метод независимых компонент - для визуального разделения данных
6. Каковы недостатки метода главных компонент?
    1. координаты объектов в новом пространстве определены не однозначно
    2. проблема с вычислением собственных векторов ковариационной матрицы, при большом количестве данных
    3. существует произвол в выборе координат объектов в новом пространстве
    4. общая сложность алгоритма
7. Выберите сферы применения метода главных компонент
    1. Визуализация данных
    2. Построение деревьев решений
    3. Обработка изображений

4. Выявление максимальной избыточности
5. Отбор признаков

8. Укажите достоинство использования метода главных компонент
   1. Простой алгоритм
   2. Координаты объектов в новом пространстве определены однозначно
   3. Легкость с вычислением собственных векторов ковариационной матрицы в случае большого количества данных
   4. все перечисленное
9. Согласно критерию каменистой осыпи число общих факторов равно ____



10. По критерию Кайзера отбираются только факторы с собственными значениями
    1. равными или большими 1
    2. меньшими 1
    3. меньше натурального логарифма

Тестовые задания по теме «Классификация многомерных наблюдений»

1. Кластеризация — …
   1. это установление зависимости непрерывной выходной переменной от входных переменных
   2. эта группировка объектов (Наблюдений, событий) на основе данных, описывающих свойства объектов
   3. выявление закономерностей между связанными событиями
   4. это установление зависимости дискретной выходной переменной от входных переменных.
2. Задача кластеризации заключается в …
   1. нахождения частых зависимостей между объектами или событиями;
   2. определения класса объекта по его характеристикам;
   3. определение по известным характеристикам объекта значение некоторого его параметра;
   4. поиска независимых групп и их характеристик во всем множестве анализируемых данных.
3. Расставьте в правильном порядке шаги алгоритма K-средних
   1. Предобработка
   2. Инициализация аномального центра
   3. Обновление аномального центра
   4. Обновление аномальной группы
   5. Выдача результатов

4. Поставьте в правильном порядке шаги алгоритма uK-средних (t), где t — порог разрешения, т. е. задаваемое пользователем минимальное количество объектов в аномальной группе, необходимое, чтобы она могла восприниматься как генератор отдельного кластера.
    1. Условие остановки
    2. Метод K-средних
    3. Настройка
    4. Аномальная группа
    5. Отбрасывание малых кластеров
5. На рисунке изображены первая (слева) и финальная (справа) итерации извлечения аномальной группы из структуры некоторого множества объектов. Малая звезда представляет центр аномальной группы.

Как называется точка, обозначенная большой звездой?
    1. центральная точка
    2. реперная точка
    3. нормальная точка
    4. аномальная точка
6. Спорный объект кластеризации — это объект, который по мере сходства …
    1. может быть отнесен к нескольким кластерам
    2. не может быть отнесен ни к одному кластеру
    3. может быть отнесен более чем к двум кластерам
7. Процедура, которая приводит значения всех преобразованных переменных к единому диапазону значений путем выражения через отношение этих значений к некой величине, отражающей определенные свойства, это – …
    1. стандартизация
    2. нормирование
    3. оба ответа верны
8. Работа кластерного анализа опирается на следующие предположения (выберите неверный ответ):
    1. рассматриваемые признаки объекта в принципе допускают желательное разбиение объектов на кластеры
    2. правильность выбора масштаба или единиц измерения признаков
    3. отнесение всех объектов к одному из предопределенных признаков
9. Иерархические агломеративные методы характеризуются …
    1. последовательным объединением исходных элементов и соответствующим уменьшением числа кластеров
    2. делением одного кластера на меньшие кластеры, в результате образуется последовательность расщепляющих групп

3. сопоставлением фиксированного числа кластеров наблюдения кластерам так, что средние в кластере максимально возможно отличаются друг от друга

10. Объект относится к кластеру, если …
    1. расстояние от объекта до центра кластера меньше радиуса кластера
    2. расстояние от объекта до центра кластера меньше диаметра кластера
    3. расстояние от объекта до центра кластера больше радиуса кластера
11. Иерархические дивизимные методы характеризуются …
    1. последовательным объединением исходных элементов и соответствующим уменьшением числа кластеров
    2. делением одного кластера на меньшие кластеры, в результате образуется последовательность расщепляющих групп
    3. сопоставлением фиксированного числа кластеров наблюдения кластерам так, что средние в кластере максимально возможно отличаются друг от друга
12. При применении кластерного анализа переменные …
    1. должны измеряться в сравнимых шкалах
    2. могут измеряться в каких угодно шкалах
    3. должны быть только числовыми
13. Характеристикой каких групп методов являются последовательное объединение исходных элементов и соответствующее уменьшение числа кластеров?
    1. иерархические агломеративные методы
    2. иерархические дивизимные (делимые) методы
    3. и тех, и других
14. Деление одного кластера на меньшие кластеры, в результате чего образуется последовательность расщепляющих групп. Характеристика каких групп методов описана выше?
    1. иерархические агломеративные методы
    2. иерархические дивизимные (делимые) методы
    3. и тех, и других

ЭКЗАМЕН

Вопросы к экзамену

1. Основные задачи, решаемые в анализе данных.
2. Этапы решения задач. Постановка задачи. Ввод данных в обработку.
3. Этапы решения задач. Качественный анализ. Количественное описание данных. Интерпретация результатов.
4. Многомерные случайные величины. Моменты второго порядка случайной величины.
5. Многомерные случайные величины. Выборочные ковариация и корреляция.
6. Введение в многомерный стохастический анализ. Исследование зависимостей. Основные прикладные цели.
7. Классификация статистических переменных. Виды статистического анализа.
8. Этапы корреляционного анализа. Причинный и статистический характер связи. Виды характеристик связи. Природа статистической взаимосвязи.
9. Количественные характеристики статистической взаимосвязи. Диаграммы рассеивания. Ковариация.
10. Коэффициент парной корреляции. Корреляционная матрица.
11. Частные коэффициенты корреляции. Коэффициент множественной корреляции
12. Корреляционная функция.
13. Характеристики многомерной статистической связи.
14. Связь порядковых переменных. Коэффициент Спирмена. Коэффициент Кендалла.
15. Анализ связей между классификационными переменными. Таблицы сопряженности. Проверка гипотезы о наличии связи.
16. Регрессионный анализ. Общая постановка задачи. Основные этапы регрессионного анализа
17. Выбор модели регрессии.
18. Уравнение регрессии. Оценка параметров модели
19. Анализ регрессии. Адекватность модели. Дисперсия адекватности и дисперсия воспроизводимости.
20. Проверка значимости коэффициентов регрессионной модели.
21. Исследование регрессионных остатков
22. Оценка точности регрессионной модели.
23. Постановка задачи МГК
24. Вычисление главных компонент.
25. Факторный анализ (ФА). Общая постановка задачи. Цели ФА.
26. Порядок выполнения факторного анализа. Основные этапы факторного анализа.
27. Определение главных факторов – факторизация. Задание числа факторов.
28. Факторный анализ. Вращение факторов.
29. Факторный анализ. Оценка значений общих факторов.
30. Кластерный анализ (КА) Общая постановка задачи. Этапы кластерного анализа.
31. Кластерный анализ. Меры сходства. Расстояние между отдельными объектами. Расстояние между классами в кластерном анализе.
32. Обзор методов кластерного анализа.
33. Методы кластерного анализа. Иерархические методы.
34. Методы кластерного анализа. Итеративные методы. Алгоритм k-средних.

35. Методы ассоциативного поиска.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.04 Глубокое обучение в проектировании

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В. В. Воронина, А. В. Михеев,

Н. Г. Ярушкина, К. В. Святов

ТЕОРИЯ И ПРАКТИКА МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Ульяновск
УлГТУ
2017

УДК 004.8 (075)
ББК 32.973-018.1я7
В12

Рецензенты:

Главный научный сотрудник ФНЦП АО «НПО «Марс», д-р техн. наук Токмаков Г. П.

Начальник расчетно-теоретического отдела ОАО «Ульяновское конструкторское бюро приборостроения», канд. техн. наук Сорокин М. Ю.

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Воронина, Валерия Вадимовна

В12 Теория и практика машинного обучения : учебное пособие / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. – Ульяновск : УлГТУ, 2017. – 290 с.

ISBN 978-5-9795-1712-4

Учебное пособие рассматривает вопросы, связанные с анализом данных: модели, алгоритмы, методы и их реализацию на языке Python. Особое внимание уделено анализу временных рядов. Книга предназначена для студентов группы направлений 09, а также для студентов других групп направлений, изучающих дисциплины, связанные с разработкой приложений в области анализа данных, в том числе TimeSeriesDataMinig и DataMining.

УДК 004.8 (075)
ББК 32.973.2-018.2я7

ISBN 978-5-9795-1712-4

Содержание

Введение

На протяжении всего периода своего развития человечество постоянно ищет способы автоматизации тех или иных видов деятельности, в итоге перекладывая на «плечи» механизмов все более обширные и сложные обязанности. Изначально мечты о роботах, выполняющих всю работу, находили воплощение лишь в литературных произведениях, однако с развитием технологий им открылся путь и в реальность. Сейчас уже не столь фантастичными звучат слова песни «До чего дошел прогресс» (к/ф «Приключения Электроника»): «До чего дошел прогресс – труд физический исчез, да и умственный заменит механический процесс». Ведь эти строки достаточно полно отражают суть такой дисциплины, как «машинное обучение», благодаря которой их воплощение в жизнь становится возможным. Однако стоит заметить, что до полной реализации описанного в песне еще очень далеко: пока машинное обучение не способно заменить умственный труд, и в ближайшее время вряд ли сможет. Но шаги в этом направлении активно совершаются.

С теоретической стороны машинное обучение – дисциплина, находящаяся на пересечении математической статистики, численных методов оптимизации, теории вероятностей, а также дискретного анализа. С помощью ее методов происходит решение задачи извлечения знаний из данных, которой занимается еще только формирующаяся область «Интеллектуальный анализ данных» (DataMining). Согласно Википедии [1]: «В теории искусственного интеллекта и экспертных систем, знание – это совокупность утверждений о мире, свойствах объектов, закономерностях процессов и явлений, а также правил логического вывода одних утверждений из других и правил использования их для принятия решений. Главное отличие знаний от данных состоит в их структурности и активности: появление в базе знаний новых фактов или установление новых связей между ними может стать источником изменений в принятии решений». Если,

опираясь на это определение знания, рассмотреть DataMining как процесс, то его сутью будет нахождение в «сырых» данных знаний, обладающих свойствами нетривиальности, интерпретируемости и практической полезности для принятия решений в различных сферах деятельности человека. Причем, эти знания ранее неизвестны.

С практической же стороны машинное обучение нацелено на создание систем, способных адаптироваться к решению различных задач без явного кодирования алгоритма, то есть систем, способных обучаться.

В данной книге рассмотрены основные модели и алгоритмы машинного обучения, в том числе для анализа временных рядов. Наряду с этим представлены и их практические реализации на языке Python. В последних разделах книги обучающемуся предлагаются контрольные вопросы по пройденным темам, а также задачи для выполнения, с помощью которых он сможет проверить себя и закрепить полученные навыки.

Книга подготовлена при реализации проекта №2.4760.2017/8.9 «Исследование и разработка моделей, методов и алгоритмов гибридизации нечетких предметных онтологий, логического вывода и интеллектуального анализа временных рядов» в рамках государственного задания Минобрнауки Российской Федерации.

О ЗАДАЧАХ МАШИННОГО ОБУЧЕНИЯ

Как было сказано выше, машинное обучение нацелено на создание систем, способных получать знания из данных, а также способных с помощью обучения улучшать показатели своей работы. Таким образом, можно утверждать, что машинное обучение является одной из областей науки о данных (DataScience). На рисунке 1 представлены ее составляющие и показано место машинного обучения среди них.

Рисунок 1. Области науки о данных

Сложность работы в сфере DataMining обуславливается неточностью, противоречивостью, разнородностью, неполнотой данных, которые при этом могут иметь гигантские объемы. Для их обработки требуются специальные программные средства: инструменты преобразования «сырых» данных в информацию, а информацию в знания. Кроме того, алгоритмы обработки данных должны иметь возможность обучаться по прецедентам. Машинное обучение (МО) как раз и занимается разрешением подобных сложностей. Причем в настоящее время усилия сконцентрированы на повышении

не столько качества решения, сколько скорости и оптимальности его получения.

Как видно из рисунка 1, машинное обучение пересекается с анализом данных, то есть имеет общие черты, но также и свою специфику. Анализ данных – это огромная и уже относительно не молодая область математики и информатики. Под нее попадает всяческая обработка данных. Например, обработка данных физического эксперимента статистическими методами для получения обобщенной информации (усредненных значений, точных значений, доверительных интервалов, оценок и тому подобное), обработка сигнала методом Фурье, даже подсчет минимального и максимального значений или отношения каких-то величин может быть анализом данных, если эти операции влекут за собой важные выводы, знания о данных и позволяют решить поставленную задачу. Так получение дисперсии некой величины является анализом данных, но не является в чистом виде машинным обучением. Хотя важно отметить, что методы статистики имеют большое значение и в машинном обучении, о чем мы еще будем говорить. Отличительной особенностью методов машинного обучения является то, что они способны решать широкий спектр задач без явного кодирования алгоритма решения. Например, метод Регрессии входит в машинное обучение. Регрессия давно применяется в экономике для прогнозирования и оценки разнообразных параметров, потому что данная математическая модель в некотором роде универсальна. Однако стоит заметить, что регрессия появилась в математике гораздо раньше, чем ее стали использовать в экономике, а вот машинным обучением это стало только тогда, когда все эти методы начали рассматриваться не как обособленные механизмы решения задач из специальных областей, а как методы преобразования информации вообще.

На рисунке 2 представлена диаграмма распределения задач, которые приходится решать в сфере машинного обучения. Необходимо отметить, что соотношение областей на данной диаграмме

отражает лишь субъективное видение отрасли авторами на момент написания книги.

Рисунок 2. Типы задач машинного обучения

Понятно, что разработка программ в области машинного обучения весьма дорога. Поэтому применять его следует лишь в случаях, когда это действительно необходимо. В таблице 1 представлены некоторые признаки, по которым можно принять решении о необходимости применения к задаче методов машинного обучения.

Таблица 1. Ключевые характеристики решаемых задач

Целесообразно применение классических методов	**Целесообразно применение методов машинного обучения**
Наличие четко формализованного алгоритма (например, поиск клиентов, которые тратят больше всего денежных средств в месяц или чаще всего покупают. Так называемая RFM сегментация).	Отсутствие четко формализованного алгоритма решения ввиду высокой вариативности решения (например, распознавание рукописного текста).
Не допускается неопределенность поведения модели (например, расчет местоположения самолета по строго заданным формулам).	Допускается неопределенность поведения модели (например, предварительный контроль качества детали, где допускается определенная доля брака).

Окончание таблицы 1

Целесообразно применение классических методов	Целесообразно применение методов машинного обучения
Экономическая целесообразность (например, методы МО дают 98% качества, классическое решение дает 90% качества, что является приемлемым, но решение на основе МО в 2 раза дороже и реализуется дольше). Поэтому здесь выбирается приоритетный критерий.	

Кроме того, для разработки решений на основе МО необходимы оцифрованные данные. Они являются ключевым аспектом, от которого зависит качество и полнота решения. Таким образом, очень важно понимать, что для применения методов машинного обучения требуются собранные и специально подготовленные данные.

Как правило, процесс машинного обучения проходит несколько этапов. На рисунке 3 представлен обобщенный алгоритм, выполняющийся при решении задач методами рассматриваемой области.

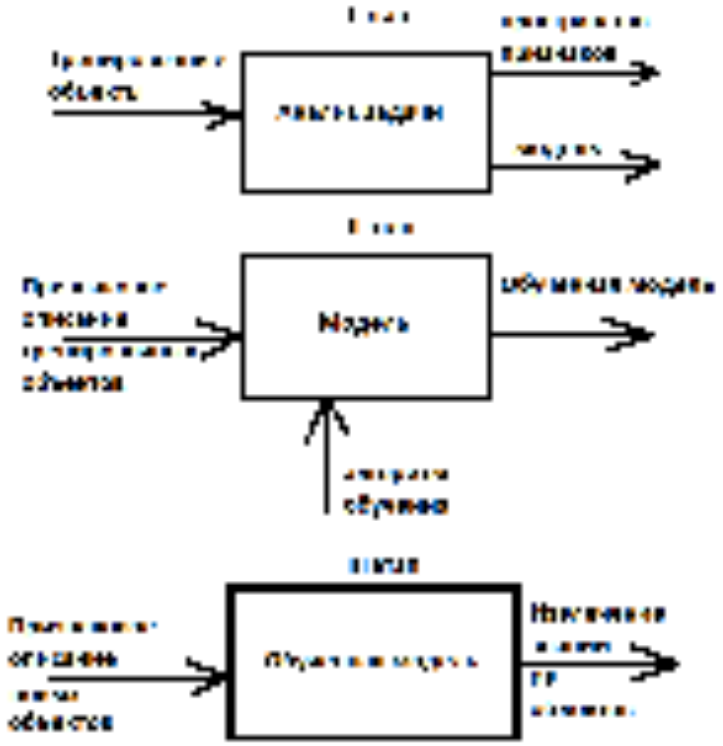


Рисунок 3. Этапы машинного обучения

Необходимо отметить, что данная иллюстрация отражает идеальный случай, когда после обучения мы сразу получили работающую модель. Однако в реальности, как правило, между вторым и третьим этапом есть промежуточное, но очень важное действие – оценка качества модели. И именно по его результатам принимается решение о переходе на следующий этап или о возврате к одному из предыдущих. Этот вопрос мы рассмотрим подробнее далее, здесь же введем определение пространства признаков, для лучшего понимания схемы с рисунка 3.

Пространство признаков

Пространство признаков – это N-мерное пространство, где N – число измеряемых характеристик объектов, выделенное для конкретной задачи. При этом объекты в пространстве признаков задаются N-мерными векторами, каждая компонента которых представляет собой значение определенной характеристики.

Рассмотрим пример. Пусть необходимо разработать функцию для сайта салона красоты, которая будет предлагать клиентам определенные косметические процедуры. Понятно, что они будут разными для разных возрастных и гендерных групп. То есть в этой задаче ключевыми будут характеристики пола и возраста людей. Графическое отображение пространства признаков, а также заданные в нем объекты A и B показаны на рисунке 4. В векторном виде объект A с характеристиками «пол» = «женский» и «возраст» = 40 будет задан как: $A= \{40, ж\}$. Объект B с возрастом 50 и мужским полом – $B= \{50, м\}$.

Как видно из рассмотренного примера, признаки объектов могут быть разными по своей природе. Существует следующая типизация:

1. Бинарные: $F \epsilon \{0,1\}$.
2. Номинальные: $|F| < \infty$.
3. Номинальные, упорядоченные: $|F| < \infty$, $F_i < F_{i+1} \ldots < F_{i+n}$.
4. Количественные: $F \epsilon R$.

Каждый тип признаков обрабатывается по-разному, о чем скажем далее. Пока же продолжим разговор о других аспектах рассматриваемой дисциплины. Как неоднократно было сказано, работа в сфере машинного обучения нацелена на создание систем, способных обучаться. Рассмотрим формальное определение понятия «обучение».

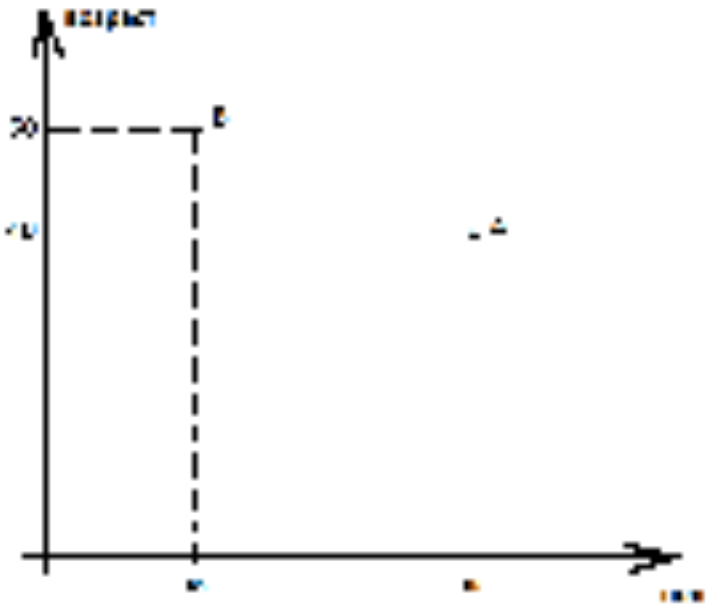


Рисунок 4. Объекты в пространстве признаков

Формальное определение понятия «обучение»

Формально понятие «обучение» специалисты определяют так: «Пусть есть некое множество задач класса *C,* для которого задано некоторое множество опыта *EX* и определена мера качества *L*. Тогда о наличии обучения на опыте *EX* относительно класса задач *C* в смысле меры качества *L* можно говорить в том случае, если при предъявлении нового опыта *EX'* возрастает качество решения задачи класса *C,* измеряемое мерой *L»*.

Существуют два основных типа обучения: с учителем и без учителя. Более подробно о них будет рассказано далее, здесь же рассмотрим формальную постановку задачи обучения для самого первого (и самого простого) типа – обучения с учителем.

Пусть X – некоторое множество объектов, по которым необходимо получить некоторое множество ответов Y. При этом предполагается, что существует некоторая зависимость $y: X \to Y$. Тогда задача обучения будет формулироваться так: дано $\{x_1 \dots x_n\} \in X$ – обучающая выборка, и $\{y_1 \dots y_n\}$ – известные ответы, причем $y_i = y(x_i)$. Найти: $a: X \to Y$ – решающую функцию, приближающую y на всем множестве X.

Данная задача в свою очередь порождает ряд подзадач:

1. Определение способа задания объектов;
2. Определение способа задания ответов;
3. Определение способа построения функции a;
4. Определение способа приближения для a и y.

Все эти подзадачи решаются последовательно, и результаты решения предыдущей, как правило, влияют на решение текущей. Поэтому самой важной среди них будет определение способа описания данных. Можно выделить следующие типы входных данных при обучении:

1. Координаты объектов в пространстве признаков;
2. Временной ряд;
3. Сигнал;
4. Изображение;
5. Видеоряд;
6. Описание взаимоотношений между объектами.

На основе описанного выше введем формальное описание приведенных на рисунке 3 второго и третьего этапов машинного обучения.

Пусть $X = \{x_1 \dots x_k\}$ – исследуемые объекты. $F = \{f_1 \dots f_N\}$ – пространство признаков, в котором эти объекты будут рассматриваться. При этом $f_i(x_j)$, $i \in [1 \dots N]$, $j \in [1 \dots k]$ – значение i-го признака j-го объекта. Тогда признаковое описание тренировочных объектов задается в виде матрицы

$$D = || f_i(x_j) ||_{k \times N} = \begin{pmatrix} f_1(x_1) \dots & f_N(x_1) \\ f_1(x_k) \dots & f_N(x_k) \end{pmatrix}.$$

Этап обучения может быть формализован следующим образом. Метод обучения $\mu: (X \times Y)_k \rightarrow A$ по выборке $XY_k=(x_i,y_i)$, $i \in [1 \ldots k]$ строит алгоритм $a=\mu(XY_k)$:

$$\begin{pmatrix} f_1(x_1) \ldots & f_N(x_1) \\ f_1(x_k) \ldots & f_N(x_k) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ y_k \end{pmatrix} \xrightarrow{\mu} a$$

Этап применения обученной модели формализуется следующим образом. Алгоритм a для новых объектов $X'=\{x'_1 \ldots x'_k\}$ выдает ответы $y_i'=a(x'_i)$, $i \in [1 \ldots k]$:

$$\begin{pmatrix} f_1(x'_1) \ldots & f_N(x'_1) \\ f_1(x'_k) \ldots & f_N(x'_k) \end{pmatrix} \xrightarrow{a} \begin{pmatrix} y'_1 \\ y'_k \end{pmatrix}.$$

Общий алгоритм машинного обучения

С понятием обучения тесно связано понятие обобщающей способности. Обобщающая способность – это свойство модели отражать исходные данные в требуемые результаты ($X \rightarrow Y$) на всем множестве исходных данных (во всех сценариях, а не только на тренировочных примерах). Величину обобщения оценивают через обратную величину – отклонение или ошибку. Ошибка – это численно выраженная разница между ответом модели и требуемым (реальным) значением. В более общем смысле обобщающая способность – способность модели найти некий «закон природы», который будет описывать неизвестную нам и скрытую взаимосвязь входных и выходных данных. Таким образом, опираясь на понятие обобщающей способности, можно выделить основные проблемные вопросы машинного обучения:

1. Достаточно ли данных для нахождения в них полезных знаний?
2. Может ли в принципе данная модель обучиться на имеющихся данных?
3. Будет ли полученная модель приближать требуемый «закон природы» на всем возможном множестве X?

4. Насколько хорошо будет работать обученная модель или насколько часто и насколько сильно будет ошибаться модель на реальных (контрольных) данных?

По сути в машинном обучении решается задача приближения алгоритма к некоторому «закону природы». С математической точки зрения задача приближения является задачей оптимизации или минимизации ошибки:

$$E=|y-y'|\rightarrow min\ .$$

Таким образом, общий алгоритм решения задач в сфере машинного обучения будет состоять из следующих шагов:

1. Понимание задачи и исходных данных;
2. Формулировка решения на математическом языке (на этом шаге важно понять, что задача формализуема, а результаты работы модели могут быть проверены);
3. Предобработка данных и выделение ключевых признаков;
4. Построение модели (или моделей);
5. Обучение модели (моделей) и оценка качества;
6. Эксплуатация модели при достижении требуемого качества, либо возврат к одному из предыдущих шагов (перенастройка модели, добыча новых данных и т. п.).

Типы задач машинного обучения

Машинное обучение применяется для решения задач в следующих областях (основные сферы применения):

1. Медицинская диагностика.
2. Техника, в частности:
    2.1. Автоматизация и управление.
    2.2. Техническая диагностика.
    2.3. Робототехника.
    2.4. Компьютерное зрение.
    2.5. Распознавание речи.
3. Экономика, в частности:
    3.1. Кредитный скоринг (credit scoring).

3.2. Предсказание ухода клиентов (churm prediction).

3.3. Обнаружение мошенничества (fraud detection).

3.4. Биржевой технический анализ (technical analysis).

3.5. Биржевой надзор (market surveillance).

4. Офисная автоматизация, в частности:

4.1. Распознавание текста.

4.2. Обнаружение спама.

4.3. Категоризация документов.

4.4. Распознавание рукописного ввода.

Если же говорить об обобщенных типах задач машинного обучения, то можно выделить следующие:

1. Регрессия (или иногда встречается термин «аппроксимация»);
2. Классификация;
3. Кластеризация.

Помимо указанных видов существуют и другие, но остановимся на обозначенных, так как они наиболее распространены. Рассмотрим формальную постановку этих задач.

Задача регрессии – приближение неизвестной целевой зависимости на некотором множестве данных. Пусть X – множество данных – описаний некоторых объектов. Y – множество возможных ответов для X.

В задаче регрессии предполагается, что существует неизвестная целевая зависимость $y: X \to Y$, чьи значения известны только на объектах обучающей выборки $XY=\{(x_1,y_1)\dots (x_n,y_n)\}$, $x \in X$, $y \in Y$. Необходимо получить алгоритм $a: X \to Y$, приближающий целевую зависимость как на множестве XY, так и на X. То есть решить задачу регрессии – значит найти алгоритм, обладающий способностью к обобщению эмпирических фактов (способностью к выводу общих знаний из частных наблюдений, прецедентов).

Задача классификации – распределение некоторого множества объектов по заданному множеству групп (классов). При этом есть некоторое подмножество объектов, для которых распределение по классам известно, классовая принадлежность остальных – неизвестна.

Требуется построить алгоритм, который указывал бы классовую принадлежность для любого объекта из исходного множества.

Формально постановку задачи классификации можно описать следующим образом. Пусть X – множество данных – описаний некоторых объектов. Y – конечное множество классов, отмеченных метками. Существует неизвестная целевая зависимость – отображение $y: X \to Y$, чьи значения известны только на объектах обучающей выборки $XY = \{(x_1, y_1) \dots (x_n, y_n)\}$, $x \in X$, $y \in Y$. Необходимо получить алгоритм $a: X \to Y$, способный классифицировать произвольный объект $x \in X$.

Как можно заметить, данная задача схожа с предыдущей. Однако главная особенность задачи регрессии заключается в том, что функция $a: X \to Y$ является непрерывной вещественной функцией. Задача классификации отличается от этого тем, что Y – дискретное множество. Кроме того, в отличие от задачи аппроксимации у задачи классификации выделяют несколько типов. По количеству классов можно выделить:

1. Классификацию на два класса: множество Y содержит всего две метки.
2. Классификацию на множество классов: Y содержит от трех до нескольких тысяч меток.

По характеру разделения объектов на классы можно выделить:

1. Классификацию на непересекающиеся классы: один объект принадлежит только одному классу.
2. Классификацию на пересекающиеся классы: один объект может принадлежать нескольким классам.
3. Классификацию на нечеткие множества: объект принадлежит всем классам с определенной степенью принадлежности.

Задача кластеризации – разделение некоторого множества объектов на непересекающиеся группы (кластеры) таким образом, чтобы каждая группа состояла из схожих объектов, а объекты разных кластеров существенно отличались.

Формально постановку задачи кластеризации можно описать следующим образом. Пусть X – множество данных – описаний некоторых объектов. Y – множество кластеров, отмеченных метками. Определена функция расстояния между объектами из исходного множества X: $f(x, x')$, и есть некоторая обучающая выборка объектов $X_o = \{x_1 \dots x_n\}$, $x \in X$. Необходимо разбить обучающую выборку на кластеры, приписав каждому x номер кластера y_i, так, чтобы близкие по метрике f объекты принадлежали одному кластеру, а объекты разных кластеров существенно отличались по метрике f. То есть необходимо построить алгоритм $a: X \rightarrow Y$, который любому $x \in X$ ставит в соответствие номер кластера $y \in Y$. Причем, иногда множество Y известно заранее, но чаще все-таки ставится задача получить оптимальное число кластеров, исходя из характера данных. Оптимальность оценивается по какому-либо критерию качества кластеризации.

Задача кластеризации сложнее аппроксимации и классификации. Это обусловлено следующими причинами:

1. Нет однозначного критерия качества кластеризации. Существует ряд эвристических критериев, а также ряд бескритериальных алгоритмов, выполняющих вполне осмысленную кластеризацию, но дающих на одних и тех же данных разные результаты.
2. Число кластеров, как правило, заранее неизвестно и задается субъективно.
3. На результат кластеризации существенное влияние оказывает выбранная метрика расстояния, которая, как правило, также выбирается субъективно.

Однако, несмотря на описанные выше сложности, кластеризация помогает достичь следующие цели:

1. Улучшить понимание данных за счет выявления их кластерной структуры: разбиение объемной выборки на группы схожих объектов может упростить дальнейшую обработку

данных за счет применения к каждому кластеру своих методов анализа.

2. Осуществить сжатие данных. В данном случае подразумевается сокращение объемной выборки за счет работы с наиболее яркими представителями кластеров (групп схожих объектов).
3. Выявить новизну в массиве данных: обнаружить нетипичные объекты, которые не удается отнести ни к одному кластеру.
4. Решить задачу таксономии: построить древообразную иерархическую структуру, упорядочивающую исходные данные. Ее построение достигается за счет дробления крупных кластеров на более мелкие, которые в свою очередь также дробятся на еще более мелкие. Визуально таксономия отображается в виде графика – дендрограммы.

Способы обучения и оценки его качества

Как было сказано выше, основная характеристика систем, разрабатываемых с помощью методов машинного обучения, – способность к обучению. В зависимости от видов решаемых задач применяют различные алгоритмы реализации этой ключевой особенности. В рамках данного пособия рассмотрим три основных вида обучения, а также определим классы задач, подходящие для каждого из этих видов.

Первый тип – обучение с учителем. Формально он был описан ранее, поэтому здесь не будем на этом останавливаться подробно. Вкратце же обучение с учителем можно описать следующим образом: дано некоторое множество объектов и множество возможных реакций системы на эти объекты. При этом ответы и объекты связаны между собой некоторой неизвестной зависимостью. Есть конечная совокупность пар объект-ответ (прецедентов), называемая обучающей выборкой. На ее основе необходимо выявить алгоритм, который впоследствии для любого объекта из исходного множества даст достаточно точный ответ. Для измерения точности ответов используется один

из функционалов качества, как правило, завязанный на вычислении отклонения полученного ответа от ожидаемого, то есть вычислении ошибки. Рассмотрим некоторые их виды. В приведенных ниже формулах используются следующие обозначения: $XY=\{(x_1,y_1)\dots(x_n,y_n)\}$ – обучающая выборка, n – количество прецедентов, y_i – фактическое значение (ожидаемый ответ) в *i-м* прецеденте ($y_i \in XY$), $\hat{y}_i$ – выданный системой ответ для x_i ($x_i \in XY$).

1. Средняя ошибка представляет собой усреднение ошибок для каждого образца и вычисляется по формуле

$$\text{СО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}.$$

2. Средняя абсолютная ошибка представляет собой усреднение абсолютных ошибок на каждом шаге и вычисляется по формуле

$$\text{САО} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}.$$

3. Среднеквадратическая ошибка вычисляется как сумма средних квадратов ошибок. Формула:

$$\text{СКО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}.$$

4. Корень из среднеквадратической ошибки вычисляется по формуле

$$\text{КСКО} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}.$$

5. Средняя относительная ошибка вычисляется как среднее относительных ошибок:

$$\text{CO} = \frac{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)}{y_i}}{n} \times 100\%.$$

6. Средняя абсолютная относительная ошибка вычисляется как среднее относительных ошибок по модулю:

$$\text{MAPE} = \frac{\sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|}{n} \times 100\% .$$

7. Симметричная средняя абсолютная относительная ошибка вычисляется как

$$\text{SMAPE} = \frac{\sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{(y_i + \hat{y}_i)/2} \right|}{n} \times 100\% .$$

У всех представленных мер качества есть свои достоинства и недостатки. Например, у первой и пятой недостаток заключается в том, что положительные и отрицательные ошибки аннулируют друг друга, поэтому в некоторых случаях они не являются достаточно хорошими индикаторами качества. В связи с этим чаще всего используется третья или четвертая меры.

Обучение с учителем используется при решении задач аппроксимации и классификации. В первом случае ответы являются действительными числами или векторами, во втором – выбираются из конечного множества меток-классов. Необходимо отметить, что приведенные выше формулы подходят только для случаев, когда ответ системы и требуемый ответ – действительные числа, получаемые при решении задачи аппроксимации. В задачах классификации же оценка качества чаще всего завязана на соотношении количеств правильно и неправильно отнесенных к классам объектов.

Второй тип обучения – обучение без учителя. Формально постановку задачи обучения без учителя можно описать следующим образом. Пусть X – множество данных – описаний некоторых

объектов. Необходимо найти множество Y, состоящее из взаимосвязей $f: (x, x')$ между объектами из X $(x, x' \in X, f \in Y)$. Качество выявления взаимосвязей проверяется некоторой метрикой, выбранной исходя из решаемой задачи.

Обучение без учителя используется для решения следующих типов задач:

1. Задача кластеризации.
2. Поиск правил ассоциации.
3. Сокращение размерности данных.
4. Визуализация данных.

В определенной степени каждая из последних трех задач является производной от первой или ее частным случаем. Рассмотрим подробнее формулировки названных задач.

Под задачей поиска правил ассоциаций подразумевается выявление в признаковых описаниях объектов (исходных данных) таких наборов и значений признаков, которые особенно часто (неслучайно часто) встречаются в исходных данных. Если же проводить аналогию с первой задачей, то каждое правило в данном случае может быть представлено как кластер.

Задача сокращения размерности данных состоит в следующем. Существует большой (значительно большой) объем признаковых описаний объектов. Причем этот объем обуславливается внушительным количеством измерений признакового пространства. Необходимо представить те же данные в пространстве меньшей размерности, при этом минимизировав потери информации. Группировка по кластерам как раз и будет одним из вариантов решения проблемы.

Задача визуализации данных является по сути частным случаем предыдущей: ее цель – представить исходные данные в отображаемом пространстве, то есть пространстве размерности 2 или 3.

Как следует из описанного выше, обучение без учителя в какой-то мере так или иначе сводится к кластеризации. Поэтому для оценки качества обучения данным способом, как правило, используют метрики качества кластеризации. Причем при их выборе учитывается,

что эти метрики не должны зависеть от исходных данных, а только от результатов разбиения. Все оценки качества можно разделить на внешние и внутренние. Первые используют внешнюю информацию об истинном разбиении объектов на кластеры, вторые опираются только на набор исходных данных, то есть данные метрики могут работать с неразмеченной выборкой, когда заранее не известно истинное разбиение объектов на группы. И именно с их помощью определяют оптимальное число кластеров.

Приведем в качестве примера метрики, выделенные компанией ODS [2]:

1. Adjusted RandIndex (*ARI*). Данная метрика относится к группе внешних: предполагается, что истинные метки объектов известны (например, заданы экспертом), однако от самих значений меток она не зависит. Только от разбиения выборки на кластеры.

Рассчитывается мера следующим образом: пусть n – число объектов в выборке, a – число пар объектов, имеющих одинаковые метки и находящихся в одном кластере, b – число пар объектов, имеющих различные метки и находящиеся в различных кластерах. Тогда можно посчитать долю объектов, для которых исходное и полученное в результате кластеризации разбиения согласованы:

$$\boldsymbol{RI} = \frac{\boldsymbol{2(a+b)}}{\boldsymbol{n(n-1)}} .$$

Полученная величина называется RandIndex (*RI*) и выражает схожесть двух разных кластеризаций одной и той же выборки. Чтобы этот индекс давал значения, близкие к нулю, для случайных кластеризаций при любом n и числе кластеров, необходимо произвести его нормирование, то есть получить AdjustedRandIndex:

$$\boldsymbol{ARI} = \frac{\mathbf{RI} - \mathbf{E[RI]}}{\boldsymbol{max}(\mathbf{RI}) - \mathbf{E[RI]}} ,$$

где E – математическое ожидание.

Мера *ARI* симметрична и не зависит от перестановок и значений меток. По сути этот индекс является мерой расстояний между различными разбиениями выборки. Его область значений – [-1,1]. Интерпретировать ее интервалы можно следующим образом: для «независимых» разбиений на кластеры – отрицательные значения, для случайных разбиений – близкие к нулю, для схожих разбиений – положительные значения, причем, *ARI*=1 говорит о совпадении разбиений.

2. Adjusted MutualInformation (*AMI*). Данная метрика схожа с предыдущей: она также симметрична и не зависит от значений и перестановок меток. Для ее определения используется функция энтропии. Разбиения выборки интерпретируются как дискретные вероятности: вероятность отнесения к кластеру равна доле объектов в нем. Как и в предыдущем случае, для данной метрики необходимо вычислить специальный индекс – MutualInformation (*MI*). Данная величина определяется как взаимная информация для двух распределений, соответствующих разбиениям выборки на кластеры. Интерпретировать это можно как долю информации, общей для обоих разбиений: насколько информация об одном из них уменьшает неопределенность относительно другого. Этот индекс рассчитывается по следующей формуле:

$$\boldsymbol{MI(X,Y)} = \sum_{\boldsymbol{y}\in\boldsymbol{Y}}\sum_{\boldsymbol{x}\in\boldsymbol{X}}\boldsymbol{p(xy)}\log\left(\frac{\boldsymbol{p(xy)}}{\boldsymbol{p(x)p(y)}}\right),$$

где *p(x, y)* – совместная функция распределения вероятностей для *X* и *Y; p(x)* и *p(y)* – функции распределения предельной вероятности для *X* и *Y* соответственно.

Областью значения индекса *AMI* является диапазон [0,1]. Интерпретируется он следующим образом: значения, близкие к нулю, говорят о независимости разбиений, а близкие к единице – об их схожести (или совпадении при *AMI*=1).

3. Гомогенность, полнота, V-мера. Данные метрики рассматривают разбиение выборки как дискретные распределения и определяются с использованием функции энтропии и условной энтропии.

Гомогенность определяет, насколько каждый кластер состоит из объектов одного класса, и рассчитывается по формуле

$$h = 1 - \frac{H(C|K)}{H(C)},$$

где K – результат кластеризации, C – истинное разбиение, H – функция энтропии. При этом

$$\boldsymbol{H(X) = -\sum_{i=1}^{n} P(x_i)\log_b P(x_i)},$$

$$\boldsymbol{H(X|Y) = \sum_{j} p(x_i y_j)\log \frac{p(y_j)}{p(x_i y_j)}}.$$

Полнота измеряет, насколько объекты одного класса относятся к одному кластеру и определяется по формуле

$$c = 1 - \frac{H(K|C)}{H(K)}.$$

Эти меры принимают значения в диапазоне [0,1]. Интерпретировать их можно следующим образом: чем больше значение, тем более точна кластеризация. Эти меры не являются симметричными и не являются нормализованными, в отличие от рассмотренных ранее, и поэтому они зависят от числа кластеров. Согласно ресурсу [2]: «Случайная кластеризация не будет давать нулевые показатели при большом числе классов и малом числе объектов. В этих случаях предпочтительнее использовать *ARI*. Однако при числе объектов более 1000 и числе кластеров менее 10 данная проблема не так явно выражена и может быть проигнорирована».

Чтобы учесть значения обеих величин, вводится симметричная *V*-мера, показывающая, насколько кластеризации схожи между собой. Ее расчет происходит по формуле

$$v = 2\frac{hc}{h+c}.$$

Силуэт. Данная метрика относится к типу внутренних: она не предполагает знания об истинном разбиении и первоначально рассчитывается для каждого объекта следующим образом. Пусть a – среднее расстояние от данного объекта до объектов из того же кластера,

b – среднее расстояние от данного объекта до объектов из ближайшего кластера (но не того, в котором находится сам объект). Тогда силуэтом данного объекта будет величина, вычисляемая по формуле

$$s = \frac{b - a}{\max(ab)}.$$

Силуэтом же всей выборки будет средняя величина силуэтов ее объектов. Интерпретировать метрику можно следующим образом: она показывает, насколько среднее расстояние до объектов своего кластера отличается от среднего расстояния до объектов других кластеров. Область значения данной величины – диапазон [-1,1]. При этом значения, близкие к левой границе, соответствуют плохим (разрозненным) кластеризациям; значения, близкие к середине, говорят о пересечении и наложении кластеров; значения, близкие к правой границе, соответствуют «плотным», четко выделенным кластерам. Согласно [2]: «чем больше силуэт, тем более четко выделены кластеры, и они представляют собой компактные, плотно сгруппированные облака точек».

Данная величина может быть использована для выбора оптимального числа кластеров, если оно заранее неизвестно: оно будет равно числу, максимизирующему значение силуэта. В отличие от рассмотренных ранее метрик, данная величина зависит от формы кластеров. Силуэт достигает больших значений при более выпуклых кластерах, которые получаются при помощи алгоритмов, основанных на восстановлении плотности распределения.

Типовые задачи при подготовке данных и обучении моделей

В разделе «Общий алгоритм машинного обучения» были представлены шаги разработки решения. Если задача типовая и не требует специального исследования или разработки специального метода, то существенная работа начинается с п. 3, который посвящен подготовке данных (обучающей и тестовой выборке). Исходные данные в подавляющем большинстве случаев требуют предварительной обработки перед тем как на них будет обучаться модель. Как правило, предварительная обработка подразумевает следующую работу: учет

пропусков, кодирование нечисловых данных, приведение данных к единому масштабу и стандартизация данных, разметка данных. Рассмотрим их подробнее.

Учет пропусков

В реальных задачах данные могут содержать пропуски. Это может быть вызвано тем, что клиенты не заполняли все поля в анкете или аккаунте, или тем, что не все параметры были оцифрованы за все время работы системы.

Таким образом, встает вопрос о том, как интерпретировать пропуски. Простейший вариант заключается в исключении объектов, имеющих неполные сведения (то есть удаление тех строк из матрицы признаков, в столбцах которых есть пропуски), или исключении признаков, содержащих неполные сведения (то есть удаления тех столбцов, в которых есть пропуски). Плюсы этого варианта в том, что он очень прост и его можно сразу опробовать на какой-либо модели. Минусы вполне очевидны. Если много объектов будут иметь пропуски, то мы рискуем удалить важные аспекты закономерности, которая сокрыта в данных, и, как следствие, получим низкое качество аппроксимации. Если мы удалим столбец, который содержит крайне важный признак, мы рискуем вовсе потерять ключевую информацию о разделении объектов.

Второй вариант борьбы с пропусками заключается в том, чтобы заменить их при помощи интерполяции. Это может быть среднее или медианное значение по столбцу. В случае если признак является функцией от времени, как и отдельные объекты, то можно интерполировать пропуски только по соседним временным значениям (например, если объект представляет собой вероятность покупки квартиры, а признак – среднюю заработную плату программиста в этот год, то за пропущенный год цифру можно приблизить по двум соседним).

Третий вариант можно также рассмотреть на примере с квартирой, который был описан выше. Если речь идет о каких-то открытых

экономических, социальных или других параметрах, то их можно найти в других источниках и тем самым дополнить данные.

Четвертый вариант заключается в том, чтобы закодировать пропуски специальным числовым значением (это может быть число, не встречающееся у других объектов) или категориальным значением (представление категориальных признаков будет рассмотрено ниже). Такой подход, как минимум, позволит внести информацию о том, что эти объекты по этому признаку отличаются от других (то есть мы не сообщим точной информации, но не удалим объекты полностью, как в первом подходе).

Пятый подход заключается в привлечении эксперта в соответствующей теме, который сможет сказать, как именно лучше всего интерполировать данные (на основе специфичных математических моделей, которые, вероятно, существуют или могут подойти). Например, для каких-то признаков можно сгенерировать псевдослучайные значения, подчиняющиеся некому распределению с учетом других известных признаков. Сюда же можно отнести генерацию синтетических данных (то есть искусственных) на основании собственного понимания предметной области. Долгий и тщательный анализ данных может прояснить поведение скрытого параметра, а также выявить способы, как сэмулировать его с определенной достоверностью. Однако этот подход опасен тем, что неверное понимание данных приведет к тому, что мы заложим в данные другие закономерности и потом их же и найдем при помощи методов машинного обучения. Иными словами, вместо решения исходной задачи мы решим искусственно созданную.

В сложных случаях на практике применяется некоторая комбинация всех вышеперечисленных подходов. Какие-то объекты или столбцы полностью исключаются, потому что их слишком сложно интерполировать или сэмулировать. Как правило, это объекты или признаки с большим числом пропусков (например, где пропусков больше, чем реальных значений). Какие-то признаки помечаются

особым кодом, какие-то данные находятся извне, а какие-то эмулируются синтетическими данными.

Многие нюансы зависят от специфики задачи. Но если никакие идеи не работают, то, вероятно, данных просто недостаточно и нужно получать еще непосредственно исходные данные (проводить физические эксперименты, опрашивать клиентов, измерять какие-то статистические метрики поведения пользователей на веб-сайтах и тому подобное).

Замечание: кроме пропусков данные могут содержать ошибки и выбросы (аномальные отклонения), вызванные неверным заполнением данных или ошибкой измерения. Решение этих проблем выходит за рамки данного учебного пособия и рекомендуется к самостоятельному изучению.

Кодирование нечисловых признаков

Очевидно, что далеко не все признаки объектов естественно описываются численным значением. Если говорить о размерах объекта или стоимости какого-то товара, то эти признаки, несомненно, будут числовыми. Если же речь идет о цвете, типе товара (категории) или вообще о текстовом описании некоторого объекта, то подобные признаки, как правило, поступают неоцифрованными.

Нечисловые признаки с неупорядоченными значениями (в которых между значениями не определена дистанция, то есть нельзя сказать, что больше или меньше) называют категориальными, или номинальными. Типичный подход к их обработке – кодирование категориального признака с m возможными значениями с помощью m бинарных признаков. Каждый бинарный признак соответствует одному из возможных значений категориального признака и является индикатором того, что на данном объекте он принимает данное значение. Такой подход иногда называют one-hot-кодированием. Например, у нас есть три варианта материала изделия: дерево, пластик, сталь. Причем мы не знаем наверняка, что лучше, а что хуже и как это материал влияет на итоговое качество товара. Тогда вместо

того, чтобы закодировать набор матриалов в один столбец как значения 1, 2, 3, one-hot-кодирование даст три столбца и следующие коды: 001, 010, 100. Заметьте, что это не двоичное кодирование.

Если речь идет не просто о категориях, а о целых предложениях или больших текстах на естественном языке, то задача существенно усложняется. Кодирование текста побуквенно в большинстве случаев ничего не даст, так как разнообразие слов и смыслов настолько велико, что на таком уровне абстракции модель не построит нужную функцию. Обработка и понимание текста – это во многом направление для исследований.

Тем не менее на сегодняшний день существует ряд подходов, которые позволяют решать реальные задачи. Первый вариант – отнестись к тексту как к категориям. Допустим, в текстовых данных всего встречается 10 тыс. уникальных слов (не считая союзов, предлогов и тому подобное). Тогда каждое отдельное текстовое описание (присущее конкретному объекту) есть такая категория, где встречаются соответствующие слова, то есть каждый текст будет кодироваться в вектор из 10 тыс. элементов, где на месте соответствующих слов-элементов будут стоять единицы, а на месте слов, которых нет в данном тексте – нули. В итоге получим довольно разреженную матрицу (состоящую в основном из нулей) и при этом довольно большую. Из-за этого возникает проблема ее хранения и обработки. На сегодняшний день существует ряд техник оптимизации работы с разреженными матрицами как на фундаментальном уровне (например, сингулярное разложение), так и на уровне библиотек (например, в пакете scipy).

Однако учитывать все слова часто бывает избыточно и даже бессмысленно. Поэтому на практике применяется подсчет статистических характеристик текста, таких как TF-IDF. Подробнее о статистическом анализе текстов можно прочитать в [3].

Сегодня также актуально кодирование текста методом word2vec на основе машинного обучения. Ключевой особенностью метода является то, что большой массив слов отображается в вещественные

векторы небольшой размерности (100-200 элементов), причем, похожие слова имеют близкие друг к другу векторы (то есть вводится дистанция между словами).

Приведение данных к единому масштабу и стандартизация

«Сырые» данные имеют разный масштаб и разное распределение по каждому признаку. Например, какой-то химический показатель смеси может иметь значения в диапазоне от 0.0001 до 0.2, а другой показатель от –100 до 100. Или, скажем, возраст клиентов может быть от 16 до 40, причем гораздо больше клиентов имеет возраст от 18 до 25, иными словами, математическое ожидание смещено относительно центра распределения. Подобные различия в признаках могут вносить существенную ошибку для множества моделей (например, для регрессии, нейронных сетей), и потому требуется привести все признаки к единому виду.

Существует некоторая путаница в терминах «стандартизация» и «нормализация». Очень часто под стандартизацией и нормализацией понимаются разные вещи, а иногда стандартизацию рассматривают как часть нормализации. Поэтому важно понять общую суть и цель этих методов.

Стандартизация данных – это процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

Нормализация данных – это процесс масштабирования вектора каждого признака, то есть приведение его к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

Так, стандартизация матрицы X:

[[1., –1.,2.],
[2., 0., 0.],
[0., 1., –1.]],

даст следующий результат:

[[0. –1.22 1.34]
[1.22   0.   –0.27]
[–1.22 1.22 –1.07]]

Видно, что значения вектора сместились (выровнялись относительного единого центра в нуле), а также произошло выравнивание разброса. Уже такого преобразования достаточно, чтобы повысить качество данных. Однако видно, что значения разных векторов не будут в одинаковом диапазоне (от –1 до 1, например). Они будут лишь иметь стандартный разброс в рамках вектора\столбца\признака.

Для того чтобы достичь одинакового масштаба всех векторов, необходима нормализация. Есть несколько видов норм и, соответственно, нормализации.

Самый очевидный и простой метод – это max норма. Чтобы все значения лежали в одном диапазоне, нужно найти максимальное из возможных значений и все остальные поделить на него. Таким образом, максимальное значение будет единицей, а все остальные лягут в диапазон от 0 до 1. Но это при условии, что нет отрицательных значений.

Менее очевидный способ – это L1 норма и нормализация. Формула следующая:

$$x'_i = \frac{x_i}{\|x\|_1} = \frac{x_i}{\sum_j |x_j|},$$

где $\|x\|_1$ и есть L1 норма, а вся формула целиком отображает процесс нормализации вектора x.

Еще один способ – это L2 норма. Формула нормализации вектора x:

$$x'_i = \frac{x_i}{\|x\|_2} = \frac{x_i}{\left(\sum_j x_j^2\right)^{1/2}},$$

где $\|x\|_2$ и есть L2 норма, а вся формула целиком отображает процесс нормализации вектора x.

Исследователи говорят, что лучшие результаты дают L2 и max нормализация, хотя любой вариант лучше, чем использование исходных данных.

По поводу плюсов-минусов этих способов можно сказать следующее: max нормализация не дает запас на неизвестные новые значения (то есть если заранее неизвестен весь диапазон данных, лучше не использовать эту норму), а L2 норма вычислительно дольше, но чаще всего дает оптимальный результат. L1 норма может дать слишком большой запас при большом разбросе данных, что может удалить нужную информацию из данных.

Разметка данных

Разметка данных – последняя из рассматриваемых нами (по порядку, но не по важности) операция с данными. Если речь идет о том, чтобы обучить модель прогнозировать будущие показатели по прошлому опыту (например, прогноз средней выручки по количеству посетителей магазинов), то такие данные, как правило, приходят с известной целевой переменной Y (то есть средней выручкой). Эти данные необходимо будет обработать в соответствии с пунктами выше, но их не потребуется дополнительно размечать. Однако не редкость, когда специалисту по анализу данных необходимо будет самостоятельно размечать выборку. Эта потребность может возникнуть из-за отсутствия размеченных исходных данных, так и может быть обусловлена экспериментами, возникающими в ходе решения общей задачи. Например, для распознавания визуальных образов машин или светофоров потребуется создать специальный файл, в котором будут проставлены ассоциации образов по имени файлов (то есть 001.png – «светофор»). В реальном бизнесе эту задачу может решать и не специалист по анализу данных или машинному обучению, а другие сотрудники (например, со стороны заказчика), но важно отметить, что этот этап может также возникнуть в ходе разработки решения, и его нельзя избежать в случае обучения с учителем.

Однако стоит отметить, что в случаях обучения и без учителя может потребоваться частичная разметка данных с целью тестирования и оценки качества модели, так как в противном случае мы просто получаем «черный ящик» без понимания того, как мы решили поставленную задачу.

Переобучение

Нами были рассмотрены типовые задачи по подготовке данных, теперь рассмотрим одну из основных проблем алгоритмов машинного обучения – переобучение.

Мы помним, что одной из важных характеристик алгоритмов машинного обучения является обобщающая способность. Однако с ней связаны еще два понятия: недообучения и переобучения. При подготовке данного пункта использовался материал (в том числе иллюстративный) из источника [4].

Недообучение возникает при обучении по прецедентам и характеризуется тем, что алгоритм не дает удовлетворительно малой средней ошибки на обучающем множестве. Как правило, это явление появлется вследствие использования недостаточно сложных моделей.

Противоположное этому явлению – переобучение. Его суть состоит в том, что вероятность ошибки натренированного алгоритма на объектах тренировочной выборки оказывается существенно меньше, чем на объектах тестовой. Чаще всего переобучение появляется из-за использования слишком сложных моделей.

Рассмотрим график, иллюстрирующий эффект переобучения (рисунок 5).

Точки на графике с рисунка 5 соответствуют разным способам обучения, и каждая из них получена усреднением по большому числу разбиений исходной выборки объемом в 72 образца на тестовую и обучающую части. Как видно из рисунка, точки имеют постоянное смещение вверх относительно диагонали графика. То есть наблюдается эффект переобучения: ошибки на тестовой выборке появляются чаще, чем на обучающей.

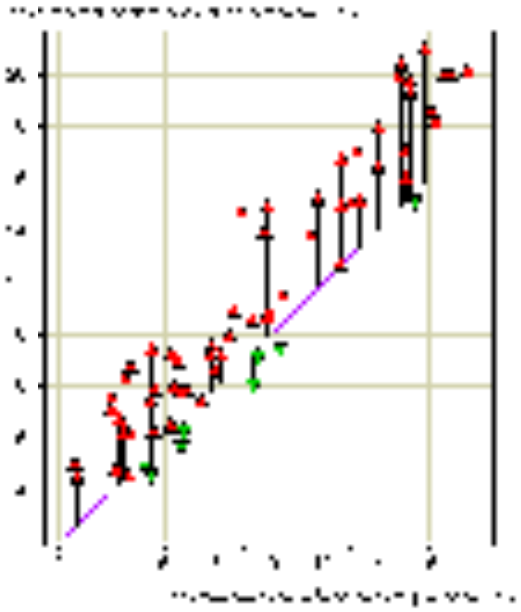


Рисунок 5. Иллюстрация переобучения

Чаще всего при построении алгоритмов обучения используется метод минимизации эмпирического риска (средней ошибки алгоритма на обучающей выборке). Его суть состоит в том, чтобы для текущей модели подобрать алгоритм, минимизирующий значение средней ошибки на данной обучающей выборке.

С переобучением метода минимизации эмпирического риска связаны три утверждения, объясняющие его причину:

1. Минимизация эмпирического риска не является гарантией малой вероятности ошибки на тестовых данных. Можно легко построить алгоритм, который минимизирует эмпирический риск до нуля, однако не будет способен к обучению. Суть состоит в том, что этот алгоритм запоминает обучающую выборку, потом сравнивает запомненный образец с предъявляемым. При совпадении предъявленного объекта и образца обучающей выборки алгоритм выдаст правильный ответ, иначе – выведется произвольный. То есть эмпирический риск равен нулю, однако обобщающей способности у алгоритма нет.

2. Согласно [4]: «Переобучение появляется именно вследствие минимизации эмпирического риска. Пусть задано конечное множество из D алгоритмов, которые допускают ошибки независимо и с одинаковой вероятностью. Число ошибок любого из этих

алгоритмов на заданной обучающей выборке подчиняется одному и тому же биномиальному распределению. Минимум эмпирического риска – это случайная величина, равная минимуму из D независимых одинаково распределенных биномиальных случайных величин, ожидаемое значение которой уменьшается с ростом D. Соответственно, с ростом D увеличивается переобученность – разность вероятности ошибки и частоты ошибок на обучении.

В данном модельном примере легко построить доверительный интервал переобученности, так как функция распределения минимума известна. Однако в реальной ситуации алгоритмы имеют различные вероятности ошибок, не являются независимыми, а множество алгоритмов, из которого выбирается лучший, может быть бесконечным. По этим причинам вывод количественных оценок переобученности является сложной задачей, которой занимается теория вычислительного обучения. До сих пор остается открытой проблема сильной завышенности верхних оценок вероятности переобучения».

3. Переобучение появляется в связи с избыточной сложностью модели. Всегда можно найти оптимальное значение сложности модели, при котором переобучение будет минимальным. Для примера приведем несколько графиков, на которых будет видна зависимость переобучения от сложности модели. При степени 2 полинома (рисунок 6) модель является недообученной. При степени 40 (рисунок 7) – переобученной и неустойчивой, а вот степень 20 (рисунок 8) – оптимальна.

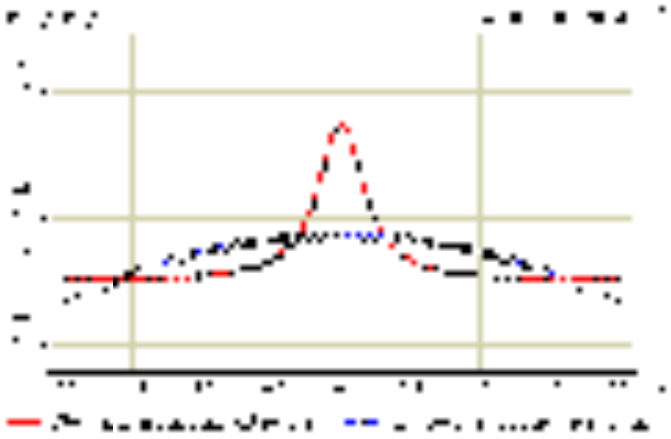


Рисунок 6. Недообученная модель

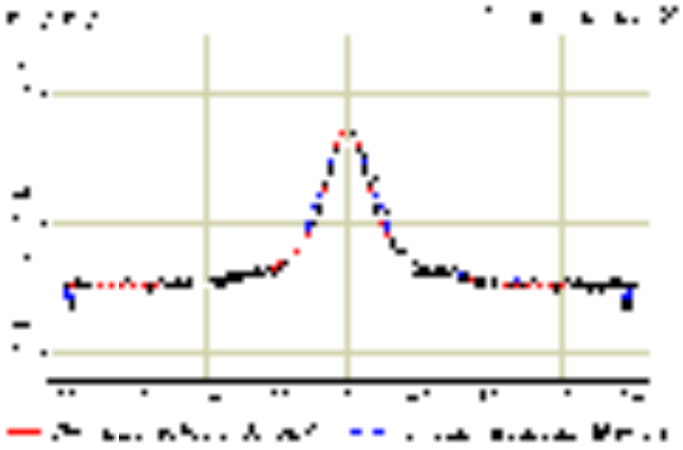


Рисунок 7. Оптимальная модель

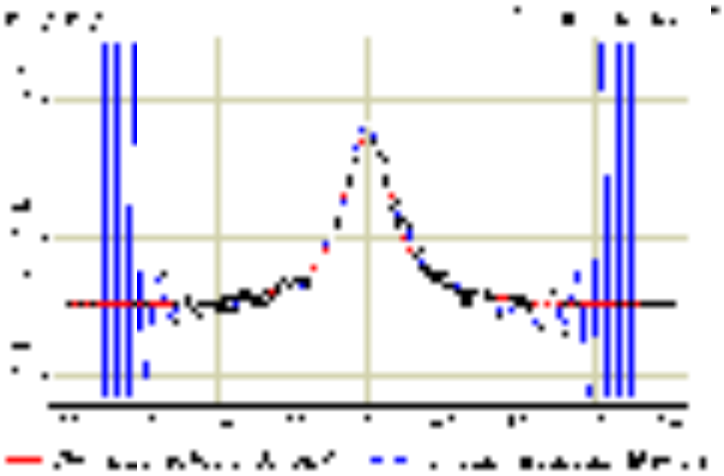


Рисунок 8. Переобученная модель

Одним из способов измерения вероятности переобучения является эмпирический метод Монте-Карло, или метод скользящего контроля. В литературе также встречаются названия кросс-проверка, или кросс-валидация. Суть его в следующем: производится некоторое количество разбиений исходной выборки на обучающую и контрольную. Для каждого разбиения происходит обучение алгоритма на обучающей подвыборке и оценка средней ошибки на контрольной. Затем вычисляется оценка скользящего среднего, как средняя по всем разбиениям величина вычисленной ошибки. Для независимой выборки этот показатель дает несмещенную оценку вероятности ошибки. Метод скользящего контроля является стандартным способом сравнения и оценивания алгоритмов классификации, регрессии, прогнозирования.

При использовании кросс-валидации можно сделать следующие выводы:

1. Если ошибка большая на большинстве участков, то скорее всего проблема в модели.

2. Если данные обучающей выборки характеризуются сильно смещенными математическим ожиданием и дисперсией, то уровень обобщения будет низким, что может быть связано с переобучением.

3. Если найдены сильные отклонения на определенных подвыборках, то, вероятно, проблема в этих участках данных или модель недообучается.

4. При малом объеме обучающей выборки кросс-валидация может стать способом борьбы с переобучением.

Если же говорить о прямых способах борьбы с переобучением, то можно выделить следующие:

1. Упрощение модели.

2. Подготовка большего числа обучающих данных (возможно, с помощью генерации).

3. Регуляризация.

Остановимся подробнее на последнем. Регуляризация представляет собой добавление некоторой дополнительной информации к условию минимизации ошибки. Выполняется это, чтобы решить некорректно поставленную задачу или предотвратить переобучение. Чаще всего добавляемая информация принимает вид штрафа за сложность модели. Например, введенные ограничения по норме векторного пространства или гладкости результирующей функции. Или же, с байесовской точки зрения, добавленные априорные распределения на параметры модели.

Согласно [5]: «Существуют следующие основные виды регуляризации:

L1-регуляризация (англ. Lasso regression):

$$\text{RSSR}(w) = \frac{1}{2}\sum_{i=1}^{n}(f(x_i, w) - y_i)^2 + \lambda\sum_{j=1}^{m}|w_j|,$$

где w – вектор весов полинома;

λ – коэффициент регуляризации.

L2, или Регуляризация Тихонова (в английской литературе – ridge regression или Tikhonov regularization), для интегральных уравнений позволяет балансировать между соответствием данным и маленькой нормой решения:

$$L(y, \hat{y}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \frac{\lambda}{2}\|w\|^2.$$

где $\|W\|^2$ – квадратичная норма вектора весов (сумма квадратов каждого веса).

Иными словами, переобучение в большинстве случаев проявляется в том, что в получающихся многочленах слишком большие коэффициенты. Соответственно и бороться с этим можно довольно естественным способом: нужно просто добавить в целевую функцию штраф, который бы наказывал модель за слишком большие коэффициенты».

По поводу применения той или иной регуляризации приведем материал источника [6]: «Регуляризацию можно применять с любым методом МО-классификации, который основан на математическом уравнении. Примеры включают линейную, логистическую регрессию и нейронные сети. Поскольку это уменьшает величину весовых значений в модели, регуляризацию иногда называют сокращением весов. Основное преимущество применения регуляризации в том, что оно часто приводит к созданию более точной модели. Главный недостаток заключается во введении дополнительного параметра, значение которого нужно определить, – весового значения регуляризации. В случае логистической регрессии это не слишком серьезно, так как в этом алгоритм обычно используется лишь параметр скорости обучения, но при использовании более сложного метода классификации, в частности нейронных сетей, добавление еще одного так называемого гиперпараметра может потребовать массы дополнительной работы для подбора комбинации значений двух параметров.

L1- и L2-регуляризация – процессы схожие. Какой же из них лучше? В принципе, исследователями сформулированы кое-какие теоретические правила насчет того, какая регуляризация лучше для определенных задач, но на практике придется поэкспериментировать, чтобы найти, какой тип регуляризации лучше в вашем случае и стоит ли вообще использовать какую-либо регуляризацию.

Применение L1-регуляризации иногда может давать полезный побочный эффект, вызывающий стремление одного или более весовых значений к 0.0, а это означает, что соответствующий признак не оказывает значимого влияния на результирующий, то есть включение его в модель требуется. Это одна из форм того, что называют «селекцией признаков». В отличие от L1, L2-регуляризация ограничивает весовые значения модели, но обычно не приводит к полному обнулению этих значений. Поэтому может показаться, что L1-регуляризация лучше L2-регуляризации. Однако недостаток применения L1-регуляризации в том, что этот метод не так-то просто использовать с некоторыми алгоритмами машинного обучения. Например, с теми, в которых используются численные методы для вычисления так называемого градиента. L2-регуляризацию можно использовать с любым типом алгоритма обучения.

Таким образом, можно сделать вывод, что L1-регуляризация иногда дает полезный побочный эффект удаления ненужных признаков, присваивая связанным с ними весам значение 0.0, но L1-регуляризация стабильно работает не со всеми формами обучения. L2-регуляризация работает со всеми формами обучения, но не обеспечивает неявной селекции функций. На практике же следует использовать метод проб и ошибок, чтобы определить, какая форма регуляризации (если она вообще нужна) лучше для конкретной задачи».

Рассмотренные методы борьбы с переобучением подойдут для регрессии или нейронных сетей. Для борьбы же с переобучением в моделях Деревьев Решений используют pruning (так называемое усечение) – удаление наименее информативных узлов для упрощения модели. Дело в том, что сразу нельзя построить оптимальное

(маленькое) дерево, так как в дереве малого размера будет мало информативных параметров, чтобы корректно обработать все входные образы (модель будет неполной, неинформативной). Поэтому сперва делают большое, но информативное дерево, а затем удаляют наименее информативные узлы и подтягивают дерево, уменьшая его размер\глубину.

Модели и алгоритмы машинного обучения

Как было сказано во введении, машинное обучение находится на стыке математической статистики, численных методов оптимизации, теории вероятностей и дискретного анализа. Эта дисциплина комбинирует и использует различные методы в построении математических моделей объектов и явлений для последующего их изучения и добычи знаний. В данном разделе предлагается рассмотреть, основные элементы математики, использующиеся в машинном обучении, а также изучить методы анализа данных, которые заложили основу рассматриваемой дисциплины.

Первое, что необходимо отметить, – любая дисциплина, использующая математический аппарат, так или иначе занимается математическим моделированием – заменой реального объекта его абстрактным, идеализированным представлением и использованием полученного представления для изучения объекта и добычи знаний. То есть основное, с чем работает машинное обучение, – это математические модели. Их можно разделить на различные типы по некоторым признакам. С точки зрения математического вида функции, составляющей модель, выделяют линейные и нелинейные модели. По количеству переменных в модели они делятся на сосредоточенные и распределенные. С точки зрения учета случайности модели делятся на детерминированные и стохастические, а с точки зрения изменчивости во времени – на статические и динамические. По природе используемых в модели параметров и переменных их можно разделить на дискретные и непрерывные.

В общем виде математическая модель может быть представлена следующим образом:

$$Y=F(X,W_c,W_d,W_s),$$

где Y – выходные данные; X – входные данные; F – некоторая функция; W_c – константные параметры модели; W_d – динамические параметры модели; W_s – статические параметры модели.

При построении моделей элементы множеств *W*, как правило, заранее неизвестны и требуют нахождения. Есть два способа решения этой задачи: аналитический и численный. Первый предполагает нахождение корней, второй – приближение результата к некоторому удовлетворительному значению. Удовлетворительность в данном случае оценивается некоторым заранее определенным критерием. При этом для решения задачи численным методом нередко происходит ее переформулировка для определения и введения условия, определяющего качество решения. Необходимо отметить, что численные методы чаще всего опираются на отклонение получаемого результата от желаемого и стремятся минимизировать это отклонение. Причем данный подход будет работать, даже если неизвестна точная формула для получения выходного значения, однако обязательно существует способ измерения отклонения, а также можно найти значения, которые он должен принимать.

Появление численных методов оптимизации было обусловлено, во-первых, тем, что не все задачи можно решить аналитически. Классический пример подобной задачи – гравитационная задача *N* тел. Согласно Википедии [1], ее формулировка звучит так: «В пустоте находится *N* материальных точек, массы которых известны $\{m_i\}$. Пусть попарное взаимодействие точек подчинено закону тяготения Ньютона, и пусть силы гравитации аддитивны. Пусть известны начальные на момент времени $t=0$ положения и скорости каждой точки $\mathbf{r}_i|_{t=0} = \mathbf{r}_{i0}$, $\mathbf{v}_i|_{t=0} = \mathbf{v}_{i0}$. Требуется найти положения точек для всех последующих моментов времени». И согласно тому же источнику [1]: «На данный момент в общем виде задача *N* тел для $N>3$ может быть решена только численно, причем для $N=3$ ряды Зундмана даже при современном уровне компьютеров использовать практически невозможно».

Во-вторых, даже возможные аналитические решения систем могут обладать существенной сложностью. Матричные решения имеют вычислительную сложность *O(N^3)*, тогда как численные – линейную.

В-третьих, при противоречивости или мультиколлинеарности данных аналитические решения могут давать неопределенный результат. В отличие от них численные методы всегда сходятся пусть и к локальному, но все же определенному решению.

В основном, при решении задач минимизации ошибки, в машинном обучении используются градиентные методы, составляющие особый класс алгоритмов оптимизации. Градиент в данном случае – это (согласно Википедии [1]): «вектор, своим направлением указывающий направление наибольшего возрастания некоторой величины φ, значение которой меняется от одной точки пространства к другой (скалярного поля), а по величине (модулю) равный скорости роста этой величины в этом направлении.

Например, если взять в качестве φ высоту поверхности земли над уровнем моря, то ее градиент в каждой точке поверхности будет показывать «направление самого крутого подъема», и своей величиной характеризовать крутизну склона.

С математической точки зрения на градиент можно смотреть как на:

1. Коэффициент линейности изменения значения функции многих переменных от изменения значения аргумента.

2. Вектор в пространстве области определения скалярной функции многих переменных, составленный из частных производных.

3. Содержимое Матрицы Якоби. Ее строки содержат градиенты составных скалярных функций, из которых состоит векторная функция многих переменных.

Пространство, на котором определена функция и ее градиент, может быть как обычным трехмерным пространством, так и пространством любой другой размерности, любой физической природы или чисто абстрактным (безразмерным)».

Метод градиентного спуска – нахождение локального экстремума (минимума или максимума) функции путем движения вдоль градиента в направлении наискорейшего спуска, задаваемого антиградиентом. Существуют следующие типы градиентного спуска:

1. С постоянным шагом.
2. С дроблением шага.
3. Наискорейшего спуска.
4. Стохастический.

Более подробно о градиентном спуске мы поговорим далее, здесь же рассмотрим аналитические методы, используемые в машинном обучении, так как именно они составляют первое поколение алгоритмов, применяемых для анализа данных.

Методы теории вероятностей

Первый срез методов, который необходимо рассмотреть, – методы теории вероятностей. Теория вероятностей – раздел математики, в котором изучаются случайные величины и события, их свойства и возможные операции над ними. Таким образом, ключевое понятие, лежащее в основе этой дисциплины, – вероятность события P_i. Опираясь на него, можно дать определение полной группы событий. Она определяется как система случайных событий, которая обладает следующими свойствами:

1. В результате случайного эксперимента непременно произойдет одно и только одно из составляющих ее событий.
2. Сумма вероятностей всех событий полной группы равна 1.

Достаточно часто при исследовании данных (выборок) и подсчете определенных характеристик выборки считаются полными группами событий.

С помощью методов теории вероятностей можно проводить как простые, так и сложные операции по анализу данных. Среди простых можно выделить подсчеты вероятностных характеристик выборки: медианы, математического ожидания, дисперсии, а также величины среднеквадратического отклонения. Рассмотрим их подробнее.

Медиана – число, характеризующее выборку по среднему из ее значений. То есть, если все данные выборки различны, и она упорядочена по возрастанию, то ровно половина из элементов выборки

будет меньше медианы, и ровно половина – больше. Формально величину можно выразить следующим образом:

Если $X=\{x_{1\ldots}x_n\}$ – характеризуемая выборка, то x_k – медиана, если $x_j<x_k$, при $j \in [1,k)$ и $x_k<x_i$, при $i \in (k,n]$ и $k=n/2$.

Рассмотрим пример. Пусть выборка X = {1, 2, 3, 4, 5, 6, 7, 8, 9}, тогда ее медианой будет число 5.

Математическое ожидание – среднее значение вероятностных элементов выборки. Формально она рассчитывается так:

$$M|X|=\sum x_i p_i \ .$$

Рассмотрим пример. Пусть выборка X = {1, 2, 3, 4, 5, 6, 7, 8, 9}, тогда ее математическим ожиданием будет величина, равная 5. Вычисляется она следующим образом: выборка рассматривается как полная группа событий с равными вероятностями. То есть p_i для каждого элемента равно 0,11. Тогда 1×0,11+2×0,11+3×0,11…=5.

Дисперсия – мера разброса элементов выборки относительно ее математического ожидания. Рассчитывается данная величина по формуле

$$D|X|=\sum(x_i - M|X|)^2 p_i \ .$$

При этом p_i рассчитывается как $1/(n-1)$, где n – количество элементов выборки. Это выполняется для того, чтобы не учитывать отклонение самого математического ожидания от себя. То есть, для выборки X = {1, 2, 3, 4, 5, 6, 7, 8, 9} дисперсия будет рассчитываться следующим образом:

(1–5) ×(1–5) ×0,125+(2–5) ×(2–5) ×0,125+(3–5) ×(3–5) ×0,125…=7,5.

Среднеквадратическое отклонение – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания. Формула расчета

$$\sigma = \sqrt{\frac{1}{n}\sum(x_i - \bar{x})^2},$$

где n – количество элементов в выборке; $\bar{x}$ – математическое ожидание.

Интерпретировать данную величину можно следующим образом: чем больше значение среднеквадратического отклонения, тем

больший разброс значений в представленном множестве (относительно его средней величины). Малое значение среднеквадратического отклонения говорит о том, что элементы выборки сгруппированы вокруг ее среднего значения. Если говорить о более практическом применении величины, то в экономике, например, она характеризует доходность портфеля и его риск.

Однако рассмотренные показатели характеризуют случайную величину лишь с какой-то одной стороны. Наиболее же полно и исчерпывающе ее описывает закон распределения – функция, определяющая для выборки X вероятность попадания в некоторый интервал или вероятность получения определенного значения x_i. Если какой-либо закон распределения описывает случайную величину, то говорят, что она ему подчиняется или по нему распределена. Иными словами, закон распределения описывает область значений случайной величины и вероятности их получения.

Среди законов распределения наиболее часто используется закон нормального распределения или распределения Гаусса, который характеризует большинство процессов, встречающихся в мире. Отсюда и произошло его название (нормальное). Закон (плотность распределения вероятности) описывается следующей формулой:

$$\boldsymbol{f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}},$$

где σ – среднеквадратическое отклонение; μ – математическое ожидание.

С математической точки зрения можно заметить, что данная функция зависит от двух параметров: математического ожидания и среднеквадратического отклонения, поэтому по сути данный закон представляет собой семейство распределений. В качестве примера приведем изображение из источника [7], показывающее графики нормальных распределений с разными параметрами (рисунок 9).

Обратите внимание!

- *График с параметрами σ2=1 и μ=0 соотвествует стандартному нормальному распределению.*

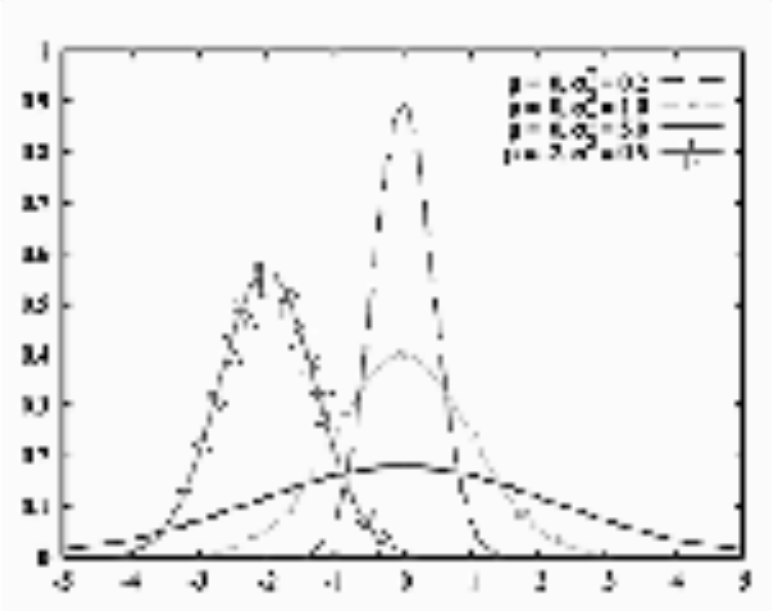


Рисунок 9. Плотность нормального распределения

Согласно Википедии [1]: «Важное значение нормального распределения во многих областях науки (например, в математической статистике и статистической физике) вытекает из центральной предельной теоремы теории вероятностей. Если результат наблюдения является суммой многих случайных слабо взаимозависимых величин, каждая из которых вносит малый вклад относительно общей суммы, то при увеличении числа слагаемых распределение центрированного и нормированного результата стремится к нормальному. Этот закон теории вероятностей имеет следствием широкое распространение нормального распределения, что и стало одной из причин его наименования».

Кроме описанных выше величин и характеристик для анализа данных очень часто используется одна из основных теорем теории вероятностей – теорема Байеса.

Она позволяет определить вероятность наступления какого-либо события при условии, что произошло другое событие, статистически

взаимосвязанное с исследуемым. Ключевыми понятиями в данной теореме являются понятия априорной и апостериорной вероятностей. Рассмотрим их подробнее.

Априорная вероятность – назначенная событию вероятность, при условии отсутствия знаний, поддерживающих его наступление. Апостериорная вероятность – назначенная событию вероятность при условии наличия знаний, поддерживающих его наступление и полученных опытным путем.

Теперь перейдем непосредственно к теореме Байеса. Формулируется она следующим образом:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

где $P(A|B)$ – апостериорная вероятность справедливости гипотезы A при наступлении B; $P(B|A)$ – вероятность наступления события B при истинности гипотезы A; $P(A)$ – априорная вероятность гипотезы A; $P(B)$ – вероятность наступления события B.

Метод Байеса имеет свои достоинства и недостатки. К первому можно отнести возможность использования экспертных знаний, возможность точного описания явления и хорошую интерпретируемость результатов. К недостаткам же относится следующее:

1. Метод не ставит целью минимизацию ошибки классификации.
2. Метод требует работы эксперта.
3. Сильная зависимость результатов от выбора модели.
4. Плохая работа при малом количестве и высокой размерности данных.
5. Метод дает плохое обобщение, особенно на высокоуровневых признаках.
6. Метод дает плохие результаты при взаимозависимости признаков.

Таким образом, можно сделать вывод, что данный метод хорошо работает в тех случаях, где признаки и результат сильно завязаны на их частотных характеристиках.

Деревья решений

Следующий метод анализа данных, который нам необходимо рассмотреть, – применение деревьев решений. Дерево решений – средство поддержки принятия решений для прогнозных моделей. Суть его работы заключается в последовательном разбиении множества данных на непересекающиеся классы, которые в свою очередь также подвергаются разбиению по каким-либо критериям с оценкой эффективности разбиения. Как правило, дерево решений состоит из «узлов», «листьев» и «веток». «Ветки» содержат записи атрибутов, от которых зависит целевая функция, «листья» – значения целевой функции, а «узлы» – остальные атрибуты, по которым происходит классификация. Чаще всего выделяют два типа деревьев: для классификации (в этом случае предсказываемый результат – класс, которому принадлежат данные) и для регрессии (результат – прогнозируемое значение целевой функции).

Обобщенный алгоритм построения дерева решений по обучающей выборке состоит из следующих шагов:

1. Берем следующий атрибут и помещаем его в корень.
2. Для всех значений этого атрибута – оставляем в «листьях» данной «ветки» только те значения, которые соответствуют определенному условию.
3. Продолжаем строить дерево среди оставленных на предыдущем шаге «листьев».

Для выбора следующего атрибута может быть использован один из следующих основных алгоритмов:

1. ID3. Атрибут выбирается на основе прироста информации и минимизации энтропии. Напомним, что под энтропией в данном случае подразумевается мера неупорядоченности системы. Чем меньше ее величина, тем более упорядочены составляющие системы.
2. C4.5 – улучшенная версия предыдущего алгоритма, в которой используется нормализованный прирост информации.

3. CART – алгоритм, используемый для построения бинарных деревьев, производящий разбиение на основе модальных значений признаков.

Рассмотрим работу по построению дерева решений с использованием первого алгоритма. Более подробно об этом можно прочитать в [8, 9].

Кратко алгоритм ID3 может быть описан тремя шагами:

1. Посчитать энтропию разбиваемого множества.
2. Выбрать признак с минимальной энтропией и, соответственно, максимальной информационной выгодой.
3. На основе полученного признака создать узел дерева и повторить процедуру.

Рассмотрим построения дерева решений для классификации следующего множества из семи объектов: {красный круг, зеленый квадрат, красный квадрат, зеленый треугольник, зеленый круг, красный треугольник, красный прямоугольник}. Как мы видим, каждый из объектов характеризуется двумя признаками: цвет и форма, то есть именно по ним мы будем проводить разбиение. Для меры энтропии выберем формулу энтропии Шеннона:

$$H = -\sum_{i=1}^{N} p_i \times \log p_i$$

В таблице 2 представлены расчеты энтропии всей системы. Подсчеты энтропии для каждого из признаков приведены в таблицах 3 и 4, соответственно.

Таблица 2. Подсчет энтропии системы

Объект	**pi**	**log(pi)**	**pi*log(pi)**
красный квадрат	0,142857	-0,8451	-0,12073
красный прямоугольник	0,142857	-0,8451	-0,12073
красный круг	0,142857	-0,8451	-0,12073
зеленый квадрат	0,142857	-0,8451	-0,12073
зеленый треугольник	0,142857	-0,8451	-0,12073
зеленый круг	0,142857	-0,8451	-0,12073
красный треугольник	0,142857	-0,8451	-0,12073
Энтропия			0,845098

Таблица 3. Подсчет энтропии для признака «Цвет»

Объект	pi	log(pi)	pi*log(pi)
красный	0,571429	-0,24304	-0,13888
зеленый	0,428571	-0,36798	-0,1577
Энтропия			0,296583

Таблица 4. Подсчет энтропии для признака «Форма»

Объект	pi	log(pi)	pi*log(pi)
круг	0,285714	-0,54407	-0,15545
квадрат	0,285714	-0,54407	-0,15545
треугольник	0,285714	-0,54407	-0,15545
прямоугольник	0,142857	-0,8451	-0,12073
Энтропия			0,587072

Как мы видим, первым признаком, на основе которого будет происходить разбиение, станет цвет, а вторым – форма. Классификация исходного множества с помощью построенного дерева показана на рисунке 10.

Рисунок 10. Классификация фигур с помощью дерева

Стоит отметить, что чаще всего деревья решений – бинарные. В узлах они содержат условия, а ветви соответствуют истинности или ложности этого условия. Поэтому, если дерево с рисунка 10 преобразовать в классическое дерево решений, то получится изображение, представленное на рисунке 11.

Теперь рассмотрим использование этого метода для решения задачи из источника [10]: спрогнозируем исход матча для футбольной команды исходя из следующих параметров:

- выше ли находится соперник по турнирной таблице;
- дома ли играется матч;
- пропускает ли матч кто-либо из лидеров команды;
- идет ли дождь.

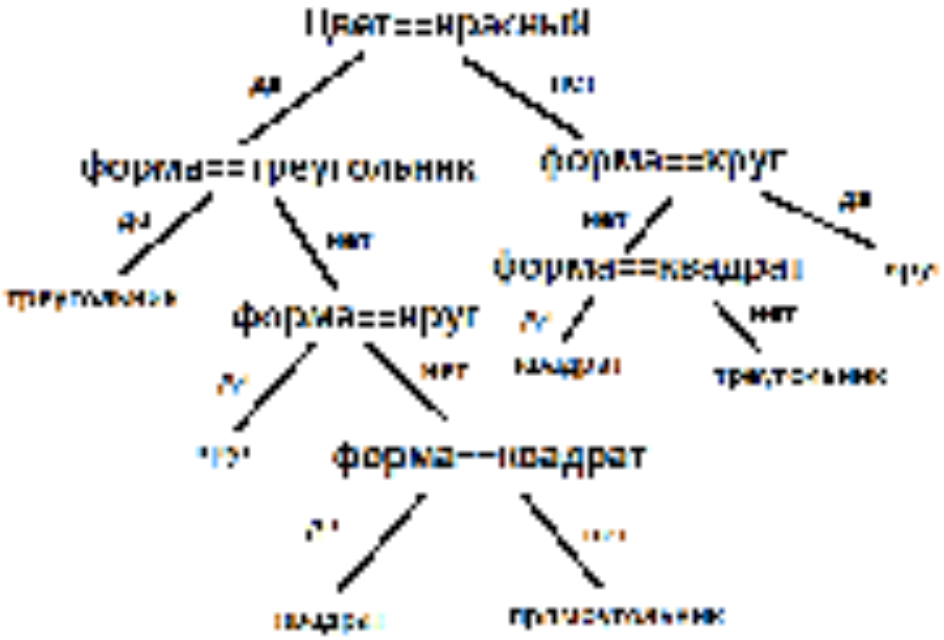


Рисунок 11. Классическое дерево решений

Прогноз предлагается выполнить на основе статистики, представленной в таблице 5.

Таблица 5. Статистические данные для задачи

Соперник	**Играем**	**Лидеры**	**Дождь**	**Победа**
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	???

Первое, что мы рассчитываем, – энтропию для признаков. Результаты можно увидеть в таблице 6, причем данные приведены в округленном виде.

Таблица 6. Энтропия для признаков

Соперник	Играем	Лидеры	Дождь	Победа
0,29	0,26	0,29	0,29	0,29

Согласно таблице 6, первый признак, по которому будет происходить деление, – место игры. По нашим данным получатся два множества: со значением «дома» и со значением «в гостях». Если мы рассмотрим второе множество, то увидим, что следующий целевой признак с нулевой энтропией – исход матча: все элементы множества «в гостях» имеют значение поля победа «нет». Следовательно, по нашим статистическим данным с помощью деревьев решений мы прогнозируем поражение.

Если говорить о достоинствах деревьев решений, то можно выделить следующие. Во-первых, простота понимания и интерпретации. Во-вторых, минимальные требования к подготовке данных, а также способность работы с большими объемами данных. В-третьих, метод одинаково хорошо работает с разными видами признаков. В-четвертых, является надежным методом и позволяет оценить модель статистическими тестами. К недостаткам же можно отнести следующее:

1. Достаточно сложно построить оптимальное дерево решений.
2. Подверженность переобучению.
3. Не для всех задач может быть получено решение удовлетворительного качества.

Статистические модели и методы

Мы рассмотрели методы машинного обучения, предоставляемые теорией вероятностей, а также модель дерева решений. Теперь обратимся к статистике и кратко опишем ее модели и методы, которые могут быть использованы для решения задач машинного обучения, а именно методы регрессионного анализа, подробно рассмотренные в следующих источниках: [11,12,13].

В контексте машинного обучения под регрессионным анализом понимается процесс построения математической модели, описывающей зависимость некоторой целевой характеристики объекта или процесса от других его характеристик. Например, зависимость числа новых клиентов от величины зарплаты работающего на улице промоутера.

В задаче регрессионного анализа всегда есть обучающая выборка, состоящая из входных параметров и откликов, а также начальная параметрическая модель, в самом простом случае – линейная, однако не обязательно таковая. Для задачи из примера эта модель может иметь вид $y=\beta_0+\beta_1\times x$, где x – размер зарплаты промоутера; y – количество новых клиентов; β_0 и β_1 – параметры модели. Задача регрессии – оценить их, то есть найти такие значения β_0 и β_1, чтобы полученная модель отражала зависимость между входом и выходом с требуемой точностью.

После получения адекватной модели мы можем решать задачу прогнозирования, подставляя в полученную формулу величину x и вычисляя величину y (с удовлетворяющей нас погрешностью). Приведенный выше пример модели – модель парной линейной регрессии, но помимо нее существует и множественная, и нелинейная регрессии. Начнем рассмотрение с самого простого.

Допустим, мы хотим описать зависимость между двумя факторами моделью вида $y=\beta_0+\beta_1\times x$. Первое, что необходимо учесть, – построенная линия никогда не будет точно проходить по опытным точкам, поэтому истинный вид регрессионной модели будет $y=\beta_0+\beta_1\times x+e$, где e – ошибки наблюдений. Второе – прежде чем переходить к оцениванию параметров модели, целесообразно построить диаграмму рассеяния, чтобы убедиться, что выбранная модель действительно может описать зависимость между факторами. На диаграмме рассеяния каждой паре «зависимый-влияющий параметр» соответствует точка на плоскости. Как правило, зависимый фактор откладывается по оси ординат, а второй – по оси абсцисс. Модель парной линейной регрессии графически представляет собой линию, следова-

тельно, использовать ее для описания зависимости целесообразно, если на диаграмме рассеяния точки располагаются вокруг (и достаточно близко) какой-либо прямой. Отклонение реальных точек от модельных – остатки или ошибки наблюдений, которые обозначаются *e*. Пример диаграммы рассеяния показан на рисунке 12.

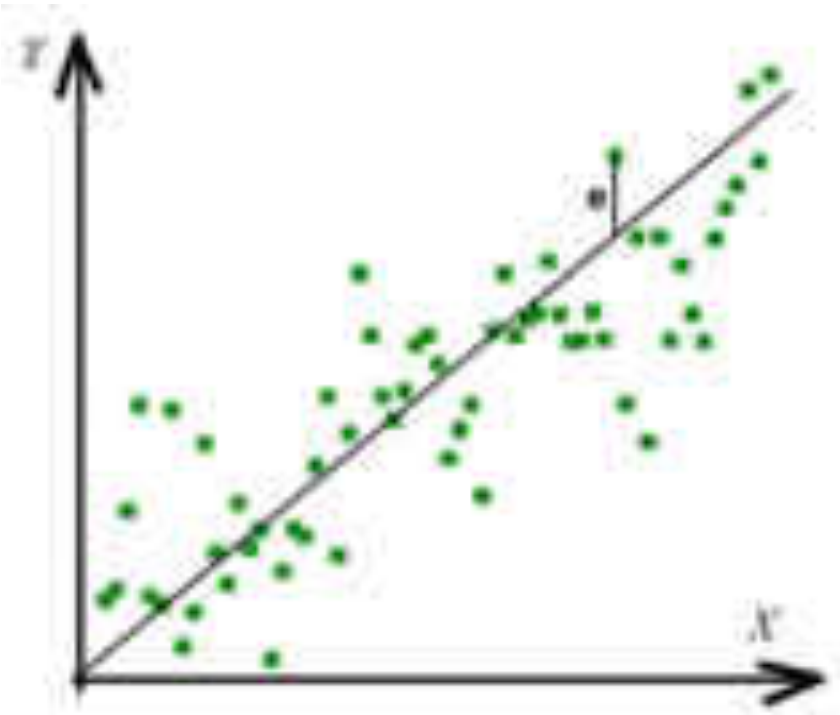


Рисунок 12. Диаграмма рассеяния

Если после анализа диаграммы рассеяния принимается решение использовать линейную парную регрессию для моделирования зависимости, то следующим шагом будет нахождение параметров модели β_0 и β_1. Для решения этой задачи в девяносто девяти процентах случаев используется метод наименьших квадратов, предложенный Гауссом более двухсот лет назад. Суть данного метода заключается в минимизации суммы квадратов отклонений опытных данных от модельных. Формально эта задача описывается так:

$$Q = \sum e_i^2 \rightarrow min$$

В данном случае e_i вычисляется следующим образом:

$$e_i = \beta_0 + \beta_1 * x_i - y_i,$$

где y_i – реальное выходное значение для входного значения x_i.

Чтобы решить указанную задачу оптимизации, решают систему уравнений:

$$\left.\begin{aligned}\frac{\partial Q}{\partial \beta_0} &= 0\\ \frac{\partial Q}{\partial \beta_1} &= 0\end{aligned}\right\}$$

Найдя необходимые частные производные и выполнив преобразования по упрощению выражений, получают нормальную систему уравнений для парной линейной регрессии:

$$\left.\begin{aligned} n * \beta_0 + \beta_1 * \sum x_i - \sum y_i &= 0\\ \beta_0 * \sum x_i + \beta_1 * \sum x_i^2 - \sum x_i y_i &= 0\end{aligned}\right\}$$

Рассмотрим для примера построение парной линейной регрессии для задачи зависимости ежемесячного количества новых клиентов от почасовой зарплаты промоутера по данным, представленным в таблице 7.

Таблица 7. Данные для анализа

Зарплата (*x*)	100	120	130	150
Количество клиентов (*y*)	70	100	120	140

Подставим данные в систему уравнений и получим следующий ее вид:

$$\left.\begin{aligned} 4 \times \beta_0 + \beta_1 \times 500 - 430 &= 0\\ \beta_0 \times 500 + \beta_1 \times 63800 - 55600 &= 0\end{aligned}\right\}$$

Выразим β_0 из первого уравнения:

$$\beta_0 = 107,5 - \beta_1 \times 125$$

Подставим во второе и найдем β_1:

$$53750 - 62500 \times \beta_1 + \beta_1 \times 63800 - 55600 = 0$$

$$\beta_1 = \frac{1850}{1300}.$$

В итоге получим уравнение регрессии:

$$y = 1,423\,1x - 70,385\ .$$

Полученные коэффициенты можно интерпретировать следующим образом. β_0 – значение при x=0. То есть, если у нас не будет работать промоутер (то есть мы не будем платить ему зарплату), ежемесячно мы прогнозируем отток 70 клиентов. Содержательная интерпретация второго коэффициента следующая: каждый рубль в зарплате промоутера дает 1,4 нового клиента в месяц.

Необходимо отметить, что коэффициенты для парной линейной регрессии можно найти средствами табличного процессора Excel. Для этого строится обычная точечная диаграмма, а затем отображается линия тренда и уравнение, как показано на рисунке 13.

Рисунок 13. Работа с парной линейной регрессией в Excel

Там же можно отобразить величину достоверности аппроксимации (или коэффициент детерминации R^2), которая показывает, насколько модельные значения близки к реальным. Чем ближе эта величина к 1, тем лучше модель описывает реальность. То есть, по сути, это критерий качества модели. Рассчитывается он по формуле

$$R^2 = \frac{\sum \tilde{y}_i^2 - n\bar{y}^2}{\sum y_i^2 - n\bar{y}^2},$$

где $\bar{y}=\frac{\sum y_i}{n}$ – среднее значение реальных результатов, $\tilde{y}$ – прогнозное значение.

Областью значения данной величины будет отрезок от 0 до 1.

Кроме расчета величины достоверности аппроксимации есть еще несколько способов оценить качество регрессионной модели. Однако в данном пособии не будем подробно останавливаться на этих методах, а лишь перечислим их:

1. Анализ диаграммы рассеяния;
2. Проверка статистических гипотез о значимости модели;
3. Анализ остатков.

По третьему пункту поясним: метод наименьших квадратов справедлив, если остатки обладают следующими свойствами:

1. Нормальным распределением.
2. Взаимонезависимостью.
3. Нулевым математическим ожиданием.
4. Постоянной дисперсией.

То есть, если остатки разбросаны хаотично, то метод наименьших квадратов можно использовать для решения поставленной задачи.

Если не хаотично, то нужно смотреть: если остатки растут при росте *x*, то есть их график напоминает диаграмму рассеяния, показанную на рисунке 12, это говорит о гетероскедастичности остатков. В данном случае необходимо использовать взвешенный метод наименьших квадратов. Если график остатков похож на отрезок параболы, то скорее всего была выбрана неправильная модель и необходимо выбрать другую. Если график остатков напоминает, например, синусоиду, то есть они циклически колеблются вверх-вниз, то это свидетельствует об автокорреляции остатков, которая может быть оценена количественно с помощью критерия Дарбина-Уотсона. В этом случае метод наименьших квадратов неприменим, и необходимо использовать другие подходы.

Как было сказано выше, парная линейная регрессия не всегда хорошо описывает данные. Тогда зависимость можно попробовать смоделировать либо кусочно-линейной моделью, то есть построить несколько разных линейных моделей для разных диапазонов значений величины *x*, либо использовать нелинейные модели: гиперболическую, параболическую, степенную и обратную.

В общем случае при втором подходе оптимизационная задача формулируется так:

$$Q = \sum(f(x_i) - y_i)^2 \to min,$$

где $f(x_i)$ – некоторая нелинейная функция.

Для ее решения можно воспользоваться, например, методом Ньютона-Рафсона. Но в некоторых случаях при работе с парной нелинейной регрессией используют прием преобразования модели к линейному виду.

Рассмотрим способ преобразования более подробно. Например, мы выбрали гиперболическую модель, задаваемую уравнением вида

$$y = \beta_0 + \beta_1/x\ .$$

Тогда, если мы введем замену z=1/*x*, то модель преобразуется к привычному нам линейному виду.

Для параболической модели вида

$$y = \beta_0 + \beta_1 \times x + \beta_2 * x^2$$

замена не выполняется. При ее решении используется подход, аналогичный случаю линейной модели, но в итоговой системе будет три уравнения.

Обратная модель вида

$$y = 1/(\beta_0 + \beta_1 * x)$$

приводится к линейной через замену z=1/y.

Степенная модель вида

$$y = \beta_0 \times x^{\beta_1}$$

приводится к линейной логарифмированием:

$$\ln(y) = \ln(\beta_0) + \beta_1 \ln(x).$$

Вводя замены, получаем уравнение:

$$z = \beta_0{}' + \beta_1 \times t.$$

Однако, выполняя подобные преобразования, мы рискуем исказить взаимосвязи и понизить адекватность модели, поэтому линеризацию нужно применять крайне аккуратно. К тому же данный подход работает не для всех моделей. В таком случае, как было сказано выше, требуется использовать численный метод Ньютона-Рафсона. Опишем кратко его идею, не вдаваясь в подробности. Поиск решения в нем происходит посредствам построения последовательных приближений, то есть на основе простой итерации. Первым шагом задается стартовое приближение около предполагаемого корня. Затем в точке приближения строится касательная к графику функции, для которой находится пересечение с осью абсцисс. Эта точка считается следующим приближением, для которого повторяется процесс. Алгоритм продолжает свою работу до тех пор, пока не будет достигнута необходимая точность.

Мы рассмотрели различные модели парной регрессии, то есть случаи, когда отклик зависит от одного фактора. Однако в некоторых задачах на выходную переменную (отклик) влияет несколько факторов. В таком случае эта зависимость описывается множественной регрессией. Чаще всего в этом случае используются линейные модели. Общий вид зависимости описывается следующим образом:

$$y_i = \beta_0 + \beta_1 \times x_{1i} + \cdots + \beta_k \times x_{ki} + \varepsilon_i,$$

где $i=1...n$, n – количество наблюдений; k – количество факторов.

При работе с множественной регрессией, как правило, переходят к матричной записи системы:

$$Y=X\times\beta+\varepsilon,$$

где $Y = \begin{pmatrix} y_1 \\ ... \\ y_n \end{pmatrix}$ – вектор откликов; $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ ... \\ \varepsilon_n \end{pmatrix}$ – вектор ошибок;

$\beta = \begin{pmatrix} \beta_1 \\ ... \\ \beta_n \end{pmatrix}$ – вектор параметров; $X = \begin{pmatrix} 1 & x_{11} \cdots & x_{k1} \\ 1 & x_{12} \cdots & x_{k2} \\ \cdots & \cdots \cdots & \cdots \\ 1 & x_{1n} \cdots & x_{kn} \end{pmatrix}$ – регрессионная матрица; n – количество наблюдений; k – количество факторов.

В данном случае задача формулируется следующим образом: для известных X и Y необходимо найти такие β, чтобы $Q=\sum \varepsilon_i^2 \rightarrow \min$.

В матричном виде последнее выражение можно переписать так:

$$Q = \varepsilon^T \varepsilon.$$

При этом

$$\varepsilon = Y - X\beta.$$

Тогда аналогом нормальной системы уравнений парной линейной регрессии в нашем случае будет следующее выражение:

$$\tilde{\beta} = (X^T X)^{-1} X^T Y.$$

Однако при работе с этой формулой могут возникнуть проблемы, связанные с тем, что обратную матрицу $(X^T X)^{-1}$ сложно посчитать. Как правило, это происходит, если ее столбцы сильно коррелированы. Например, если столбцы практически линейно зависят один от другого (имеет место мультиколлинеарность). Если коэффициент корреляции столбцов будет больше 0,8, то обратную матрицу уже не посчитать. Для решения этой проблемы либо исключают один из взаимозависимых факторов, либо прибегают к использованию гребневой регрессии (Ridge regression). Ее суть состоит в том, что для решения проблемы плохой обусловленности матрицы $(X^T X)$ к ней добавляется некое число λ. То есть обращается матрица $(X^T X + \lambda I)$. Полученные с помощью гребневой регрессии оценки параметров являются смещенными (в отличие от МНК-оценок), но доказано, что существует такое λ при котором оценки гребневой регрессии более эффективны, чем МНК-оценки. Однако четких рекомендация относительно выбора числа λ нет.

Продолжая тему проблем множественной регрессии, необходимо отметить еще одну. В случае непарной регрессии диаграмма рассеяния не применима для отслеживания выбросов, так как пространство будет многомерно. Для решения этой проблемы используют робастные методы. Основную информацию о них можно найти в источнике [14].

Если же говорить об общем алгоритме работы с множественной регрессией, то можно выделить следующие шаги:

1. Подготовка исходных данных, как правило, в виде таблицы (см. таблицу 8):

Таблица 8. Шаблон подготовки данных

№ опыта	***y***	x_1	x_2	**…**	x_k
1					
…					
n					

2. Оценивание параметров модели по формуле

$$\tilde{\beta} = (X^T X)^{-1} X^T Y,$$

или при использовании гребневой регрессии – по формуле

$$\tilde{\beta} = (X^T X + \lambda I)^{-1} X^T Y).$$

3. Проверка значимости модели с использованием критерия Фишера. Если по критерию Фишера линейная модель множественной регрессии оказалась незначима, то можно перейти к неполной квадратичной модели и проверить ее. Для двух факторов данная модель записывается следующим образом:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 .$$

Далее производят замену $\beta_3 = \beta_{12}$ и $x_3 = x_1 x_2$ и проверяют значимость получившейся модели. Если и она не значима, то переходят к полной квадратичной модели, затем (при ее незначимости) – к неполной кубической и полной кубической. Если же и последняя оказывается незначимой, то переходят к степенной, но здесь возрастает риск возникновения мультиколлинеарности. В принципе, даже кубическая модель уже дает высокую мультиколлинеарность. В случае, когда не удается найти ни одной значимой модели, признается, что регрессионными методами поставленную задачу решить нельзя, и ищется иной способ решения.

4. Проверка значимости каждого фактора по критерию Стьюдента. Если какой-то фактор оказывается незначимым, то его

убирают из расчета и повторяют его заново. Если незначимых факторов несколько, то убирают по одному.

5. Оценка качества модели с помощью коэффициента детерминации. В зависимости от области задачи приемлемы различные значения коэффициента детерминации. Например, для техники значение 0,9 будет пороговым. Все, что выше его, – хорошее, ниже – не очень. Для экономики порогом будет значение 0,5.

Теперь рассмотрим еще один особый вид регрессии – логистическую регрессию. Данный вид используется, если значение отклика (*y)* должно быть ограничено каким-либо диапазоном. Например, если *y* – вероятность некоторого события, то она должна лежать строго в диапазоне от 0 до 1. Рассмотренные ранее модели никак этого не учитывают, поэтому в подобных задачах используют логистическую регрессию. Чаще всего она применяется в задачах бинарной классификации. При $y < 0{,}5$ – один класс, иначе – другой.

В основе данного вида регрессии лежит следующая функция:

$$y = \frac{1}{1+e^{-x}}.$$

Ее график представлен на рисунке 14.

Соответствующая этой функции регрессионная модель будет иметь вид

$$y = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_k x_k)}}.$$

Теоретически, выполнив преобразование линерализации, ее можно привести к линейной модели множественной регрессии, произведя следующую замену:

$$z = -\ln\left(\frac{1}{y-1}\right).$$

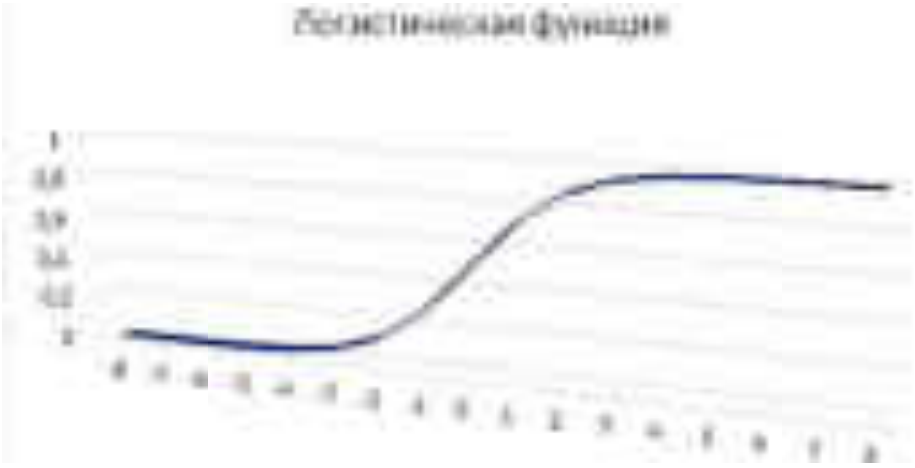


Рисунок 14. График логистической функции

Но так как в процессе преобразований нарушаются основные предпосылки МНК, то в практике этот подход не используется. Вместо этого для оценки параметров модели применяют метод максимального правдоподобия, с которым вам предлагается ознакомиться самостоятельно при необходимости.

Мы закончили рассмотрение статистических методов анализа данных. Теперь перейдем к более абстрактным моделям и алгоритмам, а именно к нечеткой логике.

Модели и методы нечеткой логики

Природа нечетких объектов обусловлена использованием экспертных оценок, которым свойственна неопределенность класса нечеткости. В отличие от стохастической неопределенности, нечеткость затрудняет или даже исключает применение статистических методов и моделей, но может быть использована для принятия предметно-ориентированных решений на основе приближенных рассуждений человека. Основные проблемы, решаемые в нечеткой логике, связаны с моделированием интеллектуальных операций приближенных рассуждений человека (эксперта), а также объектов, над которыми эти операции выполняются:

1. Объектами интеллектуальных операций, используемых в приближенных рассуждениях человека, являются переменные нового

класса – лингвистические переменные, значениями которых являются нечеткие множества. Важным является тот факт, что наименования лингвистической переменной и ее значений должны соответствовать словам, которые использует человек при решении прикладных задач. Таким образом, операндами и результатом интеллектуальных операций являются значения особого вида – *нечеткие множества*.

2. Основные интеллектуальные операции строятся с помощью *операций нечеткой логики*.

3. Алгоритмы вычисления нечетких значений предназначены для манипулирования со значениями, представленными нечеткими множествами на основе операций нечеткой логики, поэтому они классифицируются как нечеткие системы логического вывода. Часто используют сокращенную форму обозначенного класса моделей – *нечеткие модели или нечеткие системы.*

Нечеткие множества

Теория нечетких множеств, введенная Л. Заде [15], – это раздел прикладной математики, посвященный методам анализа неопределенных данных, в которых описание неопределенностей реальных явлений и процессов проводится с помощью понятия о множествах, не имеющих четких границ.

В дальнейшем для указания неопределенности экспертных оценок будем использовать эквивалентное выражение – лингвистическая неопределенность. Л. Заде предложил по аналогии с теорией вероятностей использовать в качестве математической модели лингвистической неопределенности объекта $x \in X$ функцию вида

$$Y = \mu\,(x,B),$$

где Y – результат вычисления функции, выражающий меру неопределенности (нечеткости) для конкретного объекта $x \in X$;

μ – непрерывная функция, такая, что $\mu: X \to [0,1]$. Содержательно функция μ определяет распределение неопределенности на X;

X – область определения функции μ. Область определения задается упорядоченным множеством значений произвольной природы, называемым *универсальным множеством (или универсумом*). Носителем функции $\mu(x,B)$ является подмножество $w \subset X$, на котором функция $\mu(x,B)$ принимает значение, отличное от нуля. В качестве универсального множества обычно задается множество действительных чисел;

B – вектор параметров функции, обычно числовых.

Функциональная модель лингвистической неопределенности получила название *нечеткого множества*, так как указанная функция μ рассматривается как характеристическая функция, определенная на множестве объектов X. Таким образом, с математической точки зрения, нечеткое множество моделируется параметрической функцией особого класса, называемого классом *функций принадлежности*. В том случае, если значения функции принадлежности нечеткого множества представлены точными числовыми значениями, такие нечеткие множества относят к *нечетким множествам типа 1*. Если значения функции принадлежности нечеткого множества моделируются другими нечеткими множествами, то такое нечеткое множество относят к *нечетким множествам типа 2*.

На практике используют несколько способов задания функции принадлежности. Среди них выделим следующие:

Структурный способ. Данная форма определения нечетких множеств основана на табличном представлении функций. В случае, если известен вектор параметров B, табличное представление функции принадлежности может быть задано явно путем табулирования функции $Y = \mu(x,B)$ на множестве значений w, являющемся ее носителем. При неизвестном векторе параметров B – путем прямого перечисления множества пар в виде

$$Y = \{\mu_1/x_1, \mu_2/x_2, .., \mu_n/x_n\}.$$

Данная форма удобна для графического отображения нечеткого множества и используется часто в тех случаях, когда затруднительно

задать математический вид функции $Y = \mu(x,B)$, например, если X не является множеством чисел.

Рассмотрим пример записи нечеткого множества в явной форме.

Пусть $w = \{x_1, x_2, x_3, x_4, x_5\}$, A – нечеткое множество, для которого $\mu_A(x_1) = 0{,}3; \mu_A(x_2) = 0; \mu_A(x_3) = 1; \mu_A(x_4) = 0{,}6; \mu_A(x_5) = 0{,}9$.

Тогда A можно представить в виде

$$A = \{0{,}3/x_1;\ 0/x_2;\ 1/x_3;\ 0{,}6/x_4;\ 0{,}9/x_5\}.$$

Функциональный способ. При этом предполагается, что форма функции принадлежности, моделирующей нечеткое множество, известна и определена на множестве действительных чисел X. Для представления $Y = \mu(x,B)$ используют различные функции (показанные, например, на рисунке 15: а) – треугольная, б) – трапецеидальная, в) – Гауссова).

Гауссова функция принадлежности описывается вектором параметров $B=\{\sigma, c\}$ и формулой

$$\mu(x, B) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right),$$

где c – среднее значение;

σ – среднее квадратичное отклонение.

Треугольная функция принадлежности характеризуется тройкой чисел, $B=\{a,b\ , c\}$, и вычисляется по формуле

$$\mu(x, B) = \begin{cases} \dfrac{x-a}{b-a}, a \le x \le b \\ \dfrac{c-x}{c-b}, b < x \le c \\ 0, x < a, x > c, \end{cases}$$

где b – задает координату вершины треугольника;

a, c – определяют основание треугольника.

По аналогии задается и трапецеидальная функция принадлежности, которая характеризуется четверкой чисел $B = \{a, b, c, d\}$:

$$\mu(x,B)=\begin{cases}\dfrac{x-a}{b-a}, a\le x\le b\\ 1, b<x<c\\ \dfrac{d-x}{d-c}, c\le x\le d\\ 0, x<a, x>d.\end{cases}$$

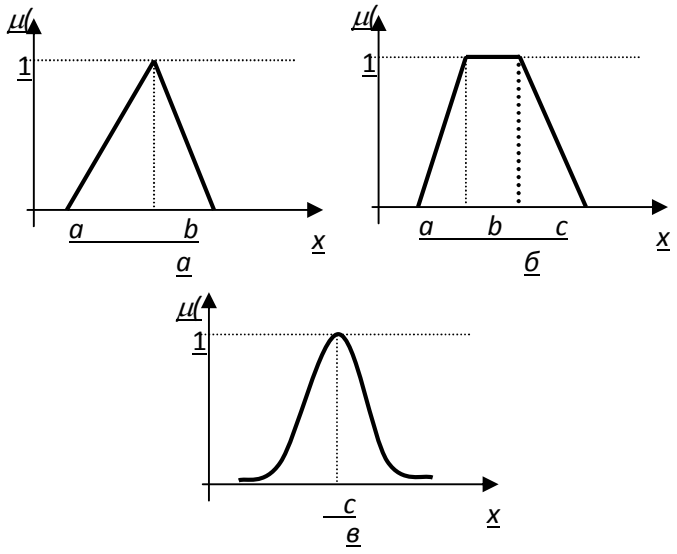


Рисунок 15. Типовые формы функций принадлежности

На практике часто параметр *B* явно не указывается для обеспечения более компактной записи функции принадлежности, то есть используется функциональная запись вида $Y=\mu(X)$ вместо $Y=\mu(x,B)$. При дальнейшем изложении в учебном пособии будем использовать компактную запись функции принадлежности.

С каждой функцией принадлежности $\mu(X)$ сопоставляется лингвистическое обозначение нечеткого множества (лингвистический терм). Тогда функция принадлежности нечеткого множества *Z* (функциональный способ) будет иметь следующую запись $Z=\mu_Z(X)$. Расширив традиционное понятие множества, Л. Заде построил «математический мостик» при описании свойств понятий в виде нечетких множеств между числом *x*, свойством *Z*, выраженным лингвистически, и степенью соответствия числа *x* свойству *Z* в виде функции

$\mu_Z(x)$. Фактически обозначение нечеткого множества через Z позволяет именовать функцию принадлежности $\mu_Z(X)$, в общем случае параметрическую, лингвистическими терминами, то есть оперировать с ней как со значениями лингвистической переменной. Часто эти два понятия – Z и $\mu_Z(X)$ – рассматриваются как эквивалентные.

Пусть $X = \{x\}$ – совокупность объектов (универсальное множество), обозначаемых через x. Пусть на X определены три нечетких множества: *A=«низкое», B=«удовлетворительное», C=«хорошее»,* обозначающие имена свойств понятия «качество». Тогда, следуя структурному способу для всех $x \in X$, нечеткое множество A может быть задано совокупностью упорядоченных пар $A = \{x, \mu_A(x)\}$, нечеткое множество B – совокупностью упорядоченных пар $B = \{x, \mu_B(x)\}$, нечеткое множество C – совокупностью упорядоченных пар $C = \{x, \mu_C(x)\}$. Используя функциональный способ записи, приведенный выше, нечеткие множества будут представлены следующим образом:

$$A = \mu_A(x),\ B = \mu_B(x),\ C = \mu_C(x).$$

Лингвистические переменные

Лингвистическая переменная – это переменная, значениями которой являются слова или высказывания естественного или искусственного языка.

Согласно [16]: «Поскольку слова в общем смысле менее точны, чем числа, понятие лингвистической переменной дает возможность приближенно описывать явления, которые настолько сложны, что не поддаются описанию в общепринятых количественных терминах… высокая точность несовместима с высокой сложностью. Таким образом, быть может, именно по этой причине обычные методы анализа систем и моделирования на ЭВМ, основанные на точной обработке численных данных, по существу не способны охватить огромную сложность процессов человеческого мышления и принятия решений. Отсюда напрашивается вывод о том, что для получения

существенных выводов о поведении гуманистических систем придется, по-видимому, отказаться от высоких стандартов точности и строгости, которые мы, как правило, ожидаем при математическом анализе четко определенных механистических систем, и относиться более терпимо к иным подходам, которые являются приближенными по своей природе».

Любая переменная описывается множеством допустимых значений, а лингвистические понятия описываются набором присущих им свойств. Л. Заде расширил понятие обычной лингвистической переменной, допустив, что в качестве ее значений (термов) выступают нечеткие переменные [16]. Пример лингвистической переменной, заимствованный из [17], представлен на рисунке 16.

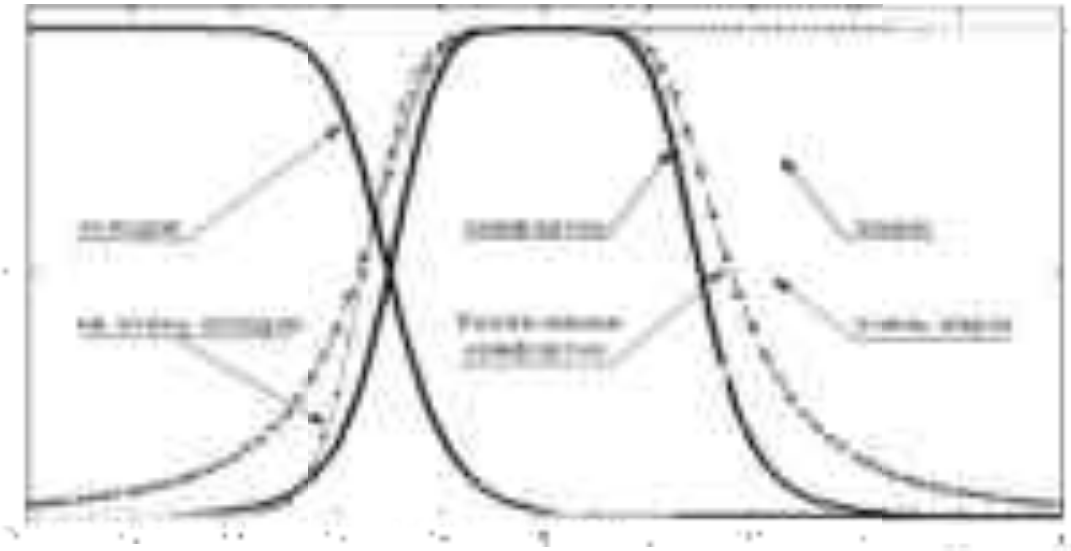


Рисунок 16. Пример лингвистической переменной «Температура»

Формально лингвистическая переменная описывается набором

$$< Name, \widetilde{X}, X, G, P >,$$

где *Name*– наименование лингвистической переменной;

X – универсальное множество объектов x;

$\widetilde{X}$ – базовое терм-множество, образующее совокупность термов лингвистической переменной, например, $\widetilde{X}$ ={«Отличный», «Хороший», «Плохой», «Удовлетворительный» и др.};

G – синтаксические правила вывода (порождения) новых термов $\tilde{X}^*$, не входящих в базовое терм-множество, задаваемые обычно на основе контекстно-свободной грамматики;

P – семантические правила, контекстно-зависимый способ вычисления смысла на основе функций принадлежности каждого терма из $\tilde{X} \cup \tilde{X}^*$.

Операции нечеткой логики

Нечеткая логика – это логика, оперирующая нечеткими высказываниями и рассуждениями на базе частичной истинности.

В основе операций нечеткой логики лежит понятие нечеткого множества, выраженного функцией принадлежности. Поэтому операндами и результатами операций нечеткой логики являются также функции, определяющие новые нечеткие множества.

В нечеткой логике для моделирования основных логических связок И ($\wedge$), ИЛИ ($\vee$) над нечеткими множествами используют триангулярные нормы [18].

Триангулярной нормой (t-нормой) называют отображение $T:[0,1]\times[0,1]\to[0,1]$, удовлетворяющее следующим условиям:

T(0, 0)=0; T(x, 1)= x; T(1, x)= x – ограниченность;

$T(x,y)\le T(a,b)$, если $x\le a$, $y\le b$ – монотонность;

$T(x,y)=T(y,x)$ – коммутативность;

$T(x,T(y,z))\le T(T(x,y),z)$ – ассоциативность.

Триангулярной конормой (s-конормой) называют отображение $S:[0,1]\times[0,1]\to[0,1]$, удовлетворяющее следующим условиям:

S(1, 1)=1; S(x, 0)= x; S(0, x)= x – ограниченность;

$S(x,y)\ge S(a,b)$, если $x\ge a$, $y\ge b$ – монотонность;

$S(x,y)=S(y,x)$ – коммутативность;

$S(x,S(y,z))\le S(S(x,y),z)$ – ассоциативность.

t-норма и s-конорма в определенном смысле являются двойственными понятиями. Эти функции могут быть получены друг из

друга, например, с помощью инволютивного отрицания и законов Де Моргана следующим образом:

$$S(x,y) = n(T(n(x),n(y))),\ T(x,y) = n(S(n(x),n(y))).$$

Простейшими примерами t-норм и s-конорм, взаимно связанных этими соотношениями для *n(x)=1-x*, являются следующие (таблица 9).

Таблица 9. Примеры t-норм и s-конорм

Формула	**Интерпретация**
T(x, y) = min{x,y}	*(минимум)*
S(x, y) = max{x,y}	*(максимум)*
T(x, y) = xy	*(произведение)*
S(x, y) = x+y-xy	*(вероятностная сумма)*
T(x, y)=max{x+y-1, 0}	*(t-норма Лукасевича)*
S(x, y)=min{x+y,1}	*(t-конорма Лукасевича ограниченная сумма)*

Основные операции с нечеткими множествами

1. Операция эквивалентности

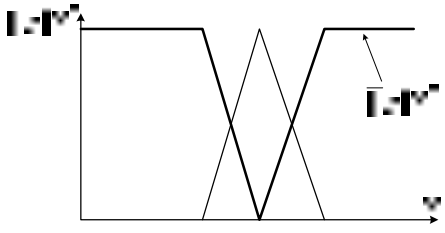
$$A \equiv B \Leftrightarrow \forall x \in X \ \ \mu_A(x) = \mu_B(x)$$

2. Операция включения

$$A \subseteq B \Leftrightarrow \mu_A(x) \le \mu_B(x), \forall x \in X$$

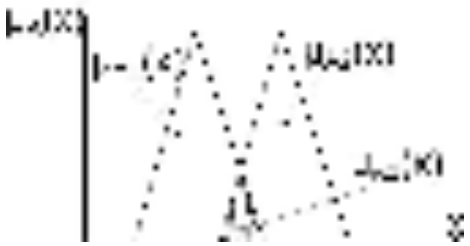
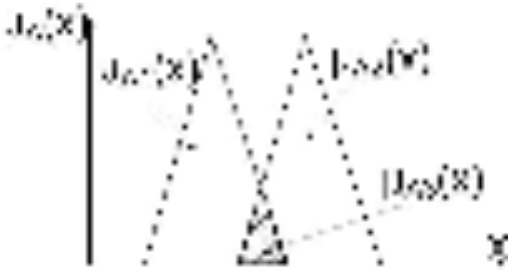
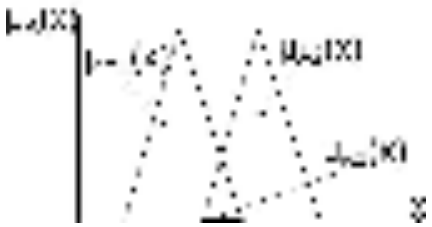
4. Нечеткая операция «НЕ» (дополнение) показана в таблице 10.

Таблица 10. Нечеткое «НЕ»

Иллюстрация	**Формула**
	$\overline{\mu_A}(x) = 1 - \mu_A(x)$ – нечеткая операция НЕ по Заде $\overline{\mu_A}(x) = \frac{1-\mu_A(x)}{1+\lambda \cdot \mu_A(x)}$ – НЕ по Сугено

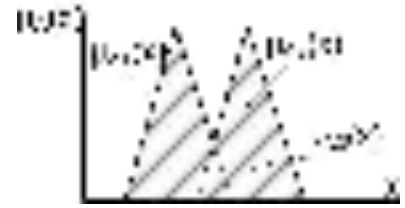
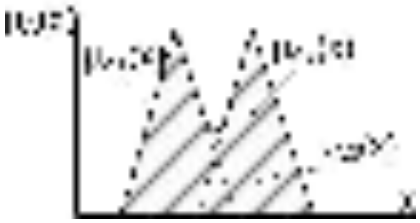
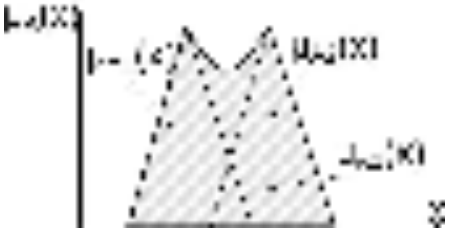
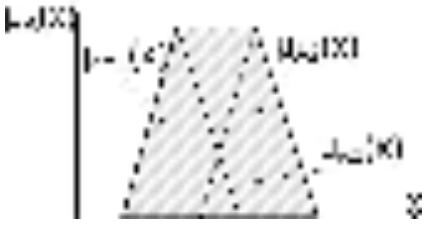
5. Нечеткая операция «И» показана в таблице 11.

Таблица 11. Нечеткое «И»

Иллюстрация	Формула
логическое произведение (Заде)	
	$\mu_{A_3}(x) = \mu_{A_1 \wedge A_2}$ $= \min(\mu_{A_1}(x), \mu_{A_2}(x))$ min – конъюкция
алгебраическое произведение (Бандлер, Коходт)	
	$\mu_{A_3}(x) = \mu_{A_1}(x) \cdot \mu_{A_2}(x)$
граничное произведение (Лукасевич, Гринс)	
	$\mu_{A_3}(x) = \mu_{A_1 \otimes A_2}$ $= \max(\mu_{A_1}(x) + \mu_{A_2}(x) - 1, 0)$
драстическое произведение (Вебер)	
	$\mu_{A_3}(x) = \mu_{A_1}(x) \triangle \mu_{A_2}(x) = \begin{cases} \mu_{A_1}(x), если\ \mu_{A_2}(x) \\ \mu_{A_2}(x), если\ \mu_{A_1}(x) \\ 0, иначе \end{cases}$ $0 \le \mu_{A_1 \triangle A_2} \le \mu_{A_1 \otimes A_2} \le \mu_{A_1 \cdot A_2} \le \mu_{A_1 \wedge A_2}$

6. Нечеткая операция «ИЛИ» показана в таблице 12.

Таблица 12. Нечеткое «ИЛИ»

Иллюстрация	**Формула**
логическая сумма (Заде)	
	$\mu_{A_3}(x) = \mu_{A_1 \vee A_2} = \max(\mu_{A_1}(x), \mu_{A_2}(x))$ max-дизъюнкция
алгебраическая сумма	
	$\mu_{A_3}(x) = \mu_{A_1}(x) + \mu_{A_2}(x) - \mu_{A_1}(x) \cdot \mu_{A_2}(x),$ $где\ \forall x \in X$
граничная сумма	
	$\mu_{A_3}(x) = \mu_{A_1 \oplus A_2} = \min(\mu_{A_1}(x) + \mu_{A_2}(x), 1)$
драстическая сумма	
	$\mu_{A_3}(x) =$ $\mu_{A_1}(x) \nabla \mu_{A_2}(x) = \begin{cases} \mu_{A_1}(x), если\ \mu_{A_2}(x) = 0 \\ \mu_{A_2}(x), если\ \mu_{A_1}(x) = 0 \\ 1, иначе \end{cases}$

По существу, все человеческие понятия являются нечеткими, так как они получаются в результате группировки (clumping) точек или объектов, объединяемых по сходству. Тогда нечеткость подобных

групп (clumps) есть прямое следствие нечеткости понятия сходства. Простыми примерами таких групп являются понятия «средний возраст», «деловая часть города», «немного облачно», «бестолковый» и др. Данную группу в нечеткой логике называют *«гранулой» (granule).* В естественном языке (ЕЯ) слова играют роль меток гранул и служат для сжатия данных. Сжатие данных с помощью слов является ключевым аспектом человеческих рассуждений и формирования понятий.

В нечеткой логике гранулирование информации лежит в основе понятий лингвистической переменной и нечетких правил типа «ЕСЛИ-ТО», задаваемой операцией импликации.

Операция импликации

В качестве основного математического инструмента при определении импликации A→B для нечетких множеств *A* и *B* используют композиционное правило Л. Заде [16], являющееся обобщением правила modusponens (таблица 13).

Таблица 13. Импликация

Описание	**Формула**
Предпосылка	$A \to B$
Событие	A^*
Вывод	$A^* \circ (A \to B)$

Пусть *U* и *V* – два универсальных множества с базовыми переменными *u* и *v* соответственно. Пусть *A* и *F* – нечеткие подмножества множеств *U* и *U×V*. Тогда композиционное правило вывода утверждает, что из нечетких множеств *A* и *F* следует нечеткое множество $B = A \circ F$. Функция принадлежности результата вычисляется с помощью триангулятных норм следующим образом:

$$\mu_B(v) = S(T(\mu_A(u), \mu_F(u,v)),$$

при моделировании s-конормы и t-нормы операциями (∨) *max* и (∧) *min* соответственно:

$$\mu_B(v) = \bigvee_{u \in U} ((\mu_A(u) \wedge \mu_F(u, v)) .$$

Другие формулы, реализующие операцию импликации, и математические объекты теории нечетких множеств и нечеткой логики приведены в учебном пособии [19].

Формализация нечеткой импликации позволила задать правила «ЕСЛИ-ТО» в виде нечетких продукционных правил и заложило основу нечеткого моделирования опыта и знаний экспертов, выраженных в виде приближенных зависимостей.

Нечеткие системы

Целью построения нечетких систем сложных явлений является приближенное описание зависимости (аппроксимация некоторой функции) $Y = f(X)$,

где Y – выходная лингвистическая переменная;

X – вектор входных лингвистических переменных. Его размерность – n;

f – зависимость между X и Y, описываемая совокупностью нечетких правил.

В основе нечетких систем лежат совокупность нечетких правил «ЕСЛИ-ТО», описывающих зависимости между нечеткими переменными предметной области, композиционное правило вывода и способ вычисления значений нечетких переменных (способ нечеткого вывода).

Системой нечеткого логического вывода в теории нечетких множеств и нечеткой логики называется модель, которая описывает поведение систем на естественном (или близком к естественному) языке в виде приближенных рассуждений на основе композиционного правила вывода.

В систему нечеткого логического вывода входят следующие объекты (см. рисунок 17):

- совокупность нечетких правил (база правил);
- набор функций принадлежностей базы нечетких переменных (база переменных);

- блок фаззификации;
- блок дефаззификации;
- блок вывода.

Рисунок 17. Система нечеткого вывода

База правил хранит множество логических правил вывода, а также их порядок (иерархическую структуру) применения. База нечетких переменных содержит названия лингвистических термов и параметры их функций принадлежности. База правил вместе с базой нечетких переменных образуют *базу знаний* (БЗ) системы нечеткого вывода.

Простейшие системы нечеткого логического вывода основаны на правилах вида:

R_i: *Если X есть* A_i *и Y есть* B_i, *то Z есть* C_i,

R_i: *Если X есть* A_i *и Y есть* B_i, *то* $z=f_i(x,y)$,

где X, Y – входные нечеткие переменные;

Z – выходная нечеткая переменная;

A_i, B_i, – входные значения (функции принадлежности);

C_i – выходные нечеткие значения (функции принадлежности);

f_i – некоторые вещественные функции.

При этом должны соблюдаться следующие условия:

- Существует хотя бы одно правило для каждого лингвистического терма выходной переменной.

- Для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

Распространены пять способов реализации нечеткого логического вывода [18].

Схема 1: Алгоритм Мамдани (Mamdani). Импликация моделируется минимумом, а агрегация – максимумом.

Схема 2: Алгоритм Цукамото (Tsukamoto). Исходные посылки – как у предыдущего алгоритма, но предполагается, что функции принадлежности являются монотонными.

Схема 3. Алгоритм Суджено (Sugeno). Алгоритм предполагает, что правые части правил вывода представлены в виде линейных функций.

Схема 4. Алгоритм Ларсена (Larsen). В алгоритме Ларсена нечеткая импликация моделируется с использованием операции умножения.

Схема 5. Упрощенный алгоритм нечеткого вывода. Исходные правила в данном случае задаются в виде

Если X есть A_i *и Y есть* B_i*, то* $z = Z_i$,

где Z_i – четкое значение.

Рассмотрим алгоритм нечеткого вывода по схеме Мамдани для базы правил вида R_i*: Если X есть* A_i *и Y есть* B_i*, то Z есть* C_i*.*, $i=[1,r]$.

1. Фаззификация.

Определяются степени истинности по функциям принадлежности для левых частей каждого правила:

$a_i = \mu_{A_i}(X)$

$b_i = \mu_{B_i}(Y)$,

где a_i – степень принадлежности X к A_i;

b_i – степень принадлежности Y к B_i;

$i= [1,r]$;

r – количество правил.

2. Импликация.

Определяется сила каждого правила, t-нормой является логический минимум:

$\alpha_i = \min(a_i, b_i)$.

Модифицируются функции принадлежности переменной z в каждом правиле:

$\mu'_i(z) = \min(\alpha_i, \mu_i(z))$,

где $\mu_i(z)$ – функция принадлежности переменной Z_i.

3. Агрегация.

Объединение выходов каждого правила логическим максимумом (s-конорма):

$\mu'(z) = \max_i(\mu_i(z))$

4. Деффазификация.

Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности. Однако пригодность этого способа ограничивается одноэкстремальными функциями принадлежности. Для многоэкстремальных функций на практике часто используется метод центра тяжести.

Дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле

$$x^0 = \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx}.$$

Физическим аналогом этой формулы является нахождение центра тяжести плоской фигуры, ограниченной осями координат и графиком функции принадлежности нечеткого множества. В случае дискретного универсального множества дефаззификация нечеткого множества по методу центра тяжести осуществляется по формуле

$$x^0 = \frac{\sum_{i=1}^{k} x_i \cdot \mu(x_i)}{\sum_{i=1}^{k} \mu(x_i)}.$$

На рисунке 18 графически представлен процесс нечеткого вывода по алгоритму Мамдани.

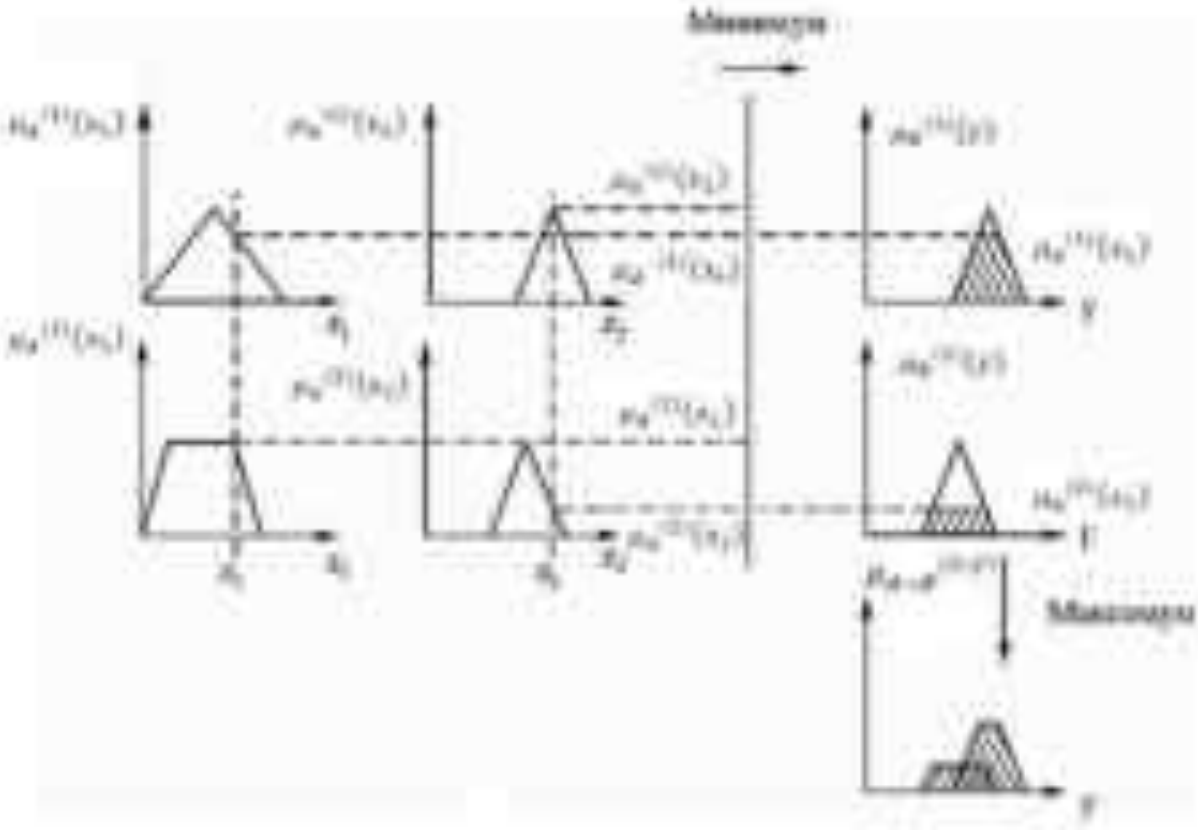


Рисунок 18. Алгоритм нечеткого вывода Мамдани

Пусть дана система управления с двумя правилами нечеткого управления:

Правило 1: IF x is A1 AND y is B1 THEN z is C1;

Правило 2: IF x is A2 AND y is B2 THEN z is C2.

Предположим, что величины x_0 и y_0, считываемые датчиком, являются четкими входными величинами для лингвистических переменных *x* и *y*, и что заданы следующие функции принадлежности для нечетких подмножеств *A1, A2, B1, B2, C1, C2* этих переменных:

$$\mu_{A1}(x) = \begin{cases} \frac{x-2}{3}, 2 \le x \le 5; \\ \frac{8-x}{3}, 5 \le x \le 8; \end{cases} \quad \mu_{A2}(x) = \begin{cases} \frac{x-3}{3}, 3 \le x \le 6; \\ \frac{9-x}{3}, 6 \le x \le 9; \end{cases}$$

$$\mu_{B1}(y) = \begin{cases} \frac{y-5}{3}, 5 \le y \le 8; \\ \frac{11-y}{3}, 8 \le y \le 11; \end{cases} \quad \mu_{B2}(x) = \begin{cases} \frac{y-4}{3}, 4 \le y \le 7; \\ \frac{10-y}{3}, 7 \le y \le 10; \end{cases}$$

$$\mu_{C1}(z) = \begin{cases} \frac{z-2}{3}, 1 \le z \le 4; \\ \frac{7-z}{3}, 4 \le z \le 7; \end{cases} \quad \mu_{C2}(z) = \begin{cases} \frac{z-3}{3}, 3 \le z \le 6; \\ \frac{9-z}{3}, 6 \le z \le 9; \end{cases}$$

Предположим, что в момент времени t_1 были считаны значения датчиков $x_0(t_1) = 4$ *и* $y_0(t_1) = 8$. Проиллюстрируем, как при этом будет вычисляться величина выходного сигнала.

Первым шагом находим -срезы для первого и второго правила на основе заданных функций принадлежности (с учетом значений x_0 и y_0). С этой целью вычисляем величины функций принадлежности в заданных точках для первого и второго правил:

$$\mu_{A1}(x_0=4)=2/3 \quad \text{и} \quad \mu_{B1}(y_0=8)=1;$$
$$\mu_{A2}(x_0=4)=1/3 \quad \text{и} \quad \mu_{B2}(y_0=8)=2/3.$$

Затем, в соответствии с правилом вывода по Мамдани (выбор минимального значения функций принадлежности), определяем:

$$\alpha_1 = min(\mu_{A1}(x_0), \mu_{B1}(y_0)) = min(2/3, 1) = 2/3;$$
$$\alpha_2 = min(\mu_{A2}(x_0), \mu_{B2}(y_0)) = min(1/3, 2/3) = 1/3.$$

Результат применения вычисленных значений α_1 и α_2 показан на рисунке 19. Окончательный результат получается путем объединения найденных значений функций принадлежности с использованием оператора максимума (с учетом стратегии вывода по Мамдани). Результирующая функция принадлежности также представлена на рисунке 19.

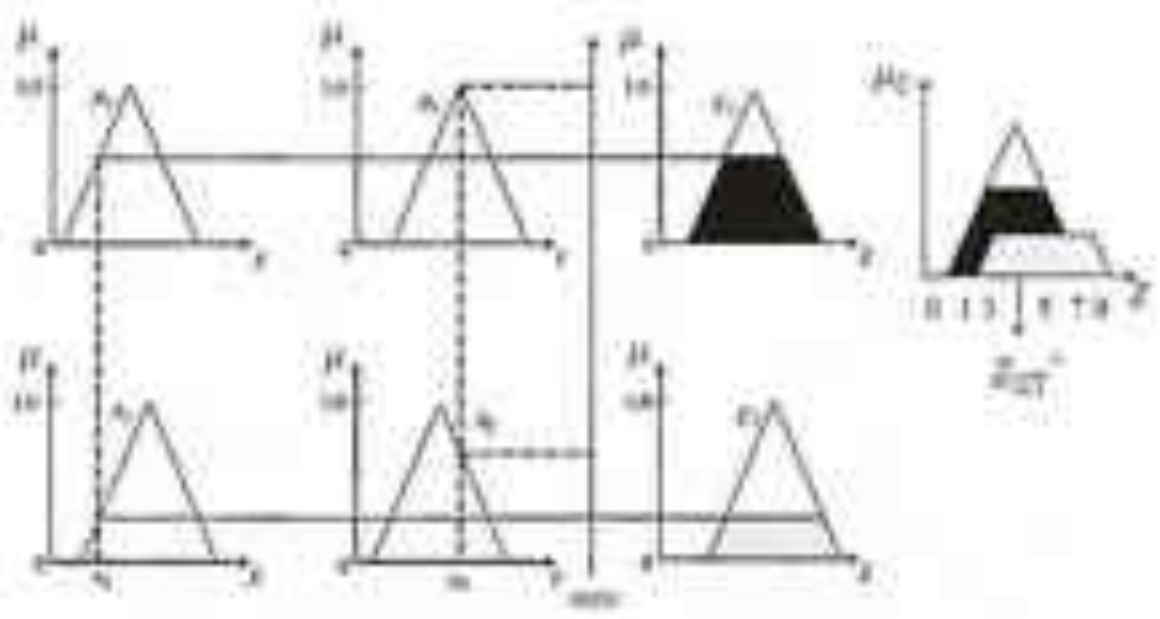


Рисунок 19. Иллюстрация нечеткого вывода по Мамдани

Для вычисления искомой выходной величины Z проводим дефаззификацию нечеткой величины μ_C. По методу центра тяжести получаем

$$Z_{\text{ЦТ}}{}^{*} = \frac{2\left(\frac{1}{3}\right) + 3\left(\frac{2}{3}\right) + 4\left(\frac{2}{3}\right) + 5\left(\frac{2}{3}\right) + 6\left(\frac{1}{3}\right) + 7\left(\frac{1}{3}\right) + 8\left(\frac{1}{3}\right)}{\left(\frac{1}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right) + \left(\frac{1}{3}\right) + \left(\frac{1}{3}\right) + \left(\frac{1}{3}\right)} = 4{,}7.$$

Нечеткая логика в анализе временных рядов

Предположим, что задан процесс, состояния которого описываются *n* значениями одной переменной. Пусть в результате наблюдения получен временной ряд этой переменной, представляющий последовательность упорядоченных в равноотстоящие моменты времени пар $\{x_i, t_i\}$, таких, что $\forall x_i \in X, X \subset R^1, t_i \in N, i \in [1, n]$. Значение x_i – *уровень* временного ряда. Тогда введем понятие нечеткого временного ряда.

Нечетким временным рядом (НВР) называют упорядоченную в равноотстоящие моменты времени последовательность наблюдений над некоторым процессом, состояния которого изменяются во вре-

мени, если значение состояния процесса в момент t_i может быть выражено с помощью нечеткой метки $\tilde{x}_i$.

Нечеткая метка $\tilde{x}_i$ может быть сформирована непосредственно экспертом в форме

$$\mu_{\tilde{x}_i}(w, x_i),$$

где $\tilde{x}_i \in \tilde{X}$, $\tilde{X}$ – множество нечетких меток;

w – носитель (интервал на X) нечеткой метки $\tilde{x}_i$, $x_i \in w$;

$\mu_{\tilde{x}_i}(w, x_i) \in [0,1]$ – функция принадлежности нечеткой метки $\tilde{x}_i$ уровню временного ряда x_i, обычно треугольной формы.

Носитель нечеткой метки $\tilde{x}_i$ – это четкое множество $w \subseteq B$ таких точек $x_i \in w$, для которых $\mu_{\tilde{x}_i}(w) > 0$, где $B \subset X$ – базовое множество нечетких меток $\tilde{X}$.

Таким образом, нечеткий временной ряд формируется в результате интервального качественного оценивания уровней числового ВР. Интервалы-носители нечетких меток, образованные на множестве *X,* обязательно пересекаются. Качественный аспект нечеткой метке придает функция $\mu_{\tilde{x}_i}(w) \in [0,1]$.

На рисунке 20 изображен абстрактный нечеткий временной ряд, где каждой нечеткой метке $\tilde{x}_i$ соответствует нечеткое множество, задаваемое функцией принадлежности.

Рисунок 20. Абстрактный нечеткий временной ряд

В отличие от традиционного временного ряда значениями нечеткого ВР являются нечеткие множества, а не действительные значения уровней ВР.

В 1993 году ученые Сонг (Song) и Чиссом (Chissom) [20] предложили нечеткие модели детерминированных (time-variant) и авторегрессионных (time-invariant) временных рядов первого порядка (fist-order) и применили разработанные модели для прогнозирования количества регистрирующихся студентов университета штата Алабама (США), фаззифицировав предварительно четкий временной ряд. Это было первое применение нечетких моделей при моделировании ВР и первое определение моделей нечетких временных рядов.

Пусть $X_t,(t=1,...) \subset R^1$ – универсальное множество, на котором определены нечеткие множества $y_t^i,(i=1,2,...)$ и Y_t – коллекция $y_t^i,(i=1,2,...)$. Тогда Y_t называется нечетким ВР.

На практике в большинстве временных рядов последовательные наблюдения зависимы, так что:

$$R=\{(y_t,y_{t-1}),(y_{t-1},y_{t-2})...\} \subseteq Y_t \times Y_{t-1},$$

где Y_t, Y_{t-1} обозначают переменные;

y_t, y_{t-1} – наблюдаемые значения этих переменных.

Наиболее частой моделью зависимости является явная функция:

$$f: Y_{t-1} \to Y_t,$$

представленная линейной функцией:

$$y_t = f(y_{t-1},\phi,\varepsilon) = \phi y_{t-1} + \varepsilon_t,$$

где ε_t – случайная ошибка, шум.

В случае нечеткого временного ряда в качестве модели авторегрессии используется нечеткое разностное уравнение:

$$y_t^j = y_{t-1}^i \circ R_{ij}(t,t-1),$$

$$y_t^i \in Y_t,\ \ y_{t-1}^i \in Y_{t-1},\ \ i \in I,\ \ j \in J,$$

где ○ – обозначает операцию композиции из теории нечетких множеств;

$R(t,t-1)=\bigcup\limits_{i,j}R_{ij}(t,t-1)$ – система нечетких отношений, которая символически может быть записана в виде $Y_t \to Y_{t-1}$.

Систему отношений R в выражении $Y_t = Y_{t-1} \circ R(t,t-1)$ называют *моделью нечеткого временного ряда* первого порядка. Данная модель – важный частный случай общей модели порядка p:

$$Y_t = (Y_{t-1} \times Y_{t-2} \times ... \times Y_{t-p}) \circ R(t,t-p),$$

$$R(t,t-p) = \max\limits_{p} \left\{ \min\limits_{j,i_1,i_2,..i_p} \left\{ y_t^j, y_{t-1}^{i_1}, ... y_{t-p}^{i_p} \right\} \right\}.$$

Метод моделирования нечетких временных рядов

Моделирование нечетких временных рядов в соответствии с нечеткой моделью, предложенной в работе [20], состоит в реализации следующих шагов:

1. Определение нечетких переменных – разбиение данных на множество интервалов (носителей нечетких множеств), определение лингвистических значений нечетких множеств и их функций принадлежности.
2. Формирование логических отношений $Y_t \to Y_{t-1}$.
3. Фаззификация входных данных – определение степени принадлежности входных данных входным нечетким переменным.
4. Вычисление результата применения нечеткого правила $R_{ij}(t,t-1)$ для каждой импликации.
5. Вычисление результирующего отношения R как объединения $\bigcup\limits_{i,j}R_{ij}(t,t-1)$.
6. Применение полученной модели к входным данным и получение выходных нечетких результатов.
7. Дефаззификация нечетких результатов.

Одной из проблем в нечетком моделировании ВР является отсутствие четких рекомендаций на первом этапе построения модели

по выбору количества и параметров нечетких множеств, моделирующих входные и выходные переменные, в частности по определению их носителей (длины интервалов). Данные задачи выполняются экспертом, и, как показывают исследования, от выбора интервалов сильно зависит результат исследования.

Пример моделирования временного ряда в нечетком подходе

Приведем пример нечеткого моделирования временного ряда для прогнозирования прямых валютных котировок ЦБ USD/RUB за июнь 2005 года, описанный в работе [21] и представляющий модификацию метода Сонга, отличающуюся (а) использованием изменений (приращений) данных прошлого вместо реальных числовых значений (регистрации или валютного курса), и (б) вычислением отношений R_j для предсказания будущих состояний.

Рассматриваемый в работе метод был изначально успешно применен к временному ряду, характеризующему количество поступающих в Алабамский университет, которые являются бенчмаркингом при сравнении методов моделирования нечетких временных рядов.

Анализ результативности предлагаемого метода по показателю точности средней относительной ошибки аппроксимации для 6 нечетких множеств (MAPE=2,42) показал, что предложенный метод превышает аналогичный показатель для этого ВР, полученный методом Сонга и Чена (рисунок 21).

Применительно к проблеме прогнозирования валютного курса USD/RUB пошаговое описание предлагаемого метода нечеткого моделирования ВР можно свести к следующему:

Шаг 1: Задание области определения (универсального множества U) проблемы, исходя из вычисленных приращений валютного курса в течение рассматриваемого интервала времени.

Имеются следующие данные. Наибольшее положительное приращение курса доллара по отношению к российскому рублю наблю-

дается в феврале 2002 года, т. е. по сравнению с январским значением рост составляет 0.3679 (более 36 копеек/месяц). В ноябре-декабре 2004 года происходит самое значительное за трехлетний период наблюдений падение котировки доллара практически на 70 копеек (–0.6949). В результате, с целью упрощения последующего разбиения на равновеликие интервалы, полученные граничные значения (–0.6949 и +0.3679) слегка корректируются. Тогда, например, в случае использования шести подынтервалов U может быть представлена отрезком [–0.7, –0.5].

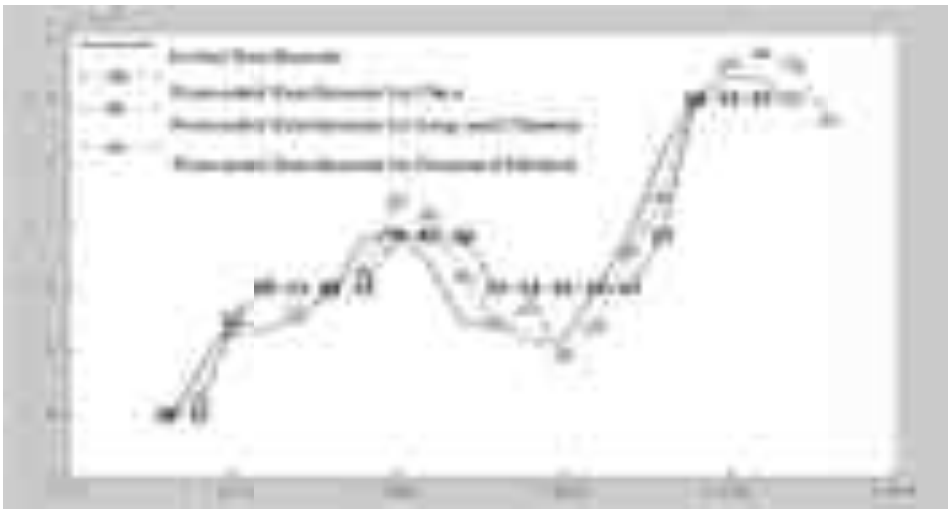


Рисунок 21. Сравнение результатов нечеткого моделирования

Шаг 2: Разбиение множества U на интервалы одинаковой длины.

Если мы оперируем с шестью нечеткими множествами, то область определения делится на 6 интервалов u_i, $i = \overline{1,6}$, $\mathrm{u}_1 = [-0.7, -0.5]$, $\mathrm{u}_2 = [-0.5, -0.3]$, ... , $\mathrm{u}_6 = [0.3, 0.5]$ (в действительности количество нечетких множеств не обязательно должно совпадать с числом интервалов разбиения).

Шаг 3: Определение нечетких множеств A_i.

Предположим, что лингвистическая переменная «изменение валютного курса» характеризуется терм-множеством, образуемым следующими значениями: A_1 (значительное уменьшение), A_2

(уменьшение), A_3 (без изменений/флэт), A_4 (увеличение), A_5 (значительное увеличение), A_6 (очень большое увеличение).

Для шести построенных выше интервалов u_i, $i = \overline{1,6}$, факт принадлежности каждого конкретного u_i определенному множеству A_j, $j = \overline{1,6}$ выражается действительным числом из единичного интервала [0,1] (предполагается, что элементы, отсутствующие в представлении множеств A_j, характеризуются нулевой степенью принадлежности):

$$A_1 = \{1/u_1 + 0.5/u_2\}$$
$$A_2 = \{0.5/u_1 + 1/u_2 + 0.5/u_3\}$$
$$A_3 = \{0.5/u_2 + 1/u_3 + 0.5/u_4\}$$
$$A_4 = \{0.5/u_3 + 1/u_4 + 0.5/u_5\}$$
$$A_5 = \{0.5/u_4 + 1/u_5 + 0.5/u_6\}$$
$$A_6 = \{0.5/u_5 + 1/u_6\},$$

где $u_i \subset U$ – элементы универсума U, а число, стоящее в числителе каждого элемента нечеткого множества, представляет собой степень принадлежности $\mu(u_i)$ этого элемента нечеткому множеству A_j, $j = \overline{1,6}$.

Шаг 4: Фаззификация приращений, полученных на шаге 1.

Считаем, что если приращение года t есть $p \in u_i$, и существует лингвистическое значение (нечеткое множество A_j) с максимальной степенью принадлежности, приходящейся на элемент u_i, тогда p фаззифицируется как A_j. Например, приращение за март 2002 по сравнению с предыдущим месяцем составляет +0.2476 – это значение попадает в интервал u_5, и фаззифицированное приращение становится равным A_5. Аналогичным образом производятся попарные сравнения каждого последующего и предыдущего месяцев, приводящие к формированию последовательности A_6 (февраль 2002), A_5

(март 2002), A_5 (апрель 2002), A_4 (май 2002), A_5 (июнь 2002), A_5 (июль 2002), A_4 (август 2002) и т. д.

Шаг 5: Формирование логических отношений $A_i \to A_j$.

Для построения последовательности логических отношений, рассматриваем попарно последовательные фаззифицированные приращения (февраль – март, март – апрель, и т. д.), определенные на шаге 4. Исключая повторяющиеся комбинации, окончательный список отношений принимает вид

$$A_1 \to A_2,\ A_1 \to A_4$$
$$A_2 \to A_1,\ A_2 \to A_2,\ A_2 \to A_3,\ A_2 \to A_4,\ A_2 \to A_5$$
$$A_3 \to A_2,\ A_3 \to A_3,\ A_3 \to A_4$$
$$A_4 \to A_2,\ A_4 \to A_3,\ A_4 \to A_4,\ A_4 \to A_5$$
$$A_5 \to A_3,\ A_5 \to A_4,\ A_5 \to A_5,\ A_5 \to A_6$$
$$A_6 \to A_5,\ A_6 \to A_4.$$

Мы предполагаем, что нечеткое импликативное отношение $D = B \to C$ для произвольных векторов B и C интерпретируется как нечеткая импликация Мамдани, следовательно, элементы матрицы D вычисляются по формуле $d_{ij} = b_i^T \times c_j = \min\left(b_i, c_j\right)$, где b_i и c_j – элементы векторов B и C, соответственно.

Шаг 6: Объединение логических отношений (шаг 5), имеющих одинаковые левые части, в группы, и вычисление отношений R_i, $i = \overline{1,6}$ для каждой сформированной группы. Можно обратить внимание на то, что группы отношений уже практически построены (см. шаг 5), и выглядят они следующим образом:

$$A_1 \to A_2,\ A_4$$
$$A_2 \to A_1,\ A_2,\ A_3,\ A_4,\ A_5$$
$$A_3 \to A_2,\ A_3,\ A_4$$
$$A_4 \to A_2,\ A_3,\ A_4,\ A_5$$
$$A_5 \to A_3,\ A_4,\ A_5,\ A_6$$
$$A_6 \to A_5,\ A_4.$$

Результирующие отношения $R_i, i=\overline{1,6}$, представляют собой объединения логических отношений, попавших в *i*-ю группу:

$$R_1 = A_1^T \times A_2 \cup A_1^T \times A_4$$
$$R_2 = A_2^T \times A_1 \cup A_2^T \times A_2 \cup A_2^T \times A_3 \cup A_2^T \times A_4 \cup A_2^T \times A_5$$
$$R_3 = A_3^T \times A_2 \cup A_3^T \times A_3 \cup A_3^T \times A_4$$
$$R_4 = A_4^T \times A_2 \cup A_4^T \times A_3 \cup A_4^T \times A_4 \cup A_4^T \times A_5$$
$$R_5 = A_5^T \times A_3 \cup A_5^T \times A_4 \cup A_5^T \times A_5 \cup A_5^T \times A_6$$
$$R_6 = A_6^T \times A_4 \cup A_6^T \times A_5.$$

Шаг 7: Прогнозирование и дефаззификация получаемых результатов.

Вычисленные отношения R_i используются в модели прогнозирования

$$A_i = A_{i-1} \circ R_i,$$

где A_i – нечеткое множество, выражающее прогнозное приращение месяца i;

A_{i-1} – известное приращение предшествующего $(i-1)$-го месяца (если $A_{i-1} = A_j$, то $R_i = R_j$, $j=\overline{1,6}$);

$\circ$ – обозначает композиционный «max-min» оператор.

Например, приращение валютного курса за февраль 2004 года при известном приращении (–0.5714) за январь месяц того же года вычисляется по формуле $F(02.2004) = A_1 \circ R_1$, где R_1 имеет вид, показанный в первой строке, а A_i – фаззифицированное приращение января 2004 года.

Шаг 8: Вычисление прогнозных валютных котировок USD/RUB.

Этот этап предусматривает преобразование полученных на шаге 7 нечетких прогнозных приращений в целые числа. В значительной степени такой процесс зависит от особенностей рассматриваемой

задачи, и одним из критериев выбора процедуры дефаззификации является ее вычислительная простота.

После того как получено числовое приращение для рассматриваемого месяца, оно суммируется с уже имеющимся значением обменного курса предыдущего месяца. Рассмотренный метод нечеткого моделирования может быть отнесен к числу полуавтоматических процедур, поскольку большинство выполняемых шагов, включая построение универсума на основании множества исходных данных задачи, могут быть эффективно воплощены в программной форме. Однако участие аналитика (эксперта) при формировании интервалов разбиения и соответствующих нечетких множеств играет также огромную роль.

Извлечение знаний из временных рядов

Исследования временных рядов, в том числе нечетких, в последние десятилетия оформились в виде отдельного направления, называемого *интеллектуальным анализом данных,* или *DataMining*, в котором анализ временных рядов получил название *интеллектуального анализа временных рядов,* или *TimeSeries DataMining (TSDM).*

Человеческое знание основано на образах и формулируется лингвистически. Вычисления со словами и образами дают методологию для управления информацией и для разработки систем, основанных на знаниях. *DataMining* интегрирует методы, основанные на образах, дает возможность для извлечения информации из баз данных в лингвистической форме, подходящей для их использования в методах принятия решений. Методы и технологии извлечения знаний с использованием временных рядов должны оперировать паттернами временных рядов, отыскивать ассоциации между ними и извлекать знания (рисунок 22). То есть необходимо представление результатов *DataMining* в форме, используемой в человеческих знаниях.

На основе новой методологии *DataMining* решается расширенная совокупность задач анализа временных рядов, определенных в работе [22]:

1) сегментация – разбиение ВР на значимые сегменты;

2) кластеризация – поиск группировок ВР или их паттернов;

3) классификация – назначение ВР или их паттернам одного из заранее определенных классов;

4) индексирование – построение индексов для эффективного выполнения запросов к базам данных ВР;

5) резюмирование (summarization) – формирование краткого описания ВР, содержащего существенные черты с точки зрения решаемой задачи;

6) обнаружение аномалий – поиск новых, нетипичных паттернов ВР;

7) частотный анализ – поиск часто проявляющихся паттернов ВР;

8) прогнозирование – получение очередного значения ВР на основе истории ВР;

9) извлечение ассоциативных правил – поиск правил, относящихся к паттернам ВР.

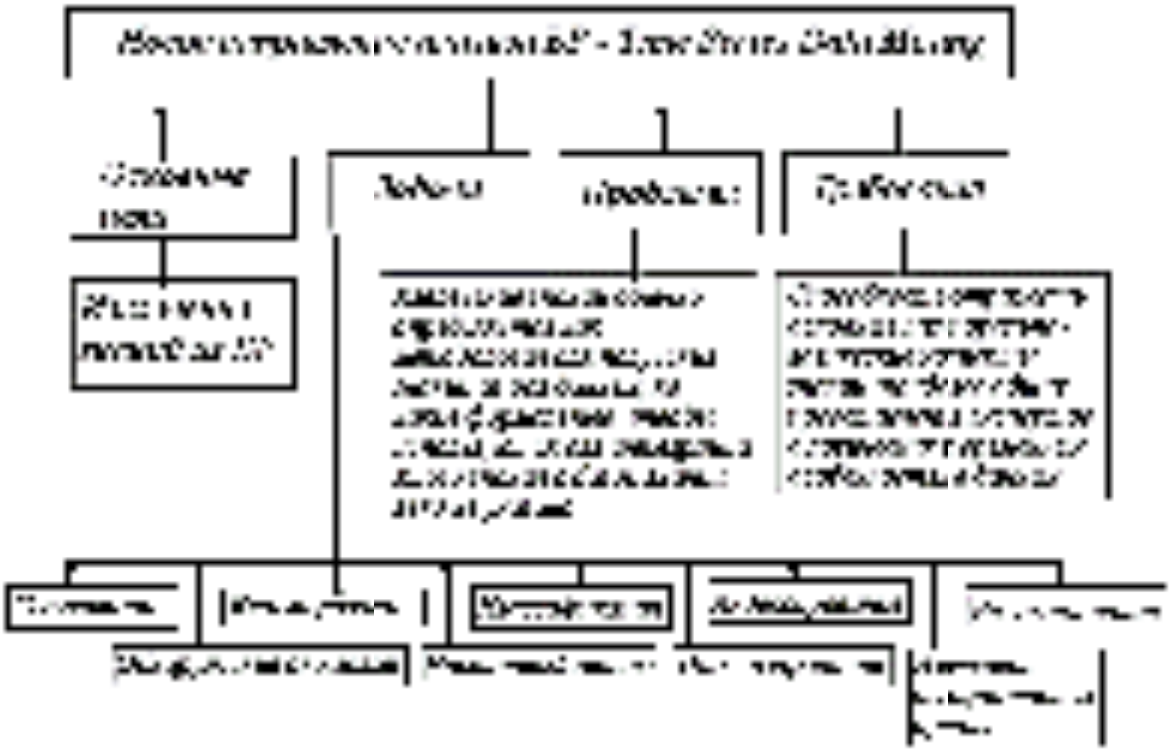


Рисунок 22. Задачи DataMining временных рядов

Традиционное выделение паттернов ВР было связано с выделением участков с постоянным знаком первой и второй производной: возрастающий и выпуклый, убывающий и гладкий и др. Различные шкалы и методы нечетких вычислений Л.Заде использовались для описания паттернов линейных трендов: рост, падение, резкий рост, медленное падение и т. д. Параметрические методы выпукло-гладкой модификации линейных функций и нечеткая грануляция выпукло-гладких паттернов позволили получить лингвистическое описание для ВР, подобное следующему: медленно убывающий и строго гладкий.

В рамках описанного выше направления TSDM акцент делается на поиск и извлечение правил из ВР, при этом полагаются на следующие основные принципы:

- Поиск правил нацелен на получение понимаемых результатов и не обязательно самых точных прогнозов;
- Важнейшим шагом на пути извлечения интерпретируемых знаний является порождение описаний фрагментов ВР в форме темпоральных образов, допускающих естественно-языковое толкование.

Требование к модели представления – способность отражать основные темпорально-логические концепты знаний (наиболее общие представления экспертов о логических и временных особенностях в данных).

Виды темпорально-логических концептов следующие:

- концепт временной продолжительности – присутствие определенного паттерна или признака ВР на определенном интервале времени;
- концепт очередности – порядок следования паттернов ВР во времени;
- концепт одновременности – совпадение во времени темпоральных событий (паттернов различных ВР);

- концепт нечеткости – нечеткость выраженности темпоральных событий и отношений.

В качестве инструментов анализа предлагается использовать нечеткие нейронные сети.

Существует множество методов прогнозирования временного ряда. Однако не создан лучший в любой ситуации метод: каждый из них имеет свои достоинства и недостатки. Поэтому одним из наиболее перспективных научных направлений стало создание комбинированных моделей.

Комбинированная модель прогнозирования – модель прогнозирования, состоящая из нескольких индивидуальных (частных) моделей, называемых базовым набором моделей. Использование комбинированной модели может повысить точность получаемого прогноза по ряду причин:

- Попытка выбрать одну модель для временного ряда с изменяющимся уровнем и динамическими свойствами приводит к выбору усредненной модели;
- При быстром изменении уровней ряда и его динамических свойств невозможно быстро производить анализ динамики и заменять одну модель прогнозирования другой;
- Любой отвергнутый из-за неоптимальности прогноз почти всегда содержит полезную независимую информацию;
- Слабые стороны одного метода прогнозирования возможно преодолеть, используя преимущества другого метода.

На кафедре «Информационные системы» Ульяновского государственного технического университета была разработана информационная система «Combination of fuzzy and exponential models». Данная система позволяет получать прогноз для временного ряда путем агрегации индивидуальных прогнозов моделей из базового набора. На момент создания системы базовый набор содержал 29 индивидуальных моделей и их модификаций, относящихся к экспоненциальным и нечетким моделям прогнозирования временных

рядов. Разработанная информационная система одержала победу на конкурсе «Computational Intelligence in Forecasting» в 2015 году [23].

Точность агрегированного прогноза напрямую зависит от выбранных из базового набора моделей. В связи с этим можно утверждать, что информационная система «Combination of fuzzy and exponential models» имеет два существенных недостатка:

- Пользователь вынужден самостоятельно выбирать модели из базового набора для каждого прогнозируемого временного ряда;
- Для всех временных рядов используется один агрегирующий метод.

Таким образом, целесообразно использовать методы машинного обучения для выбора моделей из базового набора и для выбора агрегирующего метода. Указанные задачи можно решить с помощью нейронной сети, выбирающей методы на основании значений метрик прогнозируемого временного ряда.

Для выбора метрик необходимо выбрать совокупность значимых для прогнозирования временного ряда характеристик, таких как: длина, степень выраженности тренда, степень выраженности сезонности, величина дисперсии, наличие стационарности.

При выборе используемых метрик временного ряда необходимо выполнение следующих условий:

- Метрика соответствует одной из основных характеристик временного ряда;
- Совокупность метрик максимально и разносторонне опиисывает временной ряд;
- Значение метрики принадлежит интервалу от 0 до 1.

При выборе методов прогнозирования из базового набора обучающая выборка нейронной сети представляет собой набор временных рядов, представленных в виде совокупности метрик, и соответствующих значений ошибки прогнозирования для каждой модели. Нейронная сеть, таким образом, обучается на основании значений метрик вычислять предполагаемые значения ошибки для каждой

модели из базового набора, что позволяет определять, какие модели из базового набора эффективнее применить для прогнозирования значений текущего временного ряда.

Для создания комбинированной модели прогнозирования на основании использования методов машинного обучения необходимо решить ряд задач:

- Определить состав моделей, входящих в базовый набор;
- Выявить ключевые для задачи прогнозирования метрики временного ряда;
- Определить структуру и способ обучения нейронных сетей выбора методов прогнозирования из базового набора и выбора метода агрегации.

Нечеткое сглаживание временного ряда

Очень часто при анализе временных рядов переходят от исследования самих значений ряда к исследованию тренда. Одним из методов его выделения является метод F-преобразования.

Нечеткое сглаживание временных рядов на основе нечеткого преобразования (F-преобразования) – методика, разработанная Перфильевой [24], которая может быть отнесена к методикам нечеткого приближения.

F-преобразование предполагает задание нечеткого разбиения универсального множества. В качестве последнего выбирается конечный интервал [a,b] действительной прямой. Зафиксируем значение n $(n > 2)$ узлов $x_1,...,x_n$ на [a,b] и предположим, что $x_1 \prec ... \prec x_n$, причем $a = x_1, b = x_n$.

Под нечетким разбиением [a,b] будем понимать совокупность n функций $A_1,...,A_n : [a,b] \to [0,1]$, удовлетворяющих следующим свойствам:

- $A_k : [a,b] \to [0,1], A_k(x_k) = 1$;

- $A_k(x) = 0$ если $x \notin (x_{k-1}, x_{k+1})$, где для единообразия обозначения мы положим $x_0 = a, x_{n+1} = b$;
- $A_k(x)$ непрерывна;
- $A_k(x), k = 2,...,n$ строго возрастает на $[x_{k-1}, x_k]$ и строго убывает на $[x_k, x_{k+1}]$;
- $\sum A_k(x) = 1$ для всех $x \in [a,b]$.

Функции $A_1,...,A_n$ называются базисными функциями. Базисные функции $A_1,...,A_n$ могут служить также функциями принадлежности нечетких подмножеств $A_1,...,A_n$ (обозначения функций и множеств унифицированы). Отметим, что форма базисных функций может быть уточнена дополнительно и согласована с такими требованиями к модели, как, например, гладкость.

Следующие формулы представляют нечеткое разбиение отрезка $[x_1, x_n]$, полученное совокупностью функций:

$$A_1(x) = \begin{cases} 1 - \dfrac{(x - x_1)}{h_1}, x \in [x_1, x_2] \\ 0, иначе \end{cases}$$

$$A_k(x) = \begin{cases} \dfrac{(x - x_k)}{h_{k-1}}, x \in [x_{k-1}, x_k], \\ 1 - \dfrac{(x - x_k)}{h_k}, x \in [x_k, x_{k+1}], \\ 0, иначе \end{cases}$$

$$A_{n-1}(x) = \begin{cases} 1 - \dfrac{(x - x_{n-1})}{h_{n-1}}, x \in [x_{n-1}, x_n] \\ 0, иначе \end{cases}$$

где $k = 1,...,n-1$ и $h_k = x_{k+1} - x_k$.

Предположим, что функция *f* имеет своей областью определения множество $P = \{p_1,..., p_l\} \subset [a,b]$, где $l>n$. Множество P считается плотным относительно нечеткого разбиения $A_1,...,A_n$, если выполнено условие:

$$(\forall k)(\forall j) A_k(p_j) > 0 \quad .$$

Пусть $A_k(p_j) = a_{kj}, k = 1,...,n; j = 1,...,l$, тогда матрица $A_{n\times l} = a_{kj}$ называется матрицей нечеткого разбиения для *P*, для которой справедливы свойства:

$$(\forall k)(\forall j) a_{kj} \in [0,1]$$

$$(\forall j)\sum_{k=1}^{n} a_{kj} = 1 .$$

F-преобразованием вектора *f*, определяемым матрицей нечеткого разбиения *A*, назовем вектор $F_n[f]$, где $F_n[f] = (F_1...F_n)$ и $F_i = \dfrac{\sum_{j=1}^{l} a_{ij} f_j}{\sum_{i=1}^{l} a_{ij}}$.

Координаты вектора $F_n[f]$ назовем соответственно компонентами F-преобразования. Обозначим $a_i = \sum_{j=1}^{l} a_{ij}, i = 1,..., n$; тогда $(a_1F_1,..., a_nF_n)^T = A \times f$. Компоненты F-преобразования являются точками минимума функции, задающей критерий взвешенного среднеквадратичного отклонения.

Пусть $F_n[f]$ есть F-преобразование *f*, определяемое $A_{n\times l} = a_{kj}$. Обратным F-преобразованием $F_n[f]$ назовем вектор $f_{F,n}$, вычисляемый по формуле $f_{F,n}^T = F_n[f] \times A$. Можно доказать, что если *n* возрастает, тогда $f_{F,n}(p_j)$ сходится $f(p_j), j = 1,..., N$.

F-преобразование имеет (кроме прочих) следующие свойства, важные для использования в качестве сглаживания временных рядов:

- У него прекрасные фильтрующие свойства;
- Его легко вычислять
- F-преобразование стабильно относительно выбора точек $p_1,...,p_N$. Это означает, что при выборе других точек p_k (и, возможно, изменяя их число *N*), результирующая функция $f_{F,n}$ значительно не меняется.

Прогнозирование тренда и прогнозирование числового представления временного ряда производится раздельно. Для этого необходимо вычислить так называемые остатки – разность между временным рядом и его трендом:

$$R = \{(f(p_j) - F_k)\}, \text{где } j : A_k(j) > 0 .$$

Полученный вектор *R*-вектор остатков для *k*-й компоненты тренда. Прогноз тренда и векторов остатков реализуется по формуле линейной комбинации. Прогноз компоненты тренда:

$$F_{k+1} = \alpha F_k + \beta F_{k-1}...$$

и прогноз вектора остатков:

$$R_{k+1} = \alpha R_k + \beta R_{k-1}... .$$

Для получения прогноза находится решение системы уравнений:

$$\begin{cases} F_{k-1} = \alpha F_{k-2} + \beta F_{k-3}... \\ F_k = \alpha F_{k-1} + \beta F_{k-2}... \end{cases}$$

Аналогично строится прогноз вектора остатков.

Также для построения прогноза может быть использована нейронная сеть, например, многослойный перцептрон или сеть Кохонена с выходной звездой Гроссберга. Вид нейронной сети определяет вид входных данных: на вход многослойного перцептрона подаются только абсолютные значения ($F_k, F_{k-1,...}$), на вход сети Кохонена подаются значения точек тренда, вычисленные относительно друг друга ($F_k - F_{k-1}, F_{k-1} - F_{k-2},....$).

На качество прогноза влияет количество нейронов во внутреннем слое. После получения прогнозных значений тренда и вектора остатков возможно построение числового представления прогноза временного ряда. Для этого производится сложение прогнозной компоненты тренда и прогнозного вектора остатков.

Точность прогноза зависит от количества точек временного ряда, участвующих в обучении: чем больше их, тем меньше ошибка (рисунок 23). Как видно из рисунка, вывод справедлив и для MAPE, и для SMAPE (линии на графике практически совпадают).

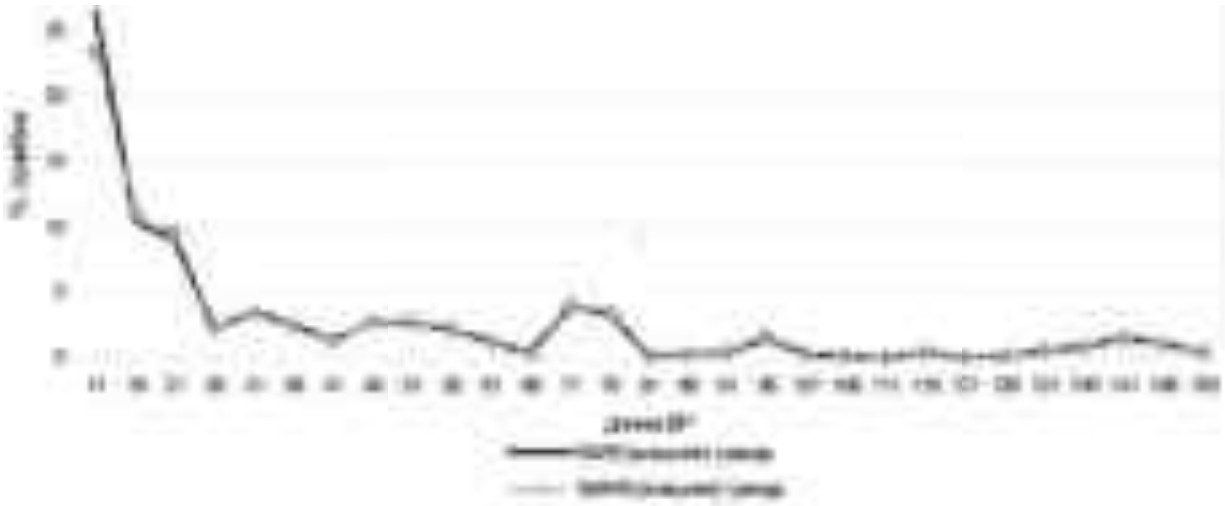


Рисунок 23. Зависимость процента ошибок от длины обучающей выборки

Процесс получения прогноза тренда и остатков зависит от некоторых параметров, которые в модели не удается задать жестко. Выбор данных параметров в значительной степени влияет на качество прогноза. Определим их:

- Степень авторегрессии при построении прогноза тренда (область значений);
- Метод, которым производится прогноз:
    - решение системы линейных уравнений;
    - нейронная сеть с абсолютными значениями предыдущих компонент на входе;
    - нейронная сеть с разностями между предыдущими компонентами на входе;

    - нейронная сеть с абсолютными значениями предыдущих компонент, а также разности между ними на входе;
- количество точек, покрываемых базисной функцией;
- сезонность (история отстоит на k точек от прогноза).

Сочетание параметров, при котором получается наилучший прогноз, получается перебором.

Нечеткая регрессия

В области прикладной статистики, анализа временных рядов и принятия решений в условиях неопределенности накоплен богатый опыт исследований и существует множество моделей, начиная от простейших линейных регрессионных моделей поиска тренда временного ряда и заканчивая сложными многоуровневыми авторегрессионными и адаптационными моделями. Регрессионный анализ, основанный на методе наименьших квадратов (Least-square), является очень удобным методом построения моделей, позволяющим численно оценивать зависимость интересующего исследователя параметра от воздействующих на него факторов. При анализе зависимости нечетких оценок от воздействующих факторов зачастую исследователям приходится иметь дело с важной информацией, которая не может быть задана точно. Некоторые наблюдения могут быть описаны только лингвистическими выражениями (типа «удовлетворительный», «хороший» и «превосходный»). Для таких данных аппаратом формализации может служить теория нечетких множеств. Были разработаны различные нечеткие регрессионные модели, основой которых является модель нечеткой линейной регрессии.

В нечеткой регрессионной модели параметры представляются триангулярными нечеткими числами и являются коэффициентами в нечеткой линейной функции. Неопределенность (vagueness) системы представляется суммарным разбросом («шириной») параметров (нечетких коэффициентов).

Построение модели состоит в нахождении оптимальных в некотором смысле коэффициентов с учетом нечеткой информации об объекте и субъективных представлений исследователя.

Базовые предположения нечеткой регрессии заключаются в том, что остатки, полученные как разность между наблюдениями и их оценками, продуцируются не случайными ошибками измерения, а неопределенностями (типа нечеткость) при вычислении параметров модели.

Большинство работ, посвященных нечеткой регрессии, были основаны на следующих базовых определениях.

Пусть дано множество наблюдений: $(y_j, x_{j1}, ...x_{jn}), j = 1,..., m$, необходимо найти нечеткую модель по следующей форме:

$$\tilde{Y} = A_0^{\sim} + A_1^{\sim} x_1 + ... + A_n^{\sim} x_n,$$

где $A_i(a_i^c, s_i^L, s_i^R), i = 1,..., n$ – триангулярные нечеткие числа;

a_i^c – среднее значение $A_i^{\sim}$;

s_i^L, s_i^R – показывают левый и правый разброс треугольной функции принадлежности соответственно.

Используются два критерия определения нечетких коэффициентов модели:

1. Для всех наблюдений принадлежность значения y_j к его нечеткой оценке $Y_j^{\sim}$ должна быть как минимум $Y_j^{\sim}(y_j) \geq h, j = 1,..., m$, где h – уровень доверия, выбранный лицом, принимающим решения.
2. Общая нечеткость предсказываемого значения зависимой переменной должна быть минимизирована. Это может быть достигнуто минимизацией суммы разбросов нечетких чисел для всех наборов данных.

Итак, проблему настройки нечеткой модели с заданными данными $(y_j, x_{j1},...x_{jn}), j = 1,..., m$ можно решить как эквивалентную задачу линейного программирования. То есть найти $a_-^c = (a_0^c,...a_n^c)$, $s_-^L = (s_0^L,...s_n^L)$, $s_-^R = (s_0^R,...s_n^R)$, которые минимизируют выражение:

$$Z = m(s_0^L + s_0^R)s_0^L + \sum_{I=1}^{n}\left[\left(s_i^L + s_i^R\right)\sum_{j=1}^{m}\left|x_{ij}\right|\right].$$

Чтобы оценить качество настройки нечеткой регрессии, используют метод наименьших квадратов. Для нечеткой регрессии среднеквадратичное отклонение (MSE) определяется следующим образом:

$$MSE = \frac{1}{m}\sum_{j=1}^{n}[y_j - def(Y_j)]^2 ,$$

где $def(Y_j)$ – дефаззифицированное значение зависимой переменной.

Исследователями были выделены различные варианты методов на основе классификации «вход – выход»: «четкий вход – четкий выход» метод CICO (Crisp-Input and CrispOutput), «нечеткий вход – нечеткий выход» метод FIFO (Fuzzy-Inputs and Fuzzy-Outputs) и смешанные данные – метод CIFO (Crisp-Inputs and Fuzzy-Outputs).

ACL-шкала и нечеткая кластеризация объектов

Для задания отношений между элементами нечеткого временного ряда мы можем использовать специальную лингвистическую шкалу в качестве инструмента как абсолютного, так и сравнительного нечеткого оценивания – ACL-шкала (Absolute&Comparative Linguistic) [25]. Абсолютные оценки, полученные по ACL-шкале, будут соответствовать нечетким меткам уровней ВР, а сравнительные оценки – нечетким тенденциям НВР. Опишем ACL-шкалу [25] как алгебраическую систему вида

$$Sx = <Name\_Sx, \tilde{X}, N, X, G, P, TTend, RTend>,$$

где *Name_ Sx* – имя ACL-шкалы; $\tilde{X}$ – базовое терм-множество абсолютных лингвистических оценок (лингвистическое название градаций); *N* – мощность базового терм-множества шкалы; *X* – универсальное множество, на котором определена шкала; *G* – синтаксические правила вывода (порождения) цепочек оценочных высказываний (производные термов, не входящих в базовое терм-множество); *P* – семантические правила, определяющие функции принадлежности для каждого терма (задаются обычно экспертно); $TTend(\tilde{x}_i,\tilde{x}_j)$ – лингвистическое отношение, фиксирующее тип изменения между двумя оценками $\tilde{x}_i,\tilde{x}_j$ шкалы; $RTend(\tilde{x}_i,\tilde{x}_j)$ – лингвистическое отношение, фиксирующее интенсивность различия между двумя оценками $\tilde{x}_i,\tilde{x}_j$ шкалы.

Первое, что нужно сделать для построения шкалы, – определить базовое терм-множество. Например, определим множество мощностью 5, содержащее в себе оценки уровней тренда исходного временного ряда: {малое, ниже среднего, среднее, выше среднего, большое}. Мощность множества обуславливается необходимостью вербализации каждого понятия шкалы, таким образом, чтобы она была понятна для человека и семантически содержательна.

Далее необходимо выбрать способ построения ACL-шкалы. Этим способом может стать неравномерное разбиение зафиксированной области значения величины с помощью алгоритма кластеризации. Достоинством такого подхода является то, что разбиение происходит без участия эксперта (тогда как в остальных возможных способах обязательно участие эксперта).

При безэкспертном разбиении задача формулируется следующим образом: дано множество из *L* точек, необходимо разбить его на *k* групп. Здесь *L* – количество элементов ВР (зафиксированное множество значений), *k* – количество термов на ACL-шкале, в нашем случае равное *N*. Как мы видим по формулировке задачи, это задача кластеризации. Кластеризация – это процесс разбиения объектов на группы по степени их схожести между собой. В отличие от

классификации, где есть заданная структура групп и признаки, на основе которых объекты в эти группы помещаются, в кластеризации структура разбиения, а также характеристические признаки являются не входными, а выходными данными. Кластеризация может быть четкой и нечеткой. При четкой кластеризации предполагается, что каждый объект принадлежит только одному кластеру, при нечеткой объект принадлежит всем кластерам, но с разной степенью принадлежности. При нечеткой кластеризации мы сможем использовать собственно степени принадлежности объектов ВР кластерам для получения термов шкалы. В итоге вся зафиксированная область разобьется на N взаимопересекающихся кластеров, упорядоченных по возрастанию значений их центров. Взаимопересечение получается за счет нечеткости кластеризации.

Кластеризация может выполняться с помощью нейронных сетей, генетических алгоритмов или с помощью собственно алгоритмов кластеризации. Например, для нечеткого разбиения некого множества точек на заданное количество кластеров можно использовать fcm-алгоритм кластеризации. Рассмотрим его подробнее.

FCM-алгоритм (Fuzzy Classifier Means) кластеризации применяется для нечеткого разбиения некого множества объектов на заданное количество кластеров. Целью алгоритма кластеризации является автоматическое разбиение множества объектов, которые задаются векторами признаков в пространстве признаков. Этот алгоритм предполагает, что объекты принадлежат всем кластерам с определенной степенью принадлежности, которая определяется расстоянием от объекта до соответствующих кластерных центров. Данный алгоритм итерационно вычисляет центры кластеров и новые степени принадлежности объектов. При этом он основан на минимизации целевой функции по мере расстояния объектов от центров кластеров:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{C} (u_{ij})^m \left\| x_i - c_j \right\| \quad ,$$

где N – количество объектов; C – количество кластеров; u_{ij} – степень принадлежности объекта i кластеру j; m – любое действительное число, большее 1; x_i– i-й объект набора объектов; c_j– j-й кластер набора кластеров; $||x_i - c_j||$ – норма, характеризующая расстояние от центра кластера j до объекта i.

Объектами кластеризации при анализе текстов, например, могут являться термины. Вектор признаков объектов кластеризации в данном случае содержит только значение какой-либо статистической метрики (например, частоты термина в тексте).

Алгоритм нечеткой кластеризации выполняется по шагам. Рассмотрим каждый из шагов подробнее, приведя необходимые модификации целевой функции для упрощения дальнейшего программирования.

Первый шаг – инициализация. На этом шаге задаются параметры кластеризации и инициализируется первоначальная матрица принадлежности объектов кластерам. К параметрам относятся следующие величины. Во-первых, степень нечеткости кластеризации – параметр m. От выбора этого параметра зависит значение функции принадлежности точки кластеру. Обычно он выбирается в пределах ~1.5..2. Чем больше его значение, тем более «нечеткая» кластеризация. В приведенных ниже примерах кода эмпирически было выбрано значение = 1.6. Во-вторых, выбирается мера расстояний $||x_i - c_j||$. Она характеризует степень близости точки к центру кластера. Если вектор характеристик состоит только из одного значения и задача, по сути, сводится к выделению значимых интервалов на прямой, то мера расстояния может иметь вид

$$||x_i - c_j|| = |x_i - c_j|.$$

Эта мера характеризует удаленность точки от центра кластера на прямой.

Третий параметр, который выбирается при инициализации – параметр сходимости алгоритма ε (уровень точности). Когда разность значений целевых функций текущей и предыдущей итераций достиг-

нет этого уровня, считается, что кластеризация завершена. Обычно этот параметр равен 0.001.

Четвертый параметр задается во избежание зависания алгоритма при невозможности достижения уровня точности. Это количество итераций алгоритма. Например, 10 000.

После выбора параметров генерируется случайным образом первоначальная матрица принадлежности объектов кластерам и происходит переход ко второму шагу алгоритма.

На шаге 2 происходит вычисление центров кластеров следующим образом:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{1.6} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{1.6}},$$

где c_j – центр j-го кластера; N – количество объектов; x_i – значение i-го объекта; u_{ij} – степень принадлежности объекта i кластеру j.

На третьем шаге формируется новая матрица принадлежности с учетом вычисленных на предыдущем шаге центров кластеров:

$$u_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_l\|}\right)^{3.33}},$$

где u_{ij} – степень принадлежности объекта i кластеру j; c – количество кластеров; c_j – вектор центра j-го кластера; c_l – вектор центра l-го кластера.

При этом если для некоторого кластера j и некоторого объекта i расстояние $|x_i - c_j|=0$, тогда полагаем, что степень принадлежности u_{ij} равна 1, а для всех остальных кластеров степень принадлежности этого объекта равна нулю.

Далее на четвертом шаге вычисляется значение целевой функции, и полученное значение сравнивается со значением на преды-

дущей итерации. Если разность не превышает заданного в параметрах кластеризации значения ε, считаем, что кластеризация завершена. В противном случае переходим ко второму шагу алгоритма.

Мы рассмотрели методы нечеткой логики, теперь перейдем к следующей модели – искусственных нейронных сетей (ИНС).

Искусственные нейронные сети

Для подготовки данного раздела использовался материал (в том числе иллюстративный) из следующего источника: [26].

Особенности нейронных сетей

Прежде чем перейти к описанию того, что собой представляет модель искусственной нейронной сети, акцентируем внимание на особенностях данного подхода, а в частности его достоинствах и недостатках. Это основные моменты, которые обязательно необходимо понять и запомнить.

Достоинства ИНС:

1. Нелинейность модели, что дает возможность аппроксимировать любые нелинейные функции. Та же задача с помощью полиномиальной модели не всегда решается, либо же ее решение становится низким по качеству.

2. Локальности восприятия, заключающаяся в том, что каждый нейрон получает не весь входной вектор. Это позволяет ей сегментировать данные и работать с более сложными ситуациями.

3. Каскад слоев. В сочетании с пунктом 2 это дает способность воспринимать более абстрактные признаки.

4. Лучше работает с мультиколлинеарностью и комбинациями признаков.

5. Нейронная сеть дает несколько механизмов для контроля переобучения.

6. Нейронная сеть способна дообучаться при непротиворечивости новых образов.

Недостатки:

Нейронная сеть не интерпретируема. Поэтому не ясна логика преобразования входных данных в выходные. Не всегда можно убедиться в стабильности решения на всей области определения.

Математика процессов и некоторых аспектов функционирования еще недостаточно изучена. Часто количество нейронов и слоев приходиться подбирать экспериментально для конкретной задачи, потому что определенной методики нет.

Нет аналитического решения, а решение численными методами градиентного спуска может приводить к попаданию в локальные минимумы ошибки.

Для ИНС очень часто требуется большая выборка. Так для распознавания речи одного языка может потребоваться от 5 до 10 тыс. часов размеченных записей. Для Глубоких сетей необходимо еще большее число объектов обучающей выборки, иначе будет постоянно происходить переобучение.

Обучение большой ИНС требует больших вычислительных ресурсов и специального оборудования GPU.

Поскольку на сегодняшний день нейронные сети очень модная тема, следует отметить, что ИНС, так же как любые обобщающие модели машинного обучения, чувствительна к противоречиям в данных и, конечно же, на текущем этапе развития НС не могут содержать противоречивые «представления» об объекте, проводить какие-то внутренние размышления и перестановки. Иными словами, из неоткуда информацию ИНС не возьмет (самостоятельно не проделает ряд выводов).

Так одним из главных конкурентов ИНС в ряде задач являются решающие деревья и ансамбли над решающими деревьями (например, XGBoost). Исключением являются задачи машинного зрения и работы с изображениями вообще, в которых ИНС занимаются сегодня лидирующие позиции.

Определение модели искусственной нейронной сети

Согласно Википедии [1]: «Искусственная нейронная сеть – математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети, созданные У. Мак-Каллоком и У. Питтсом. После разработки алгоритмов обуче-ния получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.».

Искусственные нейроны сильно упрощены по сравнению с их биологическими прототипами. Каждый из них имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим нейронам. Способность решать довольно сложные задачи обуславливается тем, что все нейроны соединены в достаточно большую сеть с управляемым взаимодействием. Схему простой нейронной сети можно увидеть на рисунке 24. Литерой «I» обозначены входные нейроны, «S» – скрытые нейроны, «O» – выходной нейрон. Соединяющие линии – связи между нейронами, или синаптические связи. Именно синаптические связи играют важнейшую роль в модели ИНС, так как в основе обучения лежит механизм корректировок силы этих связей.

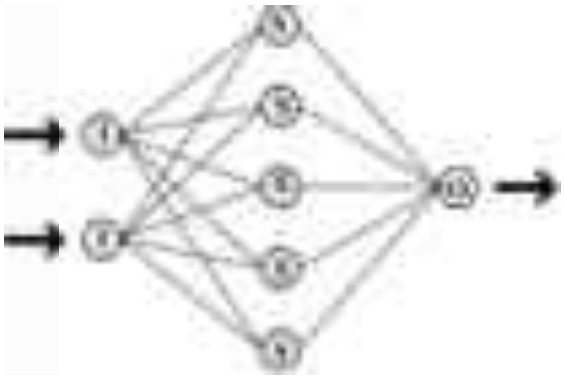


Рисунок 24. Схема простой нейросети

Нейронную сеть можно рассмотреть с разных точек зрения. Например, в машинном обучении, нейронная сеть – частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С точки зрения математики, обученная нейронная сеть – решенная многопараметрическая задача нелинейной оптимизации. Кибернетика представляет нейронную сеть как «черный ящик» или произвольную передаточную функцию, которую можно получить при обучении сети и затем воспроизводить при использовании сети. С точки же зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Очень часто именно про нейронные сети говорится, что они не программируются в привычном смысле этого слова, они обучаются и что возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Конечно же, такие трактовки и объяснения нельзя назвать точными и их можно допускать только при первом знакомстве с машинным обучением вообще.

Термин «обучение» необходимо трактовать лишь математически. В таком случае и Регрессия, и Деревья также обучаются, хотя в любой модели происходит одно и то же – корректировка весов.

Однако шумиха вокруг нейронных сетей порождает массу мифов, принижающих остальные методы, изученные нами ранее.

В чем-то эта модель напоминает уже изученные модели, такие как Регрессия и Деревья решений. В каком-то смысле Нейронные сети являются комбинацией этих методов (на уровне концепции).

В основе математической модели вычислений в ИНС лежит взвешенное суммирование, то есть, по сути, уже знакомое вам полиномиальное уравнение (в котором коэффициенты уже принято называть весами). Веса этого полинома и отражают силу синаптической связи между нейронами. Именно синаптические связи играют важнейшую роль в модели ИНС, так как в основе обучения лежит

механизм корректировок силы этих связей (подобно биологическому прототипу).

Все также вычисления завязаны на понятии ошибки и на том принципе, что корректное обобщение (отображение X=>Y) строится посредством уменьшения этой ошибки. Таким образом, они являются логическим продолжением прошлых тем.

Однако стоит отметить, что в отличие от методов Байеса и Регрессии нейронные сети изначально разрабатывались исходя из кибернетического подхода к моделям и информации. То есть нейронные модели – это некая адаптивная система, которая может прийти из одного состояния в требуемое, но при этом имеется некая случайная составляющая, которая вносит нестабильность в решения. Также нет хорошей компактной формулы, описывающей работу сети, – с самого начала суть вычислений в нейронных сетях была основана на численном (постепенном\итеративном) подходе к уменьшению ошибки.

Первая формальная модель и первая реализация нейронной сети

Согласно [27]: «Первой формальной моделью нейронных сетей (НС) и нейрона вообще была модель МакКаллока-Питтса, уточненная и развитая Клини. Впервые было установлено, что НС могут выполнять любые логические операции и вообще любые преобразования, реализуемые дискретными устройствами с конечной памятью. Эта модель легла в основу теории логических сетей и конечных автоматов и активно использовалась психологами и нейрофизиологами при моделировании некоторых локальных процессов нервной деятельности. В силу своей дискретности она вполне согласуется с компьютерной парадигмой и, более того, служит ее «нейронным фундаментом».

Пусть имеется n входных величин $x_1,\ldots,x_n$ бинарных признаков, описывающих объект x. Значения этих признаков будем трактовать как величины импульсов, поступающих на вход нейрона через n входных синапсов. Будем считать, что, попадая в нейрон, импульсы складываются с весами $\omega_1,\ldots,\omega_n$.

Если вес положительный, то соответствующий синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный импульс превышает заданный порог активации ω_0, то нейрон возбуждается и выдает на выходе 1, иначе выдается 0.

Таким образом, нейрон вычисляет n-арную булеву функцию

$$a(x) = \varphi(\sum_{j=1}^{n} w_j x^j - w_0),$$

где $\varphi(z) = [z \geq 0]$ – ступенчатая функция Хевисайда.

В теории нейронных сетей функцию φ, преобразующую значение суммарного импульса в выходное значение нейрона, принято называть функцией активации. Таким образом, модель МакКаллока-Питтса эквивалентна пороговому линейному классификатору. Схема ее вычислений показана на рисунке 25.

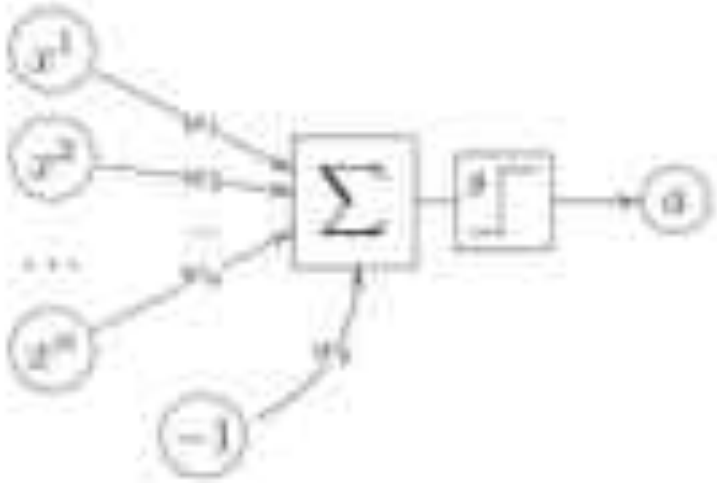


Рисунок 25. Схема вычислений в модели нейрона МакКаллока-Питтса

Получается, что нейрон производит взвешивание входных признаков (то есть рассчитывает результат с учетом их важности\веса) подобно регрессии. Однако важно понимать, что это пока всего лишь один нейрон.

Теоретические основы нейроматематики были заложены в начале 40-х годов, и в 1943 году У. МакКаллок и его ученик У. Питтс сформулировали основные положения теории деятельности головного мозга.

Ими были получены следующие результаты:

- разработана модель нейрона как простейшего процессорного элемента, выполняющего вычисление переходной функции от скалярного произведения вектора входных сигналов и вектора весовых коэффициентов;
- предложена конструкция сети таких элементов для выполнения логических и арифметических операций;
- сделано основополагающее предположение о том, что такая сеть способна обучаться, распознавать образы, обобщать полученную информацию.

Недостатком данной модели является то, что в качестве функции активации используется функция Хевисайда или униполярная пороговая функция (рисунок 26). В формализме, предложенном МакКаллоком и Питтсом, нейроны имеют состояния 0, 1 и пороговую логику перехода из состояния в состояние. Каждый нейрон в сети определяет взвешенную сумму состояний всех других нейронов и сравнивает ее с порогом, чтобы определить свое собственное состояние.

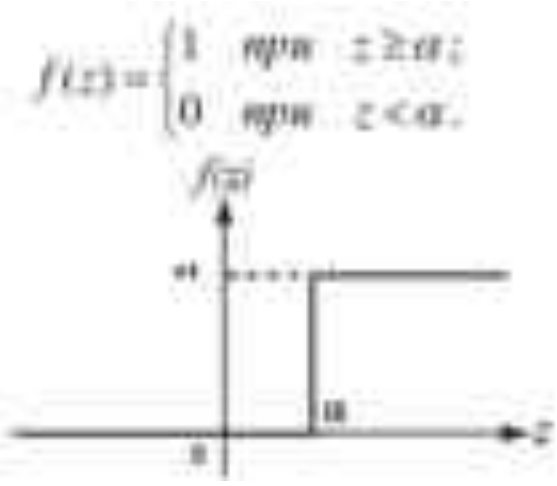


Рисунок 26. Пороговая функция, или функция Хевисайда

Пороговый вид функции не предоставляет нейронной сети достаточную гибкость при обучении и настройке на заданную задачу. Если значение вычисленного скалярного произведения, даже незначительно, не достигает до заданного порога, то выходной сигнал не

формируется вовсе, и нейрон «не срабатывает». Это значит, что теряется интенсивность выходного сигнала (аксона) данного нейрона и, следовательно, формируется невысокое значение уровня на взвешенных входах в следующем слое нейронов.

К тому же модель не учитывает многих особенностей работы реальных нейронов (импульсного характера активности, нелинейности суммирования входной информации, рефрактерности).

Несмотря на то, что за прошедшие годы нейроматематика ушла далеко вперед, многие утверждения МакКаллока остаются актуальными и поныне. В частности, при большом разнообразии моделей нейронов принцип их действия, заложенный МакКаллоком и Питтсом, остается неизменным».

Первой реализацией модели нейронной сети была созданная в 1960 электронная машина «Марк-1». Идею предложил Фрэнк Розенблатт в 1957. Согласно Википедии [1]: «Несмотря на то, что это первая модель и концепция, тем ни менее многие идеи современных нейронных сетей во многом совпадают с концепцией персептрона. Так выше была рассмотрена модель упрощенного искусственного нейрона, способного производить вычисления. Персептрон по своей сути является сетью таких нейронов. Из чего и предполагалось получать механизм для более сложных вычислений.

Несмотря на свою простоту, персептрон способен обучаться и решать довольно сложные задачи. Основная математическая задача, с которой он справляется, – это линейное разделение любых нелинейных множеств, так называемое обеспечение линейной сепарабельности. Логическая схема персептрона с тремя выходами представлена на рисунке 27.

Персептрон состоит из трех типов элементов, а именно: поступающие от датчиков сигналы передаются ассоциативным элементам, а затем реагирующим элементам. Таким образом, персептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе.

В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов. Согласно современной терминологии, персептроны могут быть классифицированы как искусственные нейронные сети:

- с одним скрытым слоем;
- с пороговой передаточной функцией;
- с прямым распространением сигнала».

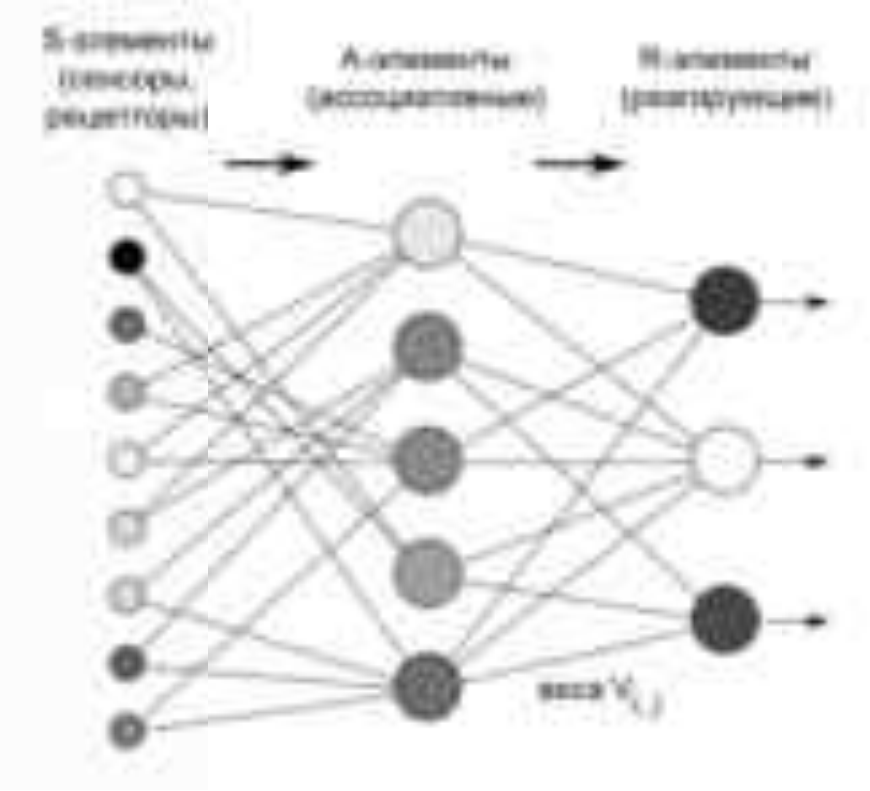


Рисунок 27. Логическая схема персептрона с тремя выходами

Многослойный персептрон (MLP)

Данный тип\архитектура является самой распространенной архитектурой искусственных нейронных сетей и в каком-то смысле классической. Очень часто, когда речь идет о нейронных сетях вообще, то имеется в виду именно многослойная сеть прямого распространения. Многие задачи классификации, аппроксимации и управления, которые решаются с помощью нейронных сетей, реша-

ются именно этим типом сети. Кроме того, принципы обучения, разработанные для этого типа сети, впоследствии стали применяться и для других типов. Так что в каком-то смысле это базовая архитектура для других типов сетей.

Если подходить к определению строго, то не совсем корректно называть такой тип сети Персептроном, ведь она отличается от него не только количеством слоев, но и тем, что:

- Нейроны имеют не ступенчатую функцию активации Хевисайда, а сигмоидальную функцию.
- Обучение производится не по правилу Хебба, а с помощью обратного распространения ошибки.

Поэтому такую архитектуру называют либо просто многослойной сетью, либо сетью прямого распространения или многослойной сетью прямого распространения. Но в то же время можно встретить и более удобное название MLP, что значит многослойный персептрон. Такое название определяет, что это все-таки не просто сеть. Структура MLP представлена на рисунке 28.

Замечание: выше уже упоминалось о том, что терминология в машинном обучении и в нейросетевых технологиях в частности еще не до конца систематизирована. Поэтому необходимо разбираться в сути моделей, допуская определенную вариативность названий. Однако со временем это перестает доставлять какие-либо трудности.

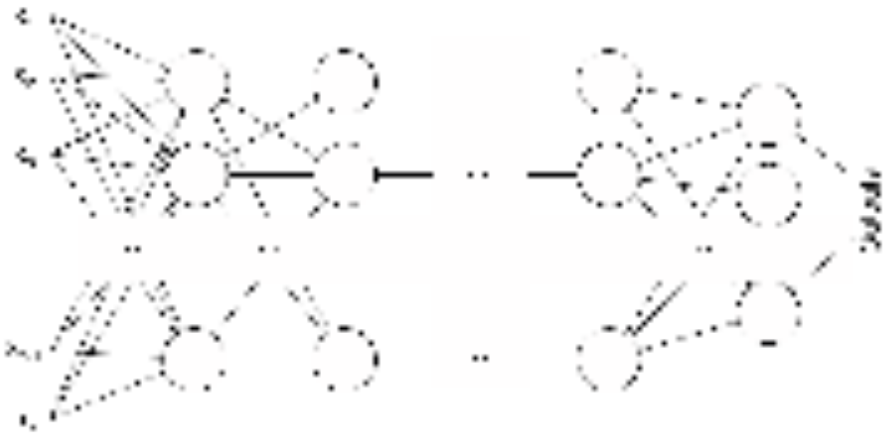


Рисунок 28. Структура многослойного персептрона

На рисунке 29 представлен график сигмоидальной функции активации.

Рисунок 29. График сигмоидальной функции

Сигмоидальная функция активации обладает следующими преимуществами по сравнению с пороговой функцией Хевисайда:

Нелинейность. Обеспечивает модели возможность обрабатывать (воспринимать) нелинейные закономерности в данных.

Фиксированные ограничения выхода. Это особенность позволяет масштабировать данные от слоя к слою.

Непрерывность и Дифференцируемость. Данные свойства позволяют обойти главный недостаток пороговой функции – резкий переход. Поскольку обучение сети происходит посредством градиентного спуска, то движение по пороговой функции слишком резкое, в отличие от сигмоидальной функции – движение по градиенту которой плавное и, как следствие, нахождение минимума ошибки становится более стабильным и вероятным.

Замечание 1: Кроме того, что нелинейная функция активации позволяет модели обрабатывать нелинейные закономерности в данных, она выполняет и другую важную задачу. Нелинейные выходы одного слоя нейронов позволяют подключить к нему второй, третий и т. п. слои. Если сделать много слоев в Персептроне с линейной функцией активации или с пороговой (суть в том, что она не непрерывна), то можно свести все эти слои всего лишь к одному слою

(с математической точки зрения последовательность взвешенных сумм (полиномов) можно легко свести к одному другому полиному).

Замечание 2: Подобно регрессионным моделям обучение модели ИНС подразумевает правку только весовых коэффициентов. Никакие другие процессы или изменения в модели не происходят.

Алгоритм обратного распространения ошибки как специальный метод градиентного спуска, разработанный именно для ИНС, будет подробнее разобран далее.

Сверточные (ConvolutionalNeuralNet) и Глубокие (DeepNet) Сети

Одна из базовых способностей людей и животных заключается в том, что мы легко распознаем визуальные образы под любым углом и в любом положении. Буква «А» останется для нас буквой «А» в какой бы области видимости наших глаз мы ее не увидели. Конечно же, при условии того, что выдержан определенный порог зашумленности, освещенности и т. п. Даже если текст расположен вверх ногами, то довольно быстро можно приноровиться его читать. А уж небольшие наклоны или разные шрифты на рекламных щитах вообще не представляют для нас проблемы.

А вот системы по распознаванию текста, к сожалению, не обладают такими возможностями. Либо они заточены распознавать определенный тип шрифта, либо они очень чувствительны к сдвигам или наклонам, либо к масштабу, либо вообще ко всему. И несмотря на то, что нейронные сети в качестве прототипа использовали принципы работы мозга, долгое время не было хорошего решения этой проблемы. Данный недостаток называется проблемой чувствительности к пространственным искажениям. А добиться нужно так называемой инвариативности к таким искажениям трех основных типов: смещениям, поворотам, масштабированию.

По поводу пространственных искажения и появления сверточной сети, в источнике [28] сказано следующее:

«Дело в том, что работа зрительной коры гораздо сложнее: в ней происходит анализ и цвета, и текстуры, и движения; информация от

двух глаз объединяется для осуществления стереозрения; работает множество других механизмов. Высшие зрительные функции все еще остаются загадкой. И самое главное, до сих пор не известно, как происходит обучение. До конца неясно даже, что в структуре зрительной системы заложено генетически, а что формируется под влиянием опыта. Хотя структура зрительной системы у человека продолжает формироваться до 4–5 лет, это может быть, как реализацией генетической программы, лишь немного адаптирующейся к окружению, так и детальным обучением, в результате которого создаются основные связи зрительного тракта.

Пытаясь разработать ИНС, которая была бы еще ближе к биологическому прототипу, японский ученый-компьютерщик Кунихика Фукусима предлагает принципиально новую модель Когнитрона в 1975 г., а в 1980 г. модификацию этой модели – «Неокогнитрон». Главное отличие этих сетей от MLP заключалось в том, что слои нейронов упаковывались не в линию, а в двумерную плоскость, чтобы информация циркулировала не только от слоя к слою, но еще сохраняла определенную пространственную ориентацию (рисунок 30). Кроме того, в этих сетях использовались принципы самоорганизации, то есть обучение происходило без учителя».

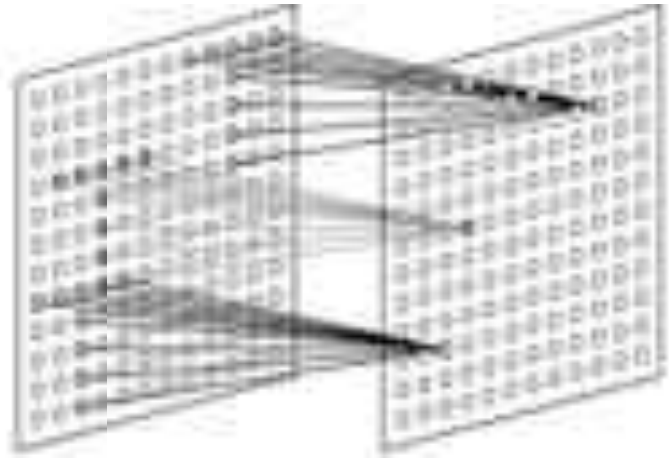


Рисунок 30. Рецептивные поля (квадратные плоскости) простых клеток, настроенных на поиск выбранного паттерна в разных позициях

В итоге подобная архитектура дала хорошие результаты при распознавании символов и даже рукописного текста относительно классических нейронных сетей. Однако по сравнению другими специализированными алгоритмами (не машинного обучения) на тот момент неокогнитрон «не дотягивал». Требовалось усложнять модель, но тогда были явные недостатки по скорости обучения и работы.

Однако общая концепция пространственно-ориентированных карт была очень ценной. И французский ученый и специалист в области обработки сигналов и компьютерного зрения Ян ЛеКун предлагает более упрощенную модель сети. Он убрал из неокогнитрона функции, которые нужны были только для того, чтобы быть похожим на реальный мозг. Он также показал, как можно использовать метод обратного распространения ошибки для обучения сетей, архитектура которых, как и у неокогнитрона, отдаленно напоминает строение коры мозга. И этот способ обучения, уже хорошо проверенный на тот момент, оказался куда эффективнее самоорганизации сетей. Так в 1995 появились нейронные сети Сверточного типа.

Структура сверточных сетей представлена на рисунке 31.

Нейроны здесь также упакованы не только в слои, но и двумерные карты. Информация циркулирует слева на право по нейронам такого же типа (как и в MLP). Но главная особенность заключается в том, что сеть не полносвязная, то есть каждый нейрон имеет свою небольшую область видимости (на верхнем рисунке область видимости показана пунктирными линиями). Такое локальное восприятие и обобщение от слоя к слою и дает решение проблемы чувствительности к пространственным искажениям, о которых говорилось выше. Иными словами, Сверточная Нейронная Сеть (СНС) способна обрабатывать пространственную топологию.

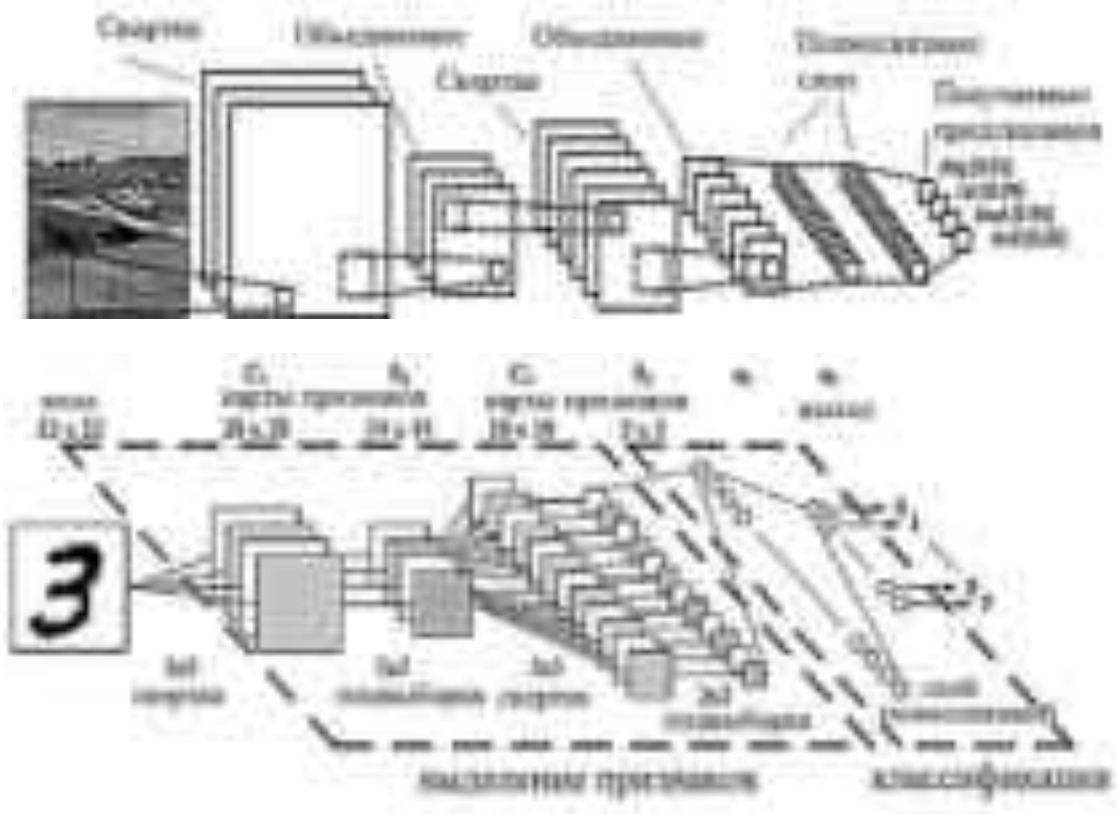


Рисунок 31. Структурная схема Сверточных нейронных сетей

Именно эта архитектура ИНС легла в основу так называемых Глубоких сетей, что породило понятие глубокого обучения (Deep Learning). Именно сеть этой архитектуры произвела такую шумиху в 2007 г., приблизив качество распознавания текста к человеческому уровню. Безусловно, данный метод не лишен недостатков, но во многих задачах именно визуального распознавания СНС является лучшим решением.

Глубокие сети, по сути, являются уже четвертой вехой развития когнитрона и их главная особенность заключается в многослойности (рисунок 32).

Но кроме этого важную роль играет разнообразное комбинирование определенных блоков карт. В глубоких сетях очень много разных модулей и ответвлений (может быть и такое, что сеть имеет несколько входов на разных уровнях).

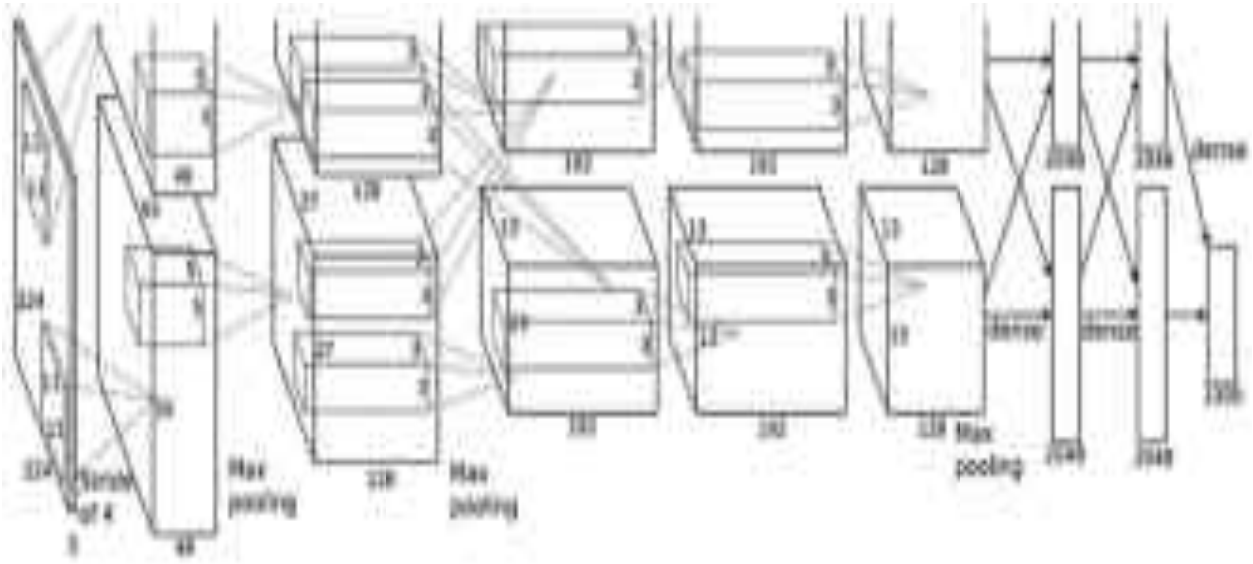


Рисунок 32. Схематичное представление Глубокой сети

Однако наращивание слоев приводит к целому ряду трудностей, как с объемом вычислений, так и с обучением сети, из чего складываются особенности, присущие именно глубокому обучению.

Карты (ART, SFAM)

Данный тип нейронных сетей разрабатывался как решение проблемы «пластичности-стабильности». Как известно, взрослый человек может понять какую-либо информацию с одного или нескольких объяснений. Речь, конечно, не идет о сложных вещах вроде курса математического анализа, но, тем не менее, мозг человека способен понять какие-то концепции даже за одно объяснение. Если переводить это на язык машинного обучения, то это означает, что нам не нужно многократное предъявление схожих примеров, нам хватит одного-двух объектов обучающей выборки. Конечно, мозг способен на такие вещи не только или, правильнее будет сказать, не столько благодаря быстро обучающимся нейронам, а потому что, воспринимая чье-то объяснение (например, когда вы читаете инструкцию к телевизору) мозг уже имеет представление о многих вещах. Иными словами, мозг уже обучен, у него есть предыдущий опыт. Кроме того,

разум способен держать контекст и т. п. Если объяснять упрощенно, то мозг дообучается, а не переобучается в таких случаях.

Но даже если не принимать во внимание контекст и априорный опыт, то возможно ли сделать обучение сети столь же пластичным, ну или близким к этому? Первая задача разработчиков сетей ART заключалась в том, чтобы сделать процесс обучения быстрым, чтобы не проходить итеративный градиентный спуск для правки весов. Эта проблема и есть проблема пластичности ИНС.

Но была и вторая задача. Опять же, вдохновляясь биологическим прототипом, разработчики понимали, что если человек узнает, что по какому-то вопросу у него на протяжении некоторого времени была неверная точка зрения, то он достаточно просто скорректирует отдельные представления о мире. У него не разрушится вся картина мира, он не сойдет с ума и не потеряет другие знания. Конечно, тут мы не берем в расчет психологические факторы, которые помешают ему это сделать.

Но если не рассматривать такие ситуации, то можно принять тот факт, что мозг очень пластичен, то есть может обучится\дообучиться с малого количества раз и в то же время стабилен – одни знания могут быть заменены на другие без разрушения другой информации.

В то же время информация в классических ИНС (в MLP) распределяется по весам всей сети. Иными словами, нет одного четкого места, которое бы отвечало за определенный блок информации. Таким образом, если дообучение происходит на объектах тех же классов, которые использовались во время обучения, то многослойную сеть прямого распространения можно дообучить (но опять же медленно). Но если взять обученную сеть на одних классах и попытаться до обучить на новых, то вся старая информация начнет разрушаться.

Например, ИНС типа MLP была обучена для классификации квадратов и кругов по визуальным образам. Допустим, что сеть стала показывать хорошие показатели качества после предъявления различных образов квадратов и кругов в количестве 10 тыс. штук. Если

позже дообучить сеть на новых вариантах квадратов и кругов, то сеть может еще улучшить показатели. Но если начать обучать ее еще и на треугольниках, то под этот новый класс не выделится определенное место в памяти сети, а распределенная по весам информация, наоборот, начнет меняться, разрушая обученное состояние.

Это и есть «проблема стабильности ИНС». Она была второй задачей разработчиков сетей типа ARTMAP (Творческие карты, или просто Карты).

Первые варианты Карт (ARTMAP 1, ARTMAP 2) были очень сложными, избыточными и во многом были инженерным творением в виде электрических схем, а не алгоритмом.

После возвращения интереса к нейронным сетям этот тип сетей был существенно оптимизирован и модифицирован. Один из самых успешных и эффективных вариантов называется Simplified FUZZY ARTMAP (упрощенная нечеткая Карта). Под нечеткостью здесь понимается то, что вычисления основаны на нечеткой логике. Детали работы нечеткости в этом подразделе не приводятся.

Структура такой сети представлена на рисунке 33.

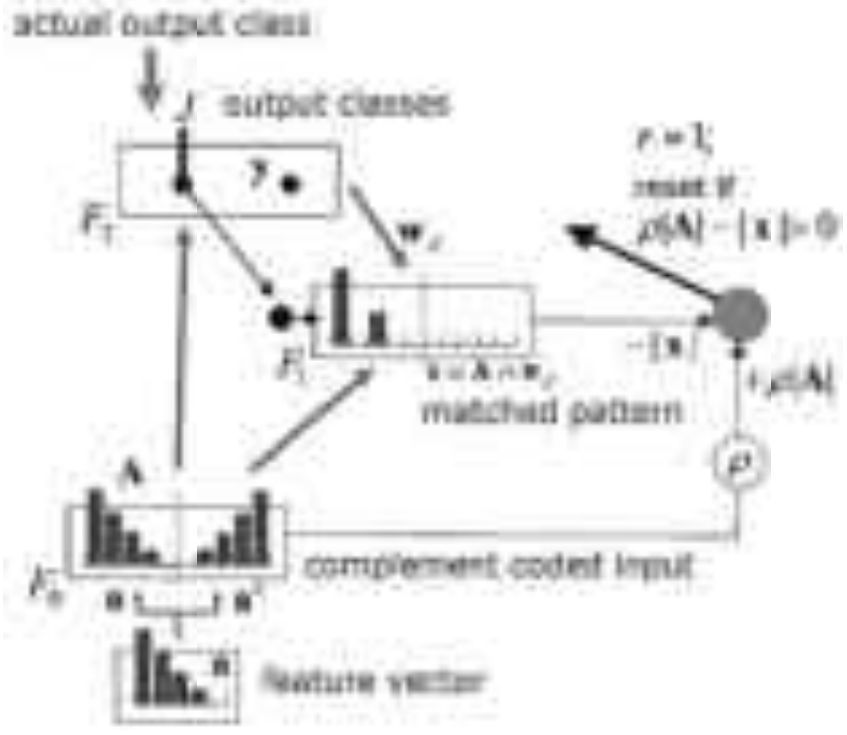


Рисунок 33. Структура основных модулей системы ARTMAP

Собственно, правильнее будет назвать это системой, содержащей нейроны, так как в структуре есть множество сторонних блоков (узлов), которые не очень красиво вписываются в биологическую концепцию мозга (хотя и не противоречат ей).

Итак, рассмотрим представленную на рисунке структуру снизу-вверх. В самом низу имеем вектор признаков «a» (как и во всех других методах ML). Данный вектор обязательно должен быть нормализован (отмасштабирован), то есть сеть не просто очень чувствительна к ненормализованным данным, а вообще не работает без нормализации. Соответственно если природа каких-то признаков неизвестна и не ясно, какой диапазон может быть в данных, то применение этой сети затрудняется (нужно специальным образом контролировать данные на вход) или вовсе отменяется.

Далее нормализованный вектор *a* дополняется комплементарной (обратной) парой a^c: там, где в исходном векторе было значение 0, в обратной паре будет 1, и наоборот (при условии нормализации от 0 до 1). Для этого и была необходима нормализация.

Далее, полученный вектор сравнивается со всеми шаблонами в базе нейронной сети. В данной случае это именно база, а не просто распределенная информация. Один нейрон в такой архитектуре представляет собой одну единицу информации, один конкретный шаблон\образ. То есть один нейрон выражает какой-то образ обучающей выборки (например, квадрат или круг). При обучении каждый поступающий вектор сравнивается с уже имеющейся базой образов, которые сохранены в нейронах сети.

Если близких образов нет, то образ сохраняется как новый, то есть создается новый нейрон и вектор входного образа полностью копируется в веса нового нейрона. Так с одного раза происходит полное корректное запоминание. При этом за нейроном закрепляется метка y (то есть это по-прежнему обучение с учителем, и для каждого входного вектора имеется ответ).

Если в базе уже имеется образ, похожий на входной образ, то происходит следующее. Образ в базе, выраженный нейроном и его

весами, подтягивается\приближается ко входному. Допустим, у сети в базе уже имеется некое представление о квадрате, и мы подаем еще один похожий образ квадрата, тогда после подтягивания в базе будет храниться нечто среднее, какой-то усредненный образ этих двух квадратов. Соответственно надо четко понимать, по какими принципам рассчитывается это усреднение. Ведь это все-таки не искусственный интеллект, и по одному образу человека и дельфина, сеть не усреднит их в образ млекопитающих. Надо понимать, что это лишь векторное усреднение. То есть если входной образ (0.3, 0.7), а образ в базе (0.35, 0.8), то итоговый будет (0.325, 0.75). Что с точки зрения гипотезы компактности позволяет решать задачу обобщения.

Замечание: напомним, что гипотеза компактности говорит нам о том, что объекты одного класса обладают близкими по значению признаками (или какой-либо комбинацией признаков), из-за чего их вообще можно отделить от объектов другого класса. Если пойти еще дальше, то можно сказать, что объекты сходных классов располагаются в некотором N-мерном пространстве признаков также близко. Тут можно сказать, что объекты потому и принадлежат какому-то одному смысловому классу, потому что по какому-то признаку или группе (комбинации) похожи.

Итак, данная сеть способна сохранять образ с одного предъявления, что решает проблему пластичности. В сети сразу появляется образ с меткой класса, к которому принадлежит этот образ. Таким образом, вместо минимизации ошибки данная сеть обучается путем создания похожих прототипов и сравнением входного образа с прототипами. В рабочем режиме сеть выдает ответ Y в виде метки того нейрона, образ которого сильнее всего похож на текущий входной образ.

При этом данная сеть также решает задачу стабильности, так как теперь есть четкие хранилища информации и система организации новых нейронов. Так что даже при предъявлении новых классов сеть будет стабильно работать.

Данный тип сети хорошо зарекомендовал себя в медицине и при распознавании звуковых сигналов. Однако есть и ряд минусов данной сети:

- при равной степени обобщения сеть ARTMAP потребует существенно большее количество памяти и вычислений, чем MLP, если речь идет о входных векторах большой размерности и очень больших обучающих выборках;
- сеть не способна учитывать топологию пространства и сложные абстрактные признаки (так как слоев по сути нет).

Рекуррентные сети (Recurrent Neural Network)

Рекуррентные нейронные сети (RecurrentNeuralNetwork; RNN) – вид нейронных сетей, в которых имеется обратная связь. При этом под обратной связью подразумевается связь от логически более удаленного элемента к менее удаленному (рисунок 34). Наличие обратных связей позволяет запоминать и воспроизводить целые последовательности реакций на один стимул.

В сетях такого типа возникает эффект памяти и способности воспринимать не только статичный образ, но и динамику образов (так как есть возможность учитывать историю через обратную связь).

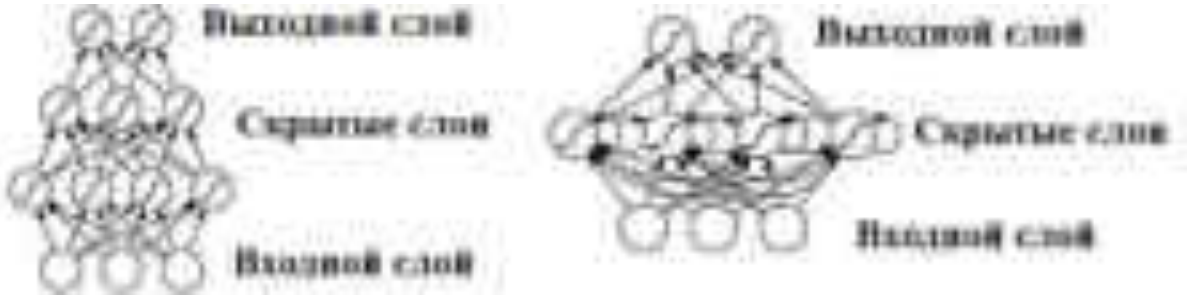


Рисунок 34. Сеть прямого распространения (слева) и рекуррентная сеть (справа)

Согласно Википедии [1]: «Долгая краткосрочная память (англ. Longshort-termmemory; LSTM) – разновидность архитектуры рекуррентных нейронных сетей (RNN), предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером. Как и большинство рекур-

рентных нейронных сетей, LSTM-сеть является универсальной в том смысле, что при достаточном числе элементов сети она может выполнить любое вычисление, на которое способен обычный компьютер, для чего необходима соответствующая матрица весов, которая может рассматриваться как программа. В отличие от традиционных рекуррентных нейронных сетей, LSTM-сеть хорошо приспособлена к обучению на задачах классификации, обработки и прогнозирования временных рядов в случаях, когда важные события разделены временными лагами с неопределенной продолжительностью и границами.

Относительная невосприимчивость к длительности временных разрывов дает LSTM преимущество по отношению к альтернативным рекуррентным нейронным сетям, скрытым Марковским моделям и другим методам обучения для последовательностей в различных сферах применения. Из множества достижений LSTM-сетей можно выделить наилучшие результаты в распознавании несегментированного слитного рукописного текста и победу в 2009 году на соревнованиях по распознаванию рукописного текста (ICDAR). LSTM-сети также используются в задачах распознавания речи, например, LSTM-сеть была основным компонентом сети, которая в 2013 году достигла рекордного порога ошибки в 17,7% в задаче распознавания фонем на классическом корпусе естественной речи TIMIT. По состоянию на 2016 год, ведущие технологические компании, включая Google, Apple, Microsoft и Baidu, используют LSTM-сети в качестве фундаментального компонента новых продуктов».

Самоорганизующиеся карты (Self-organization map, SOM)

Согласно ресурсу [29]: «Такие сети представляют собой соревновательную нейронную сеть с обучением без учителя, выполняющую задачу визуализации и кластеризации. Является методом проецирования многомерного пространства в пространство с более низкой размерностью (чаще всего, двумерное), применяется также для решения задач моделирования, прогнозирования и др. Является одной из версий нейронных сетей Кохонена. Самоорганизующиеся

карты Кохонена служат, в первую очередь, для визуализации и первоначального («разведывательного») анализа данных.

Геометрическая суть алгоритма такая, что близкие объекты в многомерном пространстве (схожие животные по тысяче признаков) будут расположены близко и на двумерной карте, где объекты будут представлены точками. Собственно, обучение сети это и есть процесс укладки таких точек (итеративного оттягивания таких точек от начальных позиций, см. рисунок 35).

Сигнал в сеть Кохонена поступает сразу на все нейроны, веса соответствующих синапсов интерпретируются как координаты положения узла, и выходной сигнал формируется по принципу «победитель забирает все», то есть ненулевой выходной сигнал имеет нейрон, ближайший (в смысле весов синапсов) к подаваемому на вход объекту. В процессе обучения веса синапсов настраиваются таким образом, чтобы узлы решетки «располагались» в местах локальных сгущений данных, то есть описывали кластерную структуру облака данных, с другой стороны, связи между нейронами соответствуют отношениям соседства между соответствующими кластерами в пространстве признаков».

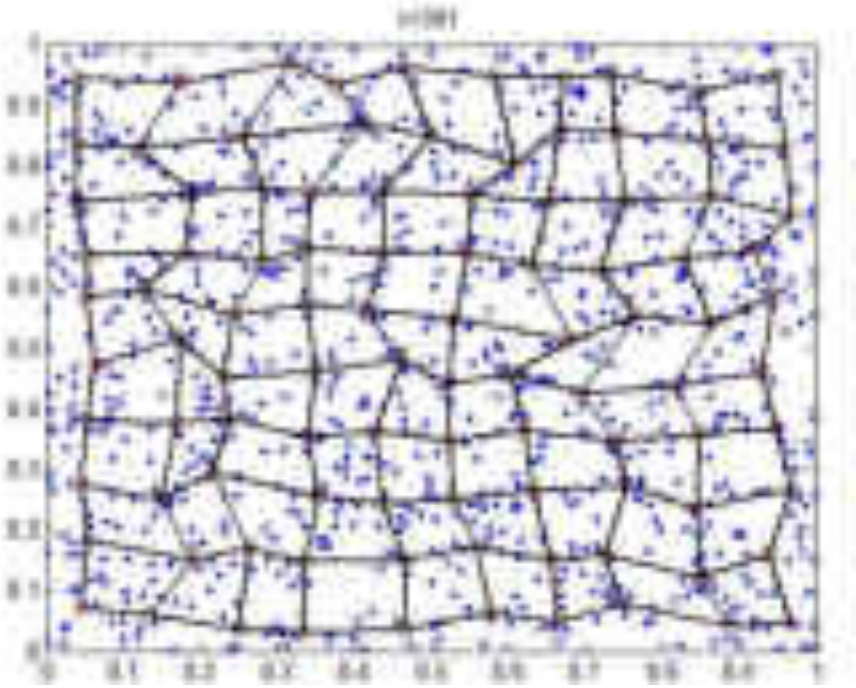


Рисунок 35. Визуализация двумерной плоскости с точками (проекциями) объектов в SOM

Автокодировщики (AutoEncoder)

Согласно Википедии [1]: «Автокодировщики (AutoEncoder) – специальная архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки. Простейшая архитектура автокодировщика – сеть прямого распространения, без обратных связей, наиболее схожая с персептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от персептрона, выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой (см. рисунок 36).

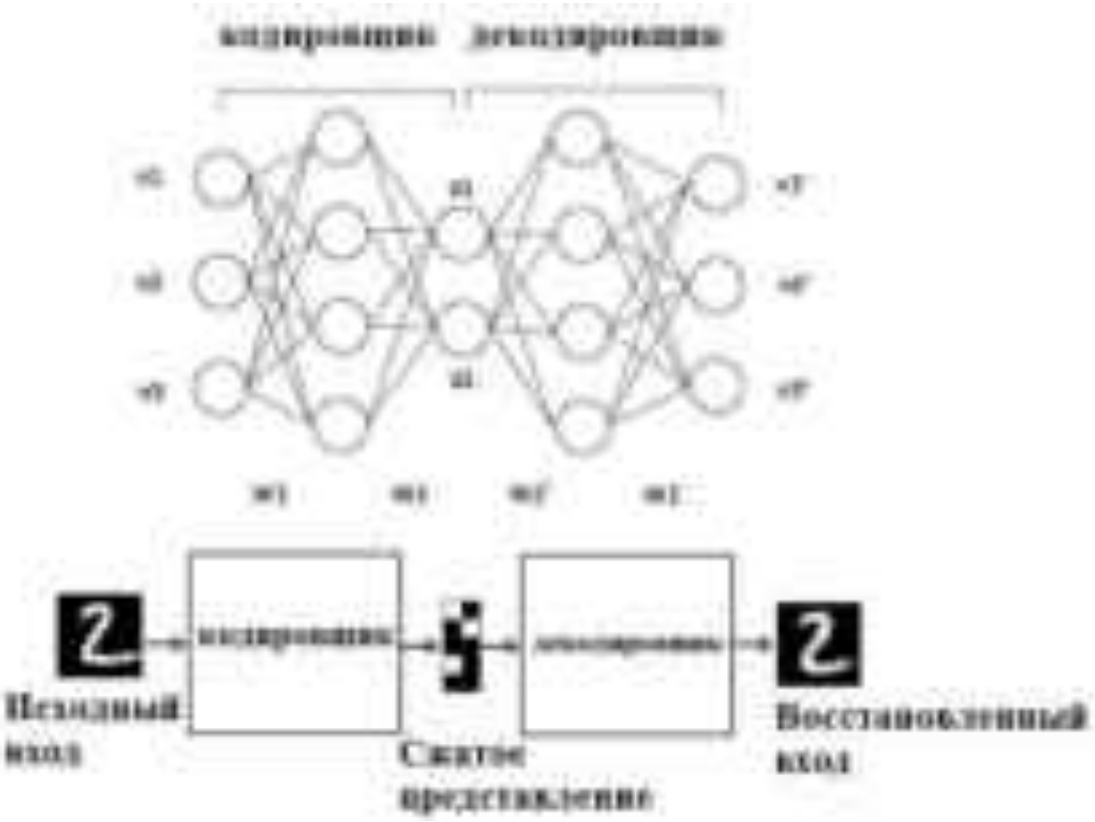


Рисунок 36. Схема сети типа Автокодировщик

Основной принцип работы и обучения сети автокодировщика – получить на выходном слое отклик, наиболее близкий к входному. Чтобы решение не оказалось тривиальным, на промежуточный слой автокодировщика накладывают ограничения: промежуточный слой должен быть или меньшей размерности, чем входной и выходной

слои, или искусственно ограничивается количество одновременно активных нейронов промежуточного слоя – разреженная активация. Эти ограничения заставляют нейросеть искать обобщения и корреляцию в поступающих на вход данных, выполняя при этом их сжатие. Таким образом, нейросеть автоматически обучается выделять из входных данных общие признаки, которые кодируются в значениях весов сети. Так, при обучении сети на наборе различных входных изображений, нейросеть может самостоятельно обучиться распознавать линии и полосы под различными углами.

Какое-то время автокодировщики применялись каскадно для обучения глубоких (многослойных) сетей, а именно для предварительного обучения глубокой сети без учителя. Для этого слои обучаются друг за другом, начиная с первых. К каждому новому необученному слою на время обучения подключается дополнительный выходной слой, дополняющий сеть до архитектуры автокодировщика, после чего на вход сети подается набор данных для обучения. Веса необученного слоя и дополнительного слоя автокодировщика обучаются при помощи метода обратного распространения ошибки. Затем слой автокодировщика отключается и создается новый, соответствующий следующему необученному слою сети. На вход сети снова подается тот же набор данных, обученные первые слои сети остаются без изменений и работают в качестве входных для очередного обучаемого автокодировщика слоя. Так обучение продолжается для всех слоев сети за исключением последних. Последние слои сети обычно обучаются без использования автокодировщика при помощи того же метода обратного распространения ошибки и на маркированных данных (обучение с учителем).

В последнее время автокодировщики мало используются для описанного «жадного» послойного предобучения глубоких нейронных сетей. После того как этот метод был предложен в 2006 г. Джеффри Хинтоном и Русланом Салахутдиновым, достаточно быстро оказалось, что новые методы инициализации случайными весами с определенным видом распределения оказываются эффективными

для улучшения сходимости. А предложенная в 2014 г. пакетная нормализация позволила обучать еще более глубокие сети, предложенный же в конце 2015 г. метод остаточного обучения позволил обучать сети произвольной глубины».

Основными практическими приложениями автокодировщиков остаются уменьшение шума в данных, а также уменьшение размерности многомерных данных для визуализации. Кроме того, с определенными оговорками, касающимися размерности и разреженности данных, автокодировщики могут позволять получать проекции многомерных данных, которые оказываются лучше тех, что дает метод главных компонент либо какой-либо другой классический метод. Помимо этого можно использовать такие сети, как алгоритм сжатия данных.

Крайне важно отметить, что если же взять архитектуру автокодировщика, но обучать его с учителем, то есть Y будет отличаться от X, то получится сеть для преобразования одного изображения в другое изображение через некоторую сложную функцию. На основе данного преобразования строятся различные фильтры изображений и видео (например, наложение различных эффектов) или же можно решать сложные задачи сегментации изображения. На сегодняшний день подобные архитектуры очень распространены и хорошо решают задачи анализа сцены для беспилотного автомобиля.

Импульсные (Спайковые) сети

Согласно Википедии [1]: «Первая научная модель импульсной нейронной сети была предложена Аланом Ходжкином и Эндрю Хаксли в 1952 году. Эта модель описывала, как потенциалы действия возникают и распространяются. Импульсы, однако, как правило, не передаются непосредственно между нейронами. Связь требует обмена химическими веществами, которые называются нейротрансмиттерами, в синаптической щели.

С точки зрения теории информации, проблема заключается в отсутствии модели, которая бы объясняла, как кодируется инфор-

мация и декодируются серии последовательностей импульсов, то есть потенциалы действия. Для нейробиологии все еще открытым является ответ на вопрос: нейроны связываются с помощью частотного или временного кодирования? С помощью временного кодирования один импульсный нейрон может заменять сотни скрытых элементов частотной нейронной сети.

Что же касается отличия спайковых сетей от классических, то тут необходимо отметить следующее. Если основное отличие сверточных сетей от классических заключается в структуре и организации связей, то импульсные сети, напротив, по структуре похожи на MLP. Их главное отличие заключается в том, что нейроны обмениваются короткими (у биологических нейронов – около 1-2 мс) импульсами одинаковой амплитуды (у биологических нейронов – около 100 мВ). Является самой реалистичной, с точки зрения физиологии, моделью ИНС (см. рисунок 37)».

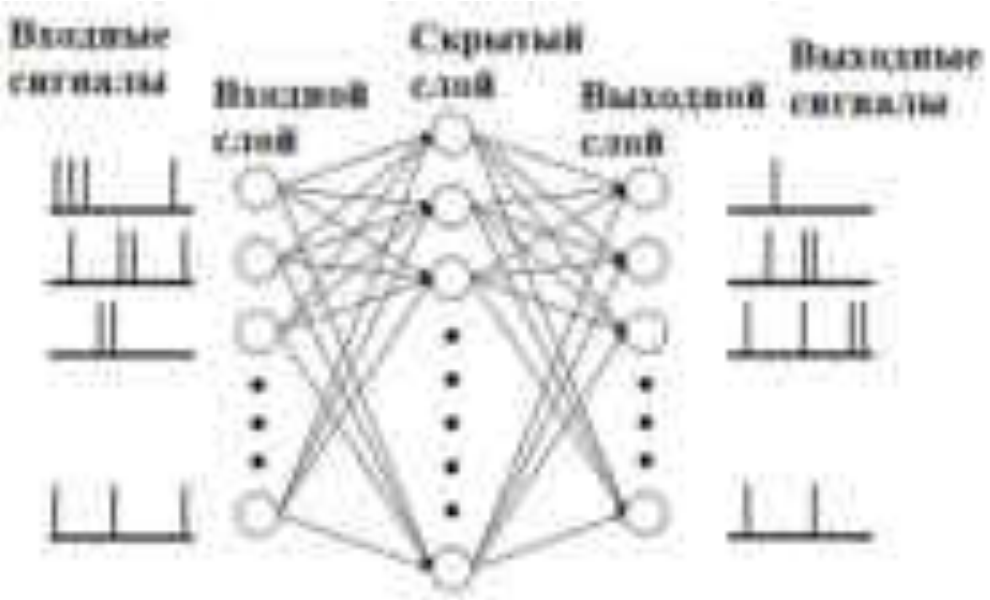


Рисунок 37. Схематичное представление структуры Импульсной нейронной сети

Данный тип сети еще находиться на стадии активного исследования. Тем ни менее ИмНС уже успешно применялась для динамического управления мобильным роботом. В качестве основных ее преимуществ можно назвать:

- Перспективность в обработке динамической информации (например, видеопотока, временных рядов и т. п.).
- Более плотное и эффективное кодирование информации по сравнению с MLP.
- Более эффективное расходование электроэнергии при физической реализации.
- Высокая степень эффективности параллельного выполнения при физической реализации.

К недостаткам можно отнести слабую изученность, сложность математической модели и вычислительную сложность при выполнении на обычном железе. А главный недостаток заключается в отсутствии хорошего алгоритма обучения.

Причины бурного развития ИНС сегодня

Казалось бы, раз ИНС такой мощный инструмент\метод и первая нейронная сеть была предложена так давно, то почему только сейчас появляется столько технологий на нейронных сетях? Тому есть две причины:

Чтобы исследовать глубокие ИНС, нужно проводить эксперименты, а они могли длиться по месяцу, а если и меньше, то нужно было все равно проводить десятки экспериментов. Таким образом, исследовать сети было сложно.

Поэтому появление соответствующих параллельных вычислителей и общее увеличение мощности компьютеров изменило эту ситуацию. Кроме того, появились обычные видео карты (GPU), на которых любой мог запускать сложные вычисления. Поэтому появилось много исследователей, которые проводили эксперименты. То есть, как ни странно, развитие тормозилось не из-за принципиальных возможностей ИНС, а из-за того, что экспериментировать было долго.

Функция активации «Relu» упростила функцию сетей в целом и сделала вычисления Глубоких сетей еще быстрее.

Психологический фактор. Извилистая история нейронных сетей и долгое пребывание в состоянии заморозки создали дурную славу нейронным сетям, и исследователи очень неохотно занимались этой сферой.

Возрастание интереса сразу сложило разные кусочки мозаики, и Сверточные сети получили вторую жизнь, а их успех дал еще больший толчок.

Стоит также отметить, что сегодня наблюдается обратный эффект. Пресса и Интернет часто преувеличивают возможности ИНС или допускают неточности, из-за чего может сложиться впечатление, что нейронные сети – это искусственный интеллект.

Борьба с переобучением в ИНС

Тема переобучения была подробно рассмотрена ранее, поэтому просто перечислим способы борьбы с этим эффектом, подходящие для большинства типов ИНС:

- Кросс-валидация. Данный метод не зависит от модели, это общий подход к тренировке моделей, помогающий контролировать и бороться с переобучением особенно на малых выборках.
- Уменьшение числа слоев\нейронов. Уменьшение слоев и нейронов явно влияет на сложность модели, однако необходимо искать компромисс между обобщающей способностью нейронов и степенью абстрактности признаков.
- Добавление L2 регуляризации. Аналогично Регрессии этот дополнительный метод штрафует высокие веса модели.
- Локальность восприятия. Задавая рецептивное поле, восприятие нейрона можно также уменьшить не только количество параметров или сложность модели, но, и более того, – определить влияющие факторы, что позволит нейрону решать локальные задачи и складывать их в глобальное решение. Это крайне важная особенность ИНС.

Обратное распространение ошибки

Итак, как и у любой другой модели в машинном обучении, у ИНС есть 2 этапа работы:

- обучение;
- использование, моделирование или получение отклика, прогноза (все это подразумевает расчет выхода по обученной модели).

При этом использование сети называют прямым распространением (forwardpropagation), потому что сигнал со входа сети распространяется к выходу (рисунок 38), то есть слева направо.

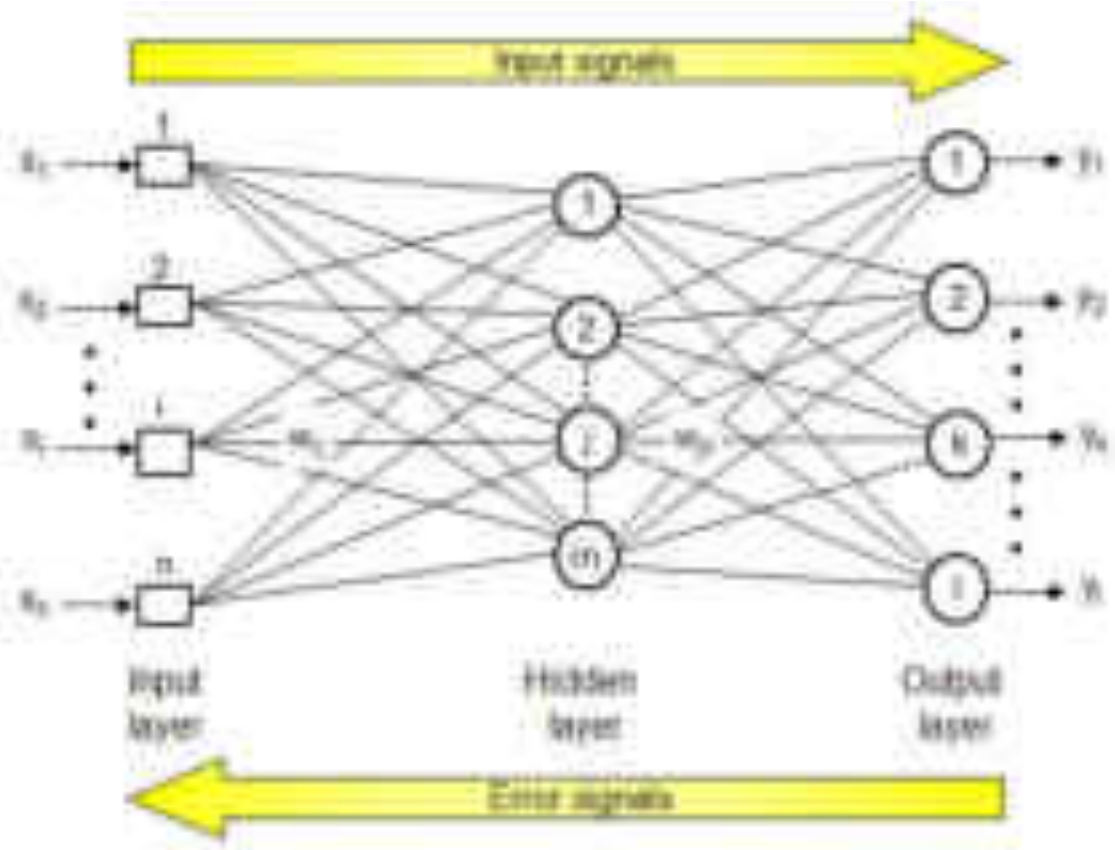


Рисунок 38. Обобщенная схема распространения сигналов по MLP

На рисунке 39 изображена принципиальная схема прямого распространения сигнала со входа на один нейрон. В случае нескольких нейронов в слое схожие вычисления будут проводиться для каждого нейрона. После чего сигнал двинется к следующему слою, и если в нем также несколько нейронов, то там будут сходные вычисления

для каждого нейрона и так далее. Каждый нейрон берет взвешенную сумму своих входов, считает функцию активации и выдает выход для следующего слоя и т. д.

Обучение называют обратным распространением (backpropagation) ошибки, потому что информация идет наоборот, с выхода сети, и информация эта – об ошибке, а не о входном образе, как при прямом распространении.

В модель нейрона на рисунке 39 включен пороговый элемент (bias), который обозначен символом b_k. Эта величина отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации. По сути, это свободный коэффициент при некотором признаке в нулевой степени (как и у Регрессии).

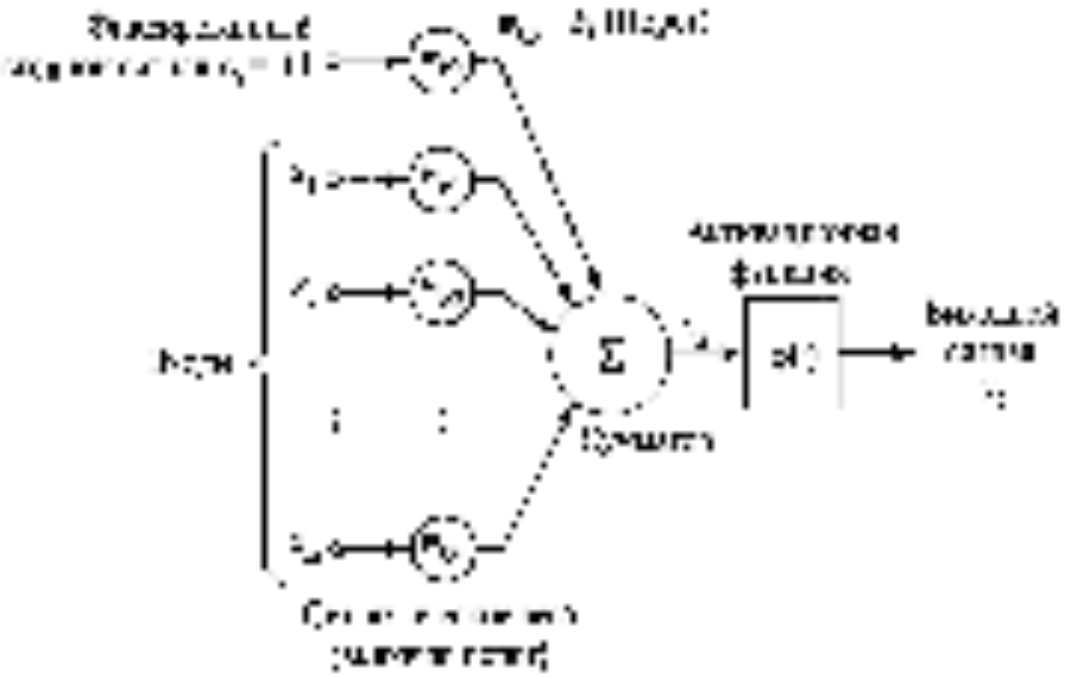


Рисунок 39. Принципиальная схема прямого распространения сигнала

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений:

$U_k = \sum_i^m w_{ki} x_i$,

$y_k = \varphi(U_k)$,

где U_k – индуцированное локальное поле, или потенциал активации.

Серия таких уравнений, вычисленных последовательно, в конце концов даст результат всей сети. Если в последнем слое сети один нейрон, то соответственно получим скаляр или число, а если нейронов несколько, то получим вектор значений для каждого входного образа X.

Теперь рассмотрим алгоритм обратного распространения ошибки для обучения искусственной нейронной сети. Он разработан в первую очередь для сетей MLP-типа. Но также подходит для сверточных сетей, автокодировщиков и, при определенных модификациях, для рекуррентных сетей.

Для начала введем определения:

Обучить сеть – минимизировать ошибку, а именно минимизировать разницу между выходом сети (реакцией на вход) и требуемой реакцией, то есть это обучение с учителем.

Минимизация ошибки производится путем итеративных правок\корректировок весов ИНС. Корректируются в ИНС только веса и больше ничего.

Минимизация ошибки происходит для каждого отдельно взятого входного образа. Повторяя такую минимизацию для каждого образа много раз, получаем общую минимизацию. Количество проходов по тренировочной выборке называется эпохами.

Начнем рассмотрение алгоритма с последнего (выходного) слоя сети (рисунок 40).

Сперва предположим, что выходной слой содержит только один нейрон. Тогда:

E, e – ошибка выхода сети для текущей пары (X, Y);

Y, y – требуемый выход для соответствующего входного образа X;

o = φ(U) – выход нейрона, рассчитывается как сигмоида от U;

U – результат взвешенной суммы, или индуцированное локальное поле;

Wij – вес, соединяющий некоторый нейрон слоя i с некоторым нейроном слоя j.

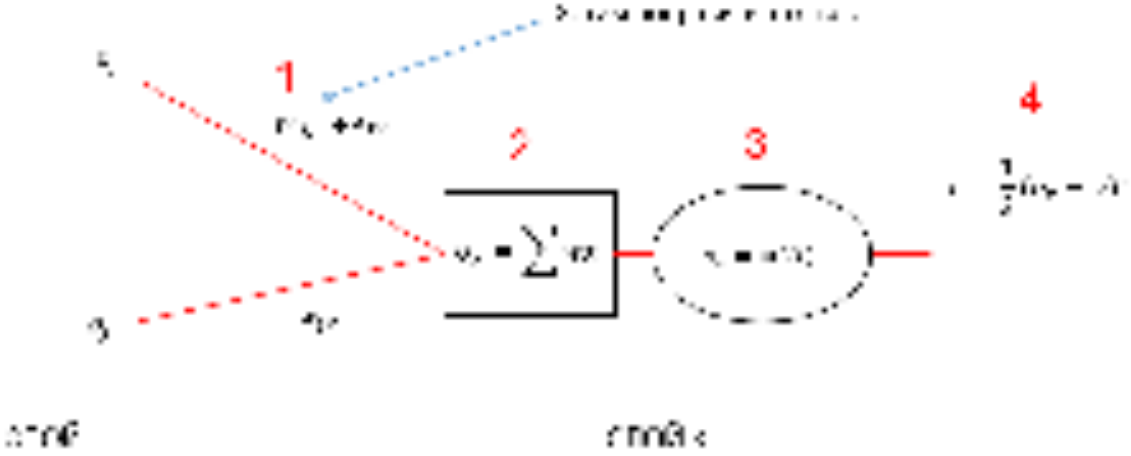


Рисунок 40. Представлена схема последнего слоя сети MLP

Индексы при символах означают индексы слоев. Так как в каждом слое имеется несколько нейронов, то о каком нейроне идет речь, когда пишется oj? О любом произвольном в этом слое, так как формула обобщенная и не вносит конкретики о номере нейрона внутри слоя.

Первая идея алгоритма\метода заключается в том, что мы двигаемся по градиенту ошибки, как и во всех численных градиентных методах. Мы не знаем, где точное решение, но если итеративно понемногу уменьшать ошибку для каждого входного образа, то мы рано или поздно придем к некому равновесному состоянию. Не обязательно минимуму ошибки, но к такому моменту, когда дальнейшие правки уже не будут коренным образом менять ситуацию, а система войдет либо в полную заморозку (изменений вообще не будет), либо в некий колебательный процесс около некоторой матрицы весов.

Собственно, двигаясь по антиградиенту ошибки $-\nabla F$, мы как раз и производим корректировки весов на некие дельта (в данном случае у весов нет никаких индексов, потому что это общая концепция правок):

$$w^{t+1} = w^t - \alpha \Delta w,$$

$$w^{t+1} = w^t - \alpha \nabla F(w^t),$$

$$\nabla F(w^t) = \frac{\partial E}{\partial w^t},$$

где w^{t+1} – вес в следующий момент времени;

w^t – вес в текущий момент времени;

Δw – правка веса на текущем шаге, в сторону уменьшения ошибки на текущем шаге;

α – размер шага, скорость движения по антиградиенту или сила правок весов.

Еще раз о том, почему используется именно градиент. Потому что нельзя рассчитать точное значение правок для всех весов сразу, ведь мы не знаем вклад каждого веса в ошибку, мы лишь знаем значение ошибки и направление. Чтобы понять суть идеи, представьте, что некая правка $\pm\Delta w$ некого веса w даст нам увеличение или уменьшение ошибки на выходе сети (см. рисунок 40). Получается, что небольшие правки весов приводят к некоторым небольшим изменениям конечной ошибки сети. Возможно ли найти функциональную зависимость между этими небольшими изменениями? Что есть отношение небольшого изменения ошибки к изменению какого-либо веса? Это производная $\frac{\partial E}{\partial w}$.

Таким образом, найдя аналитическое выражение производной ошибки по любому весу, мы определим характер воздействия правок этого веса на ошибку. А в конкретных точках этой функции (при конкретном входном образе и всех других параметрах сети) мы сможем точно подсчитать значение функции производной, то есть сможем оценить знак и значение dE, на которое будет меняться ошибка при наших правках dw. А значит, мы можем выбрать такую правку, чтобы уменьшить эту самую E. При этом все остальные веса, кроме того который правится в данный момент, замораживаем.

Итак, для последнего уровня все относительно просто. Чтобы найти производную $\frac{\partial E}{\partial w}$, необходимо взять серию производных по каждой вложенной функции (по цепному правилу дифференцирования):

$$w_{jk}^{t+1} = w_{jk}^{t} - \alpha\frac{\partial E}{\partial w_{jk}} =$$

$$= w_{jk}^{t} - \alpha\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial U_k}\frac{\partial U_k}{\partial w_{jk}} = w_{jk}^{t} - \alpha(o_k - y)o_k(1 - o_k)o_j.$$

Но что делать с внутренними слоями? Дело в том, что для нейронов внутренних слоев мы не можем явно рассчитать не то что значение правки, но и значение ошибки. Ведь чтобы рассчитать значение ошибки, надо знать, какое значение выхода должно быть у каждого нейрона внутреннего слоя. А мы не можем этого знать, ведь не знаем конкретный вклад каждого нейрона в ошибку. Мы знаем производную, то есть ближайший характер изменений ошибки от правок весов, но не можем посчитать нужные значения для произвольных точек.

Но что если пойти таким же путем, как и раньше? Допустим, что некая правка $\pm\Delta w$ некого веса w (теперь уже на скрытом слое) даст нам увеличение или уменьшение ошибки на выходе сети (см. рисунок 41). Получается, что небольшие правки этого веса тоже приводят к некоторым небольшим изменениям конечной ошибки сети при условии, что все остальные веса сети константны в рамках этого временного среза.

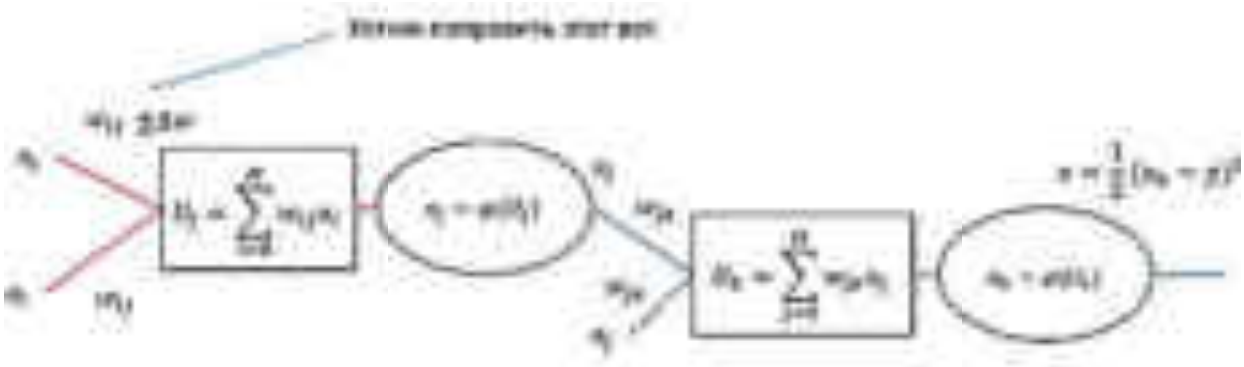
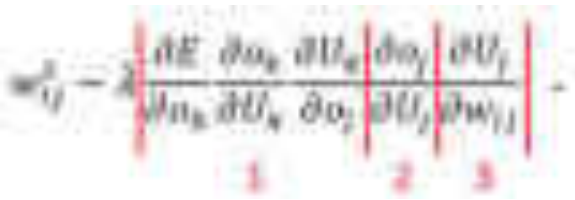


Рисунок 41. Схема последних двух слоев сети

Поэтому можно взять производную выходной ошибки по этому нейрону. Это будет такая же цепочка производных (только вес w_{ij} теперь будет константой, а не переменной, поэтому он и останется после взятия производной по o_j, а берем мы именно по o_j, чтобы пройти (протиснуть информацию) дальше).

$$w_{ij}^{t+1} = w_{ij}^{t} - \lambda\frac{\partial E}{\partial w_{ij}} = w_{ij}^{t} - \lambda\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial U_k}\frac{\partial U_k}{\partial o_j}\frac{\partial o_j}{\partial U_j}\frac{\partial U_j}{\partial w_{ij}}.$$

Теперь можно подставить каждую производную и получить формулу корректировки веса в скрытом слое. Однако выше мы делали серьезное допущение, что в последнем слое будет только один нейрон. А это может быть не так. И нас интересует именно этот общий случай и для любого слоя. Таким образом, из этой формулы надо выделить закономерность, которую можно использовать для произвольного слоя:



В формуле градиента ошибки по весу скрытого слоя присутствуют 3 составляющих. 1 – это составляющая следующего слоя, 2 – это составляющая текущего слоя и 3 – составляющая предыдущего слоя. То есть изменение выходной ошибки складывается из влияния этих составляющих, если все остальные элементы среды заморозить. А раз так, то можно сделать два вывода:

Первая составляющая в случае нескольких нейронов в следующем слое будет не одна. Понятно, что текущий нейрон j-го слоя распространяет свою ошибку на все связанные с ним нейроны слоя k. А значит, надо просуммировать вклад, который оказывает корректировка веса по всем путям. Или, иными словами, просуммировать производные по разным связям нейронов. Тогда первый блок можно будет переписать так:

$$\sum_{k=0}^{M} \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial U_k} \frac{\partial U_k}{\partial o_j} = (o_k - y)\varphi'_k w_{jk}.$$

Для сколь угодно далекого слоя можно не считать полный градиент, а брать вклад следующего слоя за основу. Ведь любой произвольный слой в первую очередь делает вклад именно в следующий слой. Таким образом, учитывая промежуточные вклады или точнее влияние всех промежуточных вкладов, мы сможем оценить итоговое влияние на ошибку. Опять же речь идет о небольших локальных приращениях $\pm\Delta w$ по конкретному w. Значит, можно выделить первую составляющую формулы в отдельную передаточную

величину вклада ошибки δ . Также ее называют производной ошибки по взвешенной сумме, по локальному индуцированному полю (по U). Тогда для последнего слоя:

$\delta_k = (o_k - y) o_k (1 - o_k)$.

А для любого другого скрытого слоя:

$\delta_j = (\sum \delta_k w_{jk}) o_j (1 - o_j)$.

Получается, что сигма рассчитывается как совокупность всех вкладов в ошибку (начиная с конца). И именно эта величина является той информацией, которая выражает ошибку от слоя к слою и распространяется обратно. Учитывая оба пункта одновременно, формулы для корректировок можно переписать следующим образом:

$\Delta w = -a\delta_k o_j = -a(o_k - y) o_k (1 - o_k) o_j$ – если корректируется вес последнего слоя.

$\Delta w = -a\delta_j o_i = a(\sum \delta_k w_{jk}) o_j (1 - o_j) o_i$ – если корректируется вес скрытого слоя.

Общий алгоритм можно представить так:

- Инициализировать веса случайным образом;
- Рассчитать выход сети прямым распространением сигнала;
- Рассчитать ошибку на выходе;
- Рассчитать корректировки весов текущего слоя по ошибке следующего;
- Обновить веса.

Условия завершения алгоритма обратного распространения ошибки:

- Требуемая величина ошибки;
- Максимальное количество итераций;
- Последние N проходов веса не правились больше чем на T.

Замечание 1: шаг градиентного спуска или скорость градиентного спуска a крайне важный параметр и нужен не только ради формальности. Дело в том, что при $a = 1$ движение по градиенту ошибки будет слишком быстрым, т.к. каждая корректировка отдельно взятого веса производиться с учетом заморозки всех остальных весов.

А следующий вес корректируется на основании исходного значения ошибки (вычисленного еще до правки первого веса). В противном случае движение по градиенту будет плавным, но это очень расточительно с точки зрения вычислительных ресурсов (такой алгоритм будет очень медленным). Кроме того, точный пересчет в рамках одного образа и не нужен, так как полное уменьшение ошибки для одного образа не гарантирует факт того, что уменьшиться общая ошибка. А уменьшить необходимо именно общую ошибку, а значит в рамках одного образа корректировки должны быть небольшими. Уточним, что на практике часто принимают $a = 0.01$.

Замечание 2: конечно, рассмотренный алгоритм не лишен недостатков. Некоторые из них очевидны и их возможно устранить. Существует множество модификаций градиентного спуска, например: Adadelta, Adam и т. п. Есть и другие методики улучшения обучения. Один из самых мощных и простых – это Batching. Но все эти методы выходят за рамки пособия, так как при необходимой фундаментальной подготовки вы можете изучить их особенности самостоятельно.

Замечание 3: алгоритм обратно распространения ошибки линеен с точки зрения вычислительной сложности.

Замечание 4: более подробное рассмотрение модификаций алгоритма обратного распространения выходит за рамки этого пособия, как и рассмотрение альтернативных функций активации (таких как ReLU, LeakyReLU, Parametric ReLU, RandomizedReLU). Поэтому вам предлагается изучить эти вопросы самостоятельно.

Нечеткие нейронные сети

Нечеткой нейронной сетью (НС) обычно называют четкую нейросеть, которая построена на основе многослойной архитектуры с использованием специальных «И»-, «ИЛИ»-нейронов.

Нечеткая нейросеть функционирует стандартным образом на основе четких действительных чисел, нечеткой является только интерпретация результатов.

Нечеткие нейронные сети осуществляют выводы на основе аппарата нечеткой логики, а параметры функций принадлежности настраиваются с использованием алгоритмов обучения НС. Поэтому для подбора параметров таких сетей применим метод обратного распространения ошибки, изначально предложенный для обучения многослойного персептрона. Нечеткая нейронная сеть, как правило, состоит из четырех слоев: слоя фаззификации входных переменных, слоя агрегирования значений активации условия, слоя агрегирования нечетких правил и выходного слоя.

Наибольшее распространение в настоящее время получили архитектуры нечеткой НС вида ANFIS и TSK [18]. Доказано, что такие сети являются универсальными аппроксиматорами. Быстрые алгоритмы обучения и интерпретируемость накопленных знаний – эти факторы сделали сегодня нечеткие нейронные сети одним из самых перспективных и эффективных инструментов мягких вычислений.

Преимущества нечетких нейронных сетей

Основным преимуществом технологии нейрокомпьютинга служит возможность выразить зависимость «выход-вход» без предварительной аналитической работы по выявлению правил, а на основе обучения на примерах. Недостатком нейросетей является невозможность объяснить выходной результат, так как значения распределены по нейронам в виде значений коэффициентов весов. Основной трудностью в применении нечетких экспертных систем служит необходимость явно сформулировать правила проблемной области в форме правил. В нечетких экспертных системах легко построить объяснение результата в форме протокола рассуждений. Поэтому в настоящее время создаются гибридные технологии.

Примером гибридной технологии служит реализация базы нечетких правил на основе нейросети. База нечетких правил для двух входных переменных имеет следующую структуру:

$R_i: if\ x_{1i}\ \ is\ \ A_{1i}\ \ and\ x_{2i}\ \ is\ \ A_{2i}\ \ then\ z_i\ \ is\ \ C_i.$

Простой реализацией базы нечетких правил служит интерпретация базы правил как таблицы определения некоторой функции, то есть базу правил можно представить обучающей выборкой:

$$\{((A_{1i}, A_{2i}), C_i)\}.$$

Например, обучающая выборка в нечетких терминах может быть сформулирована следующим образом:

{((малое, большое), около нуля)}.

Чтобы совместить две технологии: технологию нечетких систем и технологию нейрокомпьютинга, необходимо предложить способ четкого дискретного представления непрерывных функций принадлежности. Чтобы представить в четких данных непрерывные функции принадлежности, выберем максимально большой интервал $[x_1,x_2]$, в котором представлены все нечеткие множества условных частей правил. Разбиваем интервал с равным шагом, тогда любое нечеткое значение представляется четким вектором. Другой способ представления нечеткого понятия в вид четких данных состоит в представлении нечеткого множества в виде совокупности α-срезов.

α-срезом называется четкое множество, включающее все элементы x некоторого нечеткого множества X, принадлежность которых больше равна α.

$$\mu_X(x) \geq \alpha .$$

Каждое α-подмножество представляется двумя числами – левой и правой границей α^L, α^R, то есть α-срезы четко представляют непрерывную функцию принадлежности.

Изменение элемента нейросети для адаптации к нечетким системам может касаться выбора функции активации, реализации операций сложения и умножения, так как в нечеткой логике сложение моделируется любой треугольной конормой (*max, a + b – a×b...*), а операция умножения треугольной нормой (*min, a×b ,...*).

И-нейроном (AND-нейроном) называется нейрон, в котором умножение веса на вход моделируется конормой *S(w,x)*, а сложение – нормой *T(w,x)*.

ИЛИ-нейроном (OR-нейроном) называется нейрон, в котором умножение веса и входа моделируется нормой $T(w,x)$, а сложение взвешенных весов конормой $S(w,y)$ $Y = S(T(w_1,x_1),T(w_2,x_2))$.

Пример: $(max(min(w_1,x_1),min(w_2,x_2)))$.

В качестве функции активации обычно используют функцию

$F(x) = 1/(1+exp(b(x-a)))$.

Нечеткой нейросетью называют четкую нейросеть, которая построена на основе многослойной архитектуры с использованием «И-», «ИЛИ-нейронов».

Нечеткая нейросеть функционирует стандартным образом на основе четких действительных чисел. Нечеткой является только интерпретация результатов. При создании гибридной технологии, кроме объединения систем по данным, можно использовать нейрокомпьютинг для решения частной подзадачи нечетких экспертных систем, а именно настройки параметров функции принадлежности. Функции принадлежности можно сформировать двумя способами: методом экспертной оценки или на основе статистики. Гибридные технологии предлагают третий способ: в качестве функции принадлежности выбирается параметризованная функция формы (например, параметризованная Гауссова кривая), параметры которой настраиваются с помощью нейросетей. Настройка параметров может быть получена в рамках алгоритма обратного распространения ошибки.

Пусть задана следующая система нечетких правил:

If x_1 *is* A_{1i} *and ...* x_n *is* A_{ni} *then* z_i *is* C_i ,

где A_i – нечеткие числа; C_i– действительные числа. Значение $\alpha_j=\Pi_i\alpha_i$ – сила или достоверность правила, $i = 1,..,n$ – номер входной переменной, $j = 1,..,m$ – количество правил.

$$Z = \frac{\sum_{j=1}^{m} a_j z_j}{\sum_{j=1}^{m} a_j},$$

где Z – вычисленное значение выхода.

Допустим, что разработана нейросеть с n входами и одним выходом. Каким образом такая НС может аппроксимировать базу нечетких правил? Любая совокупность нечетких правил может рассматриваться как нелинейное соответствие, заданное таблицей определения $\{(x,y)\}$, где x – вектор входа; y – желаемое значение выхода; а z – значение, вычисляемое нейросетью. Тогда можно определить текущую ошибку:

$E^K=1/2\times(z^K-y^K)^2$.

То есть можно применить стандартный алгоритм коррекции ошибки на основании данного определения

$Z(t+1) = Z(t)- \zeta\times (\partial E^K/\partial Z)$,

где ζ – уровень обучения; $\partial E^K/\partial Z$ – направление градиента снижения ошибки.

$Z(t+1) = Z(t)- \zeta\times (Z^K-Y^K) \times \alpha_j / (\alpha_1 +\alpha_2 +...+\alpha_m)$.

При применении стандартного алгоритма обратного распространения ошибки для того, чтобы настроить выход, необходимо изменить параметры функции принадлежности условных частей, то есть обучение сети позволит настроить функцию принадлежности с точки зрения обучающей выборки. При практической реализации системы нечетких правил важным является вопрос о типичных представителях нечетких значений в правилах. Большинство нечетких понятий, представленных лингвистическими переменными, выражает свои значения с помощью количественных нечетких множеств: NB – отрицательное большое; NM – отрицательное среднее; NS – отрицательное малое; ZE – около нуля; PS – положительное малое; PM – положительное среднее; PB – положительное большое.

Пример использования нечеткой нейронной сети

Рассмотрим нечеткие нейронные сети на примере нечеткого регулятора для стиральной машины [18], то есть построим fuzzy-neuro контроллер. Выберем для демонстрации технологии нечетких нейронных сетей архитектуру ANFIS (Adaptive Network Based Fuzzy Inference System). Основа интеграции нейронных сетей и систем нечеткого вывода заключается в том, что оба метода представляют

нелинейное отношение в пространстве входы-выходы. Важная задача нечеткого моделирования – это настройка функций принадлежности, являющаяся по существу задачей оптимизации. Как нейронные сети, так и генетические алгоритмы используются для ее решения. Самый простой подход требует назначить определенную параметризованную функцию формы в качестве функции принадлежности и подобрать параметры на основе обучения нейронной сети. Рассмотрим простой пример с тремя нечеткими правилами вывода в базе знаний.

R_1: ЕСЛИ <количество_белья> is <много>

И <температура_воды> is <высокая> И <загрязненность> is <высокая>

ТО <длительность> is <высокая>;

R_2: ЕСЛИ <количество_белья> is <много>

И <температура_воды> is <высокая> И <загрязненность> is <низкая>

ТО <длительность> is <низкая;

R_3: ЕСЛИ <количество_белья> is <мало>

И <температура_воды> is <низкая> И <загрязненность> is <низкая>

ТО <длительность> is <малая>;

Зададим следующие функции формы для высказываний. Пусть высказывание <количество_белья> is <мало> соответствует функции $L_1(x)$ и высказывание <количество_белья> is <много> – функции $H_1(x)$.

$L_1(x)= 1/(1+exp(b_1(x-c_1))$,

$H_1(x)= 1/(1+exp(-b_1(x-c_1))$,

причем $L_1(x) + H_1(x) =1$.

Аналогично для высказывания <температура_воды> is <высокая> будем использовать функцию $L_2(t)$ и для <температура_воды> is <низкая> – функцию $H_2(t)$.

$L_2(t)= 1/(1+exp(b_2(t-c_2))$,

$H_2(t)= 1/(1+exp(-b_2(t-c_2))$,

причем$L_2(t) + H_2(t) =1$.

Для высказывания <загрязненность> is <высокая> определим функцию $L_3(z)$ и для <загрязненность> is <низкая> – функцию $H_3(z)$.

$L_3(x)= 1/(1+exp(b_3(z-c_3))$,

$H_3(x)= 1/(1+exp(-b_3(z-c_3))$,

$L_3(z) + H_3(z) =1$.

Для выходов <длительность> is <высокая> и <длительность> is <малая> аналогично определим функции

$L_4(y)= 1/(1+exp(b_4(y-c_4))$,

$H_4(y)= 1/(1+exp(-b_4(y-c_4))$,

то есть $L_4(x) + H_4(x) =1$.

Для четких значений <количество_белья>, <температура_воды> и <загрязненность>: A_1, A_2, A_3 определим релевантность (силу) правил α:

$\alpha_1 = H_1 \wedge H_2 \wedge H_3$,

$\alpha_2 = H_1 \wedge H_2 \wedge L_3$,

$\alpha_3 = L_1 \wedge L_2 \wedge L_3$.

Выходы по каждому из правил определяется с помощью обратных функций принадлежности правых частей правил.

$Y_1 = H_4^{-1}(\alpha_1)$,

$Y_2 = H_4^{-1}(\alpha_2)$,

$Y_2 = L_4^{-1}(\alpha_3)$.

Общий выход из системы нечетких правил определяется как

$y_0 = (\alpha_1 \times y_1 + \alpha_2 \times y_1 + \alpha_3 \times y_1)/(\alpha_1 + \alpha_2 + \alpha_3)$.

Далее построим нечеткую нейронную сеть, идентичную системе нечеткого вывода, и обучим функции принадлежности анцедента и консеквента правил (рисунок 42).

Слой 1. Выходы узлов – это степени, в которых заданные входы удовлетворяют функциям принадлежности, ассоциированным с этими узлами.

Слой 2. Каждый узел вычисляет силу правила. Выход верхнего нейрона $\alpha_1 = H_1 \wedge H_2 \wedge H_3$, выход среднего нейрона $\alpha_2 = H_1 \wedge H_2 \wedge L_3$, а выход нижнего – $\alpha_3 = L_1 \wedge L_2 \wedge L_3$. Все узлы помечены Т, так как

можно выбрать любую Т-норму для моделирования логического И. Узлы этого слоя называются узлами правил.

Слой 3. Каждый узел помечен N, чтобы показать, что узлы нормализуют силу правил $\beta_i = \alpha_i / (\alpha_1 + \alpha_2 + \alpha_3)$.

Слой 4. Выход нейронов – это произведение нормализованной силы правила и индивидуального выхода соответствующего правила.

$\beta_1 Y_1 = \beta_1 H_4^{-1}(\alpha_1)$,

$\beta_2 Y_2 = \beta_2 H_4^{-1}(\alpha_2)$, $\beta_3 Y_2 = \beta_3 L_4^{-1}(\alpha_3)$.

Слой 5. Одиночный выходной нейрон вычисляет выход сети

$y_0 = \beta_1 Y_1 + \beta_2 Y_2 + \beta_3 Y_2$.

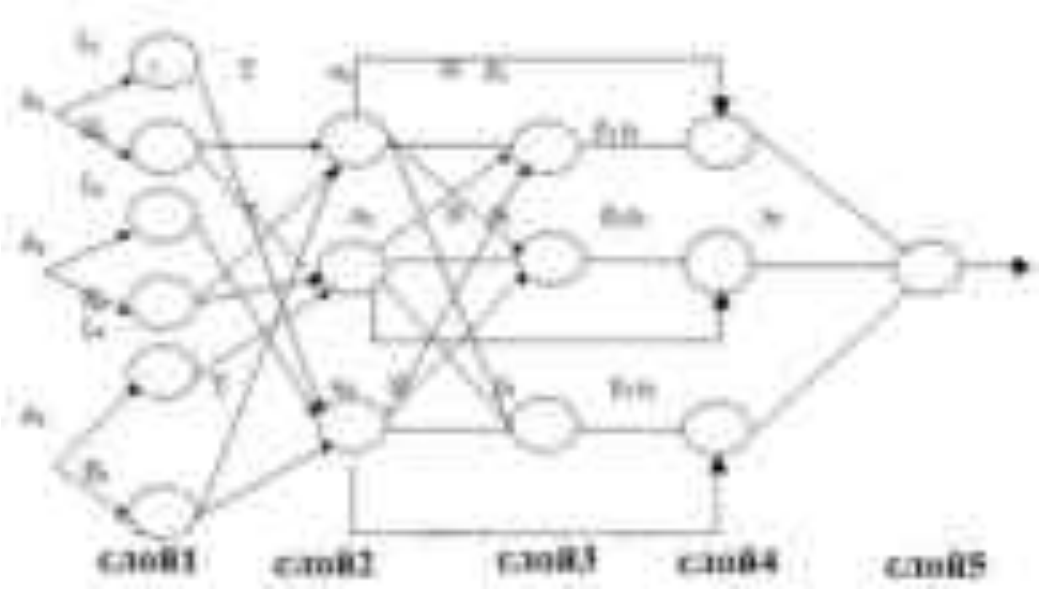


Рисунок 42. Пример нечеткой нейронной сети

Алгоритм обучения для нечеткой нейронной сети примера

Пусть задана четкая обучающая выборка

$\{(x_1,t_1, z_1 y_1), \ldots (x_k, t_k, z_k y_k)\}$, где x,t,z – входные условия количества белья, температуры воды и загрязненности, а y – длительность стирки. Ошибку на k-м образце определим как обычно:

$E^k = 1\backslash 2(O_i - Y_i)^2)$.

Используем традиционный градиентный метод для обучения параметров левой и правой частей нечетких правил. Покажем, как можно настроить параметры функции формы.

$b_4(t+1) = b_4(t) - \zeta \times (\partial E/\partial b_4)$,

$c_1(t+1) = c_1(t) - \zeta \times (\partial E/\partial c_1)$

.....

$c_4(t+1) = c_4(t) - \zeta \times (\partial E/\partial c_4)$.

Используя данные соотношения как правила изменения весов в алгоритме обратного распространения ошибки, можно настроить параметры функций принадлежности в ходе обучения нечеткой нейронной сети.

Генетические алгоритмы

Генетические алгоритмы представляют собой адаптивные методы поиска, использующиеся для решения задач функциональной оптимизации. В их основе лежит идея природной эволюции, когда популяции развиваются в течение нескольких поколений и подчиняются законам естественного отбора. Лучшая особь выбирается по принципу «выживает наиболее приспособленный». Как мы помним, он был открыт Чарльзом Дарвином. Согласно этим принципам генетические алгоритмы способны «развивать» решения реальных задач при соответствующем формальном описании. Причем в отличие от эволюции, генетические алгоритмы моделируют лишь существенные для развития процессы.

В природе действует такой механизм, что наиболее приспособленные особи имеют больше шансов на воспроизведения потомства. Причем комбинация наиболее хороших характеристик может привести к появлению максимально приспособленного потомка. Генетические алгоритмы используют прямую аналогию с этим механизмом. Множество возможных решений проблемы – это популяция. Каждое решение оценивается по степени его «приспособленности» для решения поставленной задачи.

Сфера применения генетических алгоритмов весьма обширна. Они могут использоваться при создании таких вычислительных структур, как автоматы или сети сортировки. В машинном обучении их

можно использовать для проектирования нейронных сетей или в управлении роботами. Их можно использовать для моделирования процессов в различных областях (биологические, социальные, когнитивные системы). Однако наиболее популярное приложение – оптимизация многопараметрических функций, когда нужно найти оптимальное значение, зависящее от некоторых входных параметров.

В своей работе генетический алгоритм работает с хромосомами, которые состоят из генов. Чаще всего ген – это 0 или 1, говорящая о включении или не включении признака, характеризующего решение, в хромосому. Соответственно, хромосома – это битовая строка, описывающая решение. Но иногда возможны и другие кодировки в зависимости от задачи.

При работе с генетическим алгоритмом нам необходимо определить следующее:

Кодировку хромосомы (что будет представлять собой ген, и как гены будут участвовать в характеристике решения).

Пространство гипотез (популяцию), из которых мы должны выбрать лучшую.

Функцию приспособленности, оценивающую хромосомы.

Набор и вид генетических операций (скрещивание, мутацию).

Критерий остановки алгоритма (либо желаемое оптимальное значение, либо количество шагов эволюции популяции).

Для примера рассмотрим решение задачи с помощью генетического алгоритма. Формулировка задачи: необходимо составить наиболее сбалансированный рацион питания, удовлетворяющий определенным медицинским требованиям. Известен перечень продуктов из N наименований, каждый из которых имеет такие характеристики, как содержание жиров, белков, углеводов, калорий, а также известны рамки допустимого их содержания. Подходящим считается рацион, который лучше всего удовлетворяет определенным медицинским требованиям. Например, если рекомендуемое содержание жиров – 50, белков – 60, углеводов – 70, клетчатки – 80, витамина B – 90, витамина C – 100, то лучшим будет считаться рацион, чье суммарное

отклонение минимально, но чья десятая часть, например, укладывается в рамки от 40 до 80.

Первое, что нам необходимо сделать, – выбрать кодировку хромосомы. Так как основная наша задача – подобрать список продуктов, то хромосомой может быть весь перечень из N продуктов, а 0 или 1 в ней будут говорить о включении или не включении продуктов в рацион. Для лучшего понимания рассмотрим конкретный пример. Пусть у нас есть 4 продукта, каждый из которых можно описать набором из 6 числовых параметров-характеристик. Тогда по сути получим массив чисел:

Список_объектов={Объект$_1$={20,30,40,50,60,70},

Объект$_2$={40, 40, 40, 40, 40, 40},

Объект$_3$={30, 30, 30, 30, 30, 30},

Объект$_4$={20, 30, 40, 40, 30, 30}}.

Тогда хромосома может иметь вид:

1010

Это говорит о том, что в рацион включаются первый и третий объекты. Тогда популяция может иметь вид:

0100

1010

1100

0101

Теперь определим функцию приспособленности для оценки каждой хромосомы в нашей задаче. Пусть у нас есть набор эталонных значений для каждой из характеристик объектов:

Эталон= {50,60,70,80,90,100}.

Тогда функция приспособленности будет иметь вид:

$$F = \sum_{j=1}^{N} \left(\frac{\sum_{i=1}^{m} \text{Эталон}_i - \text{Объект}_{ij}}{10} \right) \times Hrom_j \ ,$$

где $Hrom_j$– соответствующий ген в хромосоме.

Для оценки приспособленности хромосомы необходимо также проверить, укладывается ли значение функции в рамки от 40 до 80 (добавлением соответствующих условий), а для выявления наиболее приспособленной особи необходимо найти минимальную подходящую *F* в популяции.

Обратите внимание!

- *Определение вида функции приспособленности – чрезвычайно важная задача, потому что от правильности ее определения будет во многом зависеть успешность решения.*

Теперь рассмотрим генетические операции мутации и скрещивания. Если говорить формально, то мутация – это изменение одного или нескольких генов хромосомы вследствие случайного влияния. В контексте нашей задачи это может быть принудительное включение некоего продукта в набор, изменение значения вхождения продукта в набор на противоположное, принудительное исключение некоего продукта из набора и т. д. То есть, по сути, мутация – это изменение значения одного или нескольких генов хромосомы на противоположный или четко заданный. Вид и критерий применения мутации выбираются эмпирически. Допустим, для нашей задачи мы решили, что мутация будет изменять случайный ген на противоположный. Тогда, если исходная хромосома имела вид 0010, то мутировавшая может быть такой: 1010.

Операция скрещивания, или кроссовер, – операция, которая получает из двух хромосом одну, используя заданную маску. По сути из каждой хромосомы «вырезается» кусок, который помещается в новую. Существует несколько видов кроссовера. Одноточечный кроссовер: «разрез» хромосомы происходит только в одной точке, и новая особь получается путем соединения первой части первой хромосомы и второй части второй хромосомы. Двухточечный кроссовер: есть две точки «разреза», и новая хромосома получается из двух частей первой хромосомы и одной части второй.

Одноточечный кроссовер:

Хромосома$_1$ = 1001, Хромосома$_2$=1100. Точка разреза = 2. Маска потомка: 1часть_Хромосома$_1$+2часть_Хромосома$_2$. Потомок=(10)(00).

Двухточечный кроссовер:

Хромосома$_1$ = 1001, Хромосома$_2$=1100. Точки разреза = 1,3. Маска потомка: 1часть_Хромосома$_1$ + 2часть_Хромосома$_2$ + +3часть_Хромосома$_1$. Потомок=(1)(10)(1).

Теперь разберемся с таким вопросом, как отбор хромосом для воспроизведения потомства и выживание в популяции. Обычно в популяцию выбирается s наиболее приспособленных особей, которые и дают в ней потомство. Для того чтобы выбрать хромосомы в популяцию, можно использовать разные методы: турнирный, ранговый или метод рулетки. Рассмотрим эти методы подробнее.

Турнирный метод: задаем некую фиксированную вероятность выживаемости хромосомы, обозначив ее p. Случайно выбираем две особи. С вероятностью p выживает наиболее приспособлена, а 1-p – менее приспособленная.

Метод рулетки: вычисляем удельную приспособленность каждой особи относительно суммарной приспособленности популяции. Эту величину используем в качестве значения вероятности выживания.

Ранговый метод: выживает s наиболее приспособленных особей.

Таким образом, общую схему генетического алгоритма можно описать так:

Генерация начальной популяции из N особей.

Оценка приспособленности особей.

Пока не сработало условие выхода, делаем следующее:

Выберем s особей для новой популяции и кроссовера.

Выполним кроссовер и мутацию.

Оценим приспособленность итоговой популяции.

Нечеткие системы с генетической настройкой

Настройка функций принадлежности с помощью нейронной сети или генетического алгоритмы (ГА) устраняет принципиальную слабость теории нечетких систем – субъективность функций принадлежности. Применение нейронных сетей к настройке функций

принадлежности позволяет рассматривать окончательную форму функции как аппроксимацию обучающей выборки. Такую же задачу можно решить с помощью ГА, как метода стохастической оптимизации. Генетической нечеткой системой называют нечеткую систему, функции принадлежности и база правил которой спроектирована с помощью генетического алгоритма.

Применить ГА – это значит выбрать единицу кодирования, то есть хромосому; уточнить эволюционные операторы рекомбинации, мутации и селекции; сформировать функцию адаптивности (fitness-function, performanceindex). В настоящее время ГА используют либо для настройки функций принадлежности (базы данных), либо для формирования базы правил, либо для одновременного формирования и функций принадлежности и правил (а именно, базы знаний). В соответствии с объектами оптимизации выбирают единицу кодирования – хромосому. Для настройки функций принадлежности (ФП) за хромосому выбирают одно правило (Мичиганский подход), для настройки базы правил за хромосому выбирают вариант базы правил (Питтсбургский подход, подход итеративного обучения правил). В соответствии с кодированием уточняются правила генерации новых хромосом. Функция адаптивности представляет собой механизм нечеткого вывода, который для каждого варианта базы правил строит либо управление для нечеткого контроллера, либо экспертное заключение для диагностической экспертизы. Как видно из механизма нечеткого вывода, все нечеткие правила вносят вклад в окончательный результат, то есть правила сотрудничают. Но при отборе правил (хромосом) для генерации новых правил ГА накапливает правила, внесшие максимальный вклад в общий результат, то есть хромосомы конкурируют. В этом случае имеет место проблема «конкуренции и кооперации» в генетических нечетких системах (acompetition vs. a cooperation). Решение проблемы в каждом конкретном случае гене-тической нечеткой системы (ГНС) строится эвристически, например, при подходе итеративного обучения правил используют два этапа оптимизации. На первом шаге правила конку-

рируют за право войти в базу правил, а на втором – взаимодействуют при формировании общего результата.

Нечеткие нейронные сети с генетическим проектированием

Рассмотрим возможности генетических вычислений как средства структурной оптимизации нечетких нейронных сетей. Генетические вычисления можно применить на этапе проектирования нейронной сети. Возможности нейронных сетей интерполировать значения временных рядов могут быть широко использованы для оценки поведения макроэкономических показателей, в том числе индексов деловой активности или уровня ценных бумаг. Задача построения нейронного предиктора связана с принятием решений во многих точках проектирования. Необходимо исследовать входные данные и решить, по какому отрезку входных данных рационально делать предсказания, сколько точек в будущем разумно предсказать. Пространство перебора решений организации входных и выходных данных огромно для развитых рынков ценных бумаг, накопивших статистические сведения за десятки лет. Следующей точкой решения служит выбор типа нейронной сети: многослойный персептрон, радиально базисная сеть, вероятностная нейронная сеть, сеть регрессии и т. д. Для выбранного типа НС необходимо определить количество скрытых слоев, количество нейронов в них, виды функций активации и другие. Для каждой сети необходимо выбрать алгоритм обучения и его параметры, например, использование моментов (уровень моментов), уровень обученности, целевой уровень накопленной ошибки и т. д. Для нечеткой нейронной сети необходимо определить архитектуру сети, параметры функций принадлежности. В результате пространство решений при формировании нейронной сети становится необозримым для исследователя, и целесообразно применить ГА как средство эволюционного проектирования.

Генетическая оптимизация F-преобразования временных рядов

Задачи переборного типа с меньшими затратами могут быть решены путем применения эволюционных алгоритмов. Предлагается следующий классический алгоритм генетической оптимизации:

Имеется функция $F(x_1, \ldots, x_4)$ – оценка построения прогноза тренда (должна быть минимизирована), где

x_1 – степень авторегрессии при построении прогноза тренда;

x_2 – метод, которым производится прогноз;

x_3 – количество точек, покрываемых базисной функцией;

x_4 – прогноз на основании истории из предыдущих N точек.

Для выявления сезонности следует строить прогноз, отодвигая историю на t точек назад:

$$F_k = \alpha F_{k-t} + \beta F_{k-t-1} + \ldots$$

Алгоритм генетической оптимизации

1. Хромосомы имеют битовое представление (кодируем в коде Грея для получения отличия соседних хромосом в 1 бите). При этом в начале производится нормировка значений параметров к интервалу [0, 1].
2. Оператор скрещивания (возможен случайный выбор между данными вариантами оператора скрещивания).
3. Сформировать случайно начальную популяцию, состоящую из k особей $B_0 = \{A_1, A_2, \ldots, A_k\}$. При этом k определить эмпирически.
4. Вычислить приспособленность каждой особи $F_{Ai} = fit(A_i)$, $i=1 \ldots k$.
5. Выбрать двух особей A_c из популяции. $A_c = Get(B_t)$.
6. Произвести скрещивание (выбрав один из предложенных вариантов) и получить новую особь.
7. Произвести мутацию генов с некоторой вероятностью.

8. Оценить полученную особь функцией приспособленности и добавить в популяцию взамен худшего родителя (таким образом сохраняется количество особей популяции).

9. Выполнить пункты 5-8 m раз.

10. Увеличить счетчик эпох.

11. Проверить условие останова (предельное количество эпох или предельное количество особей одного генотипа в популяции).

РАЗРАБОТКА ПРИЛОЖЕНИЙ В СФЕРЕ МАШИННОГО ОБУЧЕНИЯ

Для реализации полноценного решения в сфере ПО необходимо учесть аппаратную и программную составляющую. Под аппаратной составляющей подразумевается следующее. Машинное обучение в ряде случаев требует подбор специального оборудования (или конфигурирование специальных серверов), рассчитанного на массивные вычислительные нагрузки или на тяжелые параллельные вычисления. Причем отличительной чертой можно назвать то, что мощное оборудование может потребоваться не только в производственной стадии проекта, но даже на ранних стадиях разработки для проведения экспериментов и соответствующего исследования моделей. Без мощного оборудования требуемые модели могут работать в 10-60 раз медленнее или просто не запуститься ввиду нехватки памяти (или видеопамяти).

В случае разработки встраеваемых решений может потребоваться сильная оптимизация кода и квалификация в написании кода под специальное оборудование (например, программируемые логические интегральные схемы). В случае разработки распределенных систем потребуется также применение специальных программных средств для организации распределенных вычислений.

Эти специализированные темы в данном пособии не рассматриваются.

Однако если речь не идет о разработке специализированного программно-аппаратного комплекса или встраевоемого решения, то наращивание вычислительных узлов на сегодняшний день является не столько инженерной проблемой сколько экономической.

Что же касается программной составляющей, то здесь стоит упомянуть следующее. На сегодняшний день огромное число различных алгоритмов уже реализовано в виде библиотек и пакетов, поэтому нет необходимости писать алгоритм нейронной сети или

агоритм решающих деревьев. За исключением опять же специализированных задач или исследовательских проектов.

Теоретически разработку систем МО можно вести на любом языке программирования, но при отсутствии специализированных библиотек сложность разработки может возрасти. Однако необходимо учесть, что специализированные средства создания моделей МО могут быть сложно применимы при реализации полноценных приложений. Поэтому нередко используют комбинацию технологий, когда предобработку данных и саму модель реализуют на одном языке программирования, а затем работающий прототип встраивают в приложение, написанное с использованием другой технологии. Одна из возможных архитектур реализации представлена на рисунке 43.

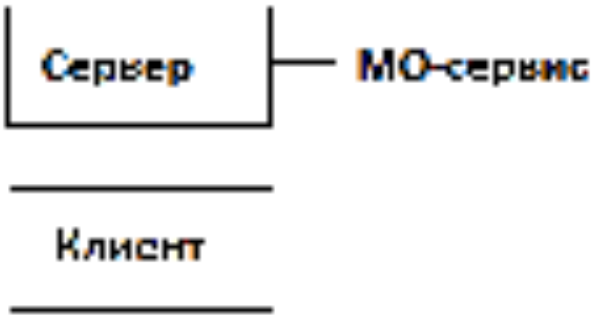


Рисунок 43. Возможная архитектура системы МО

Язык Python является одним из самых популярных языков для разработки в сфере машинного обучения и анализа данных ввиду очень простого синтаксиса, большого числа библиотек и развитого сообщества. Несмотря на то, что это интерпретируемый язык и выполнение кода на нем довольно медленное, большая часть библиотек реализована на быстрых языках типа Си, C++ и т. п. Поэтому на Python приходиться писать лишь высокоуровневое обращение к API библиотек. Что делает разработку на Python удобной, а решения – не уступающими по скорости со специальной реализацией на Си. В силу вышеизложенного будем работать именно с Python.

Однако прежде чем перейти непосредственно к Python, необходимо сказать, что при разработке систем МО, как правило, приходится пройти несколько типовых этапов. Таким образом, можно сформулировать общий алгоритм создания систем МО, который может быть изменен в силу особенностей конкретного проекта.

Алгоритм состоит из следующих шагов:

1. Сбор и предобработка данных;

2. Выбор метода решения, создание и настройка модели(подбор параметров, реализация прототипа);

3. Тестирование модели и совершенствование ее до достижения необходимой точности. При невозможности достижения точности – возврат к шагу 2;

4. Сохранение модели и интеграция ее в оболочку взаимодействия с пользователем;

5. Эксплуатация и сопровождение полученной системы и ее модернизация (при необходимости).

Основы работы с Python

Теперь перейдем к рассмотрению возможностей, предоставляемых Python для разработки систем МО. Общие сведения об этом языке можно посмотреть здесь: [30], а справочную информацию по синтаксису – здесь: [31]. В нашей же книге мы рассмотрим лишь то, что касается возможностей Python в плане создания приложений в сфере машинного обучения. Все примеры кода подготовлены на Python версии 2.7, но они могут быть адаптированы и под более старшие версии языка.

Как было сказано выше, сейчас Python является одним из наиболее распространенных языков программирования. Одним из его преимуществ является большое количество пакетов, решающих самые разные задачи. В данном пособии мы рекомендуем использовать библиотеки Pandas, NumPy и SciPy, которые существенно упрощают чтение, хранение и обработку данных. Вы также познако-

митесь с пакетом Scikit-Learn, в котором реализованы многие алгоритмы машинного обучения.

Необходимо отметить, что основными отличиями Python являются:

- Отсутствие завершающих символов в конце строки (точек, запятых и т. п.), что делает написание линейных конструкций очень «приятным» и быстрым занятием (а большинство программ для машинного обучения – это все-таки линейная математика, а не сложные многоуровневые системы).
- Нестрогая типизация. Не нужно объявлять тип данных при объявлении переменной, что опять же ускоряет процесс разработки модели.
- Язык Python интерпретируемый, кросс-платформенный и обладает хорошими средствами отладки.

Для разработки приложений нам нужно следующее:

- Язык, среда для разработки кода (IDE) и исполняемая среда. В нашем случае это Anaconda и PyCharm.
- Набор основных библиотек: Scikit-learn, Numpy, Pandas, Matplotlib, Theano, Keras.

Ниже представлены инструкции по установке и настройке компонент. Важный момент заключается в том, что все компоненты для работы в своем исходном состоянии «не родные» для систем Windows, поэтому в первую очередь эти инструкции относятся к пользователям Windows.

Инструкции по установке основных компонент:

1. Скачайте и установите AnacondaEnvironment (среда свободно распространяется на официальном сайте [32]). Anaconda необходима по большей части не как IDE, а как среда, в которой будет сразу установлено множество необходимых библиотек (развернутых соответствующим образом под Windows), таких как pip, numpy, matplotlib и еще пара десятков других. Также будет установлен и сам Python 2.7.

Обратите внимание!

- *Для корректной работы всех компонент аккаунт пользователя Windows, из под которого будет проводиться установка, должен содержать только латинские символы в своем имени.*

2. Рекомендуется установить PyCharm как основную IDE для разработки (хотя можно пользоваться самой SpiderAnaconda, которая будет установлена на предыдущем шаге) или JupyterNotebook. Однако если вы новичок в программировании или в Python, то рекомендуется именно PyCharm, так как эта среда сильно облегчает программирование и навигацию по коду. CommunityEdition распространяется бесплатно и скачать его можно здесь: [33].

3. Для проверки того, что все компоненты установлены корректно, запустите IDE, создайте новый файл (команда Alt+Insert или через меню File и пункт New) под именем testc кодом на Python, как показано на рисунке 44. Вставьте в файл код из листинга 1 и выполните его через пункт меню Run, как показано на рисунке 45.

Листинг 1

```
from sklearn import preprocessing
import numpy as np
X = np.array([[ 1., -1., 2.], [ 2., 0., 0.], [ 0., 1., -1.]])
X_scaled = preprocessing.scale(X)
print(X_scaled)
```

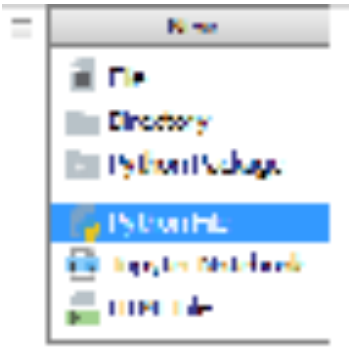


Рисунок 44. Новый файл Python

Обратите внимание!

- *Если в первом пункте меню Run нет имени вашего файла, то выберите третий пункт и в открывшемся окошке найдите имя вашего файла.*

Рисунок 45. Запуск кода на выполнение

Если все установлено верно, то в консоли вы должны увидеть результат, представленный на рисунке 46.

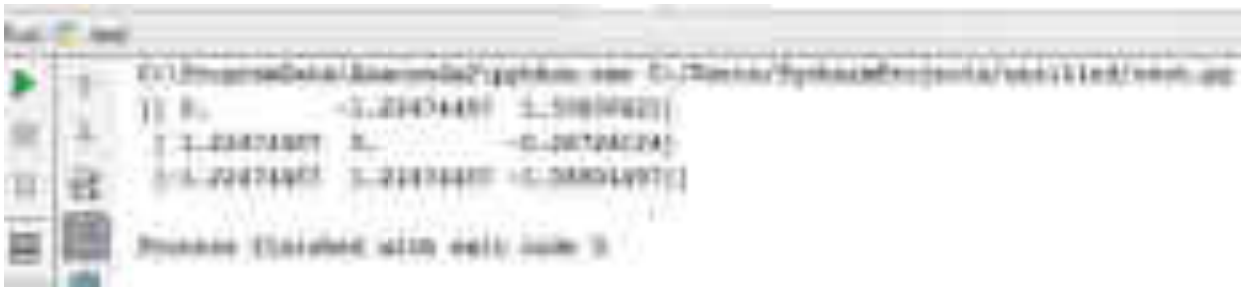


Рисунок 46. Корректная работа кода

Кроме описанного выше, нам понадобится установить и настроить еще ряд компонентов и библиотек. Инструкции по их установке следующие:

1. Скачайте и установите TDM-GCC (специальное средство компиляции для ОС Windows);
2. Откройте консоль Anacondaprompt или обычную командную строку Windows через команду cmd;
3. Выполните в консоли conda update conda;
4. Выполните в консоли conda update --all;
5. Выполните в консоли conda install mingw libpython;

6. Выполните в консоли conda install Theano;

7. Выполните в консоли pip install keras;

Обратите внимание!

- *Keras может быть настроен на tensorflowbackend, тогда надо внести изменения в файл C:/Users/AccauntName/.keras/keras.json и в строке backend=”tensorflow” написать theano;*

8. Запустите IDE, создайте новый файл с кодом на Python и выполните код из листинга 2 для проверки того, что все компоненты установлены корректно.

Листинг 2

```
import numpy as np
import sklearn.linear_model as lin
from keras.layers.core import Dense, Activation
from keras.models import Sequential
from keras.optimizers import RMSprop

x_train = np.array([[0.1, 0.3], [0.2, 0.2], [0.7, 0.8], [1.0, 0.9]])
y_train = np.array([0, 0, 1, 1]).reshape(4, 1)
x_test = np.array([[0, 0], [0.3, 0.3], [0.6, 0.7], [1, 1]])
model = Sequential()
model.add(Dense(2, init='lecun_uniform', input_shape=(2,)))
model.add(Activation('relu'))
model.add(Dense(1, init='lecun_uniform'))
model.add(Activation('linear'))
rms = RMSprop(lr=0.01)
model.compile(loss='mse', optimizer=rms)
epochs = 200
model.fit(x_train, y_train, batch_size=1, nb_epoch=epochs, verbose=0)
y_predict = model.predict(x_test, batch_size=1)
print(y_predict)
```

В результате в Output консоли не должно быть ошибок (код выхода равен 0), и должна будет распечататься информация, представленная в листинге 3.

Листинг 3

```
[[ 0.00339771]
[ 0.14396997]
[ 0.67507285]
[ 1.1803354 ]]
```

Мы установили основные, необходимые нам, компоненты. Теперь перейдем непосредственно к рассмотрению конкретных приемов работы с ними.

В данном пособии будет приведено описание только тех операторов и функций, которые непосредственно используются при решении конкретной задачи. Для получения прочей информации по Python рекомендуется воспользоваться справочником или же поисковой системой Google.

Обратите внимание!

- *Если у вас есть конкретный вопрос, например, о том, как выбрать последний элемент в одномерном или двумерном массиве, или о том, как отсортировать массив и т. п., то лучшее решение – написать этот вопрос в Google на английском;*
- *Если ваш уровень английского недостаточен, то просто сформулируйте вопрос на русском максимально коротко, затем вставьте в GoogleTranslate и затем – в поисковик. С вероятностью 90% на такие конкретные вопросы вы найдете очень конкретные примеры кода на сайте StackOverflow.*

Элементарные операции с данными

Рассмотрим работу по предобработке данных и поиску простых закономерностей в них средствами Python, а именно средствами пакетов Numpy и Pandas. Более подробно с их функциями можно ознакомиться, например, здесь: [34].

Для импорта модуля Numpy необходимо написать следующую строку кода (листинг 4).

Листинг 4

```
Import numpy as np
```

Обратите внимание!

- *Numpy импортируется с псевдонимом np, через который в дальнейшем будет обращение к модулю.*

Основными единицами данных в машинном обучении являются векторы и матрицы. С точки зрения программирования такие структуры представляют собой одномерные и двумерные массивы или специальные объекты-таблицы.

Обратите внимание!

- *Если в тексте написано просто – вектор или матрица, то значит идет речь о одномерном или двумерном массиве Numpy. В других случаях будет специально написано в каком формате представлены данные (например, DataFrame, о чем речь пойдет ниже).*

Для того чтобы рассмотреть способы работы с данными, нам необходимо получить данные для работы. Так как модели, разрабатываемые на Python, в 99% случаев используют полученные откуда-то данные или же генерируют тестовые, то мы опишем три способа получения данных: прямое объявление, случайная генерация и загрузка из csv-формата.

Начнем с первого. Объявить матрицу можно, используя код из листинга 5. Как видите, ничего сложного здесь нет: ни предварительного объявления данных, ни выделения памяти, ни объявления типа данных.

Листинг 5

```
Z = np.array([[4, 5, 0],
              [9, 9, 9]])
```

Теперь рассмотрим второй способ: сгенерируем случайную матрицу, состоящую из 6 строк и 5 столбцов. Элементы ее будут являться случайными числами из нормального распределения. Функция для генерации таких чисел: np.random.normal.

Ее параметры:

- loc: среднее нормального распределения (в нашем случае 1);
- scale: стандартное отклонение нормального распределения (его значение в нашем случае равно 10);
- size: размер матрицы (в нашем случае (6, 5)).

Код генерации матрицы X и ее распечатки представлены в листинге 6.

Листинг 6

```
X = np.random.normal(loc=1, scale=10, size=(6, 5))
print X
```

Далее рассмотрим способ загрузки данных из файла в csv-формате, который предназначен для хранения табличных данных. Как правило, в файлах этого формата столбцы разделяются запятой, а первая строка содержит их имена.

Допустим, у нас есть файл «titanic.csv», содержащий данные в виде, представленном на рисунке 47.

Обратите внимание!

- *Небольшие файлы csv (до 1 Гб) можно достаточно удобно посмотреть в Excel. Чтобы корректно загрузить CSV файл, нужно сначала открыть Excel, создать пустую книгу, а затем вызвать соответствующего мастера загрузки через вкладку «Данные»=>«Из текста»..*

Рисунок 47. Файл с данными

Для загрузки этих данных нам необходимы средства библиотеки pandas. Код ее импорта, а также код загрузки и распечатки загруженных данных представлены в листинге 7. Там же представлен код записи данных в csv-файл.

Листинг 7

```
# -*- coding: utf-8 -*-
import csv
import pandas
#чтение из файла
data = pandas.read_csv('titanic.csv', index_col='PassengerId')
print data
#запись в файл данных в исходном виде
data.to_csv('q.csv')
#запись в файл данных как массива значений
with open('test.csv', 'w') as csvfile:
  spamwriter = csv.writer(csvfile, delimiter=';',quotechar='|', quoting=csv.QUOTE_MINIMAL)
  spamwriter.writerow (data.as_matrix())
```

Обратите внимание!

- *Символ # - знак комментария.*
- *Для того чтобы закомментировать (или раскомментировать) несколько строк, необходимо выделить их и нажать сочетание клавиш Ctrl+правый слеш.*
- *Для того чтобы писать комментарии в Python на русском языке, необходимо добавить следующую строку в начало файла с кодом:# -*- coding: utf-8 -*-*

При выполнении кода из листинга 7 данные будут загружены в виде DataFrame, с помощью которого можно удобно работать с ними. Параметр index_col='PassengerId' означает, что колонка PassengerId задает нумерацию строк данного DataFrame.

DataFrame – первичная структура данных pandas, представляющая собой двумерную, изменяемую по размеру, потенциально гетерогенную структуру табличных данных с маркированными осями (строками и столбцами).

В виде DataFrame над данными удобно производить различные манипуляции сортировки, выборки, применения функций построчно или по колонкам и т. п. Однако DataFrame – это комплексный объект высокого уровня и его нельзя использовать напрямую для обучения моделей. Перед этим его нужно перевести в обычную цифровую матрицу, например, следующим образом (листинг 8), и дальше работать с ним как с обычным двумерным массивом данных.

Листинг 8

```
x_test= data[['ColumnName_1', 'ColumnName_2']].as_matrix()
```

Вернемся к листингу 7. В нем представлены два способа записи в файл. Первый – через метод самого DataFrame, второй – через метод библиотеки csv. Необходимо отметить, что второй способ может быть использован не только для работы с DataFrame, но и с любыми данными, которые надо записать в csv-файл.

Мы рассмотрели основные способы получения данных, теперь перейдем к работе с ними и продемонстрируем некоторые возможности библиотеки Numpy по операциям с матрицами на конкретных задачах из практики.

Допустим, перед нами стоит следующая задача: есть данные о ежедневной выручке по филиалам компании за первые 12 дней месяца, записанные в виде таблицы (рисунок 48) и сохраненные в файле «data.csv». Выведем номер филиала, преодолевшего по прибыли порог, равный 1000.

Первое, что нам необходимо сделать, – загрузить данные. Однако, чтобы pandas корректно обработал файл, нам нужно убрать из него кириллицу и убедиться, что разделители в csv действительно запятые. Для этого мы можем поместить файл в папку проекта pyCharm, двойным щелчком мыши на нем (в окне проекта) открыть его и привести к виду, показанному на рисунке 49. Теперь для чтения данных, преобразованию их в матрицу и распечатки полученного массива мы можем использовать код из листинга 9.

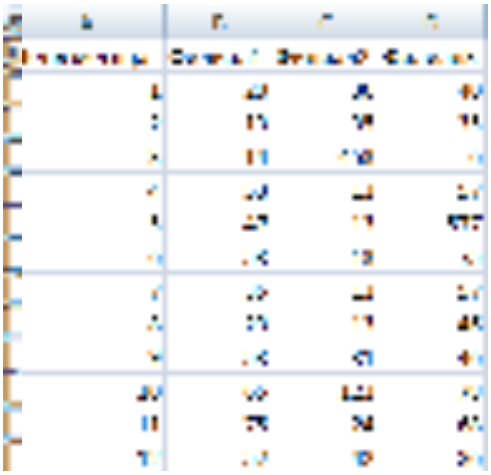


Рисунок 48. Файл с данными для задачи

Рисунок 49. Корректный файл с данными

Обратите внимание!

- *Если исходный файл создан через Excel, то разделителями могут быть точки с запятой. Заменить их на запятые можно при редактировании файла в PyCharm через цепочку пунктов меню «Edit»=>«Find» =>«Replace».*

Листинг 9

```
import pandas
data= pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
print x_test
```

Если все сделано верно, в консоли должен появиться результат, показанный на рисунке 50.

Рисунок 50. Загруженные данные

Обратите внимание!

- *При преобразовании к матрице мы указываем в кавычках имена столбцов с данными по филиалам и не указываем столбец Day, так как содержащиеся в нем данные для задачи не нужны.*

Теперь перейдем непосредственно к решению поставленной задачи. С точки зрения работы с матрицей, для получения ответа нам

необходимо посчитать сумму по каждому столбцу и вывести номера тех, чья сумма превысит 1000.

Функция для подсчета суммы: np.sum. В качестве параметров она принимает матрицу, для которой необходимо посчитать сумму и измерение (строки или столбцы), которое необходимо суммировать. Измерение (axis) задается цифрой (для двумерной матрицы 0 – строки, 1 – столбцы).Если этот параметр не задать, то результат функции будет рассчитан для всей матрицы целиком. Результатом выполнения операции будет массив с соответствующими суммами. Библиотека Numpy предоставляет возможности применения к матрицам (и массивам) логических операций, причем применяемых поэлементно. Соответственно, результатом такой операции будет матрица такого же размера, в ячейках которой будет записано либо True, либо False (удовлетворяет текущий элемент условию или нет). Индексы элементов со значением True можно получить с помощью функции np.nonzero. Функция в качестве параметра принимает матрицу, в которой необходимо отыскать ненулевые элементы. Заметим, что в нашем случае мы можем сразу передать логическое выражение, составленное из массива сумм и самого условия, тогда элементы со значением False будут интерпретироваться как нулевые. Код решения представлен в листинге 10. Результат работы кода – на рисунке 51. Ответ к задаче: третий филиал.

Листинг 10

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
#print x_test
r = np.sum(x_test, axis=0)
print (r)
print np.nonzero(r> 1000)
```

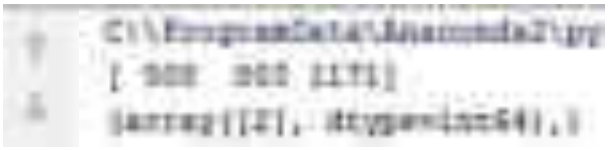


Рисунок 51. Решение задачи

Теперь решим следующую задачу: определим, в какой день суммарная выручка по филиалам превысила 500. Код решения представлен в листинге 11, а результат – на рисунке 52.

Рисунок 52. Решение задачи

Листинг 11

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.sum(x_test, axis=1)
print (r)
print np.nonzero(r>500)
```

Код нахождения филиала с максимальной средней выручкой представлен в листинге 12, а результат показан на рисунке 53.

Листинг 12

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.mean(x_test, axis=0)
print (r)
print np.nonzero(r== np.max(np.mean(x_test, axis=0)))
```

Рисунок 53. Номер филиала с максимальной средней выручкой

Номер филиала с максимальной величиной стандартного отклонения прибыли можно получить, используя код из листинга 13.

Далее нам необходимо более подробно рассмотреть работу со структурой DataFrame, но перед этим упомянем еще две полезные функции Numpy, которые могут понадобиться в последующем: функцию генерации единичной матрицы (np.eve) и функцию вертикальной стыковки матриц (np.vstack). Первая в качестве параметра принимает количество строк (оно же количество столбцов), вторая – матрицы, которые нужно объединить. В листинге 14 показан пример генерации и объединения двух единичных матриц.

Листинг 13

```
import numpy as np
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
x_test= data[['Fil1', 'Fil2', 'Fil3']].as_matrix()
r = np.std(x_test, axis=0)
print (r)
print np.nonzero(r== np.max(np.std(x_test, axis=0)))
```

Листинг 14

```
import numpy as np
A = np.eye(3)
B = np.eye(3)
print A
print B
AB = np.vstack((A, B))
```

Работа с DataFrame

Мы рассмотрели основные моменты работы с матрицами, поэтому теперь вернемся к другой структуре данных: DataFrame. Как мы помним, она является таблицей, где колонки имеют заголовки, а строки, кроме номеров, могут иметь дополнительные индексы в виде цифр или имен (по сути, индексы представляют собой отдельную, специальную колонку в таблице DataFrame, которая не относится к данным). Для того чтобы посмотреть, что представляют собой данные, можно воспользоваться несколькими способами.

Если мы хотим распечатать первые 5 строк данных, то можем воспользоваться кодом из листинга 15, в случае, если указан только один индекс, или же воспользоваться методом head() DataFrame, как показано в листинге 16.

Листинг 15

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data[:5]
```

Листинг 16

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data.head()
```

Если же нас интересует содержимое конкретного столбца, то можно использовать квадратные скобки и название столбца, как показано в листинге 17.

Листинг 17

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data['Fil1']
```

Обратите внимание!

- *Код из листинга 16 не применим к индексному столбцу.*

Для подсчета некоторых статистик (количества, среднее, максимум, минимум) можно также использовать методы объекта DataFrame. В листинге 18 приведен пример кода, считающего количество повторов каждого из значений в столбце с именем 'Fil1'. Результат работы кода показан на рисунке 54. Значения суммы, минимума, максимума, среднего, количества и среднеквадратического отклонения вычисляются аналогично с использованием соответствующих функций (sum, min, max, mean, count, std).

Листинг 18

```
import pandas
data = pandas.read_csv('data.csv', index_col='Day')
print data['Fil1'].value_counts()
```

Рисунок 54. Результат работы кода из листинга 18

Теперь рассмотрим еще несколько более сложных функций работы с DataFrame на примере некоторой таблицы, представленной на рисунке 55.

В листинге 19 представлены интересующие нас фрагменты кода с комментариями. Как можно заметить, возможности DataFrame весьма обширны, поэтому здесь приведены лишь основные моменты, а с остальным вам предлагается ознакомиться самостоятельно.

Рисунок 55. Структура таблицы с данными

Обратите внимание!

- *Для лучшего понимания кода листинга 19 вам рекомендуется почитать про Лямбда функции в Python, например, здесь: [35];*
- *Более подробно со списком методов DataFrame можно познакомиться в документации: [36];*
- *Ссылка на уроки по Pandas: [37].*

Листинг 19

```
#-*- coding: utf-8 -*-
# Выбор элементов DataFrame с использованием iloc
# Строки:
data.iloc[0] # выбирает первую строку таблицы.
data.iloc[1] # выбирает вторую строку таблицы
data.iloc[-1] # выбирает последнюю строку таблицы

# Столбцы:
data.iloc[:,0] # выбирает первую колонку таблицы
data.iloc[:,1] # выбирает вторую колонку таблицы
data.iloc[:,-1] # выбирает последнюю колонку таблицы

# Множественный выбор
data.iloc[0:5] # выбирает первые пять строк таблицы
data.iloc[:, 0:2] # выбирает первые две колонки таблицы со всеми строками

# Выбирает строки с индексами 'Andrade' и 'Veness' (в случае текстовых индексов),
#со всеми колонками между колонок с именем 'city' и 'email'
data.loc[['Andrade', 'Veness'], 'city':'email']
```

Окончание листинга 19

```
# Выбирает те же строки, только с колонками 'first_name', 'address' и 'city'
data.loc['Andrade':'Veness', ['first_name', 'address', 'city']]

# Выбирает строки с firstname равным Antonio и выбирает только колонки между
#колонками 'city' и 'email'
data.loc[data['first_name'] == 'Antonio', 'city':'email']

# Выбирает строки со всеми колонками, где в колонке email встречаются ячейки,
#заканчивающиеся на 'hotmail.com'
data.loc[data['email'].str.endswith("hotmail.com")]

#Выбирает строки, где first_name равно одному из следующих значений
data.loc[data['first_name'].isin(['France', 'Tyisha', 'Eric'])]

# Выбирает строки, где в колонке first_name встречается Antonio и в колонке email
#встречается gmail.com
data.loc[data['email'].str.endswith("gmail.com") & (data['first_name'] == 'Antonio')]

# Выбирает строки, у которых в колонке id есть значения от 100 до 200, и возвращает
#только колонки 'postal' и 'web'
data.loc[(data['id'] > 100) & (data['id'] <= 200), ['postal', 'web']]

#Лямбда функция, которая возвращает True/False переменную.
# Выбирает строки, где ячейка в колонке company_name имеет 4 слова.
data.loc[data['company_name'].apply(lambda x: len(x.split(' ')) == 4)]

#Распечатка заголовков
print list(data)
```

Предобработка данных. Стандартизация и нормализация

Мы рассмотрели основные моменты работы с данными, теперь перейдем к такому вопросу, как предобработка данных. Прежде чем обучать и использовать большинство моделей, необходимо привести исходные данные к единообразному виду (к единому диапазону). Единый диапазон определяется через математическое ожидание выборки и разброс (дисперсию). То есть нужно, например, чтобы все данные были в диапазоне от 0 до 1 или от –1 до 1. Как было сказано

в одном из предыдущих разделов, достигается такое приведение посредством двух процессов\методов: стандартизации данных и нормализации.

Можно провести стандартизацию самостоятельно, используя Numpy. Для этого вычтите из каждого столбца его среднее значение, а затем поделите на его стандартное отклонение. Или же можно использовать библиотеку scikit-learn, как показано в листинге 20.

Однако, чаще всего, кроме нормировки одной матрицы, потребуется нормировать несколько матриц по одним и тем же шкалам (для каждого столбца\признака будут свои коэффициенты смещения и масштабирования).

Так если тестовая\рабочая выборки поступают в систему онлайн, то их потребуется нормировать по тем же коэффициентам, что и тренировочную (рассчитанным на тренировочной выборке). Иначе нормировка будет некорректной.

Листинг 20

```
from sklearn import preprocessing
import numpy as np
X = np.random.normal(loc=1, scale=10, size=(6, 5))
X_scaled = preprocessing.scale(X)
print X
print X_scaled
```

Вместо того чтобы хранить матрицы коэффициентов нормировки, гораздо удобнее воспользоваться классом StandardScaler из scikit-learn. Сперва нужно «обучить» его на тренировочных данных (методом fit), а потом преобразовывать разные матрицы одинаковым образом. Работа с этим классом показана в листинге 21.

Листинг 21

```
from sklearn import preprocessing
import numpy as np
X = np.random.normal(loc=1, scale=10, size=(6, 5))
X_test = np.random.normal(loc=1, scale=10, size=(6, 5))
```

Окончание листинга 21

```
scaler = preprocessing.StandardScaler().fit(X)
train_data_scaled = scaler.transform(X)
test_data_scaled = scaler.transform(X_test)
print train_data_scaled
print test_data_scaled
```

Как было сказано в одном из предыдущих разделов, стандартизация смещает значения векторов (выравнивает относительного единого центра в нуле), а также производит выравнивание разброса. Однако значения разных векторов не будут в одинаковом диапазоне (от –1 до 1, например). Они будут лишь иметь стандартный разброс в рамках вектора\столбца\признака. Для того чтобы достичь одинакового масштаба всех векторов, необходима нормализация.

Средствами scikit-learn нормализацию можно выполнить следующим образом (листинг 22).

Листинг 22

```
#-*- coding: utf-8 -*-
from sklearn import preprocessing
import numpy as np
X=[[1., -1.,2.],
  [2., 0., 0.],
  [0., 1., -1.]]

#нормализация max-нормой
X_normalized_max = preprocessing.normalize(X, norm='max', axis=0)

#нормализация нормой l1
X_normalized_max = preprocessing.normalize(X, norm='l1', axis=0)

#нормализация нормой l2
X_normalized_max = preprocessing.normalize(X, norm='l2', axis=0)
```

Более подробную информацию о методах библиотеки scikit-learn по предобработке данных можно найти здесь: [38].

Работа с деревьями решений

Мы рассмотрели методы предобработки данных. Теперь перейдем к следующему – к деревьям решений. Как было сказано в одном из предыдущих разделов, решающие деревья относятся к классу логических методов. Их основная идея состоит в объединении определенного количества простых решающих правил, благодаря чему итоговый алгоритм является интерпретируемым. Как следует из названия, решающее дерево представляет собой бинарное дерево, в котором каждой вершине сопоставлено некоторое правило вида «j-й признак имеет значение меньше b». В листьях этого дерева записаны числа-предсказания. Чтобы получить ответ, нужно стартовать из корня и делать переходы либо в левое, либо в правое поддерево в зависимости от того, выполняется правило из текущей вершины или не выполняется.

Одна из особенностей решающих деревьев заключается в том, что они позволяют получать важности всех используемых признаков. Важность признака можно оценить на основе того, как сильно улучшился критерий качества благодаря использованию этого признака в вершинах дерева.

Рассмотрим данные о пассажирах «Титаника» (их можно скачать здесь: [39]). Решим на них задачу классификации, в которой по различным характеристикам пассажиров требуется найти у выживших пассажиров два наиболее важных признака (из четырех рассматриваемых: пол, класс, возраст, цена билета).

В библиотеке scikit-learn решающие деревья реализованы в классах sklearn.tree.DecisionTreeClassifier (для классификации) и sklearn.tree.DecisionTreeRegressor (для регрессии). Обучение модели производится с помощью функции fit. Найти важность признаков можно, имея уже обученный классификатор: его поле feature_importances_ содержит массив «важностей» признаков. Индекс в этом массиве соответствует индексу признака в данных. Стоит обратить внимание, что данные могут содержать пропуски.

Pandas выгружает такие значения как nan (not a number). Для того чтобы проверить, является ли число nan'ом, можно воспользоваться функцией numpy isnan.

Перейдем к решению задачи. Первое, что нам необходимо сделать, – внимательно посмотреть на данные перед загрузкой. Ранее мы уже предположили, что среди них могут быть пропуски, и определились с методом решения этой проблемы, но при изучении информации мы должны увидеть еще одну: признак Sex имеет строковые значения, тогда как все остальные – числовые. Следовательно, нам необходимо привести и его к числовому виду. Например, заменим male на 0, остальное – на 1.

Для этого определим функцию, которую применим к нашему массиву данных. В Python функции начинаются с ключевого слова def, и их операторы (тело функции) обязательно имеют отступ от начала строки. Применение функции к объекту выполняется оператором apply. Код функции представлен в листинге 23.

Листинг 23

```
def Sex_to_bool(sex):
    if sex == "male":
        return 0
    return 1
```

Алгоритм решения задачи будет следующим:

1. Загрузить выборку из файла titanic.csv с помощью пакета Pandas.
2. Оставить в выборке четыре признака: класс пассажира (Pclass), цену билета (Fare), возраст пассажира (Age) и его пол (Sex). Привести пол к числовому виду и убрать из выборки пустые значения.
3. Выделить целевую переменную (она записана в столбце Survived).

4. Обучить решающее дерево с параметром random_state=241 и остальными параметрами по умолчанию (речь идет о параметрах конструктора DecisionTreeClassifier).
5. Вывести важности признаков.

Код решения задачи (с комментариями) – в листинге 24. Результат работы – на рисунке 56.

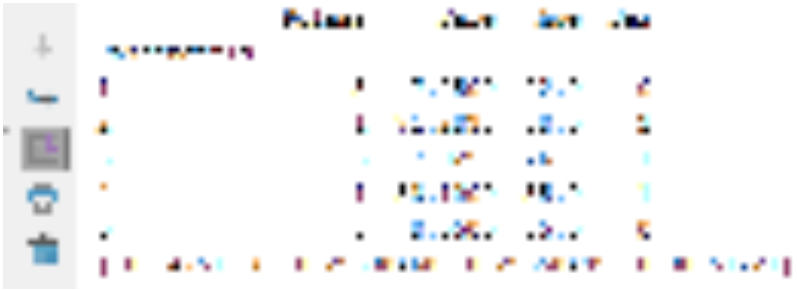


Рисунок 56. Важности признаков

Листинг 24

```
-*- coding: utf-8 -*-
import pandas
from sklearn.tree import DecisionTreeClassifier
import numpy as np

data = pandas.read_csv('titanic.csv', index_col='PassengerId')
#функция для приведения пола к числу
def Sex_to_bool(sex):
    if sex == "male":
        return 0
    return 1
#приведение пола к числу
data['Sex'] = data['Sex'].apply(Sex_to_bool)
#выгрузка непустых данных
data=data.loc[(np.isnan(data['Pclass'])==False) & (np.isnan(data['Fare'])==False)
&(np.isnan(data['Age'])==False) & (np.isnan(data['Sex'])==False)&
(np.isnan(data['Survived'])==False)]
#отбор нужных столбцов
corr = data[['Pclass', 'Fare', 'Age', 'Sex']]
#респечатка первых 5 строк данных
print corr.head()
```

Окончание листинга 24

```
#определение целевой переменной
y = data['Survived']
#создание и обучение дерева решений
clf = DecisionTreeClassifier(random_state=241)
clf.fit(corr, y)
#получение и распечатка важностей признаков
importances=clf.feature_importances_
print importances
```

Таким образом мы видим, что наиболее важными признаками будут пол и цена билета. Однако по описанному решению стоит сделать одно важное замечание. При преобразовании пола к числовому виду мы исказили характер данных: ранее объекты столбца «пол» не могли быть математически сравнимы между собой, а после наших преобразований эта характеристика у них появилась. Это привело к неоднозначности итогового решения: при male=0 массив ответа имеет вид: [0.14751816 0.29538468 0.25658495 0.30051221], а при female=0 – [0.14000522 0.30343647 0.2560461 0.30051221]. Конечно, в нашей ситуации изменение незначительное и не влияет на итоговый ответ, но в другой ситуации это может быть не так. Поэтому вам предлагается самостоятельно подумать, как избежать подобного искажения данных.

Работа с линейной регрессией

Мы рассмотрели работу с решающими деревьями, теперь же перейдем к рассмотрению регрессии. Начнем с линейных моделей, приведя в качестве примера решение задачи прогнозирования уровня зарплаты по данным вакансий.

Линейные методы хорошо подходят для работы с разреженными данными. К таковым относятся, например, тексты. Это можно объяснить высокой скоростью обучения и небольшим количеством параметров, благодаря чему удается избежать переобучения.

Линейная регрессия имеет несколько разновидностей в зависимости от того, какой регуляризатор используется. В рамках предлагаемой задачи мы будем использовать гребневую регрессию, где применяется квадратичный, или L2-регуляризатор. Эта модель реализована в классе sklearn.linear_model.Ridge. Более подробно о реализации гребневой регрессии можно почитать здесь: [40].

Исходные данные объемом 60 000 записей хранятся в файле «salary_train.csv», тестовые данные (2 записи) хранятся в файле «salary_test_mini.csv» (скачать их можно отсюда: [41]). Оба файла имеют вид, представленный на рисунке 57.

В данном случае столбцы A-C будут содержать значения входных признаков, а последний столбец – целевой переменной.

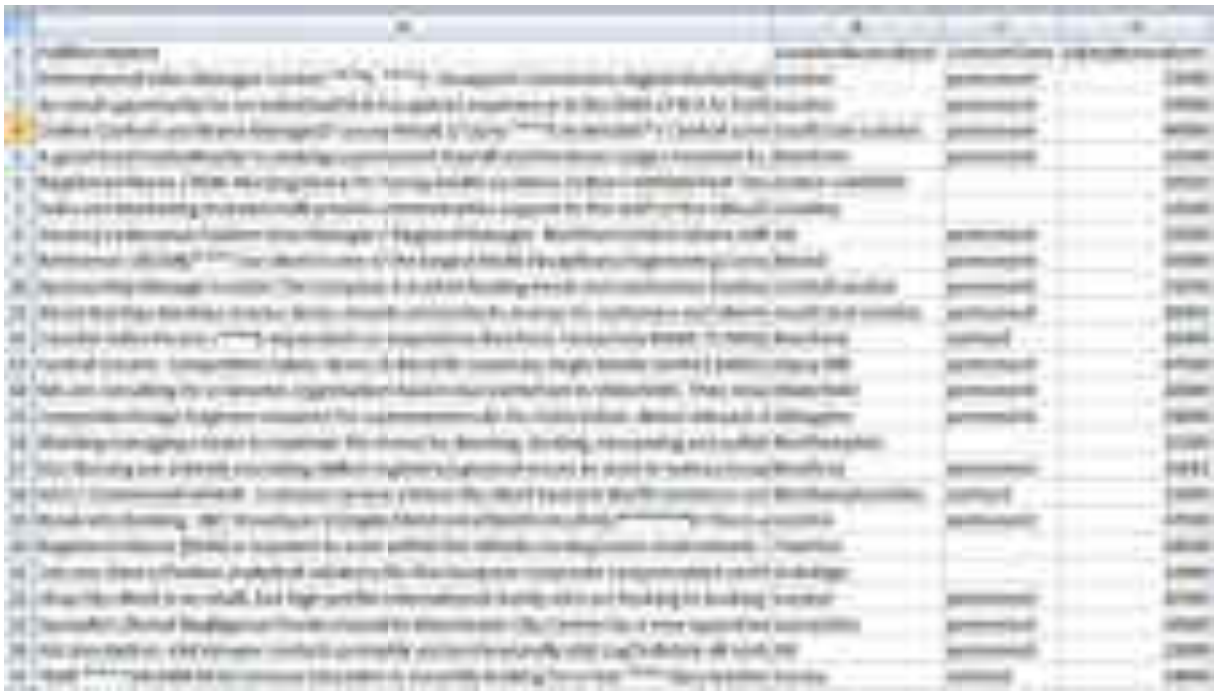


Рисунок 57. Исходные данные

Как можно заметить, первый столбец содержит объемный текст, который необходимо предобработать для загрузки в модель. Например, извлечь TF-IDF-признаки, воспользовавшись классом sklearn.feature_extraction.text.TfidfVectorizer, который преобразует массив текста в матрицу, содержащую TF-IDF-признаки. Более

подробно об этом классе можно почитать здесь: [42]. Использование его для решаемой задачи показан в листинге 25.

Листинг 25

```
vectorizer = TfidfVectorizer(min_df=10)
train_text_feature_matrix = vectorizer.fit_transform(data_train['FullDescription'])
idf = vectorizer.idf_
print dict(zip(vectorizer.get_feature_names(), idf))
```

В данном случае поле idf_ класса TfidfVectorizer содержит вектор с глобальными весовыми коэффициентами. Строка кода print dict(zip(vectorizer.get_feature_names(), idf)) выведет в консоль массив данных вида: {u'pre': 5.6339293213815242, u'paperless': 9.0576775285655771, u'peoplewithchemistry': 9.6042212349336467, u'rfps': 9.1117447498358519, u'yellow': 8.957594070008593...}.

Теперь, что касается оставшихся столбцов-признаков: LocationNormalized и ContractTime являются строковыми, и поэтому с ними нельзя работать напрямую. Применим к ним one-hot-кодирование. Для рассматриваемого нами языка Python оно реализовано в классе sklearn.feature_extraction.DictVectorizer, который преобразует списки сопоставленных пар признак-значение в массивы, с которыми может работать Numpy, или в разреженные матрицы (scipy.sparse-matrices) для использования с классами-«измерителями качества» scikit-learn (scikit-learnestimators). Когда значения признаков строковые, этот трансформер выполняет бинарное one-hot-кодирование: для каждого из возможных строковых значений признака строится признак с логическим значением, говорящий, принимает данный признак указанное строковое значение или нет. Например, пусть на входе есть признак «c», который может принимать значения «кот» и «дог». На выходе будет два признака: один – «c = кот», другой – «c = дог». Признаки, которые не представлены в обучающей выборке, будут иметь нулевое значение в результирующей матрице или результирующем массиве.

Подробнее о DictVectorizer можно прочитать здесь: [43]. Пример использования этого класса для указанных данных приведен в листинге 26.

Листинг 26

```
# -*- coding: utf-8 -*-
from sklearn.feature_extraction import DictVectorizer
.....
enc = DictVectorizer()
train_dic = data_train[['LocationNormalized', 'ContractTime']].to_dict('records')
test_dic = data_test[['LocationNormalized', 'ContractTime']].to_dict('records')
X_train_categ = enc.fit_transform(train_dic)
X_test_categ = enc.transform(test_dic)
```

При этом считается, что в data_train загружены данные обучающей выборки, а в data_test – данные, на которых будет проверяться работа модели.

В результате работы кода из листинга 26 train_dic (также как и test_dic) будет содержать множество записей, таких как: {['LocationNormalized': 'UK', 'ContractTime': 'permanent'], ['LocationNormalized': 'Bradford', 'ContractTime': 'permanent']}. X_train_categ будет содержать данные, представленные на рисунке 58.

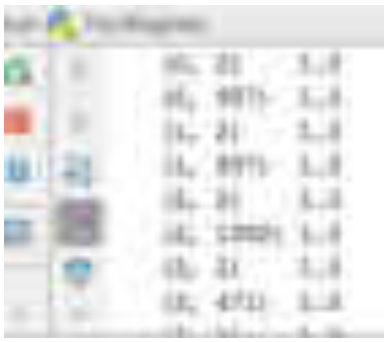


Рисунок 58. Преобразованные данные

Интерпретировать их можно следующим образом. X_train_categ – это по сути матрица, но только не в классическом формате. Если точнее, то X_train_categ – это разреженная матрица, которая в сокращенном виде описывает некую полную матрицу (назовем ее X_Full).

В X_train_categ указано, в каких позициях X_Full стоят единицы. Так (0, 2) 1.0 – означает, что в нулевой строке и втором столбце стоит единица. Во всех остальных ячейках матрицы X_Full, которые не обозначены в X_train_categ, стоят нули. X_Full – это матрица размера M на N, где M – число строк, которое определяется количеством объектов в выборке, а N – число уникальных слов в столбцах ['LocationNormalized', 'ContractTime'] исходной таблицы.

Теперь вернемся к предобработке данных. Так как в строковых данных есть пропуски, то нам потребуется заменять их на специальные строковые величины (например, 'nan'). Для этого можно использовать код из листинга 27.

Листинг 27

```
data_train['LocationNormalized'].fillna('nan', inplace=True)
```

С учетом изложенного выше, алгоритм решения поставленной задачи будет следующим:

1. Загрузить данные об описаниях вакансий и соответствующих годовых зарплатах из файла salary-train.csv.
2. Провести его предобработку:
   2.1. Привести тексты к нижнему регистру (text.lower()).
   2.2. Заменить все, кроме букв и цифр, на пробелы для облегчения дальнейшего разделение текста на слова. Для такой замены в строке text подходит следующий вызов: re.sub('[^a-zA-Z0-9]', ' ', text). Также можно воспользоваться методом replace у DataFrame, чтобы сразу преобразовать все тексты, например, так: train['FullDescription'] = train['FullDescription'].replace('[^a-zA-Z0-9]', ' ', regex = True)
   2.3. Применить TfidfVectorizer для преобразования текстов в векторы признаков. Оставить только те слова, которые встречаются хотя бы в 5 объектах. Для этого использовать параметр min_df у TfidfVectorizer.

2.4. Заменить пропуски в столбцах LocationNormalized и ContractTime на специальную строку 'nan' (листинг 27).

2.5. Применить DictVectorizer для получения one-hot-кодирования признаков LocationNormalized и ContractTime.

2.6. Объединить все полученные признаки в одну матрицу «объекты-признаки».

Обратите внимание!

- *Матрицы для текстов и категориальных признаков являются разреженными. Для объединения их столбцов нужно воспользоваться функцией scipy.sparse.hstack.*

2.7. Обучить гребневую регрессию с параметрами alpha=1 и random_state=241, помня, что целевая переменная записана в столбце SalaryNormalized. Обучение производится вызовом метода fit.

3. Построить прогноз для двух примеров из файла salary-test-mini.csv. Прогноз строится методом predict.

Код решения задачи (с комментариями) – в листинге 28. Результат работы – два прогнозных числа для двух записей из тестовой выборки: [56876.4573163 37557.00551031].

Листинг 28

```
# -*- coding: utf-8 -*-
import pandas
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Ridge
from scipy.sparse import hstack

#Загрузка данных
data_train = pandas.read_csv('salary-train.csv')
data_test = pandas.read_csv('salary-test-mini.csv')

# Приведение текста к нижнему регистру
data_train['FullDescription'] = data_train.FullDescription.str.lower()
data_test['FullDescription'] = data_test.FullDescription.str.lower()
```

Окончание листинга 28

```
# Удаление ненужных символов
data_train['FullDescription'] = data_train['FullDescription'].replace('[^a-zA-Z0-9]', ' ',
regex=True)
data_test['FullDescription'] = data_test['FullDescription'].replace('[^a-zA-Z0-9]', ' ',
regex=True)
# Преобразование текста в вектор признаков, используя TdifVectorizer из sklearn
vectorizer = TfidfVectorizer(min_df=10)
train_text_feature_matrix = vectorizer.fit_transform(data_train['FullDescription'])
test_text_feature_matrix = vectorizer.transform(data_test['FullDescription'])
# Заполнение пустых ячеек
data_train['LocationNormalized'].fillna('nan', inplace=True)
data_train['ContractTime'].fillna('nan', inplace=True)
data_test['LocationNormalized'].fillna('nan', inplace=True)
data_test['ContractTime'].fillna('nan', inplace=True)
# Преобразование категориальных признаков в числовые
enc = DictVectorizer()
train_dic = data_train[['LocationNormalized', 'ContractTime']].to_dict('records')
test_dic = data_test[['LocationNormalized', 'ContractTime']].to_dict('records')
X_train_categ = enc.fit_transform(train_dic)
X_test_categ = enc.transform(test_dic)
# Здесь по горизонтали объединяется разреженная матрица с признакими текста (из
#столбца FullDescription) и матрица с закодированными категориями (из столбцов
#Location Normalized и Contract Time)
x_train = hstack((train_text_feature_matrix, X_train_categ))
y_train = data_train['SalaryNormalized'].values
x_test = hstack((test_text_feature_matrix, X_test_categ))
#Создание и обучение регрессии
ridge_regression = Ridge(alpha=1, random_state=241)
ridge_regression.fit(x_train, y_train)
#Получение и распечатка прогнозных значений
y_test = ridge_regression.predict(x_test)
print y_test
```

Сохранение и загрузка обученной модели

Как вы могли заметить, работа программы идет довольно долго. Большую часть этого времени занимает процесс обучения, поэтому при работе с обученной моделью используют практику сохранения и загрузки ее параметров в\из файл(а). Рассмотрим следующий

пример. Пусть у нас есть некоторый массив данных о двух факторах и зависимой переменной, сохраненные в файле «data-logistic.csv» (скачать можно отсюда: [41]). Выполним следующее: обучим на этих данных модель линейной регрессии, сохраним ее в файл, создадим еще одну новую модель линейной регрессии, обучим ее на части данных, затем загрузим ранее обученную модель из файла и сравним результаты предсказания обеих моделей. Сохранение и загрузку мы будем выполнять с помощью модуля pickle, и для удобства вынесем код загрузки и сохранения в отдельные функции LoadFromObject и SaveToObject. Решение задачи показано в листинге 29, а ее результат – на рисунке 59.

Листинг 29

```
# -*- coding: utf-8 -*-

import pandas
import pandas.core.series as Series
import numpy as np
from scipy.sparse import hstack
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
import sys
import matplotlib.pyplotasplt
from sklearn import linear_model
import pickle

#Определяем функцию сохранения
def SaveToObject(obj, filename):
  with open(filename, 'wb') as output:
  pickle.dump(obj, output)

#Опредлеляем функцию загрузки
def LoadFromObject(filename):
   f = open(filename, 'rb')
   loaded_obj = pickle.load(f)
   f.close()
   return loaded_obj
```

Окончание листинга 29

```
# Загружаем данные
data_train = pandas.read_csv('data-logistic.csv')
X = data_train.ix[:,1:3].values
y = (data_train.ix[:,0].values + 1)/2
#Для лучешго понимания представления данных распечатайте формы массивов:
print X.shape
print y.shape
# Создание, обучение модели и получение ее прогноза
regression = linear_model.LogisticRegression()
regression.fit(X, y)
ans = regression.predict(X)
# Оценка качества созданной модели
metric = roc_auc_score(y, ans)
print "AUC Trained: ",  metric.real
#Сохранение и загрузка модели в файл с именем model
SaveToObject(regression, "model")
loaded_model = LoadFromObject("model")
#Создание, обучение новой модели, получение ее прогноза и оценка качества
not_fitted = linear_model.LogisticRegression()
not_fitted.fit(X[0:2,:], y[0:2])
ans = not_fitted.predict(X)
metric = roc_auc_score(y, ans)
print "AUC not Trained: ", metric.real
#получение прогноза и оценки качества загруженной модели
ans = loaded_model.predict(X)
metric = roc_auc_score(y, ans)
print "AUCLoaded: ", metric.real
```

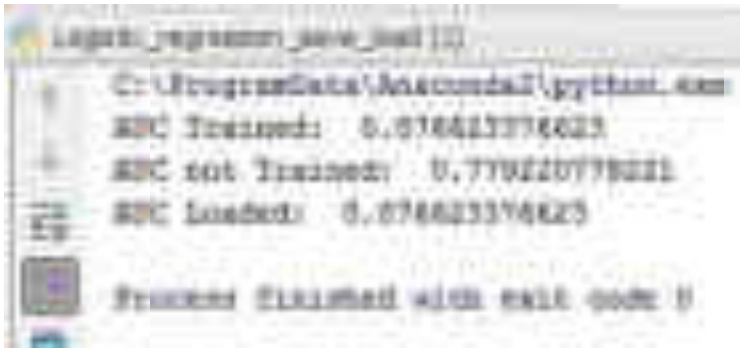


Рисунок 59. Результат работы кода из листинга 29

Как видно из рисунка 59, модель, обученная на части данных, дала более плохой результат, тогда как качество загруженной модели не изменилось. То есть можно говорить о том, что сохранение и загрузка прошли корректно.

Работа с логистической регрессией

Мы рассмотрели применение регрессии для решения задачи предсказания, теперь перейдем к задаче классификации и познакомимся с еще одной моделью – логистической регрессией. Это один из видов линейных классификаторов. Ее особенность заключается в том, что результатом является вероятность принадлежности объекта к классу N, тогда как большинство линейных классификаторов могут выдавать только номера классов.

Логистическая регрессия использует достаточно сложную функцию для оценки качества, которая не допускает записи решения в явном виде (в отличие от, например, линейной регрессии). Тем не менее логистическую регрессию можно настраивать с помощью градиентного спуска.

Для примера мы выполним следующее: сгенерируем некоторую случайную выборку и каждому ее элементу поставим в соответствие число 0 или 1, которое будет обозначать принадлежность к первому или второму классу. Затем обучим логистическую модель на этих данных и линейную модель. После – построим график и сравним результаты. Для генерации случайной выборки из 100 элементов используем следующий код из листинга 30.

Листинг 30

```
import numpy as np
…..
n_samples = 100
np.random.seed(0)
X = np.random.normal(size=n_samples)
```

Обратите внимание!

- *В дальнейших листингах операторы импорта библиотек будут появляться только при первом вызове импортируемых элементов, так как предполагается, что все представленные ниже листинги формируют единый код, который вам предлагается собрать в единый файл самостоятельно.*

Для формирования множества откликов Y определим функцию, реализующую правило: Y=1 для всех X>0 и Y=0 – в остальных случаях. Код функции можно увидеть в листинге 31.

Листинг 31

```
Y = (X > 0).astype(np.float)
```

Следующим шагом нам необходимо выполнить преобразования нашего одномерного массива X: во-первых, добавить небольшой разброс и смещение, чтобы избавиться от прямой функциональной зависимости. Во-вторых, нужно преобразовать его в двумерный массив. Последнее выполняется для функции обучения регрессии: на вход она требует данные в формате двумерного массива. Преобразование выполняется кодом, представленным листинге 32.

Листинг 32

```
X[X > 0] *= 4
X += .3 * np.random.normal(size=n_samples)
X = X[:, np.newaxis]
```

Теперь мы можем объявить и обучить модели регрессии, как показано в листинге 33.

Листинг 33

```
from sklearn import linear_model
log_reg = linear_model.LogisticRegression()
log_reg.fit(X, Y)
lin_reg = linear_model.LinearRegression()
lin_reg.fit(X, Y)
```

Подробнее о реализации логистической модели можно прочитать здесь: [44], а линейной – здесь: [45].

Следующим шагом нам необходимо сгенерировать тестовые данные и выполнить предсказание. Последнее будет выполняться методом predict. Он возвращает бинарные метки классов (0,1). Кроме этого существует еще метод predict_proba (для логистической регрессии). Он возвращает вероятность принадлежности объекта к классу 0 или 1 соответственно. Поскольку у нас два класса, то в результате predict_proba получается двумерный массив (один столбец отражает принадлежность к классу 0, а другой к классу 1). Для отображения мы выберем лишь первый столбец.

Обратите внимание!

- *Сумма элементов массивов (столбцов, векторов) с одним индексом равна единице по правилу полной вероятности.*

Код генерации тестовых данных и вызовы методов прогноза представлен в листинге 34.

Листинг 34

```
X_test = np.linspace(start=-5, stop=10, num=300)
X_test = X_test[:, np.newaxis]
y_log_label = log_reg.predict(X_test)
y_lin = lin_reg.predict(X_test)
y_log_probabilty = log_reg.predict_proba(X_test)[:,1]
```

Следующим шагом нам необходимо вывести график для отображения результатов. Создавать полотно для рисования графиков и отображать их мы будем с помощью библиотеки Matplotlib, как показано в листинге 35.

Листинг 35

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
....
```

Окончание листинга 35

```
# рисуем исходные точки
plt.figure(1, figsize=(4, 3))
plt.scatter(X.ravel(), y, color='black', zorder=20)
# рисуем график меток логистической регрессии
plt.plot(X_test, y_log_label, color='red', linewidth=3)
# рисуем график линейной регрессии
plt.plot(X_test, y_lin, linewidth=1)
# добавляем линию уровня, по которому точки будут относится к 1 или 2 классу
plt.axhline(.5, color='green')
# рисуем логистическую кривую для вероятностей принадежности объектов к
#классам
plt.plot(X_test, y_log_probabilty, color='blue', linewidth=3)
# настраиваем оси и выводим график
plt.ylabel('y')
plt.xlabel('X')
plt.xticks(range(-5, 10))
plt.yticks([0, 0.5, 1])
plt.ylim(-.25, 1.25)
plt.xlim(-4, 10)
plt.legend(('LabelLogisticRegressionModel', 'LinearRegressionModel',
'ProbabiltyLogRegression'), loc="lowerright", fontsize='small')
plt.show()
```

В результате после выполнения кода появится график, как показано на рисунке 60.

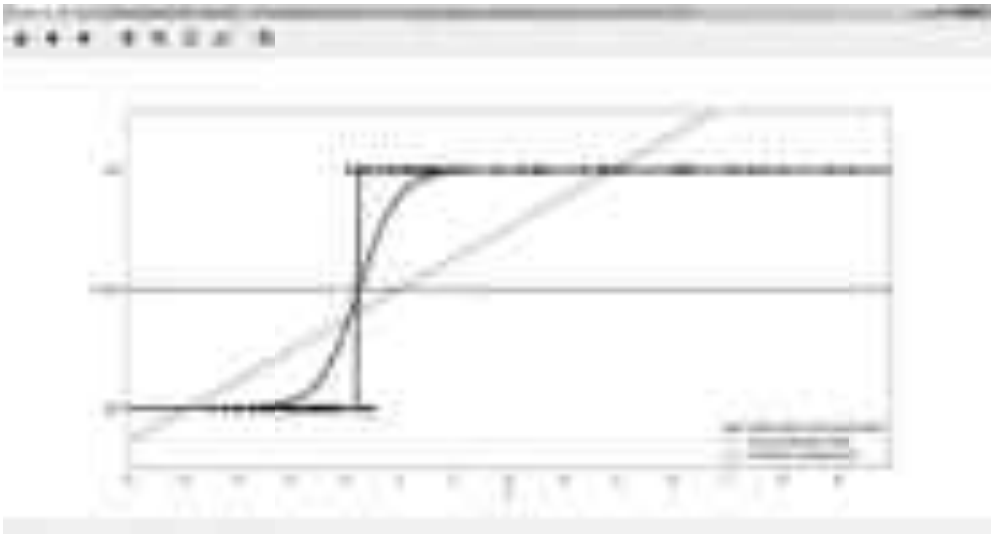


Рисунок 60. Графики регрессий

Для лучшего понимания вам рекомендуется воспроизвести код и посмотреть на график, который вам выведет программа. На нем должно быть видно, что красная кривая (результат логистического предсказания\классификации) лучше приближает распределение точек (лишь несколько точек вылетает). Красная линия построена лишь по меткам, полученным от log_reg.predict, поэтому она имеет такой резкий переход. Такой «округленный» результат легко использовать в дальнейшем, хотя в таком случае мы не получаем информацию о вероятностях. Линейная регрессия (голубая линия) приближает данные не так хорошо.

Важно понимать, что линейная регрессия отнесет все точки (x<1) к классу 0, так как результат функции линейной регрессии будет меньше 0.5. А все точки (X>1) будут отнесены к классу 1, так как результат функции линейной регрессии будет больше 0.5. Для этого на графике выведена разделяющая линия Y=0.5. Граница разделения классов по логистической регрессии проходит примерно в точке X=0.25, что разделяет классы гораздо точнее. Толстая синяя линия отображает вероятностную логистическую кривую, которая дает больше детальной информации о предсказании. Поскольку результат представляет собой не бинарные метки классов, а вероятность принадлежности к выбранном классу, то кривая получается гладкой.

Для точной оценки качества моделей воспользуемся функциями score и accuracy_score в scikit-learn. Поскольку линейная регрессия выдает вещественные значения, а y_test сгенерирована как бинарный массив, то для оценки качества нужно использовать функцию model.score(), которая способна читать и бинарные, и вещественные ответы. Для подсчета качества логистической регрессии можно воспользоваться измерителем качества классификации accuracy_score. Но перед этим нам необходимо получить реальную принадлежность данных тестовой выборки к классам. Сделать это можно с помощью определенной нами функции для обучающей выборки, как показано в листинге 36.

Листинг 36

```
y_test = (X_test > 0).astype(np.float)
```

Теперь можно вывести значения качества классификации для обеих моделей, как показано в листинге 37. Для линейной регрессии это выполняется ее методом score, а для логистической – методом accuracy_score из sklearn.metrics.

Листинг 37

```
from sklearn.metrics import accuracy_score
....
# оцениваем точность решений
lin_score = lin_reg.score(X_test, y_test)
print "Linear regression accuracy: ", lin_score
log_score = accuracy_score(y_log_label, y_test)
print "Logistic regression accuracy: ", log_score
```

В итоге в консоль будут выведены следующие результаты: Linearregression accuracy=0,52442, Logisticregression accuracy=0,98333. Они подтверждают, что логистическая регрессия выполнила классификацию более качественно.

Решение задачи ранжирования признаков

Следующее, что нам необходимо рассмотреть, – применение регрессионных моделей для определения важности признаков. Здесь в качестве примера возьмем регрессионную проблему Фридмана, когда на вход моделей подается 14 факторов, выход рассчитывается по формуле, использующей только пять факторов, но факторы 1-5, а также 10-14 взаимозависимы. Наша задача – посмотреть, как разные виды регрессий оценят важности факторов и какой из них будет иметь наибольшую среднюю значимость по всем моделям. Для работы мы возьмем три модели: линейную регрессию, гребневую и лассо (линейную регрессию с L1-регуляризатором). О реализации в scikit-learn последней модели можно подробнее прочитать здесь: [46].

Первое, что нам необходимо сделать для решения поставленной задачи, – подключить необходимые модули, как в листинге 38. Вторая строка кода из листинга 38 – импорт класса, выполняющего масштабирование данных до заданного диапазона (например, так, чтобы все значение находились в диапазоне от 0 до 1). Подробнее о реализации класса можно прочитать здесь: [47]. Необходимость его использования объясняется следующим: каждая модель регрессии дает оценки важности признаков в своем диапазоне. Для того чтобы найти признак с максимальной средней важностью по трем моделям, нам необходимо привести выданные ими оценки к одному виду, в чем и поможет нам указанный класс.

Листинг 38

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import MinMaxScaler
import numpy as np
```

После импорта необходимых классов мы должны сгенерировать исходные данные, как показано в листинге 39.

Листинг 39

```
#генерируем исходные данные: 750 строк-наблюдений и 14 столбцов-признаков
np.random.seed(0)
size = 750
X = np.random.uniform(0, 1, (size, 14))
#Задаем функцию-выход: регрессионную проблему Фридмана
Y = (10 * np.sin(np.pi*X[:,0]*X[:,1]) + 20*(X[:,2] - .5)**2 +
    10*X[:,3] + 5*X[:,4]**5 + np.random.normal(0,1))
#Добавляем зависимость признаков
X[:,10:] = X[:,:4] + np.random.normal(0, .025, (size,4))
```

Теперь создаем и обучаем модели (листинг 40).

Листинг 40

```
#линейная модель
lr = LinearRegression()
```

Окончание листинга 40

```
lr.fit(X, Y)
#гребневая модель
ridge = Ridge(alpha=7)
ridge.fit(X, Y)
#Лассо
lasso = Lasso(alpha=.05)
lasso.fit(X, Y)
```

Наконец, нам нужно выгрузить в единый массив размера 3×14 (количество_моделей и количество_признаков) все оценки моделей по признакам. Найти средние оценки и вывести результат в формате списка пар {номер_признака – средняя_оценка}, отсортированном по убыванию. Получить оценки признаков можно через поле coef_ у каждой из наших моделей. Для удобства отображения данных поместим их в конструкцию вида: [имя_модели : [{имя_признака : оценка},{имя_признака : оценка}]]. То есть верхним уровнем у нас будет словарь, где ключом будет имя модели. В нем будут располагаться три записи из четырнадцати пар каждая. Пример содержимого списка представлен в листинге 41.

Листинг 41

```
{'Lasso': {'x13': 0.0, 'x14': 0.0, 'x10': 0.0, 'x8': 0.0, 'x9': 0.0, 'x11': 0.0, 'x2': 0.72, 'x3': 0.0,
'x12': 0.0, 'x1': 0.69, 'x6': 0.0, 'x7': 0.0, 'x4': 1.0, 'x5': 0.29},
'Ridge': {'x13': 0.0, 'x14': 0.92, 'x10': 0.01, 'x8': 0.08, 'x9': 0.0, 'x11': 0.59, 'x2': 0.75, 'x3': 0.06,
'x12': 0.67, 'x1': 0.76, 'x6': 0.08, 'x7': 0.03, 'x4': 1.0, 'x5': 0.61},
'Linear reg': {'x13': 0.48, 'x14': 0.12, 'x10': 0.01, 'x8': 0.03, 'x9': 0.0, 'x11': 0.59, 'x2': 0.61, 'x3':
0.51, 'x12': 0.19, 'x1': 1.0, 'x6': 0.02, 'x7': 0.0, 'x4': 0.69, 'x5': 0.19}}
```

Первое, что нам необходимо сделать, чтобы реализовать описанное, – создать список вида ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x12', 'x13', 'x14'], содержащий имена признаков. Выполнить это можно с помощью кода из листинга 42.

Листинг 42

```
names = ["x%s" % i for i in range(1,15)]
```

Затем нам необходимо объявить функцию, которая на вход будет получать сформированный выше список и список оценок по признакам. Выходом же функции должен быть словарь, составленный из попарно соотнесенных элементов списков (то есть словарь из пар имя_признака: оценка_признака, где первое – ключ, а второе – значение). Причем оценки внутри функции должны быть приведены к единому диапазону от 0 до 1 и округлены до сотых.

В теле функции нам потребуются следующие объекты. Во-первых, класс MinMaxScaler, о котором было сказано выше. Во-вторых, функция abs пакета Numpy для получения абсолютных значений оценок(модуля). В-третьих, функция map, позволяющая применить операцию округления (round) к каждому элементу массива посредствам оператора lambda. В-четвертых, нам понадобятся функции rashape и ravel. Первая – для переформирования входного массива признаков из одной строки и четырнадцати столбцов в четырнадцать строк и один столбец, вторая – для преобразования полученного двумерного массива в одномерный. В-пятых, нам потребуются функции zip и dict. Первая попарно соотнесет имена признаков и оценки, вторая – преобразует полученные пары в словарь. Примеры использования функций zip, map и lambda можно найти здесь: [48]. Код реализации описанных действий можно найти в листинге 43.

Листинг 43

```
def rank_to_dict(ranks, names):
    ranks = np.abs(ranks)

    minmax = MinMaxScaler()

    ranks = minmax.fit_transform(np.array(ranks).reshape(14,1)).ravel()
    ranks = map(lambda x: round(x, 2), ranks)
    return dict(zip(names, ranks))
```

Вызов функции для наших моделей представлен в листинге 44.

Листинг 44

```
ranks["Linear reg"] = rank_to_dict(lr.coef_, names)
ranks["Ridge"] = rank_to_dict(ridge.coef_, names)
ranks["Lasso"] = rank_to_dict(lasso.coef_, names)
```

Последним шагом мы должны сформировать среднее по каждому признаку, загрузив их в отдельный список. Затем отсортировать его по убыванию и вывести. Реализовать это можно, используя код из листинга 45.

Листинг 45

```
#Создаем пустой список для данных
mean = {}

#«Бежим» по списку ranks
for key, value in ranks.iteritems():
#«Пробегаемся» по списку значений ranks, которые являются парой имя:оценка
for item in value.iteritems():
#имя будет ключом для нашего mean
#если элемента с текущим ключем в mean нет - добавляем
if(item[0] not in mean):
mean[item[0]] = 0

#суммируем значения по каждому ключу-имени признака
      mean[item[0]] += item[1]

#находим среднее по каждому признаку
for key, value in mean.iteritems():
   res=value/len(ranks)
   mean[key] = round(res, 2)

#сортируем и распечатываем список
mean = sorted(mean.iteritems(), key=lambda (x, y): y, reverse=True)

print "MEAN"
print mean
```

Результат работы программы представлен в листинге 46.

Листинг 46

```
MEAN
[('x4', 0.9), ('x1', 0.82), ('x2', 0.69), ('x11', 0.39), ('x5', 0.36), ('x14', 0.35), ('x12', 0.29), ('x3', 0.19), ('x13', 0.16), ('x8', 0.04), ('x6', 0.03), ('x10', 0.01), ('x7', 0.01), ('x9', 0.0)]
```

Мы видим, что по средней оценке наиболее важным оказался четвертый признак, затем первый и потом второй. Сравним эти данные с показателями каждой отдельной модели. Для этого отсортируем массив ranks также по убыванию оценок и выведем его, используя код из листинга 47. Результат работы кода представлен в листинге 48.

Листинг 47

```
for key, value in ranks.iteritems():
    ranks[key] = sorted(value.iteritems(), key=lambda (x, y): y, reverse=True)
for key, value in ranks.iteritems():
    print key
    print value
```

Листинг 48

```
Lasso
[('x4', 1.0), ('x2', 0.72), ('x1', 0.69), ('x5', 0.29), ('x13', 0.0), ('x14', 0.0), ('x10', 0.0), ('x8', 0.0), ('x9', 0.0), ('x11', 0.0), ('x3', 0.0), ('x12', 0.0), ('x6', 0.0), ('x7', 0.0)]
Ridge
[('x4', 1.0), ('x14', 0.92), ('x1', 0.76), ('x2', 0.75), ('x12', 0.67), ('x5', 0.61), ('x11', 0.59), ('x8', 0.08), ('x6', 0.08), ('x3', 0.06), ('x7', 0.03), ('x10', 0.01), ('x13', 0.0), ('x9', 0.0)]
Linear reg
[('x1', 1.0), ('x4', 0.69), ('x2', 0.61), ('x11', 0.59), ('x3', 0.51), ('x13', 0.48), ('x12', 0.19), ('x5', 0.19), ('x14', 0.12), ('x8', 0.03), ('x6', 0.02), ('x10', 0.01), ('x9', 0.0), ('x7', 0.0)]
```

Как можно увидеть, метод Lasso отобрал признаки x1, x2, x4, x5 как значимые параметры, а все остальные совсем незначимые (так как они равны нулю). Что довольно близко к истине, так как параметры x1-x5 заданы как независимые случайные величины. Но взвешивание по такому методу довольно грубое, резкое.

Метод Ridge сделал более мягкое взвешивание. Но несмотря на то, что x3 также упущен, его влияние может быть учтено через скорре-

лированные параметры x14, x12, x11 (которые были также отобраны как важные). Поэтому данный метод может быть очень полезным.

Метод линейной регрессии также отдал предпочтение многим действительно важным параметрам (причем x3 здесь имеет гораздо больший вес).

В заключение можно сказать, что учет оценок от разных методов позволит более оптимально выделить значимые признаки и отделить от менее значимых. Но, конечно, без реальных экспериментов, проверяющих выбранное множество параметров, невозможно гарантировать идеального разбиения для всех задач.

Кроме рассмотренных выше моделей для отбора признаков еще можно использовать:

1. RandomizedLasso из sklearn.linear_model (оценки признаков будут находиться в поле scores_). Это так называемое рандомизированное лассо работает, разделяя тренировочные данные на подвыборки и вычисляя Лассо-оценки признаков, где штраф случайного подмножества коэффициентов масштабирован. Применяя такую двойную рандомизацию несколько раз, метод присваивает наивысшие оценки признакам, выбор которых повторялся в процессе рандомизации. Это называется «выбором стабильности». Если же сказать короче, то признаки, которые выбираются наиболее часто, считаются значимыми. Подробнее об этом классе можно почитать здесь: [49].

2. RFE из sklearn.feature_selection (оценки признаков будут находиться в поле ranking_, однако в отличие от всех остальных моделей, класс выдает не веса при коэффициентах регрессии, а именно ранг для каждого признака. Так наиболее важные признаки будут иметь ранг – «1», а менее важные признаки ранг больше «1». Коэффициенты остальных моделей тем важнее, чем больше их абсолютное значение). Класс RFE выполняет ранжирование признаков через рекурсивный отбор путем отсева признаков. Целью рекурсивного устранения элемента (RFE) является выбор признаков путем рекурсивного рассмотрения меньших и меньших наборов признаков, учитывая внешнюю оценку, которая присваивает признакам веса

(например, коэффициенты линейной модели). Сначала оценщик обучается на начальном наборе признаков и ассоциированных с ними весами. Затем признаки, абсолютные веса которых являются наименьшими, удаляются из текущего набора. Эта процедура рекурсивно повторяется на оставшемся множестве до тех пор, пока в конечном итоге не будет достигнуто желаемое количество признаков. То есть работа с этим классом строится, как показано в листинге 49. Подробнее о нем можно почитать здесь: [50].

Листинг 49

```
lr = LinearRegression()
lr.fit(X, Y)

rfe = RFE(lr)
rfe.fit(X,Y)
```

3. RandomForestRegressor из sklearn.ensemble (оценки признаков будут находиться в поле feature_importances_). Регрессор на основе Случайного леса – это мета-оценщик, который обучает несколько классификационных деревьев решений на разных подвыборках данных и использует усреднение для улучшения точности и контроля переобучения. Размер подвыборки всегда соответствует размеру входной выборки, но образцы из выборки перемешиваются (если параметр bootstrap=True, что является значением по умолчанию). Подробнее об этом классе можно посмотреть здесь: [51].

4. f_regression из sklearn.feature_selection (для получения оценок см.код из листинга 50. Оценки будут находиться в объекте f). Это быстрая линейная модель, позволяющая оценить эффект одного признака и применяющаяся последовательно к множеству признаков. В основе ее работы лежит вычисление взаимной корреляции между каждым признаком и выходной переменной. Более подробно можно посмотреть здесь: [52].

Листинг 50

```
f, pval  = f_regression(X, Y, center=True)
```

Работа с полиномиальной регрессией

Напоследок, заканчивая разговор о регрессионных моделях, мы познакомимся с полиномиальной регрессией, а также посмотрим на практике такое явление, как переобучение модели. Рассматривать мы это будем на следующем примере: сгенерируем точки функцией косинуса и попробуем аппроксимировать их линейной регрессией, полиномиальной регрессией со степенью 4 и 15, а также гребневой регрессией и созданными на основе нее полиномами 4 и 15 степеней. Затем рассчитаем точность моделей, выведем шесть графиков и сравним результаты.

Итак, первым шагом создадим обычную функция косинуса от аргумента, записанную в виде lambda-выражения (листинг 51). Особенности Python позволяют использовать в качестве аргумента не только числа, но и массивы\векторы (так что использование циклов не требуется).

Листинг 51

```
import numpy as np
true_fun = lambda X: np.cos(1.5 * np.pi * X)
```

Представленная в листинге 51 функция будет использоваться как истинная закономерность в некоторых данных. То есть именно то, что аналитики и ученые пытаются выяснить, проводя измерения некоторого объекта природы или бизнеса. Собственно, поэтому название функции – true_fun. В дальнейшем эту функцию можно будет вызывать подобно функции, которая определена через def, как показано в листинге 52.

Листинг 52

```
result= true_fun(something_data)
```

Далее, для корректного проведения эксперимента необходимо создать некую выборку, которая будет служить аналогом измерений истинной закономерности. Для этого мы создадим распределение\выборку

на основе объявленной ранее функции с небольшим случайным смещением по X и Y (листинг 53).

Листинг 53

```
np.random.seed(0)
n_samples = 30
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1
```

Полиномиальная регрессия задается в несколько шагов. Сперва необходимо определить коэффициенты полинома (которые автоматически генерируются с помощью функции PolynomialFeatures) и модель для основы (в нашем случае линейную или гребневую). Затем необходимо соединить вместе базовую регрессию и полиномиальные коэффициенты (вектор или матрицу преобразования) как последовательность трансформаций некоторых данных. Итоговый объект и будет являться полиномиальной моделью. Трансформацию данных можно выполнить с помощью объекта Pipeline из sklearn.pipeline (подробнее о нем можно почитать здесь: [53]).

Для удобства работы и сокращения кода сделаем следующее: объявим массив с нужными нам степенями полинома, как показано в листинге 54.

Листинг 54

```
degrees = [1, 4, 15]
```

Затем объявим функцию, которая будет принимать в качестве параметров номер степени (индекс элемента в массиве dergees), полотно, на котором нужно отобразить график, базовую модель и уровень графика на полотне (первый или второй ряд по вертикали). Внутри тела функции будет создаваться и обучаться нужная полиномиальная модель, результат работы которой будет выведен на возвращаемом функцией графике. Кроме точек на графике будет отображена точность модели, вычисленная методом кросс-валидации:

средние оценки качества по метрике «neg_mean_squared_error» (среднеквадратичная ошибка). Эти оценки будут получены с помощью объекта cross_val_score из sklearn.model_selection. Подробнее о кросс-валидации в scikit можно почитать здесь: [54, 55].

Объявление функции показано в листинге 56, с работой которого вам предлагается разобраться самостоятельно. Единственно, поясним назначение параметра cv в cross_val_score. Параметр cv определяет, на сколько групп будет разбита тренировочная выборка. Каждая группа будет содержать тренировочную часть А и тестовую Б. Причем эти множества (А и Б) не будут пересекаться. Так для следующих данных: входа X = np.array([[11, 2], [3, 14], [1, 2], [3, 4]]), выхода y = np.array([1, 2, 3, 4]) cross_val_score с параметром cv=2 даст следующее разбиение на 2 группы\итерации (листинг 55).

Листинг 55

```
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
```

Массив TRAIN содержит это индексы тех объектов, которые входят в тренировочную выборку. То есть TRAIN: [2 3] означает, что в тренировочную выборку войдут второй и третий объекты из X: [1,2], [3, 4]. Все остальные – в тестовую.

Листинг 56

```
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import cross_val_score
def MakeExample(index, plt, model,k):
   polynomial_features = PolynomialFeatures(degree=degrees[index],
                                            include_bias=False)
```

Окончание листинга 56

```
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", model)])
    pipeline.fit(X[:, np.newaxis], y)
    scores = cross_val_score(pipeline, X[:, np.newaxis], y,
                             scoring="neg_mean_squared_error", cv=10)
    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    if k == 1:
      plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    if k==0:
     plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(degrees[i], -scores.mean(),
scores.std()))
    else:
     plt.title("MSE = {:.2e}(+/- {:.2e})".format(-scores.mean(), scores.std()))
    return plt
```

Код вызова функции из листинга 56 (и, соответственно отрисовка графиков) представлен в листинге 57, а результат работы кода – на рисунке 61.

Листинг 57

```
plt.figure(figsize=(14, 6))
for i in range(len(degrees)):
    linear_regression = LinearRegression()
    plt.subplot(2, len(degrees), i+1)
    plt = MakeExample(i, plt, linear_regression,0)
for i in range(len(degrees)):
    ridge = Ridge(alpha=0.02)
    plt.subplot(2, len(degrees), len(degrees) + i + 1)
    plt = MakeExample(i, plt, ridge,1)
plt.show()
```

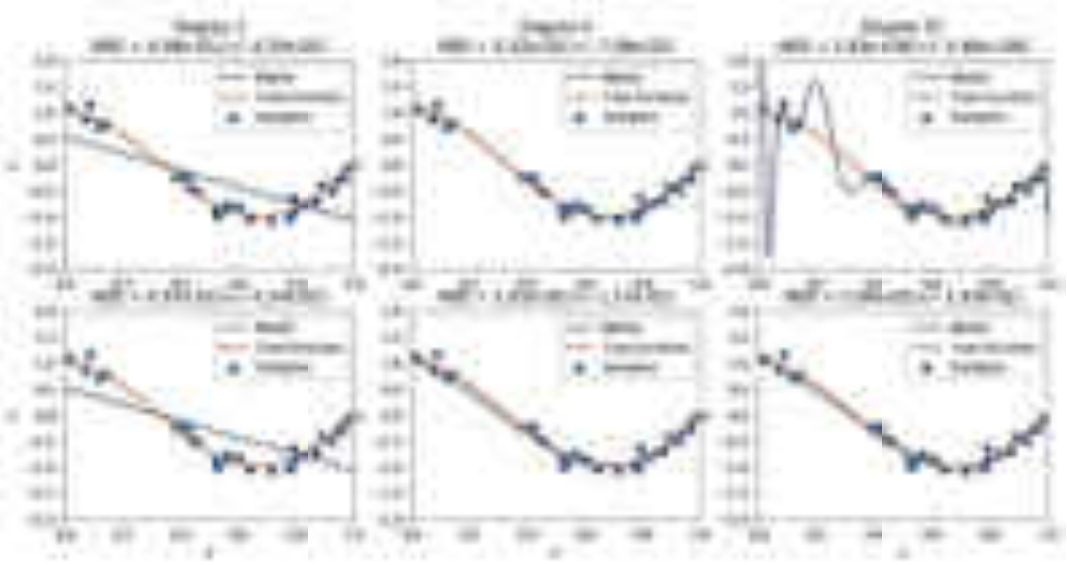


Рисунок 61. Результаты работы с разными моделями

Для лучшего понимания вам рекомендуется воспроизвести код и посмотреть на рисунок, выданный вашей программой. Если говорить о моделях, построенных на основе линейной регрессии, то, сравнивая первый и второй графики (полином 1 и 4 степени), можно сделать вывод, что повышение степени положительно влияет на качество модели. Но третий график (полином 15 степени) демонстрирует большую ошибку. Как мы видим, на тренировочной выборке модель показала хороший результат (кривая проходит через все точки выборки). Но на тестовой выборке модель «ведет себя плохо» (плохо приближает true_fun). Это и означает, что при повышении сложности модели есть риск перейти за критическую точку и переобучить модель. Однако, как мы видим на графиках гребневой регрессии, регуляризация позволяет улучшить качество решения.

Работа с простейшими моделями нейронных сетей

Следующее, с чем нам необходимо познакомиться, – работа с простейшими моделям нейронных сетей, которые содержатся в библиотеке Sklearn (в дальнейшем будет рассмотрена работа с другими библиотеками, которые содержат более сложные алго-ритмы и позволяют создавать другие нейроклассификаторы и модели). Однако Sklearn не специализируется на нейросетевых моделях и не содержит более сложные алгоритмы.

Начнем мы с обычного персептрона, являющегося одним из вариантов линейных моделей. Это распространенный класс моделей, которые отличаются своей простотой и скоростью работы. Их можно обучать за разумное время на очень больших объемах данных, и при этом они могут работать с любыми типами признаков: вещественными, категориальными, разреженными. Рассмотрим решение следующей задачи: пусть у нас есть данные вида, показанного на рисунке 62, и сохраненные в файлах perceptron-train.csv и perceptron-test.csv (скачать файлы можно отсюда: [41]). Целевая переменная записана в первом столбце, признаки – во втором и третьем. Выполним следующее: применим к этим данным однослойный персептрон (о реализации модели Perceptron более подробно здесь: [56]), многослойный персептрон (о реализации модели MLPClassifier более подробно здесь: [57]), сравним точности предсказания моделей, затем нормализуем данные и повторим сравнение. Причем данные мы будем прогонять 100 раз, и на каждом прогоне выполнять 2000 эпох обучения. В качестве показателей выведем минимальные, максимальные, средние значения accuracy и стандартное отклонение, полученные на 100 прогонах. А также выведем график, визуализирующий результаты.

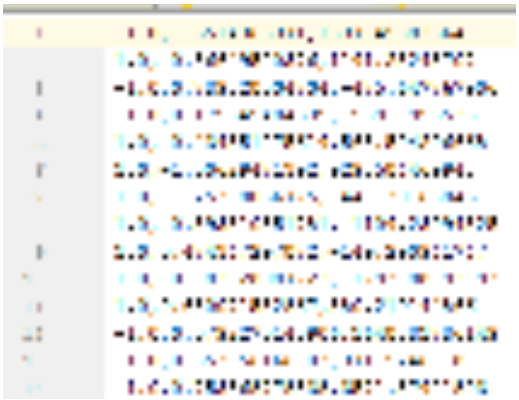


Рисунок 62. Данные для тестовой задачи

Поясним необходимость нормализации. Как и в случае с метрическими методами, качество линейных алгоритмов зависит от некоторых свойств данных. В частности, признаки должны иметь одина-

ковый масштаб. Если это не так, и масштаб одного признака сильно превосходит масштаб других, то качество может резко упасть.

Один из способов достижения нужного свойства признаков заключается в их стандартизации. Для этого берется набор значений признака на всех объектах, вычисляется их среднее значение и стандартное отклонение. После этого из всех значений признака вычитается среднее, и затем полученная разность делится на стандартное отклонение. В ходе нашего эксперимента мы выясним, улучшает ли нормализация качество работы модели. Код решения задачи (с комментариями) представлен в листинге 58. Результат программы, выдаваемый в консоль, представлен на рисунке 63, а построенный график – на рисунке 64.

Листинг 58

```
# -*- coding: utf-8 -*-
#Необходимые импорты
import numpy as np
import pandas
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#функция расчета медианы
def median(lst):
    return np.median(np.array(lst))
#Загрузка тренировочных данных
data_train = pandas.read_csv('perceptron-train.csv')
X_train = data_train.ix[:, 1:3].values
y_train = data_train.ix[:, 0].values
#Загрузка тестовых данных
data_test = pandas.read_csv('perceptron-test.csv')
X_test = data_test.ix[:, 1:3].values
y_test = data_test.ix[:, 0].values
```

Продолжение листинга 58

```
#Инициализация массива для счетчика итераций
rs = np.linspace(0,100,num=100)
#Инициализация списков для сохранения accuracy моделей
acc_p = []
acc_pn = []
acc_mlp = []
acc_mlpn = []
#Цикл прогона моделей
for i in rs:
    i = int(i)
#Распечатка номера итерации
   print "Random: ", i
#Создание модели персептрона
    clf = Perceptron(random_state=i, alpha=0.01, n_iter=2000)
#Обучение модели
   clf.fit(X_train, y_train)
#Получение прогноза
    predictions = clf.predict(X_test)
# Расчет показателя accuracy
    acc = accuracy_score(y_test, predictions)
#Распечатка результата
    print "Perceptron: ", acc
#Добавление оценки в список оценок для модели персептрона
    acc_p.append(acc)
#Нормализация данных
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
#Работа пресептрона с нормализованными данными
    clf = Perceptron(random_state=i, alpha=0.01, n_iter=2000)
    clf.fit(X_train_scaled, y_train)
    predictions = clf.predict(X_test_scaled)
    acc = accuracy_score(y_test, predictions)
    print "Perceptron with normalization: ", acc
    acc_pn.append(acc)
#Создание многослойного классификатора
    mlp = MLPClassifier(random_state=i, solver="sgd", activation="tanh", alpha=0.01,
hidden_layer_sizes=(2, ), max_iter=2000, tol=0.00000001)
```

Окончание листинга 58

```
    mlp.fit(X_train, y_train)
#Работа с ненормализованными данными
    predictions = mlp.predict(X_test)
    acc = accuracy_score(y_test, predictions)
    print "MLP: ",  acc
    acc_mlp.append(acc)
#Работа с нормализованными данными
    mlp = MLPClassifier(random_state=i, solver="sgd", activation="tanh", alpha=0.01,
hidden_layer_sizes=(2, ), max_iter=2000, tol=0.00000001)
    mlp.fit(X_train_scaled, y_train)
    predictions = mlp.predict(X_test_scaled)
    acc = accuracy_score(y_test, predictions)
    print "MLPwith Norm: ",  acc
    acc_mlpn.append(acc)
#Распечатка итоговых результатов
print "Perceptron: ", min(acc_p), median(acc_p), max(acc_p), np.std(acc_p)
print "Perceptron with Norm: ", min(acc_pn), median(acc_pn), max(acc_pn),
np.std(acc_pn)
print "MLP: ", min(acc_mlp), median(acc_mlp), max(acc_mlp), np.std(acc_mlp)
print "MLP with Norm: ", min(acc_mlpn), median(acc_mlpn), max(acc_mlpn),
np.std(acc_mlpn)
#Расчет минимума и максимума для графика
X = np.concatenate((X_train, X_test), axis=0)
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
#Построение графика
figure = plt.figure(figsize=(17, 9))
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
ax = plt.subplot(1, 1, 1)
# Точки из обучающей выборки
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright)
# Тестовые точки
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.6)
ax.set_xlim(x_min, x_max)
ax.set_ylim(y_min, y_max)
ax.set_xticks(())
ax.set_yticks(())
plt.show()
```

Рисунок 63. Результат работы программы

Воспроизведите код, посмотрите на полученный график. Он должен быть похож на изображенный на рисунке 64. Обратим внимание на один важный технический момент. На графике можно увидеть, что принадлежность к классу (то есть по сути Y) отображается не отдельным измерением, а цветом. Такой механизм позволяет рисовать двумерные графики вместо трехмерных, что проще и нагляднее.

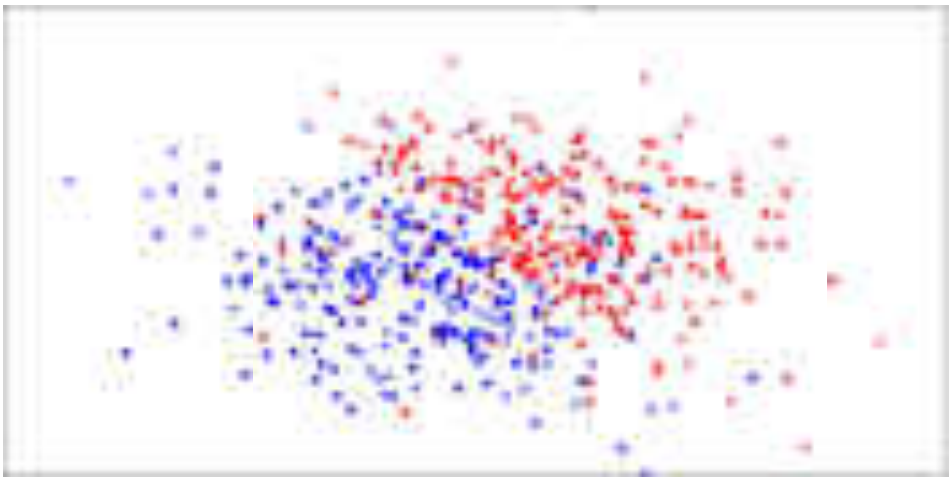


Рисунок 64. Итоговый график

Содержательный анализ результатов работы программы и доработку кода вам предлагается выполнить самостоятельно (см. практическое задание «Многослойный персептрон»).

Реализация алгоритма обучения нейронной сети

Данный пункт посвящен алгоритму обратного распространения ошибки как классическому подходу к обучению нейронных сетей. Поскольку нейронные сети сегодня – это важный класс моделей для построения сложных иерархических признаков, то возникает необходимость детального понимания протекающих в них процессов. Важность именно понимания нейронных сетей обуславливается еще тем, что сложнейшие на сегодняшний день сети еще плохо описаны математически, и требуется проделать много исследовательской работы в этом направлении.

Задача состоит в том, чтобы реализовать обучение полносвязной трехслойной сети прямого распространения на каких-то простых синтетических данных.

Если вы собираетесь заниматься именно исследованием в ML, то подобного рода задачи – это ваша обычная работа. Если вы собираетесь заниматься решением конкретных проблем бизнеса, то с такими задачами вам сталкиваться вряд ли придется, и основная работа будет посвящена подготовке и интерпретации данных. Подготовке данных было уже уделено много внимания, поэтому сейчас рассмотрим исследовательскую задачу. Однако прежде чем перейти непосредственно к ее реализации, рассмотрим применение матричной записи данных в ней.

Так, если вы хотите взвесить некоторые значения X по некоторым весам W и получить сумму, то не обязательно писать выражение $w_1 \times x_1 + w_2 \times x_2$. Ту же самую операцию можно реализовать в виде матричного умножения. Первая матрица A будет представлять из себя X, где столбцы – признаки, а строки – традиционно объекты (если вы обрабатываете один объект за раз, то это может быть матрица с одной строкой). Вторая матрица B будет представлять из себя матрицу весов, где каждая колонка будет отождествляться с нейроном следующего слоя, а каждая строка будет отвечать за конкретный вес (значение веса). Однако даже матрица с одной строкой

в Python будет представлена как двумерный массив. Вы можете поэкспериментировать самостоятельно, выводя различные массивы\матрицы в консоль. После перемножения таких двух матриц получим новую матрицу C, где каждая строка будет представлять собой взвешенный набор признаков $X (x_1, ..., x_n)$ по $W (w_1, ..., w_n)$. Схематично это выглядит так, как показано на рисунке 65 [1].

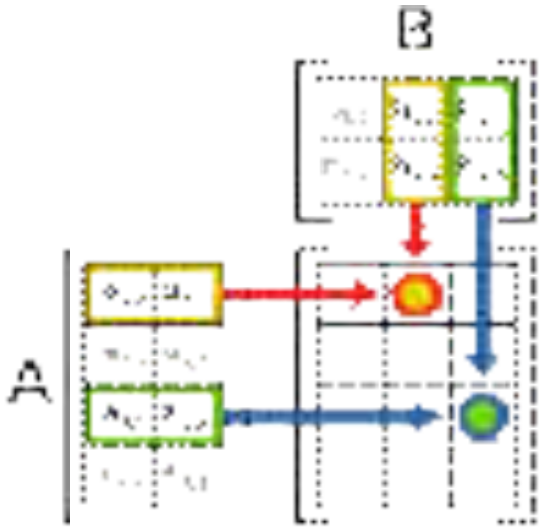


Рисунок 65. Умножение матриц

Необходимо отметить, что взвешивание значений с точки зрения матрицы происходит так, что из матрицы A берутся строки, а из матрицы B берутся колонки. Поэтому иногда массивы\матрицы нужно будет транспонировать. Напомним суть операции транспонирования, приведя в качестве примера рисунок 66.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Рисунок 66. Транспонирование матриц

Объявления функций активации и ее производной представлены в листинге 59.

Листинг 59

```
# -*- coding: utf-8 -*-
#Необходимый импорт
import numpy as np
#Определяем функцию активации
def activation(x):
    return 1 / (1 + np.exp(-x))
#Определяем производную от функции активации
def sigma_derivative(x):
    return x * (1 - x)
```

Реализация алгоритма обраного распространения ошибки с подробными комментариями приведена в листинге 60. Забегая вперед, скажем, что в практической части вам предлагается задача модификации данного кода и интерпретации его результатов.

Листинг 60

```
# X - это матрица, где столбцы это признаки,
# а строки это объекты выборки.
X = np.array([[0, 0, 1],
              [0, 1, 0],
              [1, 0, 0],
              [1, 1, 1]])
# Y - это матрица, где столбцы это признаки
# (в данном случае один целевой признак),
# а строки это объекты выборки.
y = np.array([[0],
              [1],
              [1],
              [0]])
# Зафиксируем генератор случайных чисел, чтобы получать каждый раз
#предсказуемый (одинаковый) результат
np.random.seed(4)
# Нотация слоев - L1, L2, L3.
# В каждом слое есть n1, n2, n3 нейронов.
# Строки в матрицах индексируются как i, столбцы как j.
# W_1_2 - это матрица весов между первым и вторым слоем.
# Строки определяют левый нейрон (т.е. нейрон первого слоя) и соответственно
количество строк равно n1.
```

Продолжение листинга 60

```
# Столбцы определяют правый нейрон (т.е. нейрон второго слоя) и соответственно количество столбцов равно n2.
# На пересечении i-й строки и j-го столбца получаем ячейку с конкретным весом, который
# связывает i-ый левый нейрон (слоя L1) и j-й правый нейрон (слоя L2)
W_1_2 = 2 * np.random.random((3, 2)) - 1
# W_2_3 - это матрица весов между вторым и третьим слоем. Все остальное как в W_1_2
W_2_3 = 2 * np.random.random((2, 1)) - 1
#скорость движения по антиградиенту
speed = 1.1
#цикл прогона модели
for j in range(100000):
    # lo, l1, l2 - матрицы определенного слоя сети. Каждая строка матрицы это реакция #на i-й объект входа. Каждая колонка матрицы это реакция j-го нейрона #соответствующего слоя на разные входные образы.
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной #реакцией j-го нейрона на конкретный вход.
    # Первый слой нейронов полностью принимает значения входа X (т.е. все реакции #единичные).
    l1 = X
    # Второй слой нейронов рассчитывается как функция активации по каждому #элементу матрицы U
    # По сути l2 это матрица 4 на 4, где в каждой ячейке результат активации нейрона #второго слоя для всех 4-х входных образов
    # Детально:
    # матрица U = np.dot(l0, W_0_1) является результатом
    # матричного произведения выходов нейронов предыдущего слоя на веса между 1 #и 2 слоем.
    # Строки матрицы U отвечают за конкретный входной образ (объект).
    # Столбцы матрицы U отвечают за нейроны правого слоя (L2).
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной #взвешенной суммой
    # для i-го входного образа и j-го нейрона слоя (l2). Иными словами, для каждого #нейрона слоя L2 и для каждого входа
    # считается U = W1X1 + W2X2 + W3X3 (поскольку входной нейрон содержит 3 #нейрона).
    l2 = activation(np.dot(l1, W_1_2))
    # тоже самое проделываем для Третьего слоя
```

Продолжение листинга 60

```
    # По сути l3 это матрица 4 на 1, где в каждой ячейке результат активации нейрона
#третьего слоя (а он один)
    #  для всех 4-х входных образов
    l3 = activation(np.dot(l2, W_2_3))
    # рассчитывает ошибку на выходе
    l3_error = y - l3
# рассчитывает модуль средней ошибки
    if (j % 10000) == 0:
        print "Error:" + str(np.mean(np.abs(l3_error)))
    # sigma - есть локальный градиент ошибки
    # l3_sigma рассчитывается как ошибка выхода всей сети на производную функции
#активации всех нейронов L3.
    # На пересечении i-й строки и j-го столбца получаем ячейку с конкретной сигмой
    # для i-го входного образа и j-го нейрона слоя (l3). Иными словами, для каждого
#нейрона слоя L3
    # и для каждого входа рассчитывается производная по функции активации
#умноженная на ошибку.
    # То есть l3_sigma будет матрица 4 на 1 (поскольку в L3 только один нейрон).
    l3_sigma = l3_error * sigma_derivative(l3)
    print l3_sigma
    # Ошибка L2 слоя оценивается через взвешенную сигму слоя L3 по весам между
#L2 и L3
    # Поскольку l3_sigma это матрица 4 на 1 и матрица W_2_3 4 на 1, то последнюю
#матрицу
    # надо Транспонировать, чтобы выполнить правило умножения матриц и взвесить
#элементы l3_sigma
    # по элементам W_2_3.
  # Тогда итоговая матрица l1_error будет 4 на 4, где на пересечении i-й строки и j-го
#столбца
    # будет ячейка с конкретной ошибкой j-го нейрона слоя L2 для i-го входного
#образа.
    l2_error = l3_sigma.dot(W_2_3.T)
    print l2_error
    # l2_sigma рассчитывается как ошибка слоя L2 на производную функции активации
#всех нейронов L2. На пересечении i-й строки и j-го столбца получаем ячейку с
#конкретной сигмой для i-го входного образа и j-го нейрона слоя (L2). Иными
#словами, для каждого нейрона слоя L2 и для каждого входа рассчитывается
#производная по функции активации, умноженная на ошибку.
```

Окончание листинга 60

```
    # То есть l2_sigma будет матрица 4 на 4 (поскольку в L2 четыре нейрона).
    l2_sigma = l2_error * sigma_derivative(l2)
    print l2_sigma
   # обновляем веса
    # l2 это матрица 4 на 4, где в каждой ячейке результат активации нейрона второго
#слоя для всех 4-х входных образов (по строкам).
    # A l3_sigma это матрица 4 на 1, где в каждой строке локальный градиент ошибки
#для 4-х входных образов.
    # Чтобы взвесить столбец матрицы l2 по локальному градиенту (l3_sigma), нужно
#транспонировать матрицу l2, чтобы
    # результат активации каждого нейрона в слое L2 по каждому входному образу
#вытянулся в строку.
    # Тогда соответствующая строка умножится на столбик l3_sigma.
    # Заметьте, что транспонирование l3_sigma не даст результата, так как тогда в
#каждом столбце
    # l3_sigma будет по одному значению, а в каждой строке l2 по 4 значения, а значит
#такое перемножение матриц не пройдет. Применяем транспонирование к l2.
    W_2_3 += speed * l2.T.dot(l3_sigma)
    # аналогично W_2_3
    W_1_2 += speed * l1.T.dot(l2_sigma)
# Прямое распространение для тестовых данных
X_test = np.array([[0, 0, 0],
                   [0, 1, 1],
                   [1, 0, 1],
                   [1, 1, 0],
                   [0.5, 0.5, 0],
                   [0.5, 0.5, 1]])
# Y_test должен получиться [1, 0, 0, 1, 1, 0]
l1 = X_test
l2 = activation(np.dot(l1, W_1_2))
l3 = activation(np.dot(l2, W_2_3))
print l3
```

Регуляризация и сеть прямого распространения

В отличие от предыдущих пунктов здесь не будет кода целиком, а только общие рекомендации по выполнению экспериментов, позволяющих выявить влияние регуляризации на многослойную сеть прямого распространения. В ходе работы над пунктом вы научитесь

генерировать синтетические данные, проводить серии экспериментов, анализировать возможности моделей на разнотипных данных и сравнивать результаты экспериментов.

Необходимые для выполнения работы операторы import представлены в листинге 61.

Листинг 61

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
```

Первый шаг, который вам необходимо выполнить, – генерация данных. Библиотека Scikit-learn содержит множество наборов данных (sklearn.datasets). Среди этого набора есть реальные данные (выборки, которые можно загрузить при помощи функций – Loaders), так и специальные генераторы синтетических данных – Samplesgenerator. Подобные генераторы позволяют создать данные на лету по заданными распределениям, функциям, с возможностью добавления шума, случайных отклонений, смещений и т. п.

Есть несколько классических сценариев для проверки возможностей моделей в машинном обучении. Мы воспользуемся одним генератором линейных задач (datasets.make_classification) и двумя нелинейными генераторами (make_moons, make_circles). Отметим, что одной из важнейших характеристик данных является размерность. Важно понимать, что величина размерности данных соответствует количеству признаков, которые описывают исходные объекты. То есть каждая ось декартового пространства закрепляется за одним количественным признаком. Не все генераторы данных позволяют задавать размерность данных. Тем ни менее ниже приведены крайне важные распределения данных, которые иллюстрируют разные типы задач в машинном обучении. Соответственно тестирование моделей

на подобных данных позволяет без наличия реальных данных проверять какие-либо гипотезы.

Пример генераторов:

- make_classification – позволяет сгенерировать такие классы-признаки, которые будут линейно разделяться в n-мерном пространстве (см. рисунок 67. а).
- make_circles – позволяет сгенерировать такие классы-признаки, что признаки одного класса геометрически окружают признаки другого класса (см. рисунок 67.б).
- make_moons – позволяет сгенерировать такие классы-признаки, что признаки одного класса частично пересекаются с признаками другого класса (см. рисунок 67. в).

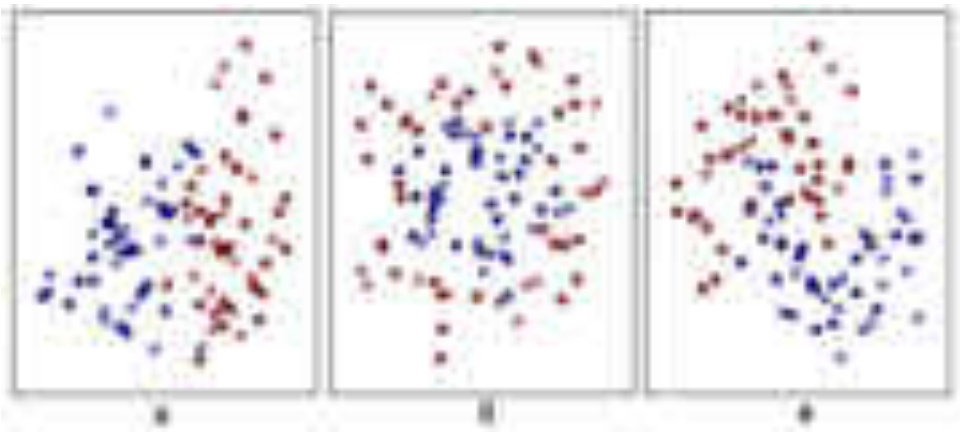


Рисунок 67. Виды распределений признаков, сгенерированных инструментами Scikit-learn

Самостоятельно найдите в Scikit информацию по различным генераторам и сгенерируйте синтетические данные для проведения экспериментов. Пример кода для создания данных трех разнотипных задач классификации приведен в листинге 62.

Для удобства дальнейшей работы рекомендуется объединить все наборы данных в список, как показано в листинге 63, тогда вы сможете организовать цикл по каждому элементу этого списка (что приведено в коде). В этом же листинге есть пример масштабирования данных и разбиения их на тренировочную тестовую выборку.

Листинг 62

```
X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
random_state=0, n_clusters_per_class=1)
rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_dataset = (X, y)
moon_dataset = make_moons(noise=0.3, random_state=0)
circles_dataset = make_circles(noise=0.2, factor=0.5, random_state=1)
```

Листинг 63

```
datasets = [moon_dataset, circles_dataset, linearly_dataset]
for ds_cnt, ds in enumerate(datasets):
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=42)
```

Функция train_test_split делит данные так, что тестовая выборка составляет 40% от исходного набора данных. Разделение происходит случайным образом (т.е. элементы берутся из исходной выборки не последовательно). В этом же цикле можно организовать обучение моделей, их тестирование и формирования графиков.

В теле цикла создайте коллекцию многослойных сетей (MLPClassifier) с различными коэффициентами регуляризации (alpha) и обучите каждую модель на каждом типе данных. Всего должно получиться 15 моделей (3 типа данных\задачи и 5 видов коэффициентов регуляризации). Для реализации используйте код из листинга 64.

Листинг 64

```
alphas = np.logspace(-5, 3, 5)
```

Следующим шагом отобразите исходные данные и результаты на графиках. Пример отображения исходных данных уже был приведен в предыдущих пунктах. Однако на этот раз необходимо построить не один график, а серию графиков в одном окне. В итоге должна получиться таблица из графиков (3 строки, 6 столбцов). Для этого изучите функцию subplot из библиотеки matplotlib. Пример кода для инициализации нового графика (если точнее, то подграфика в рамках одного полотна\окна) показан в листинге 65.

Листинг 65

```
current_subplot= plt.subplot(3, 5 + 1, i)
```

Показанным в листинге 65 способом мы можем получить один из графиков в полотне и заполнять его данными, а потом таким же образом получить следующий график (то есть объект current_subplot). В первую очередь на всех графиках необходимо отобразить исходные данные (точки\признаки). Можете отобразить тренировочную и тестовую выборку или только одну. Пример кода, в котором точки тестовой выборки рисуются полупрозрачными (alpha=0.6), приведен в листинге 66.

Листинг 66

```
current_subplot.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright)
current_subplot.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,alpha=0.6)
```

В листинге 67 cm и cm_bright это цветовые схемы для отрисовки графиков. Их можно взять как из готового набора, так и задать вручную. Это делается следующим образом (листинг 65):

Листинг 67

```
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
```

При отрисовке графиков рекомендуется придерживаться следующего. Пусть левый столбец графиков отображает только исходные данные и ничего больше. Остальные же столбцы должны отображать функции обученных моделей в пространстве исходных данных. Это необходимо, чтобы визуально оценить, как каждая модель разделила исходное пространство признаков. Вместо построения значений функции Y на отдельной оси отобразите значения Y в виде цветовых градиентов (чтобы не строить трехмерные графики). Пример визуализации показан на рисунке 68.

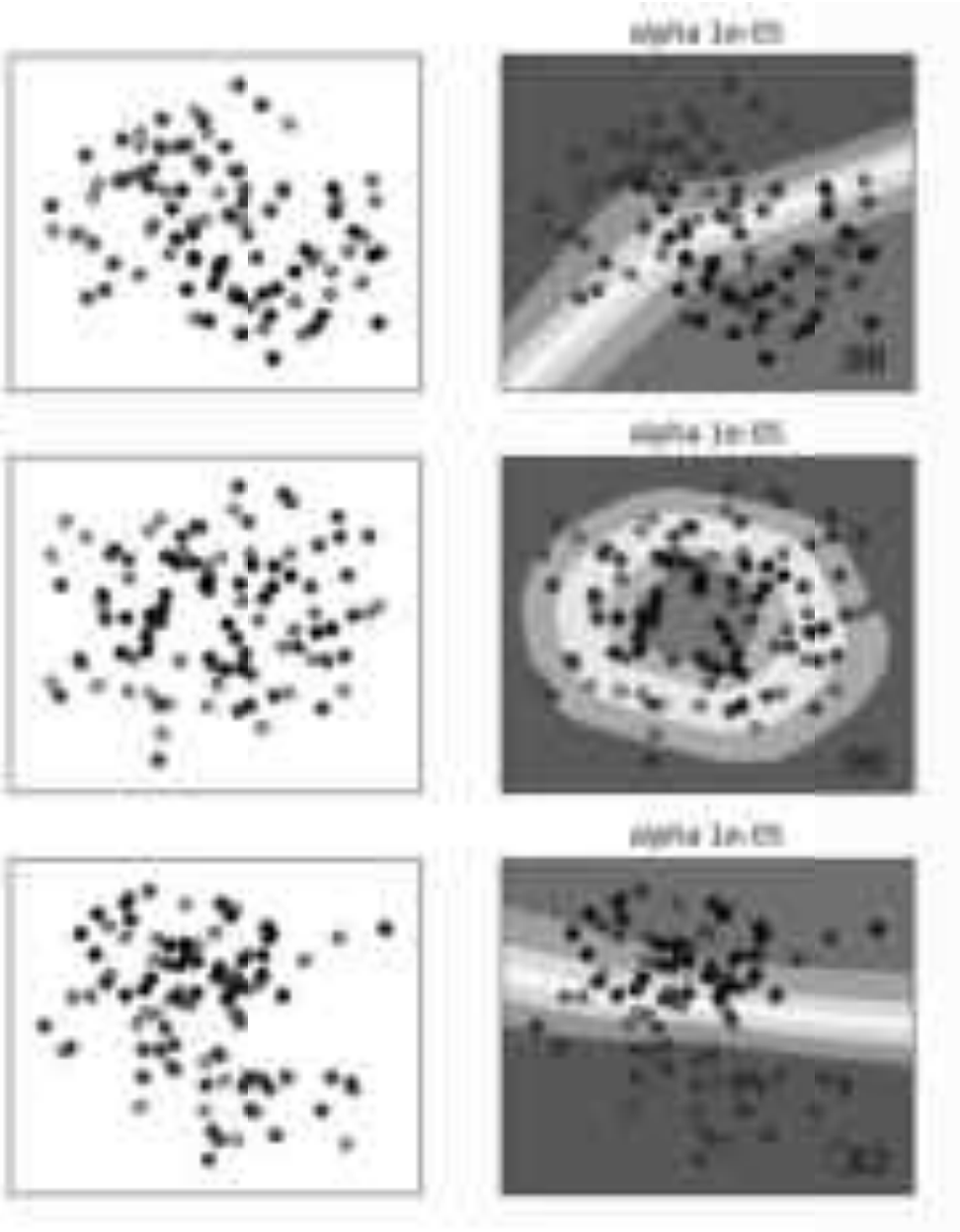


Рисунок 68. Пример визуализации качества моделей

Здесь слева представлена колонка с тремя графиками исходных распределений признаков (на каждом графике две группы точек исходных данных). Каждая строка отвечает за свою задачу классификации (moondataset, circledataset, linearseparabledataset). Правая колонка демонстрирует как конкретная модель многослойной сети MLP (с параметром alpha=0.00001) классифицирует исходное распределение. Значения модели в каждой точке признаков представлены цветовым градиентом. Это гораздо более удобный способ визуализации, чем построение трехмерных поверхностей.

Как мы видим, правый столбец отображает градиент. Чтобы получить градиент, необходимо определить множество точек, по которым будет строиться график. Это можно сделать при помощи регулярной сетки, как показано в листинге 68.

Листинг 68

```
h = .02  #шаг регулярной сетки
x0_min, x0_max = X[:, 0].min() - .5, X[:, 0].max() + .5
x1_min, x1_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx0, xx1 = np.meshgrid(np.arange(x0_min, x0_max, h), np.arange(x1_min, x1_max, h))
```

В листинге 69 представлено несколько вариантов кода для получения значений функции модели в указанных точках сетки.

Листинг 69

```
Z = clf.decision_function(np.c_[xx0.ravel(), xx1.ravel()])#1
Z = clf.predict_proba(np.c_[xx0.ravel(), xx1.ravel()])[:, 1]#2
Z = clf.predict(np.c_[xx0.ravel(),xx1.ravel()])#3
```

Сделаем некоторые пояснения к коду:

1. decision_function возвращает расстояние до разделяющей гиперплоскости в соответствующем измерении (проекционной оси).
2. predict_proba возвращает вероятность принадлежности к каждому классу (так для двух классов можно брать одну из вероятностей (один из столбцов двумерного массива) и исходя из значения раскрашивать область в соответствующий цвет).
3. Predict возвращается целочисленную оценку принадлежности к классу (как бинарная маска).

Третий вариант в данном случае выдаст округленные значения классов (0, 1 и т. д.), а функция predict_proba выдает сглаженную функцию вероятности класса. Чтобы понять разницу, попробуйте различные варианты.

Однако не все классификаторы имеют decision_function. Чтобы узнать, содержит ли текущий объект интересующую нас функцию, можно воспользоваться методом hasattr. Этот метод возвращает флаг,

указывающий на то, содержит ли объект указанный атрибут и нужен для обработки разных вариантов классификаторов (можно выстроить на этом if-else логику). Пример ее вызова показан в листинге 70.

Листинг 70

```
hasattr(clf, "decision_function")
```

Далее вам необходимо нарисовать результат, оси, заголовок и текст со значением точности модели. Выполнить это можно с помощью кода из листинга 71.

Листинг 71

```
Z = Z.reshape(xx0.shape)
current_subplot.contourf(xx0, xx1, Z, cmap=cm, alpha=.8)
current_subplot.set_xlim(xx0.min(),xx0.max())
current_subplot.set_ylim(xx0.min(),xx1.max())
current_subplot.set_xticks(())
current_subplot.set_yticks(())
current_subplot.set_title(name)
current_subplot.text(xx0.max() - .3, xx1.min() + .3, ('%.2f' % score).lstrip('0'),
size=15, horizontalalignment='right')
```

После отрисовки градиента добавьте на каждый график тестовую выборку (точки из каждой задачи для графика каждой модели). Это можно сделать таким же образом, как показано в листинге 72.

Листинг 72

```
current_subplot.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,alpha=0.6)
```

Не забудьте, что после построения каждого графика необходимо инкрементировать переменную i для функции subplot(), а также помните о необходимости применить функцию plt.show() в конце всех операций формирования графиков для вывода результата в окне ОС.

При желании вы можете настроить отступы графиков в окне просмотра при помощи следующего кода (листинг 73).

Листинг 73

```
figure.subplots_adjust(left=.02, right=.98)
```

Для задания заголовков и вывода текста на графиков можно использовать следующие функции из листинга 74.

Листинг 74

```
current_subplot.set_title("Input data")
current_subplot.text(xx0.max() - .3, xx1.min() + .3, ('%.2f' % score).lstrip('0'), size=15,
horizontalalignment='right')
```

Работа с библиотеками Keras и Theano. Настройка под Windows

Мы закончили рассмотрение простых моделей нейронных сетей и переходим к более сложным. В частности, к сверточным сетям. Однако поскольку в библиотеке scikit нет полноценной поддержки таких сетей, то в данном пункте мы рассмотрим установку и настройку библиотек Keras и Theano, потому что для пользователей Windows установка этих библиотек нетривиальна.

Для разворачивания Keras и Theano для CPU на Windows выполните следующее:

1. Откройте Anaconda prompt (командную строку Анаконды через меню «Пуск» или просто выполните стандартную команду cmd. Если все установлено верно, то Anaconda «перехватит» адресованные ей команды)
2. Выполните conda update conda
3. Выполните conda update --all
4. Выполните conda install mingw libpython
5. УстановитеTheano, pip install Theano
6. Установите Keras pip install keras
7. Выполните код для проверки корректности всех действий (он показан в листинге 75).

Листинг 75

```
import numpy as np
import sklearn.linear_model as lin
from keras.layers.core import Dense, Activation
from keras.models import Sequential
from keras.optimizers import RMSprop

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([0, 1, 1, 0]).reshape(4, 1)
x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

model = Sequential()
model.add(Dense(2, init='lecun_uniform', input_shape=(2,)))
model.add(Activation('relu'))
model.add(Dense(1, init='lecun_uniform'))
model.add(Activation('linear'))
rms = RMSprop(lr=0.01)
model.compile(loss='mse', optimizer=rms)

epochs = 200
model.fit(x_train, y_train, batch_size=1, nb_epoch=epochs, verbose=0)
y_predict = model.predict(x_test, batch_size=1)
print y_predict
```

Теперь опишем разворачивание Theano для GPU на Windows. Ведь обучение даже относительно простых сверточных сетей – крайне затратный вычислительный процесс, который может занять несколько часов на довольно мощном процессоре (здесь также стоит учитывать, что не все библиотеки и платформы поддерживают параллельное исполнение кода между ядрами процессоров, а значит, попросту не используют всю вычислительную мощь CPU).

Современные графические карты позволяют запускать очень много параллельных вычислений на аппаратном (а не только на программном) уровне. Лидером таких вычислений на сегодняшний день является компания NVidia, которая предлагает специальную платформу под названием CUDA. Это программно-аппаратная платформа, которая встроена в видеокарты и драйвера NVidia. Подобная

технология может ускорить вычисления во много раз (от 4 до 100). Конечно же, GPU вариант предпочтительнее не только в реальных проектах и промышленных системах, но и в процессе обучения, так как может существенно сэкономить время на эксперименты и выполнение учебных заданий.

Библиотека Theano может исполнять свои модели на графическом процессоре (GPU), который установлен в системе. Однако поддерживаются только графические процессоры от NVidia и только те, которые поддерживают режим CUDAComputeCapability 3.0 или выше. Узнать о том, какой режим поддерживает ваша видеокарта можно на официальном сайте NVidia.

Само разворачивание Theano состоит из следующих шагов:

1. Проверьте, что у вас есть свежий драйвер на видеокарту Nvidia. Необходима именно карта NvidiacCompute Capablity = 3.0 или выше (проверить Compute Capablity своей видеокарты можно тут [58]). Если карты от Nvidia нет, то этот сценарий реализовать не получиться и просто используйте CPU.
2. Скачайте и установите CUDA Toolkit 8.0 отсюда: [59].
3. Скачайте Cudnn 5.0 для CUDA Toolkit 8.0 отсюда [60]. Это библиотека для низкоуровневой поддержки нейронных сетей. Распакуйте архив Cudnn. Там будет несколько папок. В каждой папке по одному файлу. Эти файлы нужно переместить в место, куда был установлен CUDA Toolkit 8.0. Например, файл cudnn64_5.dll из папки cuda\bin положить в папку bin в место, куда установился CUDA Toolkit 8.0 и так далее.
4. Теперь код Theano может запускаться на GPU. Но его еще нужно скомпилировать. Для компиляции нужен VC++ компилятор от Майкрософта. Нужную версию компилятора можно получить, установив Visual Studio 2012.
5. Создайте текстовый файл с именем .theanorc, заполненный как показано в листинге 76. Описание приведено в листинге 77.

6. Созданный файл нужно положить в домашнюю или пользовательскую директорию ОС. Для Win это C:\Users\$UserName$. Чтобы четко определить директорию, можно выполнить следующий код (листинг 78).

Листинг 76

```
[global]
floatX=float32
device=gpu
mode=FAST_RUN
cnmem=0.7
 [cuda]
root=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5
 [nvcc]
flags=-LD:\Program_Files\Anaconda\libs
–compiler_bindir=C:\Program_Files_(x86)\Microsoft_Visual_Studio_11.0\VC\bin\
–fastmath=True
```

Листинг 77

```
[global]
Cnmem - объем памяти, который используется на видео карте
 [cuda]
Root - путь на корневую папку CUDAtoolkit. Например:
«root=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0»
 [nvcc]
Flags - указывает путь до библиотек Anaconda
compiler_bindir - путь до компилятора cl.exe из VisualStudio
Во всех параметрах [nvcc] не должно быть пробелов. Так «Program Files (x86)» нужно заменить на  «Program_Files_(x86)». Пример пути:
«D:\Program_Files\Visual_Studio_1\VC\bin\»
```

Листинг 78

```
os.environ['THEANO_FLAGS']="device=gpu0,lib.cnmem=0.5"
path=os.path.expanduser('~/.theanorc.txt')
```

7. Выполните проверочный код из листинга 79. Он должен вывести 'Used the gpu' и нулевой код ошибки.

Листинг 79

```
from theano import function, config, shared, sandbox
import theano.tensor as T
import numpy
import time
 vlen = 10 * 30 * 768  # 10 x #cores x # threads per core
iters = 10000
  rng = numpy.random.RandomState(22)
x = shared(numpy.asarray(rng.rand(vlen), config.floatX))
f = function([], T.exp(x))
print(f.maker.fgraph.toposort())
t0 = time.time()
for i in range(iters):
    r = f()
t1 = time.time()
print("Looping %d times took %f seconds" % (iters, t1 - t0))
print("Result is %s" % (r,))
if numpy.any([isinstance(x.op, T.Elemwise) for x in f.maker.fgraph.toposort()]):
    print('Used the cpu')
else:
    print('Used the gpu')
```

Получение данных средствами Keras

Подобно библиотеке Scikit-learn библиотека Keras также имеет встроенные функции для получения готового набора данных. Пример показан в листинге 80. На выходе у нас получаются стандартные массивы Numpy.

Листинг 80

```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Забежав немного вперед, скажем, что сверточные модели, представленные библиотекой Keras, принимают на вход немного непривычный формат данных для обучения. Так параметр Input_shape функции model.fit() имеет следующую форму: (samples, channels, rows, cols). Здесь samples – количество элементов в выборке (может

не указываться), channels – количество каналов (если речь идет об RGB-изображениях, то на каждую компоненту будет по матрице и этот параметр будет равен трем), rows и cols соответственно ширина и высота матрицы (изображения).

Исходя из этого, нам необходимо модифицировать полученные данные. Чтобы сменить форму массива Numpy, выданного load_data, воспользуйтесь функцией reshape, которая в качестве параметров принимает размер каждого измерения. Пример функции показан в листинге 81.

Листинг 81

```
X_train = X_train.Reshape(N, 1, r, c)
```

Кроме смены формы массива его элементы необходимо привести в диапазон от 0 до 1 для корректной работы с ним сверточной нейронной сетью. Сделайте это самостоятельно.

Над выборкой Y также необходимо провести преобразования. Целое число нужно перевести в бинарную маску (вектор) так, чтобы класс 2 из трех возможных классов представлял собой вектор [0,0,1] при отсчете от нуля. Подобное преобразование можно сделать при помощи функции np_utils.to_categorical.

Создание и обучение модели сверточной сети

Общая логика обучения модели такая же, как и в моделях библиотеки Scikit-Learn. Но поскольку Keras больше ориентирована на нейросетевые модели, то она позволяет более детально настраивать структуру и работу сети. В этом пункте мы рассмотрим лишь некоторые из возможностей.

Модели в Keras описываются как вычислительные графы. Поэтому, чтобы создать и обучить сверточную сеть, сначала необходимо создать последовательную модель (граф последовательных вычислений\обработчиков, «контейнер моделей»). Сделать это можно так, как показано в листинге 82.

Листинг 82

```
model = Sequential()
```

После создания граф необходимо заполнить рядом обработчиков. Описание всех возможных обработчиков есть в документации Keras, например, здесь: [61]. Ниже представлен список основных обработчиков, с которыми мы столкнемся в этом пункте.

Dense – одномерный слой обычных нейронов. По сути, этот обработчик вычисляет взвешенные суммы для каждого нейрона слоя, но не вычисляет для них функцию активации (так как она задается отдельным обработчиком). Функция, создающая обработчик, принимает на вход число нейронов. Использовать ее можно так, как показано в листинге 83.

Листинг 83

```
Dense(128)
```

ConvolutionND – сверточный слой нейронов. Данный обработчик подсчитывает взвешенные суммы для каждой карты и каждого нейрона в слое, но также не вычисляет для них функцию активации. Функция, создающая обработчик принимает на вход число карт в слое и размер ядра\фильтра, также определяется форма входа и режим обработки границ (valid означает, что все ядра поместятся в исходную матрицу и не будет выступающих границ, при этом карта будет меньше исходной матрицы; в случае same карта будет такого же размера, как и входная матрица). Можно задать и ряд других параметров, но с ними вам предлагается ознакомиться самостоятельно. Пример использования приведен в листинге 84.

Activation – обработчик функции активации. Задается после каждого слоя нейронов (Convolution или Dense), такие обработчики, как MaxPooling2D или Dropout, не являются слоями нейронов. Данный обработчик рассчитывает значение активации для взвешенных сумм предыдущего слоя. В качестве аргумента функция прини-

мает строку с названием собственно функции активации (relu, tanh, linear, sig, prelu). Пример использования – в листинге 85.

Листинг 84

```
Convolution2D(nb_filters, nb_conv, nb_conv,
border_mode='valid',
input_shape=(1, img_rows, img_cols)))
```

Листинг 85

```
Activation('relu')
```

MaxPoolingND – обработчик объединения по максимуму. Обработчик проходит по матрице, поступающей на вход, окном размера pool_size (например, 2X2) и подает на выходную матрицу одно максимальное значение из данной области входной матрицы. Иллюстрация работы функции – рисунок 69 [1].

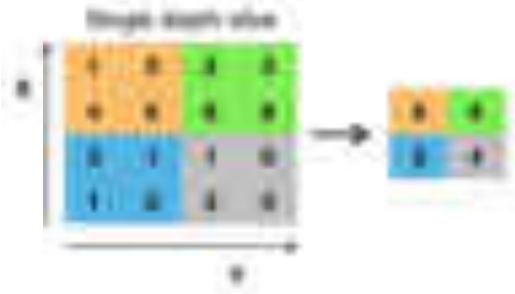


Рисунок 69. Иллюстрация действия обработчика MaxPooling2D

Как мы видим, из входной матрицы 4X4 формируется матрица 2X2 максимальных значений. Пример вызова функции приведен в листинге 86.

Листинг 86

```
MaxPooling2D(pool_size=(nb_pool, nb_pool))
```

Dropout – данный обработчик задает степень разрыва нейронных связей между соседними слоями. В качестве параметра функция принимает значение от 0 до 1, выражающее процент разрыва. В данном случае 25% связей между слоями выродятся в ноль и не будут меняться в процессе обучения.

Обратите внимание!

- *Такой возможности не было в библиотеке scikit-learn. Это важная настройка, так как она позволяет контролировать переобучение сети не только регуляризацией. Кроме того, таким способом можно добиться эффекта локальности.*

Вызов обработчика – в листинге 87.

Листинг 87

```
Dropout(0.25)
```

Flatten – выравнивающий обработчик. Преобразует связи от нейронов слоя свертки к одномерному слою обычных нейронов. Такой обработчик используется ближе к завершению сети, чтобы формировать выход модели. Пример вызова – в листинге 88.

Листинг 88

```
Flatten()
```

Если создать модель из двух сверточных слоев с функцией активации «relu», затем перейти к обычным слоям, добавить разрывы и завершите модель слоем из 10 нейронов с функцией активации «softmax», то данная функция сформирует выходной вектор таким образом, чтобы сумма всех элементов была равна единице, а отдельный элемент вектора выражал вероятность принадлежности входного образа к классу, который отображается этим элементом вектора.

Чтобы добавить тот или иной обработчик в вычислительный граф, используется функция add, как показано в листинге 89.

После формирования графа необходимо скомпилировать модель. Причем то же самое придется выполнить и после загрузки модели из файла, поскольку сохраняется лишь граф вычислений, а не готовый к исполнению объект. Пример компиляции модели приведен в листинге 90.

Листинг 89

```
model.add(<обработчик>)
```

Листинг 90

```
model.compile(loss='categorical_crossentropy',
optimizer='adadelta',
metrics=['accuracy'])
```

Обучается модель уже знакомым способом (с помощью функции fit, куда передаются тренировочные выборки X, Y). Однако в данных моделях имеет смысл рассмотреть еще несколько параметров:

- batch_size – определяет количество предъявляемых образов в рамках одного обновления весов (так если batch_size>1, то производиться обучение по группе, а не по одиночным образам). Данный параметр может повлиять на скорость обучения (особенно на CPU) и на объем требуемой памяти. Так небольшие группы по 50 образов могут потребовать меньше памяти и вычислительных ресурсов (слово «могут» используется потому, что есть и другие факторы, в том числе аппаратная реализация некоторых функций и поведение кэша).
- nb_epoch – определяет количество эпох обучения.
- verbose – определяет, выводить ли в консоль детали обучения или нет. Под деталями понимается информация о ходе обучения, времени и оценке качества за каждую итерацию.
- validation_data – определяет набор данных, по которому будет проводиться оценка качества модели.

Пример обучения модели приведен в листинге 91.

Листинг 91

```
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
verbose=1, validation_data=(X_test, Y_test))
```

Качество обученной модели проверяется кодом из листинга 92.

Листинг 92

```
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Напомним, что:

- Величина loss характеризует интегральную оценку всех среднеквадратических ошибок. Единицы этой величины совпадают с единицами Y. Чем меньше это значение, тем лучше. Однако для каждой задачи (каждого типа данных) конкретные величины будут разными, и оценивать эту величину нужно\можно, только зная характер данных и допустимые величины. Так, для оценки стоимости квартиры общая ошибка 300 000 на тренировочной выборке в несколько тысяч квартир может быть вполне приемлемой. А вот такая же величина loss для оценки уровня лейкоцитов в крови на тысячу пациентов просто неприемлема.
- Величина accuracy характеризует процент верных ответов из тестовой выборки. Чем ближе это значение к 100% (или 1.00), тем лучше.

Загрузка и сохранение сложных моделей

В одном из прошлых пунктов был рассмотрен вариант сохранения объектов посредством функций pickle.dump(obj, output) и pickle.load(f). Однако многие модели Keras имеют такую сложную структуру, что стандартный сериализатор объектов вызывает исключение из-за глубины ссылок. Поэтому для сохранения и загрузки сложных моделей нейронных сетей необходимо использовать следующий код (листинг 93).

Обратите внимание!

- *Настройки модели и веса сохраняются раздельно.*

Листинг 93

```
def SaveToJSon(model, fileName):
    model_json = model.to_json()
with open(fileName + ".net", "w") as json_file:
        json_file.write(model_json)
    model.save_weights(fileName + ".wgt")
print("model saved to disk")
def LoadFromJSon(fileName):
    json_file = open(fileName + ".net", 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    loaded_model.load_weights(fileName + ".wgt")
print("Loaded model from disk")
return loaded_model
```

Рекуррентные сети для прогнозирования временных рядов

Теперь рассмотрим пример работы с рекуррентными сетями средствами библиотеки Keras, а именно, их применение для решения задачи прогнозирования временных рядов. Пусть в файле «international-airline-passengers.csv» (скачать его можно отсюда: [41]) находятся данные о ежемесячном количестве пассажиров международного аэропорта в виде, представленном на рисунке 70. Применим рекуррентную сеть для прогнозирования количества пассажиро в будущем. Глубину прогноза возьмем равной одному месяцу, как и количество точек, по которому будем строить прогноз. Качество работы модели оценим по величине среднеквадратичной ошибки. Данная задача и код ее решения, показанный в листинге 94, взяты из [62].

Листинг 94

```
# -*- coding: utf-8 -*-
#Необходимые импорты
import numpy
import matplotlib.pyplot as plt
```

Продолжение листинга 94

```
from pandas import read_csv
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
# функция для создания выборки из исходного массива данных
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return numpy.array(dataX), numpy.array(dataY)
# фиксация состояния рандома для получения предсказуемого результата
numpy.random.seed(7)
# загрузка данных
dataframe = read_csv('international-airline-passengers.csv', usecols=[1], engine='python',
skipfooter=3)
dataset = dataframe.values
dataset = dataset.astype('float32')
# нормализация данных
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
# ручное разделение данных на обучающую и тестовую выборку
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
# Изменение исходного набора данных для того, чтобы в каждой строке X оказался
# временной ряд размером look_back, начиная от момента времени t0. A в Y
# временной ряд единичного размера от момента времени t0 + look_back.
# В данном примере, для look_back=4  X1 будет состоять из объектов:  Data[t0],
# Data[t1], Data[t2], Data[t4], а Y1 из объектов: Data[t5]. X2 начнется с элемента
# Data[t1] и т.д.
 trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
# изменение формы входных данных и приведение их к виду
# [количество объектов, размер временного ряда из объектов, количество признаков
каждого объекта]
```

Окончание листинга 94

```
trainX = numpy.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = numpy.reshape(testX, (testX.shape[0], testX.shape[1], 1))
# Создание графа вычислений (некоторой абстрактной модели)
model = Sequential()
#4 – это количество блоков LSTM, input_shape - это форма вектора входа
model.add(LSTM(4, input_shape=(look_back, 1)))
model.add(Dense(1))
#Компиляция модели
model.compile(loss='mean_squared_error', optimizer='adam')
#Обучение модели
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
# выполнение предсказания
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# так как прогнозное значение нормализовано, то необходимо его восстановление
# под восстановлением понимается приведение к исходному масштабу данных
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# вычисление ошибки прогноза
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
# подготовка данных для отрисовки
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] = testPredict
# построение графика
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Рисунок 70. Данные о пассажирах

На этом мы заканчиваем рассмотрение инструментальных средств для разработки приложений сферы машинного обучения на языке Python. Как вы могли заметить, в пособии не были рассмотрены реализации генетических алгоритмов и методов нечеткой логики. Это связано с тем, что их реализация на языке Pyhton не требует каких-либо специализированных библиотек, поэтому эту часть практического материала вам предлагается освоить самостоятельно в рамках выполнения задач по этим темам (см. раздел «Практические задания»). В случае затруднения вы можете обратиться к следующим ресурсам: реализация генетического алгоритма на Python – [63] и материалы учебного пособия [64], где рассматривается реализация генетических алгоритмов и некоторых операций нечеткой логики на языке Java.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ТЕСТОВЫЕ ЗАДАНИЯ

Тест «Общие сведения о машинном обучении»

1. Выберите верные утверждения
   a) Одна из задач машинного обучения – научиться делать прогнозы для признаков
   b) Объекты описываются с помощью признаков
   c) Одна из задач машинного обучения – научиться делать прогнозы для объектов
   d) Признаки описываются с помощью объектов
2. Какие из этих задач являются задачами классификации?
   a) Прогноз температуры на следующий день
   b) Разделение книг, хранящихся в электронной библиотеке, на научные и художественные
   c) Поиск групп похожих пользователей интернет-магазина
   d) Прогноз оценки студента по пятибалльной шкале на экзамене по машинному обучению в следующей сессии
3. Какие свойства данных препятствуют однозначному построению разделяющей поверхности?
   a) Ортогональность
   b) Мультиколлинеарность
   c) Противоречивость
   d) Категориальность
4. Какая способность людей и систем позволяет получать им новые знания по наблюдению отдельных прецедентов (примеров)?
   a) Корректировать ошибку
   b) Обобщать
   c) Запоминать
   d) Распознавать образы
5. Какая задача лучше всего подходит под следующее описание. Нахождение такой функции F, которая бы наилучшим образом отображала неизвестные ранее объекты X в конечное множество

целочисленных номеров (имен, меток), на основании обучающих пар (X, Y)?

a) Прогнозирование денежных затрат
b) Кластеризация клиентов
c) Классификация образов
d) Выявление особенностей в данных

6. Почему для обучения моделей используются такие методы, как Градиентный спуск?
   a) Потому что метод позволяет корректировать параметры модели постепенно
   b) Потому что аналитические решения не всегда дают корректное решение
   c) Потому что такой подход позволяет получать более точные решения (Глобальный экстремум в отличие от локального)
   d) Потому что при большой размерности входных данных подобные методы работают быстрее
7. Выберите верные утверждения
   a) Метод Байеса – это во многом классический подход к классификации, основанный на оценке частоты встреч объектов со схожими признаками
   b) Благодаря универсальности статистического подхода метод Байеса позволяет решать любые задачи без априорной информации
   c) Данный метод позволяет очень хорошо обобщать высокоуровневые признаки
   d) Закон, задающий распределение вероятностей, который используется в предсказательной модели, сильно влияет на способ обобщения

Проблема переобучения

1. Какие факторы влияют на переобучение?
2. Какие есть способы оценки переобучения?
3. Какие есть способы борьбы с переобучением?

Регрессия

1. Почему такая простая формула, как y=kx+b, позволяет делать прогнозы или классификацию?
2. В чем отличие линейной и логистической регрессий?
3. В чем отличие линейной от нелинейной регрессии?
4. В чем отличие линейной регрессии от полиномиальной?
5. Что позволяет делать LASSO?
6. В чем заключаются особенности Ridge регрессии?

Модели и методы нечеткой логики

1. Охарактеризуйте следующие понятия: нечеткие множества, операции нечеткой логики, нечеткие модели или нечеткие системы.
2. Сформулируйте понятие лингвистической неопределенности.
3. Напишите функцию лингвистической неопределенности некоторого объекта.
4. Дайте определение функции принадлежности.
5. К какому типу относятся нечеткие множества, если значения функции принадлежности нечеткого множества представлены точными числовыми значениями?
6. К какому типу относятся нечеткие множества, если значения функции принадлежности нечеткого множества моделируются другими нечеткими множествами?
7. Какие существуют способы задания функции принадлежности?
8. Приведите три примера функции принадлежности, задаваемых функциональным способом.
9. Напишите формулу компактной записи функции принадлежности.
10. Сформулируйте определение лингвистической переменной.
11. Опишите набор переменных, с помощью которого описывается лингвистической переменная.
12. Дайте определения следующих понятий: X – универсальное множество объектов, $\widetilde{X}$ – базовое терм-множество, G – синтакси-

ческие правила вывода (порождения) новых термов, Р – семантические правила.

13. Сформулируйте определение нечеткой логики.
14. Какие необходимые характеристики нечеткой логики ввел Л. Заде?
15. Что лежит в основе операций нечеткой логики?
16. Какие операции используются для моделирования основных логических связок И, ИЛИ над нечеткими множествами в нечеткой логике?
17. Сформулируйте определения триангулярной нормы и триангулярной конормы (t-норма и s-конорма) и докажите их двойственность. Приведите примеры.
18. Запишите композиционное правило Л. Заде.
19. Сформулируйте содержательное определение операции импликации.
20. Сформулируйте определение системы нечеткого логического вывода.
21. Какие объекты входят в систему нечеткого логического вывода?
22. Какие условия должны соблюдаться при построении правил на основе системы нечеткого логического вывода?
23. Перечислите пять способов реализации нечеткого логического вывода.
24. Подробно опишите реализацию нечеткого логического вывода с помощью алгоритма Мамдани. Нарисуйте схему процесса нечеткого вывода этого алгоритма.

Нечеткие временные ряды

1. Сформулируйте определение уровня временного ряда.
2. Сформулируйте определение нечеткого временного ряда.
3. Сформулируйте определение носителя нечеткой метки $\widetilde{x}_i$.
4. Приведите примеры прикладных задач обработки нечетких ВР.
5. Дайте определение нечеткому разбиению.

6. Опишите основные этапы алгоритма моделирования нечетких ВР в соответствии с нечеткой моделью Сонга.
7. В чем преимущество использования моделей нечетких ВР?
8. Что понимается под интеллектуальным анализом данных или Data Mining?
9. Какие задачи решаются на основе Data Mining?
10. Приведите виды темпорально-логических концептов ВР.
11. В чем заключается метод аппроксимации временного ряда на основе F-преобразования?

Нечеткая регрессия

1. Какие существуют подходы к построению моделей нечеткой линейной регрессии?
2. Какие существуют критерии для определения нечетких коэффициентов модели?
3. Какие вы знаете варианты методов на основе классификации «вход – выход»?

Генетические алгоритмы

1. Поясните происхождение термина «генетические алгоритмы».
2. Опишите сферу применения генетических алгоритмов.
3. Дайте определение гену в контексте генетических алгоритмов.
4. Дайте определение хромосоме в контексте генетических алгоритмов.
5. Дайте определение популяции в контексте генетических алгоритмов.
6. Дайте определение степени приспособленности в контексте генетических алгоритмов.
7. Дайте определение кроссовера в контексте генетических алгоритмов.
8. Дайте определение мутации в контексте генетических алгоритмов.
9. Перечислите методы отбора хромосом для кроссовера.
10. Расскажите о типовой схеме генетического алгоритма.

Нечеткая кластеризация

1. Дайте определение классификации.
2. Дайте определение кластеризации.
3. Поясните разницу между классификацией и кластеризацией.
4. Определите область применение кластеризации.
5. Определите цель алгоритма FCM-кластеризации.
6. Определите перечень входных данных для алгоритма FCM-кластеризации.
7. Перечислите шаги алгоритма FCM-кластеризации.
8. Почему кластеризация, проводимая с помощью алгоритма FCM, является нечеткой?
9. Поясните назначение параметра «степень нечеткости» в алгоритме FCM.
10. Что является выходными данными алгоритма FCM-кластеризации?

Искусственные нейронные сети и глубинное обучение

1. Какие достоинства и недостатки есть у ИНС по сравнению с Регрессией и Решающими Деревьями?
2. Сеть какого типа лучше использовать для прогнозирования?
3. Сеть какого типа можно использовать в условиях постоянного изменения данных, когда точной выборки еще не существует?

Тест «Искусственные нейронные сети»

1. Выберите верные утверждения:
   a) ИНС проще подобрать под любую нелинейную задачу. Все, что нужно сделать, это увеличивать число слоев пропорционально числу признаков
   b) ИНС позволяют обрабатывать более высокоуровневые признаки за счет нелинейной функции активации и последовательным слоям

c) По сравнению с Регрессией ИНС практически не подвержены Переобучению при любом количестве нейронов
d) С точки зрения математического аппарата ИНС – это комбинация полиномиальной регрессии высокого порядка и формулы Байеса
e) ИНС может аппроксимировать любую нелинейную непрерывную функцию, но это еще не гарантирует 100% сходимости на произвольных данных
f) ИНС в отличие от регрессии может хорошо обрабатывать высокую степень мультиколлинеартности и противоречивости в данных

2. Сеть какого типа лучше использовать для прогнозирования временных рядов?

a) Сверточную
b) ART MAP
c) Импульсную
d) MLP
e) Рекуррентную
f) Когнитрон

3. Сеть какого типа лучше использовать для обработки трехмерных сцен?

a) MLP
b) Рекуррентную
c) ART MAP
d) Сверточную
e) Когнитрон
f) Импульсную

4. Сеть какого типа лучше использовать для решения задачи классификации клиентов по одиночному вектору клиентских характеристик (с учетом того, что этот вектор содержит большое количество категориальных признаков)?

a) Автокодировщик
b) MLP

c) Когнитрон
d) ART MAP
e) Сверточную
f) Рекуррентную
g) Импульсную

5. Сеть какого типа можно использовать в условиях постоянного изменения данных, когда точной выборки еще не существует и сеть приходится постоянно дообучивать на новых классах?

a) MLP
b) Сверточную
c) Когнитрон
d) Рекуррентную
e) ART MAP
f) Автокодировщик
g) Импульсную

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Работа с файлом данных «Титаника»

Приведенные ниже задания основаны на данных 'titanic.csv', где содержатся сведения о пассажирах «Титаника».

Задачи:

1. Какое количество мужчин и женщин плыло на корабле?
2. Какой части пассажиров удалось выжить? Посчитайте долю выживших пассажиров. Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.
3. Какую долю пассажиры первого класса составляли среди всех пассажиров? Ответ приведите в процентах (число в интервале от 0 до 100, знак процента не нужен), округлив до двух знаков.
4. Какого возраста были пассажиры? Посчитайте среднее и медиану возраста пассажиров. В качестве ответа приведите два числа через пробел.
5. Коррелируют ли число братьев/сестер/супругов с числом родителей/детей? Посчитайте корреляцию Пирсона между признаками SibSp и Parch.
6. Какое самое популярное женское имя на корабле? Извлеките из полного имени пассажира (колонка Name) его личное имя (First Name). Это задание – типичный пример того, с чем сталкивается специалист по анализу данных. Данные очень разнородные и шумные, но из них требуется извлечь необходимую информацию. Попробуйте вручную разобрать несколько значений столбца Name и выработать правило для извлечения имен, а также разделения их на женские и мужские.

Работа по отбору признаков

Доработайте код отбора признаков, использовав все модели из списка:

1. Линейная регрессия (LinearRegression)
2. Гребневая регрессия (Ridge)
3. Лассо (Lasso)
4. Случайное Лассо (RandomizedLasso)
5. Рекурсивное сокращение признаков (Recursive Feature Elimination – RFE)
6. Сокращение признаков Случайными деревьями (Random Forest Regressor)
7. Линейная корреляция (f_regression)

Отобразите получившиеся значения\оценки каждого признака каждым методом\моделью и среднюю оценку. Проведите анализ получившихся результатов. Какие 4 признака оказались самыми важными по среднему значению? (Названия\индексы признаков и будут ответом на задание).Какие выводы можно сделать по результатам отдельных методов?

Многослойный персептрон

1. Доработайте код задачи классификации (листинг 58). Попробуйте настроить различные параметры сети, такие как количество нейронов в первом или втором слое, вид функции активации (activation), алгоритм градиентного спуска (solver), количество максимальных итераций при обучении (max_iter), порог точности при обучении (tol). Варьируя эти параметры, попробуйте найти наилучший вариант (возможные значения переменных можно найти в документации Scikit-learn).

2. Насколько лучше или хуже работает MLPClassifier по сравнению с персептроном? Без нормализации данных и с нормализацией данных? Предположите, почему именно так ведут себя модели.

3. Проведите серию экспериментов с различными значениями random_state. Как случайная инициализация весов влияет на Персептрон и многослойную сеть?

4. Дополнительно постройте график распределения исходных данных.

Реализация алгоритма обратного распространения ошибки

1. Модифицируйте сеть из листинга 59. Добавьте еще один слой и восстановите работоспособность алгоритма. Критерием его работоспособности может быть стремительно уменьшающаяся ошибка.

2. Дополнительно обучите старую и новую модели на следующих данных:

```
X = [[0, 0, 1], [0.3, 1, 0],
    [1, 0.3, 0],  [0.6, 0.2, 1],
    [0.6, 0.2, 1]]
```

Проанализируйте исходные данные. Какая зависимость скрыта в них? Объясните, что произошло с качеством классификации. Почему? С чем может быть связано подобное в реальных задачах и как на это можно повлиять (если это вообще возможно)?

Регуляризация и сеть прямого распространения

1. Вернитесь к задаче из пункта «Влияние регуляризации на многослойную сеть прямого распространения». Проанализируйте результат кода при количестве нейронов, равном 100, в скрытом слое MLP. Напомним, что регуляризация – это механизм снижения больших весов модели при повышении ее сложности. Этот механизм помогает избежать переобучения. Параметр регуляризации alpha регулирует размер штрафов, которые накладываются на веса. Какой коэффициент регуляризации оптимален для данной задачи?

2. Задача сравнения персептрона, многослойной сети и регрессий. В пункте «Влияние регуляризации на многослойную сеть прямого распространения» была описана общая методика по проведению

серии экспериментов и выводу результатов (понятно, что это можно сделать не только в виде графиков, но и выводом информации в консоль или записи более детальной информации в файл\таблицу для дальнейшей обработки). Соответственно каркас этой программы можно повторно использовать во многих других экспериментах не только по анализу влияния регуляризации.

По аналогии с заданием выше сгенерируйте 3 задачи классификации со следующими параметрами: make_moons (noise=0.3, random_state=rs), make_circles (noise=0.2, factor=0.5, random_state=rs), X, y = make_classification (n_samples=500, n_features=2, n_redundant=0, n_informative=2, random_state=rs, n_clusters_per_class=1) и сравните следующие модели:

- Линейную регрессию
- Полиномиальную регрессию (например, со степенью 3)
- Гребневую полиномиальную регрессию (например, со степенью 4, alpha = 1.0)
- Персептрон
- Многослойный персептрон с десятью нейронами в скрытом слое (alpha = 0.01)
- Многослойный персептрон со ста нейронами в скрытом слое (alpha = 0.01)

Создайте отдельный файл и замените только код моделей. Постройте графики и отобразите качество моделей. В результате у вас должно получиться полотно графиков, похожее на рисунок 71 (причем в данном случае представлена только первая задача классификации на Moondataset).

Проанализируйте полученные результаты, рассчитав их при параметре random_state=25 для следующих функций\объектов:

- Perceptron
- MLPClassifier
- Ridge
- make_classification

- rng = np.random.RandomState(25)
- make_moons
- make_circles
- train_test_split

Рисунок 71. Пример работы программы по сравнению нескольких моделей на задачах moon, circle, linear

Сравнение эффективности моделей из библиотеки Keras

Вы научитесь:

- Проводить самостоятельный анализ результатов экспериментов;
- Проводить серию экспериментов;
- Интерпретировать результаты моделей;
- Подбирать соответствующие данным инструменты и параметры.

Необходимые import для выполнения работы представлены в листинге 95.

Листинг 95

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils
import matplotlib.pyplot as plt
from keras.models import model_from_json
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
```

Инструкции по выполнению представлены в листинге 96 (первые 3 шага), а также ниже. Прежде чем перейти к шагу 3, сравните качество полученных трех моделей на данных MNIST. Объясните полученные результаты. Есть ли превосходство сверточной сети перед обычной или перед полиномиальной регрессией?

Листинг 96

```
#Шаг 1. Загрузите данные MNIST
(X_train, y_train), (X_test, y_test) = mnist.load_data()
#Шаг 2. Создайте модель многослойного персептрона (на основе Keras)и сверточной
#сети (на основе Keras) и полиномиальной регрессии из Scikit-Learn.
#Для задания обычного линейного слоя в качестве первого слоя сети потребуется
#указать форму входного вектора (размер). Это можно сделать следующим образом:
Dense(128, input_dim=input_size)
#Шаг 3. Далее попробуйте эти три типа модели уже на других данных.
#Для этого загрузите и подготовьте данные с фотографиями людей с помощью
#библиотеки scikit-learn (изучите код самостоятельно):
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
# introspect the images arrays to find the shapes (for plotting)
n_samples, h, w = lfw_people.images.shape
X = lfw_people.images
# the label to predict is the id of the person
y = lfw_people.target
target_names = lfw_people.target_names
n_classes = target_names.shape[0]
print("n_classes: %d" % n_classes)
# split into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

Шаг 4. Отобразите данные, чтобы удостовериться в их корректной подготовке для дальнейшего обучения.

Шаг 5. Обучите модель многослойного персептрона, полиномиальной регрессии и сверточной сети. Попробуйте разное число нейронов и слоев в сетях (10, 20, 50, 100, 200, 500 нейронов) и (1-3 слоя). Большее число нейронов и слоев не нужно.

Замечание 1: Конечно же, не следует обучать модели сперва на MNIST, а потом эти же модели (конкретные экземпляры) на фотографиях. Это совсем разные задачи, и при малых количествах слоев и таких выборках вряд ли произойдет хорошее обобщение. Имеется в виду, что те же три типа моделей теперь надо сравнить на фотографиях.

Замечание 2: Если на CPU обучение идет слишком долго, то возьмите не всю обучающую выборку, а лишь половину или одну треть.

В заключение сравните качество обученных моделей на данных фотографий. Объясните полученные результаты и ответьте а вопросы:

- Почему получилось именно так?
- На что способны и не способны используемые модели?
- Чем отличаются данные из первой части задания от данных из второй части задания?

Работа с библиотекой OpenCV

Докажите гипотезу, сформулированную по завершению предыдущего задания. Что нужно сделать с данными MNIST, чтобы работающая ранее модель стала давать на них плохие результаты? Для этого вам может понадобиться библиотека OpenCV для Python. Которая достаточно легко устанавливается. Смотрите раздел «**Installing OpenCV from prebuilt binaries»** по следующей ссылке: [65].

Согласно Википедии [1]: «OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно

использоваться в академических и коммерческих целях – распространяется в условиях лицензии BSD.

Проект OpenCV возник в 1999 году в рамках исследования Intel по высокопроизводительным приложениям по обработке 3D сцен. Это крайне мощная библиотека и стандарт де-факто для обработки изображений. Сферы применения OpenCV:

Инструмент выделения 2D и 3D признаков;

Трехмерная навигация;

Системы распознавания лиц;

Системы распознавания жестов;

Человеко-компьютерные интерфейсы;

Мобильные роботы;

Распознавание эмоций;

Идентификация объектов;

Сегментация и распознавание;

Стереозрение;

Восстановление поверхности;

Отслеживание движения;

Дополненная реальность.

Кроме того, для поддержки некоторых вышеуказанных областей, OpenCV включает в себя библиотеки статистической обработки данных и машинного обучения:

Решающие деревья;

Алгоритм максимального правдоподобия;

Алгоритм k-ближайших соседей;

Наивный Байесовский классификатор;

Искусственные нейронные сети;

Случайные леса;

Метод опорных векторов.

В библиотеке OpenCV есть методы для многих видов деформаций изображений. Для базовой проверки способностей ИНС на двумерных изображениях вполне хватит следующих:

Повороты;

Линейные сдвиги;

Масштабирование».

Ниже (в листинге 97) приведен код, который позволяет выполнить преобразование исходного изображения images[i]. Каждый метод преобразования cv2 возвращает новое изображение. Для поворотных и линейных сдвигов используются специальные матрицы деформации. Код для их инициализации также приведен ниже. Недостающие переменные – это коэффициенты и константы. Определите их самостоятельно.

Листинг 97

```
import cv2
# setrotation
rot_Matrix = cv2.getRotationMatrix2D((cols / 2, rows/ 2), ang, 1)
cv2.warpAffine(images[i], rot_Matrix, (cols, rows))
# set scaling
cv2.resize(images[i], dsize=(rows, cols), fx=scale, fy=scale,
interpolation=cv2.INTER_CUBIC)
# set translation
trans_Matrix = np.float32([[1, 0, tx], [0, 1, ty]])
cv2.warpAffine(images[i], trans_Matrix, (cols, rows))
```

После преобразования части изображений проведите соответствующие эксперименты с разными моделями. Измерьте качество получившихся моделей. Вероятно, придется проделать много экспериментов с различными параметрами, чтобы добиться такой ситуации, когда результаты будут объяснять теорию.

Нечеткая логика

1. На языке Python разработайте скрипт, позволяющий задать нечеткое множество с треугольной функцией принадлежности и отобразить его название, параметры, а также степень принадлежности вводимого пользователем объекта.

2. На языке Python разработайте скрипт, позволяющий задать нечеткое множество с трапециевидной функцией принадлежности и отобразить его параметры, а также степень принадлежности вводимого пользователем объекта.

3. На языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

4. На языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

5. На языке Python разработайте скрипт, позволяющий выполнить операцию объединения заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – объединение данных нечетких множеств.

6. На языке Python разработайте скрипт, позволяющий выполнить операцию объединения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – объединение данных нечетких множеств.

7. На языке Python разработайте скрипт, позволяющий выполнить операцию дополнения, заданного пользователем нечеткого множества с треугольной функцией принадлежности. Входными данными будут параметры функции принадлежности и четкие объекты множества. Выходными – дополнение нечеткого множества.

8. На языке Python разработайте скрипт, позволяющий выполнить операцию дополнения, заданного пользователем нечеткого множества с трапециевидной функцией принадлежности. Входными данными будут параметры функции принадлежности и четкие объекты множества. Выходными – дополнение нечеткого множества.

9. На языке Python разработайте скрипт, позволяющий выполнить операцию импликации заданных пользователем нечетких множеств с треугольными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – результат импликации данных нечетких множеств. Причем результат вывести через лингвистические переменные. Импликацию моделировать минимумом.

10. На языке Python разработайте скрипт, позволяющий выполнить операцию импликации заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – результат импликации данных нечетких множеств. Импликацию моделировать минимумом.

Генетические алгоритмы

1. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N наименований продуктов, для каждого из которых известно m характеристик. Необходимо получить самый дешевый рацион из k наименований, удовлетворяющий заданным медицинским нормам для каждой из m характеристик.

2. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано n пунктов производства продуктов и k городов, которые в них нуждаются. Каждый город может потребить x продуктов, а каждый пункт произвести y продуктов. Необходимо получить оптимальный маршрут, так, чтобы все города получили нужный им объем продуктов без сильного его превышения, а транспортные расходы были минимальными.

3. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N наименований продуктов, для каждого из которых известно m характеристик. Необходимо получить самый лучший по характеристикам рацион из k наименований, удовлетворяющий заданным ценовым рамкам. Лучшим считается рацион с минимальным отклонением от нормы.

4. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано n пунктов производства продуктов и k городов, которые в них нуждаются. Каждый город может потребить x продуктов, а каждый пункт произвести y продуктов. Необходимо получить оптимальный маршрут, так, чтобы все города получили нужный им объем продуктов с минимальным его превышением, а транспортные расходы укладывались в определенные рамки.

5. На языке Python разработайте скрипт, который с помощью генетического алгоритма решает следующую задачу. Дано N полей и k культур для посева. Для каждого поля известна характеристика урожайности каждой из k культур, а для каждой культуры – его закупочная стоимость. Необходимо получить самый лучший урожай за наименьшую стоимость.

Нечеткая кластеризация объектов

1. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере заработной платы n людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

2. На языке Python разработайте скрипт, кластеризующий загруженные данные о возрасте n людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

3. На языке Python разработайте скрипт, кластеризующий загруженные данные о стоимости n автомобилей на определенные им

кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

4. На языке Python разработайте скрипт, кластеризующий загруженные данные о росте *n* людей на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

5. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере затрат на производство *n* продуктов на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

6. На языке Python разработайте скрипт, кластеризующий загруженные данные о длительности перелетов до *n* пунктов на определенные им кластеры, обозначенные заданными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

7. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере затрат на связь среди *n* отделов на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

8. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере площадей *n* квартир на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

9. На языке Python разработайте скрипт, кластеризующий загруженные данные о размере урожая *n* сельскохозяйственных культур на определенные им кластеры, обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

10. На языке Python разработайте скрипт, кластеризующий загру-женные данные о весе *n* людей на определенные им кластеры,

обозначенные определенными в программе лингвистическими метками. Максимальное количество меток задать самостоятельно.

Анализ временных рядов

1. *Лингвистические шкалы.* Разработать скрипт на Python, позволяющий задавать с использованием функций принадлежности лингвистическую шкалу для решения задачи по варианту (таблица 14). Количество оценок в шкале и параметры функций принадлежности должны задаваться по умолчанию, если в заданном пользователем файле не указано иное. Осуществить графическое отображение функций принадлежности меток шкалы, выделив каждую отдельным цветом. Предусмотреть загрузку файла с данными и оценку их по построенной шкале.

Таблица 14. Варианты для задачи «Лингвистические шкалы»

Вариант	**Функция принадлежности**	**Назначение шкалы**
1.	Треугольная	Оценка эффективности продаж
2.	Трапециевидная	Оценка эффективности операций с валютой
3.	Треугольная	Оценка прибыли
4.	Треугольная	Оценка затрат на оплату труда
5.	Трапециевидная	Оценка рентабельности
6.	Треугольная	Оценка объемов продаж
7.	Трапециевидная	Оценка объемов закупок
8.	Треугольная	Оценка загруженности сервера и сети
9.	Треугольная	Оценка уровня зарплаты
10.	Трапециевидная	Оценка убытков
11.	Треугольная	Оценка эффективности продаж
12.	Трапециевидная	Оценка эффективности операций с валютой
13.	Треугольная	Оценка прибыли

Окончание таблицы 14

Вариант	Функция принадлежности	Назначение шкалы
14.	Треугольная	Оценка затрат на оплату труда
15.	Трапециевидная	Оценка рентабельности
16.	Треугольная	Оценка объемов продаж
17.	Трапециевидная	Оценка объемов закупок
18.	Треугольная	Оценка загруженности сервера и сети
19.	Треугольная	Оценка уровня зарплаты
20.	Трапециевидная	Оценка убытков
21.	Треугольная	Оценка эффективности продаж
22.	Трапециевидная	Оценка эффективности операций с валютой
23.	Треугольная	Оценка прибыли
24.	Треугольная	Оценка затрат на оплату труда
25.	Трапециевидная	Оценка рентабельности
26.	Треугольная	Оценка объемов продаж
27.	Трапециевидная	Оценка объемов закупок
28.	Треугольная	Оценка загруженности сервера и сети
29.	Треугольная	Оценка уровня зарплаты
30.	Трапециевидная	Оценка убытков

2. *Нечеткие временные ряды и ряды нечетких тенденций*. Реализовать скрипт на языке Python, позволяющий загружать четкий временной ряд величины из задачи по лингвистическим шкалам (задача «Лингвистические шкалы»). С помощью построенной в предыдущей работе шкалы (задача «Лингвистические шкалы») построить из четкого временного ряда временной ряд нечетких значений и нечетких тенденций. Осуществить графическое отображение нечетких меток шкалы и нечетких тенденций, выделив каждую отдельным цветом.

3. *Кластеризация и лингвистические шкалы.* Язык реализации Python. Построить шкалу из задачи по лингвистическим шкалам (работа «Лингвистические шкалы») с помощью алгоритма fcm-клас-

теризации. Количество кластеров задается по умолчанию, но может редактироваться пользователем посредствам загрузки из файла. Каждый кластер – метка на шкале с присвоенным ему лингвистическим значением либо по умолчанию, либо пользователем. Осуществить графическое отображение меток шкалы, выделив каждую отдельным цветом.

3. *Прогнозирование значения временного ряда с помощью нейронной сети.* Реализовать прогноз следующего значения четкого временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» с помощью нейронной сети, но НЕ рекуррентного типа. Графически отобразить реальный ряд и прогнозное значение, а также проиллюстрировать результат тестового прогноза.

4. *Прогнозирование значения временного ряда с помощью модифицированного метода Сонга.* Реализовать прогноз следующего значения временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» с помощью метода, описанного в пункте «Пример моделирования временного ряда в нечетком подходе». Графически отобразить реальный ряд и прогнозное значение, а также проиллюстрировать результат тестового прогноза.

5. *F-преобразование.* Для временного ряда из работы «Нечеткие временные ряды и ряды нечетких тенденций» выделить тренд методом F-преобразования.

Работа с рекуррентными сетями

1. Модифицируйте код из листинга 91: поэкспериментируйте с разной структурой сети, разным количеством точек, по которым строится прогноз, и с разным количеством LSTM-блоков. Оцените качество работы сети в разных экспериментах. Интерпретируйте результаты.

2. Решите задачу «Прогнозирование значения временного ряда с помощью нейронной сети» из блока заданий «Анализ временных рядов» с помощью рекуррентной сети. Сравните и интерпретируйте результаты.

3. Объясните работу кода из листинга 98.

Листинг 98

```
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.utils import np_utils

numpy.random.seed(7)
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
char_to_int = dict((c, i) for i, c in enumerate(alphabet))
int_to_char = dict((i, c) for i, c in enumerate(alphabet))
seq_length = 1
dataX = []
dataY = []
for i in range(0, len(alphabet) - seq_length, 1):
  seq_in = alphabet[i:i + seq_length]
  seq_out = alphabet[i + seq_length]
  dataX.append([char_to_int[char] for char in seq_in])
  dataY.append(char_to_int[seq_out])
  print seq_in, '->', seq_out
X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
X = X / float(len(alphabet))
y = np_utils.to_categorical(dataY)

model = Sequential()
model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, y, nb_epoch=500, batch_size=1, verbose=2)
scores = model.evaluate(X, y, verbose=0)
print("Model Accuracy: %.2f%%" % (scores[1]*100))
for pattern in dataX:
  x = numpy.reshape(pattern, (1, len(pattern), 1))
 x = x / float(len(alphabet))
  prediction = model.predict(x, verbose=0)
  index = numpy.argmax(prediction)
  result = int_to_char[index]
  seq_in = [int_to_char[value] for value in pattern]
  print seq_in, "->", result
```

4. Проведите два эксперимента, модифицируя код из листинга 92 следующим образом:

1. Изменив значение seq_length, сделав его равным 3 и изменив форму входных данных следующим образом:

X = numpy.reshape(dataX, (len(dataX), 1, seq_length))

2. Изменив значение seq_length, сделав его равным 3 и изменив форму входных данных следующим образом:

X = numpy.reshape(dataX, (len(dataX), 1, seq_length))

Интерпретируйте полученные результаты.

Заключение

Мы рассмотрели модели и алгоритмы, используемые в сфере машинного обучения для анализа данных, а также способы их реализации на языке Python с использованием библиотек Keras, Theano и других.

В настоящее время машинное обучение и область анализа данных являются весьма перспективными направлениями в ИТ. Задачей данной книги было познакомить вас с азами этого направления, рассмотрев наиболее важные модели, алгоритмы и их реализацию. Однако, чтобы стать специалистом в сфере анализа данных, недостаточно просто знать модели и инструменты. Необходимо понимать границы их применений и преимущества одного перед другим.

Данное учебное пособие лишь приоткрывает дверцу в мир машинного обучения, освещая небольшую его часть. Если же после работы с книгой вы поняли, что анализ данных – крайне интересная вещь, то вам непременно нужно двигаться дальше. Причем не только в изучении моделей, методов, алгоритмов и инструментов их реализации (что вы можете сделать, изучив приведенные в библиографическом списке источники), но и в совершенствовании своего мышления. Ведь пока искусственный интеллект еще не изобретен, всю семантическую нагрузку задач машинного обучения берет на себя разработчик-человек. Поэтому он непременно должен обладать острым аналитическим умом.

Перефразируя Шерлока Холмса, напомним, что «развитый мозг – это то, что всегда будет в цене». В связи с этим, закончим пособие поже-ланием: оттачивайте свой разум, покоряйте новые вершины, не останавливайтесь на достигнутом!

ГЛОССАРИЙ

DataMining – интеллектуальный анализ данных.

TimeSeries DataMining – интеллектуальный анализ временных рядов.

Аномалия временного ряда – новый, не типичный паттерн временного ряда.

Апостериорная вероятность – назначенная событию вероятность при условии наличия знаний, поддерживающих его наступление и полученных опытным путем.

Априорная вероятность – назначенная событию вероятность, при условии отсутствия знаний, поддерживающих его наступление.

Временной ряд (ВР) – последовательность упорядоченных в равноотстоящие моменты времени пар (момент_времени, значение_характеристики).

Генетические алгоритмы – адаптивные методы поиска, использующиеся для решения задач функциональной оптимизации.

Градиент – вектор, указывающий направление наибольшего возрастания некоторой величины, значение которой меняется в скалярном поле.

Дефаззификация – получение оптимального четкого значения по агрегированному нечеткому понятию.

Дисперсия – мера разброса элементов выборки относительно ее математического ожидания.

Задача классификации – распределение некоторого множества объектов по заданному множеству групп (классов).

Задача кластеризации – разделение некоторого множества объектов на непересекающиеся группы (кластеры) таким образом, чтобы каждая группа состояла из схожих объектов, а объекты разных кластеров существенно отличались.

Задача регрессии – приближение неизвестной целевой зависимости на некотором множестве данных.

Задача таксономии – задача построения древообразной иерархической структуры, упорядочивающей исходные данные.

Закон распределения – функция, определяющая для выборки вероятность попадания в некоторый интервал или вероятность получения определенного значения.

Знание – совокупность утверждений о закономерностях и свойствах процессов и явлений, а также связывающих их правил логического вывода и правил использования их при принятии решений.

Индексирование объектов временного ряда – построение индексов для эффективного выполнения запросов к базам данных ВР.

Классификация объектов временного ряда – назначение ВР или их паттернам одного из заранее определенных классов.

Кластеризация объектов временного ряда – поиск группировок ВР или их паттернов.

Комбинированная модель прогнозирования – модель прогнозирования, состоящая из нескольких индивидуальных (частных) моделей, называемых базовым набором моделей.

Концепт временной продолжительности – присутствие определенного паттерна или признака ВР на определенном интервале времени.

Концепт нечеткости ВР – нечеткость выраженности темпоральных событий и отношений.

Концепт одновременности ВР – совпадение во времени темпоральных событий (паттернов различных ВР).

Концепт очередности ВР – порядок следования паттернов ВР во времени.

Кроссовер – операция в генетическом алгоритме, позволяющая хромосомам обмениваться между собой своими частями.

Лингвистическая переменная – это переменная, значениями которой являются слова или высказывания естественного или искусственного языка.

Математическое ожидание – среднее значение вероятностных элементов выборки.

Медиана – число, характеризующее выборку по среднему из ее значений.

Метод градиентного спуска – нахождение локального экстремума (минимума или максимума) функции путем движения вдоль градиента в направлении наискорейшего спуска, задаваемого антиградиентом.

Мутация – операция в генетическом алгоритме, произвольным образом изменяющая хромосому.

Недообучение – ситуация, когда алгоритм при обучении с учителем не дает удовлетворительно малой средней ошибки на обучающем множестве.

Нечеткая логика – логика, оперирующая нечеткими высказываниями и рассуждениями на базе частичной истинности.

Нечеткий временной ряд (НВР) – упорядоченная в равноотстоящие моменты времени последовательность наблюдений над некоторым процессом, состояния которого изменяются во времени, если значение состояния процесса в каждый момент времени может быть выражено с помощью нечеткой метки .

Нечеткое множество – совокупность объектов, принадлежащих ему с определенной степенью.

Нормализация данных – процесс масштабирования вектора каждого признака к такому виду, что вектор будет иметь единичную норму (при этом есть разные способы оценки\подсчета нормы).

Обобщающая способность – свойство модели отражать исходные данные в требуемые результаты (X –> Y) на всем множестве исходных данных (во всех сценариях, а не только на тренировочных примерах).

Обучение – способность алгоритма при решении задач некоторого класса выдавать на некотором опыте лучшие результаты в смысле заданной меры качества при предъявлении нового опыта.

Ошибка – численно выраженная разница между ответом модели и требуемым (реальным) значением.

Переобучение – свойство натренированного алгоритма на объектах тренировочной выборки давать существенно меньшую вероятность ошибки, чем на объектах тестовой.

Пространство признаков – это N-мерное пространство, где N – число измеряемых характеристик объектов, выделенное для конкретной задачи.

Регуляризация – добавление некоторой дополнительной информации к условию минимизации ошибки.

Резюмирование временного ряда – формирование краткого описания ВР, содержащего существенные черты с точки зрения решаемой задачи.

Сегментация временного ряда – разбиение временного ряда на значимые сегменты.

Система нечеткого логического вывода – модель, описывающая поведение систем на естественном (или близком к естественному) языке в виде приближенных рассуждений на основе композиционного правила вывода.

Среднеквадратическое отклонение – величина, характеризующая рассеивание значений выборки относительно ее математического ожидания.

Стандартизация данных – процесс приведения вектора каждого признака к такому виду, что его математическое ожидание станет нулевым, а дисперсия – единичной.

Степень принадлежности – значение, задаваемое функцией принадлежности и находящееся в интервале от 0 до 1.

Фаззификация – процесс установки соответствия между четким значением x и нечетким f через функцию принадлежности.

Функция принадлежности – функция, отображающая базовое значение x и нечеткое f в интервал от 0 до 1.

Хромосома (в генетическом алгоритме) – битовая строка, описывающая решение.

Частотный анализ временного ряда – поиск часто проявляющихся паттернов ВР.

Энтропия – мера неупорядоченности системы.

Предметный указатель

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа». URL: http://ru.wikipedia.org/wiki/, (режим доступа – свободный), (дата обращения: 28.03.2017).

2. [Электронный ресурс]: Материалы открытого курса по машинному обучению от компании ODS. URL: https://habrahabr.ru/company/ods/blog/325654/, (режим доступа – свободный), (дата обращения: 28.03.2017).

3. Воронина, В. В. Разработка приложений для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина, В. С. Мошкин. – Ульяновск : УлГТУ, 2015. – 162 с.

4. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Переобучение, (режим доступа – свободный), (дата обращения: 28.03.2017).

5. [Электронный ресурс]: Статья по регуляризации. URL: https://ru.wikipedia.org/wiki/Регуляризация_(математика), (режим доступа – свободный), (дата обращения: 28.03.2017).

6. [Электронный ресурс]: Материалы сайта MSDN. URL: https://msdn.microsoft.com/ru-ru/magazine/dn904675.aspx, (режим доступа – свободный), (дата обращения: 28.03.2017).

7. [Электронный ресурс]: Материалы сайта machinelearning. URL:http://www.machinelearning.ru/wiki/index.php?title=Нормальное_распределение, (режим доступа – свободный), (дата обращения: 28.03.2017).

8. Паклин, Н. Б. Глава 9, // Бизнес-аналитика: от данных к знаниям : учебное пособие / Н. Б. Паклин, В. И. Орешков. – 2-е изд.. – СПб. : Питер, 2013. – С. 444-459.

9. [Электронный ресурс]: Ю. Лаходюк, Энтропия и деревья принятия решений. URL: https://habrahabr.ru/post/171759/, (режим доступа – свободный), (дата обращения: 28.03.2017).

10. [Электронный ресурс]: Статья по деревьям решений. URL: http://ru.wikipedia.org/wiki/Дерево_принятия_решений, (режим доступа – свободный), (дата обращения: 28.03.2017).

11. Клячкин, В. Н. Статистические методы анализа данных / В. Н. Клячкин, Ю. Е. Кувайскова, В. А. Алексеева. – М. : Финансы и статистика, 2016. – 240 с.

12. Клячкин, В. Н. Статистические методы в управлении качеством: компьютерные технологии / В. Н. Клячкин. – М. : Финансы и статистика, ИНФРА-М, 2009. – 304 с.

13. Валеев, С.Г. Регрессионное моделирование при обработке наблюдений / С. Г. Валеев. – М. : Наука, 1991. – 272 с.

14. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа»: Статья под названием «Робастные методы». URL: http://ru.wikipedia.org/wiki/Робастность, (режим доступа – свободный), (дата обращения: 08.07.2017).

15. Zadeh, A. Lotfi. Fuzzy Sets / Lotfi A. Zadeh // Information and Control. – 1965.

16. Заде, Л. А. Основы нового подхода к анализу сложных систем и процессов принятия решений / Л. А. Заде // Математика сегодня. – М. : Знание, 1974. – С. 5-49.

17. Штовба, С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. – М. : Горячая линия – Телеком, 2007. – 288 с.

18. Ярушкина, Н. Г. Основы теории нечетких и гибридных систем : учебное пособие / Н. Г. Ярушкина. – М. : Финансы и статистика, 2004. – 320 с.

19. Ярушкина, Н. Г. Интеллектуальный анализ временных рядов : учебное пособие / Н. Г. Ярушкина, Т. В Афанасьева., И. Г. Перфильева. – М. : ИД «ФОРУМ» : ИНФРА-М, 2012. – 160 с. – (Высшее образование).

20. Song, Q. Fuzzy time series and its models / Q. Song, B. Chissom // Fuzzy Sets and Systems. – №54 (1993) – P. 269-277.

21. [Электронный ресурс] Дегтярев, К. Ю. Применение специализированных компьютерных программ и методов, основанных на нечетких временных рядах для краткосрочного прогнозирования USB/RUB котировок / К. Ю. Дегтярев. URL: http://www.exponenta.ru/educat/news/degtyarev/paper.pdf, (режим доступа – свободный), (дата обращения: 08.07.2017).

22. Batyrshin, I. Perception Based Time Series Data Mining for Decision Making / I. Batyrshin // IFSA’07 Fuzzy Logic, Soft Computing and Computational Intelligence.

23. [Электронный ресурс]: Материалы IRAFM-2015. URL: http://irafm.osu.cz/cif2015/main.php?c=Static&page=results, (режим доступа – свободный), (дата обращения: 08.07.2017).

24. Новак, В. Математические принципы нечеткой логики / В. Новак, И. Перфильева, И. Мочкорж; пер. с англ.; под ред. А. Н. Аверкина. – М. : ФИЗМАТЛИТ, 2006.

25. Афанасьева Т. В. Модель ACL-шкалы для генерации лингвистических оценок в принятии решений / Т. В. Афанасьева // Вопросы современной науки и практики. Университет им. В. И. Вернадского. Т. 2. Серия «Технические науки». – 2008. – №4 (14). – С. 91-97.

26. [Электронный ресурс]: Саймон Хайкин - Нейронные сети. Полный курс. URL: http://neuralnetworksanddeeplearning.com, (режим доступа – свободный), (дата обращения: 08.07.2017).

27. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Модель_Мак-

Каллока_Питтса, (режим доступа – свободный), (дата обращения: 08.07.2017).

28. [Электронный ресурс]: Статья Логика мышления. Часть 3. Персептрон, сверточные сети. URL: https://geektimes.ru/post/214317/, (режим доступа – свободный), (дата обращения: 08.07.2017).

29. [Электронный ресурс]: Материалы сайта machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Самоорганизующаяся_карта_Кохонена, (режим доступа – свободный), (дата обращения: 08.07.2017).

30. [Электронный ресурс]: Материалы свободной энциклопедии «Википедиа»: Python. URL: https://ru.wikipedia.org/wiki/Python, (режим доступа – свободный), (дата обращения: 08.07.2017).

31. [Электронный ресурс]: Материалы по синтаксису Python. URL: https://habrahabr.ru/post/31180/, (режим доступа – свободный), (дата обращения: 08.07.2017).

32. [Электронный ресурс]: Официальный сайт разработчиков Anaconda. URL: https://www.continuum.io/Downloads, (режим доступа – свободный), (дата обращения: 08.07.2017).

33. [Электронный ресурс]: Официальный сайт PyCharm. URL: https://www.jetbrains.com/pycharm/download/#section=windows, (режим доступа – свободный), (дата обращения: 08.07.2017).

34. [Электронный ресурс]: Официальный сайт SciPy. URL: https://scipy.org/ (режим доступа – свободный), (дата обращения: 08.07.2017).

35. [Электронный ресурс]: Покоряем Python – уроки для начинающих. Функции lambda. URL: https://pythlife.blogspot.ru/2012/11/lambda.html (режим доступа – свободный), (дата обращения: 08.07.2017).

36. [Электронный ресурс]: Документация по DataFrame. URL: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Data

Frame.html (режим доступа – свободный), (дата обращения: 08.07.2017).

37. [Электронный ресурс]: Уроки по Pandas. URL: https://bitbucket.org/hrojas/learn-pandas (режим доступа – свободный), (дата обращения: 08.07.2017).

38. [Электронный ресурс]: Документация по Scikit-learn. URL: http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-normalization (режим доступа – свободный), (дата обращения: 08.07.2017).

39. [Электронный ресурс]: Сервис Kaggle. Массив данных о пассажирах Титаника URL: https://www.kaggle.com/prkukunoor/TitanicDataset (режим доступа – свободный), (дата обращения: 08.07.2017).

40. [Электронный ресурс]: Документация по Scikit-learn:Ridge. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge (режим доступа – свободный), (дата обращения: 08.07.2017).

41. [Электронный ресурс]: Ссылка для скачивания упоминаемых в книге файлов с данными. URL: https://drive.google.com/drive/folders/0B5yyS8oSQ0FDelpKRXg3c3lKVlU (режим доступа – свободный), (дата обращения: 20.07.2017).

42. [Электронный ресурс]: Документация по Scikit-learn:TFIDFVectorizer. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer (режим доступа – свободный), (дата обращения: 08.07.2017).

43. [Электронный ресурс]: Документация по Scikit-learn: DictVectorizer. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html (режим доступа – свободный), (дата обращения: 08.07.2017).

44. [Электронный ресурс]: Документация по Scikit-learn: LogisticRegression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (режим доступа – свободный), (дата обращения: 08.07.2017).

45. [Электронный ресурс]: Документация по Scikit-learn: LinearRegression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (режим доступа – свободный), (дата обращения: 08.07.2017).

46. [Электронный ресурс]: Документация по Scikit-learn:Lasso. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (режим доступа – свободный), (дата обращения: 08.07.2017).

47. [Электронный ресурс]: Документация по Scikit-learn: MinMaxScaler. URL: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html(режим доступа – свободный), (дата обращения: 08.07.2017).

48. [Электронный ресурс]: Функции map и zip и lambda. Python. URL: http://ninjaside.info/blog/ru/funkcii-map-i-zip-i-lambda-python/ (режим доступа – свободный), (дата обращения: 08.07.2017).

49. [Электронный ресурс]: Документация по Scikit-learn: RandomizedLasso. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RandomizedLasso.html (режим доступа – свободный), (дата обращения: 08.07.2017).

50. [Электронный ресурс]: Документация по Scikit-learn:RFE. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html(режим доступа – свободный), (дата обращения: 08.07.2017).

51. [Электронный ресурс]: Документация по Scikit-learn: RandomForestRegressor. URL: http://scikit-learn.org/stable/modules/

generated/sklearn.ensemble.RandomForestRegressor.html (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

52. [Электронный ресурс]: Документация по Scikit-learn: f_regression. URL: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html(режим доступа – свобод-ный), (дата обращения: 08.07.2017).

53. [Электронный ресурс]: Документация по Scikit-learn: Pipeline. URL: http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html(режим доступа – свободный), (дата обращения: 08.07.2017).

54. [Электронный ресурс]: Документация по Scikit-learn: Cross-validation. URL: http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation(режим доступа – свободный), (дата обращения: 08.07.2017).

55. [Электронный ресурс]: Документация по Scikit-learn: Cross_val_score. URL: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html, (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

56. [Электронный ресурс]: Документация по Scikit-learn: Perceptron. URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html, (режим доступа – свободный), (дата обращения: 08.07.2017).

57. [Электронный ресурс]: Документация по Scikit-learn: MLPClassifier. URL: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, (режим доступа – свобод-ный), (дата обращения: 08.07.2017).

58. [Электронный ресурс]: Официальный сайт NVIDIA. Проверка видеокарты. URL: https://developer.nvidia.com/cuda-gpus (режим доступа – свободный), (дата обращения: 08.07.2017).

59. [Электронный ресурс]: Официальный сайт NVIDIA. Скачивание CUDA. URL: https://developer.nvidia.com/cuda-downloads (режим доступа – свободный), (дата обращения: 08.07.2017).

60. [Электронный ресурс]: Официальный сайт NVIDIA. Скачивание CUDANN. URL: https://developer.nvidia.com/rdp/cudnn-download (режим доступа – свободный), (дата обращения: 08.07.2017).

61. [Электронный ресурс]: Документация по билиотеке Keras. URL: https://keras.io/layers/core/, (режим доступа – свободный), (дата обращения: 08.07.2017).

62. [Электронный ресурс]: Материалы сайта machinelearningmastery. URL: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/, (режим доступа – свободный), (дата обращения: 08.07.2017).

63. [Электронный ресурс]: Статья по реализации генетического алгоритма на Python. URL: http://easydan.com/arts/2016/genetic-optimization/, (режим доступа – свободный), (дата обращения: 08.07.2017).

64. Воронина, В. В. Разработка веб-сервисов для анализа слабоструктурированных информационных ресурсов : учебное пособие / В. В. Воронина. – Ульяновск : УлГТУ, 2016. – 166 с.

65. [Электронный ресурс]: Документация «Installing OpenCV from prebuilt binaries». URL: http://docs.opencv.org/3.2.0/d5/de5/tutorial_py_setup_in_windows.html, (режим доступа – свободный), (дата обращения: 08.07.2017).

Учебное электронное издание

ВОРОНИНА Валерия Вадимовна
МИХЕЕВ Александр Вячеславович
ЯРУШКИНА Надежда Глебовна
СВЯТОВ Кирилл Валерьевич

ТЕОРИЯ И ПРАКТИКА МАШИННОГО ОБУЧЕНИЯ

Учебное пособие

Редактор Н. А. Евдокимова

ЛР № 020640 от 22.10.97.

ЭИ № 1004. Объем данных 3,6 Мб.

Печатное издание
Подписано в печать 08.11.2017.
Усл. печ. л. 16.97. Тираж 100 экз. Заказ 932.

Ульяновский государственный технический университет,
432027, г. Ульяновск, ул. Сев. Венец, д. 32.

ИПК «Венец», УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.
Тел.: (8422) 778-113
E-mail: venec@ulstu.ru
venec.ulstu.ru

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.05 Архитектурное моделирование в проектировании АС

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

П.И. Соснин

Архитектурное моделирование автоматизированных систем

Учебное пособие

2008

УДК 319.766 (075)

Рецензенты: доктор технических наук,

Соснин П.И.
С66 Архитектурное моделирование автоматизированных систем: учебное пособие/ П.И, Соснин. –

Учебное пособие предназначено для студентов, обучающихся по направлению «Информатика и вычислительная техника», а также по специальностям «Вычислительные машины, комплексы, системы и сети» и «Информационные системы и технологии». Пособие может быть также использовано студентами других специальностей, профиль которых связан с разработками автоматизированных систем, интенсивно использующих программное обеспечение.

УДК 319.766 (75)
ББК Я7

Содержание

Оглавление

Введение

В соответствии со стандартами 34-ой серии автоматизированные системы (АС) представляют собой «организационно-технические системы, обеспечивающие выработку решений на основе автоматизации информационных процессов в различных сферах деятельности или в их сочетании». В современных международных стандартах (например, в стандарте IEEE 1471) такой тип систем принято называть «software-intensive systems» («системами, интенсивно использующими программное обеспечение»), что указывает на «существенное влияние программного обеспечения ПО, входящего в их состав, на проектирование, конструирование, материализацию и эволюцию АС»[89].

В разработках АС накоплен определённый опыт, который позволяет считать такой вид деятельности одним из наиболее сложных и непредсказуемых видов работ, которые слишком часто завершаются с превышением запланированного на них времени и/или бюджета. Кроме того, разработки АС часто завершаются в более бедной версии результата или же прекращаются без достижения каких-либо положительных результатов. Наиболее убедительно и с попытками выявления причин эта особенность разработок АС отражена в отчётах Standish Group, одной из наиболее известных экспертных компаний по проблемам информационных технологий [70].

К числу основных причин такого положения дел, которое по основным составляющим сохраняется по настоящее время [15], относятся:

- недостаточное вовлечение в разработку пользователей и других лиц, заинтересованных в создании АС;

- неполнота требований и спецификаций;

- неуправляемое (и часто из-за субъективных причин) изменение требований и спецификаций по ходу разработки АС;

- недостаточная автоматизированная поддержка исполняемых действий;

- использование «незрелых» (слабо проверенных на практике и не дающих устойчивые позитивные эффекты) технологий;

- недостаточность выделенных или привлекаемых ресурсов;

- нереалистичные ожидания от разработки;

- использование неявных и/или смутных целей;

- нереальные временные рамки исполнения разработок.

Опасное влияние указанных причин и ряда других причин на успешность разработок АС обусловлено, в первую очередь, их сложностью с позиций программного обеспечения. Разработка такого обеспечения для конкретной АС обычно носит длительный, дорогостоящий и уникальный характер, что не способствует целенаправленному накоплению опыта подобных разработок. По сути дела в разработках АС создаются их образцы, в каждом из которых достигнута определённая степень успешности.

Степень успешности разработки конкретной АС адекватно оценивается и подтверждается только в процессе её практического использования. Успешные АС используются как источники разнородных образцов для их повторного использования, в первую очередь как источники образцов для проектных решений при разработке программного обеспечения. Среди таких образцов важное место занимают образцы архитектурных проектных решений, без применения которых немыслима разработка программного обеспечения современных АС.

Разработка архитектуры и включение её в процесс разработки АС способствует снижению стоимости разработки (за счёт улучшения коммуникативного взаимодействия между разработчиками, открывает возможность более ранней оценки альтернатив и рисков), снижает время выхода на рынок (за счёт возможности повторного использования предыдущих решений и управляемой эволюции продукта), снижает стоимость сопровождения (за счёт объединения в группы изменений, поддающихся предвидению); способствует улучшению качества продукта и приводит к другим позитивным эффектам (раскрываются ниже).

Принципиальная роль архитектуры АС была осознана всего лишь около 15 лет назад. За эти 15 лет в этой предметной деятельности было создано так много, что этот период среди специалистов по разработкам АС получил название «золотого века архитектуры АС». Одно из достижений этого века – отдельная

профессиональная дисциплина (самостоятельная предметная область) и учебная университетская дисциплина, часто с названиями «Архитектура автоматизированных систем» или «Архитектура программных систем».

Во втором издании учебного пособия раскрывается специфика предметной области «Архитектура автоматизированных систем» по зарубежным (в основном) и российским источникам, включающим стандарты, монографии, статьи и отчёты, представленным в Интернете. Отбор материалов проводился в основном по двум-трём первым страницам Yandex, Rumbler и Google для ключей доступа, раскрывающих архитектуру программного обеспечения и архитектуру систем, интенсивно использующих программное обеспечение. Для предмета пособия в книге используется общий термин «архитектура».

Учебный материал пособия распределён по четырём главам. Содержание трёх первых глав идентично содержанию первого издания, а четвёртая глава содержит информацию о совершенствовании нормативной базы архитектурного моделирования за последнее десятилетие.

В первой главе определяется класс «автоматизированных систем, интенсивно использующих программное обеспечение», с акцентом на специфику архитектуры таких систем. Приводится ряд определений архитектуры с учётом её значимости, раскрываются место и роль архитектуры как формы концептуального существования АС. Представляется ретроспектива исследований и разработок в области архитектуры АС за последние 15 лет.

Во второй главе внимание акцентируется на архитектурных образцах, стандартах и архитектурных концептуальных схемах. Раскрывается представление архитектуры АС в форме системы архитектурных видов, согласованных с интересами групп лиц, заинтересованных в разработке АС. Обобщённо демонстрируются архитектурные схемы Дж. Захмана, DoDAF, MoDAF, TOGAF и FEAF. Проводится сопоставление рабочих архитектурных схем, используемых в технологиях разработки АС различными корпорациями.

Материал третьей главы связан с вопросами разработки архитектур АС. С позиций разработки предлагается рассматривать архитектуру как специфический вид автоматизированных систем, интенсивно использующих программное обеспечение. Представляются базовые архитектурные парадигмы (объектно-

ориентированная, компонентно-ориентированная и сервисно-ориентированная парадигмы), варианты архитектур (в том числе с ориентацией на события, модели и паттерны) и архитектурные стили.

Особое внимание уделяется вопросам качества АС, языкам описания архитектур и методам их проектирования, а также вопросам оценки и документирования результатов архитектурного моделирования. Обобщённо представляются идеи аспектно-ориентированного анализа и проектирования АС. Каждая из глав заканчивается списком контрольных вопросов, на каждый из которых приведён ряд потенциальных ответов.

В четвертой главе, расширяющей первое издание пособия, опубликованного в 2007 году, представлены материалы по нормативному совершенствованию стандарта IEEE-1471, которое закреплено в стандарте **ISO/IEC/IEEE 42010:2011** и его русифицированной версии **ГОСТ Р 57100—2016.**

В четвёртой главе раскрыты причины: расширения ответственности нормативов архитектурного моделирования **(от систем, интенсивно использующих программное обеспечение до систем любого типа**); введения дополнительных структур в концептуальную модель (фреймворк) стандарта IEEE-1471; дополнения новой версии концептуальной модели уточнениями в виде совокупности подчинённых моделей; уточнения терминологии.

Ещё одним расширением материалов первого издания является включение Приложения, которое содержит шаблон архитектурного описания системы.

Учебное пособие предназначено для студентов, обучающихся по направлению «Информатика и вычислительная техника», а также по специальностям «Вычислительные машины, комплексы, системы и сети» и «Информационные системы и технологии». Пособие может быть также использовано студентами других специальностей, профиль которых связан с разработками автоматизированных систем, интенсивно использующих программное обеспечение. Совсем близко время, когда других систем в окружении человека просто не будет.

ГЛАВА ПЕРВАЯ. АРХИТЕКТУРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ

1.1. Автоматизированные системы

В наиболее широком плане к категории АС относятся все системы, интенсивно использующие программное обеспечение. Такой тип систем обеспечивает информатизацию процессов: в рамках «Электронных государств»; систем *e*-бизнеса и *e*-систем другого типа, в том числе систем, обеспечивающих информатизацию в рамках предприятий и организаций; различного рода встроенных систем; CAD-CAM-CAE–систем, обслуживающих разработку систем различного рода, а значит, и систем типа АС, а также других систем разнородных типов. Реальность такова, что эволюция систем, используемых человеком, движется в направлении всё большей информатизации, увеличивающей класс систем АС-типа.

Каждая АС имеет ту или иную архитектуру. Если при разработке АС её архитектурные представления использовались явно, то, как показывает практика, это повышало степень успешности разработок и приводило к другим положительным эффектам. Если же такое внимание к архитектуре отсутствовало, то у АС формировалась **случайная** (accidental) архитектура [14], позитивный или негативный **вклад** которой в разработку **не планировался и не контролировался**. Случайная архитектура может оказаться и очень удачной, но вероятность этого мала. Часто (и с разными целями) случайная (удачная) архитектура после завершения разработки регистрируется, например, для целей обучения обслуживающего персонала или для целей повторного использования (разработки очередных версий или родственных систем).

В зависимости от типа АС различают «Архитектуру электронного государства», «Архитектуру предприятия», «Архитектуру системы», «Архитектуру программного обеспечения» и другие варианты архитектур систем или их подсистем и их представлений с различных точек зрения. Однако, основные про-

блемы разработки АС проявляются на уровне разработки их программного обеспечения (ПО), что привело автора к решению спуститься в учебном пособии с уровней архитектур государственных и отраслевых АС и даже с уровня архитектур «предприятия» на уровень архитектур АС без указания специфики АС. Поэтому в последующем тексте термин «архитектура» часто используется в смысле «архитектура ПО».

Акцент на архитектурах ПО обусловлен тем, что содержание вариантов употребления терминов «Архитектура ПО» и «Архитектура автоматизированных систем, интенсивно использующих ПО», сложилось за последние 15 лет. Это содержание является «молодым» по сравнению с другими вариантами содержания понятия «архитектура». В то же время новые архитектурные понятия и их материальные воплощения оказали существенное влияние на процессы разработки АС любых типов.

Вопрос специфики АС в пособии раскрыт не на уровне разработки, а на уровне использования архитектур АС. Таким образом, ниже проводится обзор архитектурных представлений АС с позиций их программного обеспечения, разумеется, в контексте АС как целого. Заметим, что достаточно часто архитектуру АС приходится строить (и использовать) с учётом её представления (рис.1.1) в контексте «предприятия».

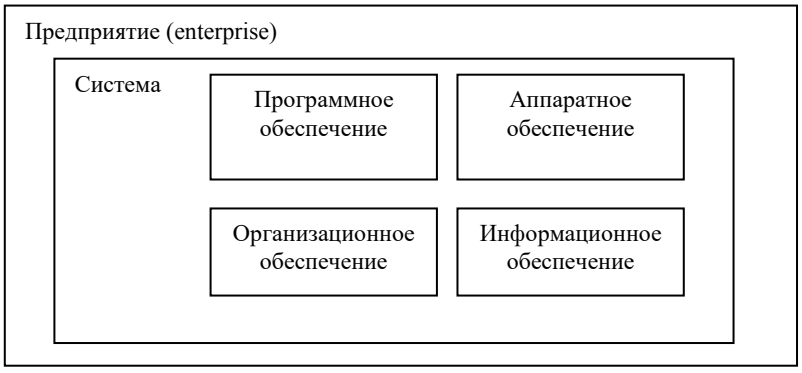


Рис.1.1. Обобщённые отношения между архитектурами разных уровней

Такой учёт, раскрытый в [25], выводит на взаимосвязанную совокупность архитектур:

- **программного обеспечения** (software architecture), на архитектурах которого и будет сосредоточено содержание обзора;

- **аппаратного обеспечения** (hardware architecture), в базисе основных единиц компьютерной и телекоммуникационной техники;

- **информационного обеспечения** (information architecture);

- **организационного обеспечения** (organizational architecture), раскрывающего связность организационных структур (в том числе на уровне персонифицированных ролей и ответственности) с бизнес-процессами;

- **предприятия** (enterprise architecture) с позиций бизнеса, в общем случае выходящего за рамки компании.

Вопросы и решения по архитектурам аппаратного, информационного и организационного обеспечения, а также по архитектурам предприятий в пособии будут затрагиваться по мере необходимости для раскрытия архитектур ПО. В то же время предварительно отметим следующее:

1. Архитектуры аппаратного обеспечения традиционная и достаточно устоявшаяся форма, используемая в представлении АС. Опосредовано она находит отражение в моделях развертывания ПО.

2. В основе архитектуры информационного обеспечения лежат аналогии с АС, интенсивно использующими данные, что находит отражение в архитектурных стилях ПО.

3. Архитектурные решения по организационному обеспечению учитываются во многих моделях и артефактах, сопровождающих разработку современного ПО.

4. Предприятия, интенсивно использующие ПО, являются определённым классом АС. Архитектуры таких систем нашли богатое представление в образцах и стандартах [77-80,89], которые используют заимствования из опыта архитектурных представлений и, в то же время служат источниками полезной информации и решений как при разработках АС других типов, так и при разработках ПО. Следует отметить, что достаточно полный обзорный материал по «enterprise architecture» представлен в [85].

5. Достаточно полный обзор по опыту «электронных государств» и используемых архитектурных решениях в АС такого типа приведён в [84].

1.2. Определения архитектуры и её значимость

Осознанному и конструктивному применению терминов «архитектура ПО» и «архитектура систем, интенсивно использующих ПО», около 15 лет [43]. За этот период его варианты употребления детализировались дифференцировались и уточнялись. Представительные коллекции вариантов употреблений термина приведены в [47]. Так, например, WEB-коллекция определений SEI (Software Engineering Institute of Carnegie Mellon) содержит около 70 определений, объединённых в разделы, включающие раздел, в котором каждое заинтересованное лицо может зарегистрировать собственное понимание и определение «архитектуры ПО».

В профессиональной литературе по программной инженерии наиболее часто ссылаются на следующие определения:

1. (SEI) Архитектура программы или АС – это структура (или набор структур) системы, которая включает программные элементы, наблюдаемые из вне свойства этих элементов, и взаимодействие между ними.

2. (RUP – Rational Unified Process) Архитектура включает: существенные решения по организации АС; выделенные структурные элементы с их интерфейсами и функциями, обеспечивающими взаимодействие элементов в рамках АС; композиции структурных и функциональных элементов в подсистемы, в соответствии с архитектурным стилем, который определяет их организацию, элементы, интерфейсы, взаимодействие и композицию в более сложные образования. Архитектура затрагивает не только структуру и поведение, но также использование, функциональность, и другие аспекты.

3. (IEEE – Institute of Electrical and Electronic Engineers) В соответствии с рекомендуемой практикой архитектура определяется как фундаментальная организация системы, связывающая её компоненты, их взаимодействие между собой и с окружающей их средой, а также принципы управления проектированием и развитием.

4. Архитектура, в первую очередь, – это совокупность принципов структурирования, которые предоставляют возможность в контексте АС как целого

представить её с помощью набора более простых систем, каждая из которых определена в соответствующем локальном контексте.

В работе [26] проведён анализ третьего (из представленных выше вариантов) определения архитектуры со следующим толкованием его отдельных позиций:

1. Архитектура в любой версии определяет структуру. Разумеется, архитектура не сводится только к структурам (деление на части, элементы, интерфейсы, связи, взаимодействие), но структурный аспект в её содержании и его материализации является наиболее существенным. Функции элемента структуры могут выполнять модуль, подсистема, процесса, библиотека и т. д., причём каждый из таких элементов может быть реализован различным образом. Например, коннектор может быть сокетом, синхронным или асинхронным, ассоциированным с определённым протоколом и т. д.

2. Архитектура определяет поведение. Архитектура определяет не только структуры АС через элементы и связи, но также и взаимодействие элементов структур, то есть взаимодействие, осуществляемое во времени. В этом плане архитектура представляет поведение АС в терминах процессов их функционирования. Для выражения динамики процессов в архитектурных описаниях используют различные средства, например «диаграммы последовательностей» на языке UML.

3. Архитектура фокусирует свои интересы на существенных элементах структуры и поведения. В построениях архитектуры АС абстрагируются до самого существенного в АС как целом. Детали целого и детали частей АС раскрываются в других формах существования АС, в первую очередь в её «концептуальном проекте», разработку которого проводят опираясь на архитектуру АС.

4. **Архитектура находит баланс для совокупности требований лиц, заинтересованных в её разработке.** В типичном случае разработки конкретной АС система требований, выявленная и сформированная до построения её архитектуры, нуждается в адаптации к реалиям средств, выделенных на разработку, и к реалиям будущего использования АС, учитывающим её сопровождение и эволюцию. Основные решения по адаптации, включающей построение сбалансированной совокупности требований, принимаются при построении архитек-

туры АС. Наиболее сложна балансировка нефункциональных требований к АС, отвечающих за её качество и/или за качество процесса разработки.

В нахождении баланса следует принимать в расчёт то, что:

- конечный пользователь заинтересован в интуитивно понятном и предсказуемом поведении АС при взаимодействии с ней, в таких характеристиках качества как «удобство», «понятность» и «обучаемость»;

- **системный администратор** заинтересован в интуитивно понятных средствах для его работы, в том числе в средствах, помогающих в задачах мониторинга состояний АС и её поведения;

- **маркетолог** заинтересован в выигрышных функциях АС на рынке родственных (для АС) систем, достаточном для маркетинга времени, ясном позиционировании среди родственных систем и цене;

- **покупатель** заинтересован в цене, стабильности работы АС и плане поставки и введения в эксплуатацию;

- **разработчик** заинтересован в ясных и непротиворечивых требованиях к АС, а также в возможности использования простого подхода к проектированию;

- **руководитель проекта** заинтересован в предсказуемости развертывания работ и их результатов, в трассируемости проекта, плане действий, производительных средствах, доступных ресурсах, в первую очередь финансовых средств;

- **лицо, ответственное за сопровождение АС**, заинтересовано в понятности АС и решений, вложенных в её разработку, а также в документированности и легкости модификации.

5. Архитектура интегрирует существенные решения на основе принципов рациональности. Существенные решения и формы их интеграции в архитектуре АС должны быть рационально представлены и обоснованы. Решения должны не только декларироваться, но и объясняться с позиций «Почему им было отдано предпочтение среди альтернатив».

6. Архитектура должна быть согласована с архитектурным стилем или их совокупностью. В разработках архитектур следует использовать накопленный опыт успешных разработок, в первую очередь опыт, вложенный в архитектурные стили, каждый из которых является образцом общих типовых решений.

Понятие архитектурного стиля, образцы стилей и их классификация представлены в пособии в п.3.4.

7. На архитектуру оказывает воздействие её среда. Конкретная АС разрабатывается в конкретной среде разработки и для конкретной среды использования. Факторы этих окружений не могут не влиять на архитектурные решения. К числу основных факторов, оказывающих влияние на архитектуру, относятся: предназначение (миссия) АС; лица, заинтересованные в разработке АС; ограничения разных типов; уровень квалификации лиц, вовлечённых в процессы использования АС.

8. Архитектура воздействует на структуру команды, которая разрабатывает АС. Для реализации того, что вложено в архитектуру, требуются вполне конкретные умения квалифицированных исполнителей. По архитектуре можно заключить, какие специалисты и в каком количестве потребуются для разработки АС, что и должно использоваться при формировании коллектива разработчиков.

9. Архитектура присутствует в любой АС. Даже если при разработке АС вопрос о создании её архитектуры, по тем или иным причинам, не поднимался, разработанная АС будет иметь архитектуру, но эта архитектура будет носить случайный (по принятым проектным решениям) характер.

10. Архитектура имеет определённые границы. Как уже отмечалось выше и было представлено на рис.1.1, различают архитектуру в разных объемах: программного обеспечения, аппаратного обеспечения, информационного обеспечения; организационного обеспечения, предприятия. И всё же архитектуры ПО являются тем центром, около которого или от которого строят архитектуры других названных объёмов.

Завершая пункт, отметим, что проведённое толкование 3-го определения архитектуры применимо и для других названных выше и неназванных определений архитектуры.

1.3. Архитектура как форма концептуального существования АС

Как уже отмечалось, число определений архитектуры велико. Одно и безоговорочно принятое разработчиками АС определение архитектуры АС отсут-

ствует. Даже представленный выше анализ построен как перечень существенных проявлений архитектуры, а не как ряд проявлений, исходящих из одной сути. Поскольку авторские определения архитектуры поощряются, представим ещё одно понимание сути архитектуры АС, конструктивное воплощение которой в разработке АС носит принципиальный характер.

На рис.1.2 представлена обобщённая картина успешности разработок АС, интегрирующая положение в этой области за 10 лет. Степень успешности, выраженная в процентах числа проектов, завершившихся в соответствии с исходными замыслами и планами, чрезвычайно низка (около 30%). Значительное число разработок либо прекращается, либо превышает запланированное время и /или средства, либо завершается в более бедной версии. Факт низкой успешности в этой области означает, что разработчики АС до сих пор не получили очень важных инструментально-технологических средств.

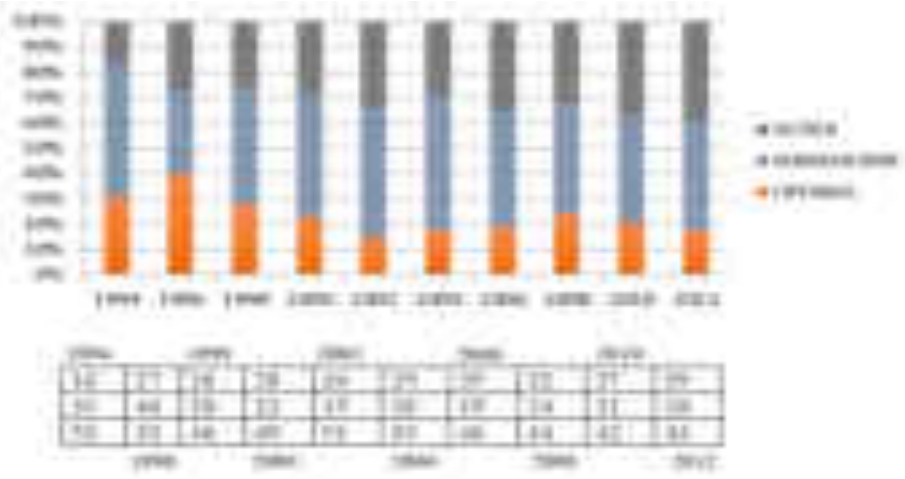


Рис.1.2. Степень успешности разработок АС (Standish Group)

Исходя из специфики проблем разработки АС, роль таких средств могли бы выполнить средства искусственного интеллекта (ИИ), поскольку слишком многое в разработке АС опирается на взаимодействия с «опытом» и «знанием», в первую очередь, для «принятия решений» и «решения задач». Обращаясь к опыту ИИ, следует ожидать, что, возможно, существующие средства ИИ придётся адаптировать к специфике разработки АС, а, возможно, придётся разрабатывать и новые средства ИИ.

По глубокому убеждению автора степень успешности разработок АС можно существенно повысить за счёт инструментально-технологических средств, позволяющих оперативно, как только в этом возникает необходимость, интегрировать интеллектуальные усилия лиц, заинтересованных в разработке АС. В основе этого убеждения лежит следующее:

1. Интеллект создавался природой для решения вполне определённых проблем сохранения вида, за счёт приобретения и использования внегенетического опыта, полученного конкретным лицом или группой лиц при взаимодействии с окружающей средой.

2. Для расширения круга задач, доступных интеллекту и, тем самым, для развития индивидуального и коллективного опыта, обеспечивающего полезное взаимодействие с окружающей средой, интеллект научился использовать различного рода средства, повышающие его «коэффициент полезного действия».

3. Специфика разработки современных АС заключается в том, что по ходу разработки встают задачи (когда и сколько – вопрос второй), для решения которых необходимо оперативно (когда в этом возникает необходимость) интегрировать интеллектуальные усилия групп лиц, заинтересованных в разработке АС. Более того, сложность задач разработки АС превышает уровень сложности задач, для решения которых у современного интеллекта уже есть инструментальные помощники.

4. Реальность разработок АС такова, что за каждую задачу процесса разработки ответственность возлагается на конкретного разработчика, и именно на базе его индивидуального интеллекта следует осуществлять оперативную интеграцию необходимых (для решения задачи) интеллектуальных усилий.

5. В интегрированном интеллекте должны поддерживаться базовые интеллектуальные активности, в первую очередь, интегрированное сознание и интегрированное понимание.

6. Феномен «сознания», отвечающий за «работу со-знанием», проявляется в процессах взаимодействия с «опытом». На «сознание» и «знание» в процессах взаимодействия с «опытом» интеллект возлагает обеспечение доступа к «прецедентам» (базовым единицам опыта, кодирующим типовое реагирование в типовых ситуациях), определённым образом закодированным и включённым в си-

стему «опыта». От осуществления такого доступа существенным образом зависит полезность обращения к «опыту».

7. Специфический вид «работ со-знанием» целесообразно контролировать, что и выполняет интеллектуальная активность, получившая название «понимание».

8 Базовой формой проявления работы сознания, в том числе и интегрированного, являются рассуждения, представления которых могут быть вынесены за пределы мозговых структур, в том числе и для инструментальной обработки.

9. Наиболее важным результатом понимания как средства контроля за работой сознания, а значит, и как средства контроля за рассуждениями, является целостное представление в образном виде (в виде схемы или «картины» другого рода) выполненной сознанием (рассуждениями) работы.

10. Коллективные «рассуждения» целесообразно считать продуктом интегрированного «сознания» только в том случае, когда для каждого индивидуального интеллекта они могут быть представлены и использованы в оперативных целях как «индивидуальные рассуждения + модель интегрированных рассуждений».

11. «Образные представления» можно считать продуктом интегрированного «понимания» только в том случае, когда для каждого индивидуального интеллекта они могут быть представлены и использованы в оперативных целях (в рассуждениях и их контроле) как «индивидуальные образные представления, которым есть место и роль в составе модели интегрированного образного представления».

12. ***Интегрированное концептуальное представление сущности АС, регистрирующее в образной форме всё необходимое для контроля (с помощью понимания) рассуждений по существенным темам, связанным с процессами разработки и использования АС, и есть основное предназначение архитектуры АС.***

Заключительный 12-й пункт перечня, представленного выше, и есть авторское определение «архитектуры АС». Другими словами, **«архитектура есть форма концептуального существования АС, обслуживающая контроль (с помощью понимания) процессов жизненного цикла АС»**.

Авторское определение будет использоваться ниже с различными целями. Сопоставляя его с тем, что представлено выше, можно отметить, что авторское определение согласовано с теми определениями архитектуры, которые уже были приведены, и с толкованиями 3-го определения.

В то же время авторское определение архитектуры акцентирует внимание на принципиальной роли активности интегрированного сознания в формах интегрированных рассуждений.

Оценивая современные технологии разработки АС с позиций инструментальной поддержки интегрированных рассуждений (в рамках оперативной интеграции интеллектуальных усилий разработчиков), можно констатировать, что эта позиция практически упущена. Таких средств в технологиях разработки АС практически нет, в то время как в искусственном интеллекте проблемам и средствам моделирования рассуждений уделяется большое внимание. Так, например, в тематике регулярных европейских конференций по искусственному интеллекту (ECCAI-XXXX) присутствует около двадцати позиций, связанных с моделированием рассуждений разных типов.

Тот факт, что инструментальной поддержке рассуждений, их моделированию и управлению рассуждениями в коллективной разработке уделяется слишком мало времени, наводит на мысль, что в построениях архитектуры не используются важные составляющие (как информационные, так и деятельностные). Учёт таких составляющих способен повысить степень успешности разработок АС.

Поясним включение в определение архитектуры ссылки на **«темы»** рассуждений. В процесс разработки в разные моменты времени включаются лица с разными интересами, способными выразить их адекватно на своём естественно-профессиональном языке. Такие интересы определяют темы для рассуждений, в которые вовлекаются не только специалисты по содержанию темы. В рамках тем происходит концептуальное смешивание различных профессиональных языков, что существенно усложняет понятийный контроль коллективных рассуждений по теме. А значит, по каждой теме следует оперативно формировать толковый словарь по теме и семантические образцы (концептуальные схемы), обслуживающие понимание.

Ещё одно пояснение по образцам для понимания. Подбирая подходящие средства, помогающие пониманию, логично использовать опыт контроля из различных предметных областей, в частности, опыт контроля из теории и практики измерений. В таком виде практик принято использовать образцы, их совокупности и системы образцов, с которыми проводят сопоставление контролируемого результата. Что из себя представляет конкретный контролируемый результат и процесс сопоставления является вторичным.

А значит, для понимания необходимы образцы, с которыми следует проводить сравнения. Следовательно, для выполнения определённой работы сначала следует создать соответствующие ей образцы понимания, их совокупности и систему образцов, с которыми в последующих действиях следует сравнивать последующие акты понимания. Функции таких образцов способны выполнить адекватные концептуальные представления того, что требуется понять. Одной из принципиальных характеристик таких представлений является их целостность.

Разработка АС представляет собой специфическую деятельность, для актов понимания в которой необходимы образцы для понимания не только АС как целого, но и её частей, а также образцы понимания процесса разработки АС и частей этого процесса.

1.4. Проблема сложности

1.4.1. Причины сложности

Процесс разработки АС и его результаты имеют принципиально сложный характер, в основном из-за сложности ПО. Именно сложность является причиной, приводящей к задачам разработки АС, для решения которых приходится интегрировать интеллектуальные усилия.

То, что «Сложность ПО является его существенным свойством, причём не случайным свойством», детально раскрывается в работе [12]. В этой публикации представлены комментарии по следующим факторам, оказывающим воз-

действие на технологии разработки систем и приводящим к сложности ПО (а значит, и АС):

1. Законы физики.
2. Законы ПО.
3. Изменение алгоритмов.
4. Трудности распределения.
5. Проблемы проектирования.
6. Проблемы функциональности.
7. Важность организации.
8. Воздействие экономики.
9. Влияние политики.
10. Недостаток воображения.

Представим каждый из этих факторов детальнее:

1.**Законы физики.** Любая конкретная АС локализована и функционирует в определённом физическом пространстве (например, в рамках распределённой компьютерной сети) и, следовательно, должна подчиняться законам физики (скорость распространения электромагнитных волн, задержки передачи информации между компьютерами, электромеханика и теплотехника компьютерных средств, возможность физических дефектов, вплоть до выхода физических средств из строя и т. д.). Зависимость от физики особенно принципиальна для АС реального времени, сложность которых в существенной мере определяется требованиями к АС, отражающими её связь с реальностью. По темам физики в процессе разработки придется рассуждать, а результаты рассуждений овеществлять в составе АС.

2. **Законы ПО**. Известно, что не все задачи, решение которых необходимо в определённых АС, имеют алгоритмическое решение. Очень часто известные алгоритмические решения «не по карману» разработчикам АС . Практически в любой разработке сталкиваются с новыми задачами, решение которых сначала необходимо построить. Достаточно типичны ситуации «комбинаторного взрыва». Программ без ошибок практически не существует. Даже в «средних» программных системах число возможных вариантов развития вычислений практически необозримо. По темам ПО в процессе разработки придётся рассуждать

больше всего, а результаты рассуждений овеществлять как программное обеспечение АС.

3. **Изменение алгоритмов.** В ситуациях потенциального комбинаторного взрыва приходится вводить управляемые изменения «точных алгоритмов», вводя в него различного рода эвристики и, тем самым, отказываясь от гарантий построения положительного результата. Кроме того, реальность разработки такова, что алгоритм приходится изменять и не раз, адаптируясь к решаемой задаче или к ресурсам, вовлечённым в процесс разработки. По устоявшемуся мнению автора «программирование отличается от других способов решения задач тем, что оно «разрешает» изменять постановки задач. В разработках ПО достаточно типична ситуация, в которой постановка решения задачи принимает свою завершённую формулировку на завершающем этапе разработки АС. Рассуждения по темам изменений являются особо опасными.

4. **Трудности распределения.** Реальность такова, что современные АС обычно распределены в пространстве (корпоративная сеть, телекоммуникационная среда) и обслуживают множество пользователей. Они относятся к классу распределённых (в сети) систем, на разработку которых неявно воздействуют следующие заблуждения: сеть надёжна, задержки равны нулю, пропускная способность каналов неограниченна, сеть защищена, топология сети не изменяется, у сети только один администратор, цена транспортировки равна нулю, сеть однородная. На самом деле это не так. Проблемы распределения приводит к дополнительным темам для рассуждений/

5. **Проблемы проектирования.** В основу проектирования АС положено упрощение процесса проектирования и его результата. В то же время простота понимается и воспринимается по-разному конечными пользователями, разработчиками, обслуживающим АС персоналом и другими заинтересованными лицами. Для конечных пользователей простота связывается с соответствием опыта пользователей и средств их взаимодействия с АС. Для разработчиков АС простота ассоциируется с простотой архитектуры АС. Для тех, кто развёртывает АС, простота связана с простотой средств инсталляции АС. Необходимо следовать известному изречению А. Эйнштейна «Любую вещь необходимо делать простой, но не проще».

В проектировании как нигде велика роль подходящего абстрагирования и образцов. В его основе лежит использование опыта проектирования, как коллективного, так и индивидуального. Основная задача опыта (и взаимодействия с ним) – помочь его владельцам и пользователям во взаимодействии с окружающим миром, в рассматриваемом случае помочь разработчикам АС и их пользователям в их действиях. Представить опыт в среде проектирования и обеспечить взаимодействие с ним является исключительно сложной задачей. Рассуждения являются основным средством взаимодействия с опытом проектирования.

6. **Проблемы функциональности.** Проблемы функциональности АС начинаются на этапе выявления требований и определения спецификаций. Достаточно часто этот этап начинается с обобщённого видения проблемы, смутной идеи и ряда описаний применения АС. Этап является наиболее критичным к возможным ошибкам, которые вводятся в разработку из-за неправильного понимания исходных потребностей, затребовавших разработку АС (неправильного понимания заказчиками АС и её разработчиками, дефектов взаимопонимания между ними и другими лицами, вовлечёнными в разработку). Такого рода проблемы функциональностей АС существуют и на последующих этапах их разработки. Рассуждения о функциональностях для процесса разработки являются особо принципиальными.

7. **Важность организации.** Как уже отмечалось разработка современных АС и их использование носит коллективный характер. Кроме того, разработка осуществляется в распределённой корпоративной среде коллективом разработчиков и другими заинтересованными лицами. Типичная группа разработчиков состоит из 4–8 членов. Но в реальный процесс разработки обычно вовлечены группа групп или более сложная структура (например, иерархическая) из групп групп и их членов. Всё это существенно усложняет продукт разработки и процесс его создания (приходится распределять информацию и функциональности, а также распределять работы и ответственности).

В типичном случае группы разработчиков специализированы, а их члены выполняют различные роли так, чтобы эффективно использовать коллективный и индивидуальный опыт. В своей работе они используют различные формаль-

ные языки и естественно-профессиональные языки (настроенные, например, на тестирование состояний процесса разработки и его продуктов или на документирование архитектуры АС). Задачи формирования таких групп и организации их работы относятся к категории очень сложных.

На их решение оказывают существенное воздействие: цена их решения; планирование работы и вклад каждого участника в общую работу, в том числе в рамках взаимодействия с другими лицами, вовлечённым в процесс разработки АС; проблемы коммуникации в группах, включая разделение знаний и опыта, полномочий и памяти; недостаток времени; время, потерянное на программное и аппаратное обеспечение, которое не работает. Каждое воздействие открывает тему для рассуждений и не одну.

8. **Воздействие экономики.** Любая разработка имеет определённую стоимость. Следует различать суммы, одна из которых соответствует финансовым средствам, которые выделены на разработку АС, а вторая соответствует средствам, потенциально необходимым для выполнения работ в рамках оговоренных требований и ограничений. Практика разработок показывает, что первая сумма (и исходно запланированное время на разработку) обычно в два раза меньше, чем реальные нужды [44].

Для оценок исполнимости разработки по усилиям или времени принято использовать формулу Б. Боэма

$$\textbf{Исполнимость} = \textbf{Сложность}^{\textbf{Процесс}} * \textbf{Команда} * \textbf{Инструменты},$$

в которой: «Исполнимость» = «Усилия» или «Время»,
«Сложность» = «Количество кода, разработанного человеком»,
«Процесс» = «Зрелость процесса или описания»,
«Команда» = «Умения, опыт и мотивации» и
«Инструменты» = «Степень автоматизации процесса разработки».

9. **Влияние политики.** Команды разработчиков и других лиц, заинтересованных в разработке АС, действуют в рамках организаций и предприятий, с их корпоративной системой отношений и организационной политикой, в контексте других политик (отраслевых, государственных, международных). Всё это вно-

сит в проблему сложности дополнительные темы для рассуждений и дополнительные составляющие в АС или её окружение.

10. **Недостаток воображения.**

Достаточно типична ситуация, когда приходится разрабатывать АС, у которой нет аналога. А значит, приходится строить её будущее образное представление, в первую очередь как образец для понимания, за счёт феномена «воображение». У того лица или лиц, которым будет поручена такая работа, их индивидуальных и коллективных возможностей воображения может просто не хватить, что, разумеется, приведёт к проблемам. Следовательно, феномену «воображения» необходимы инструментальные помощники, повышающие результативность и качество работы феномена. В число таких помощников в обязательном порядке входят рассуждения.

Названные факторы сложности и ряд других, приводящих к сложности ПО, названы также в докладе Г. Буча на конференции [14], в котором указан ряд «сил, оказывающих давление» на процесс разработки АС (рис.1.3).

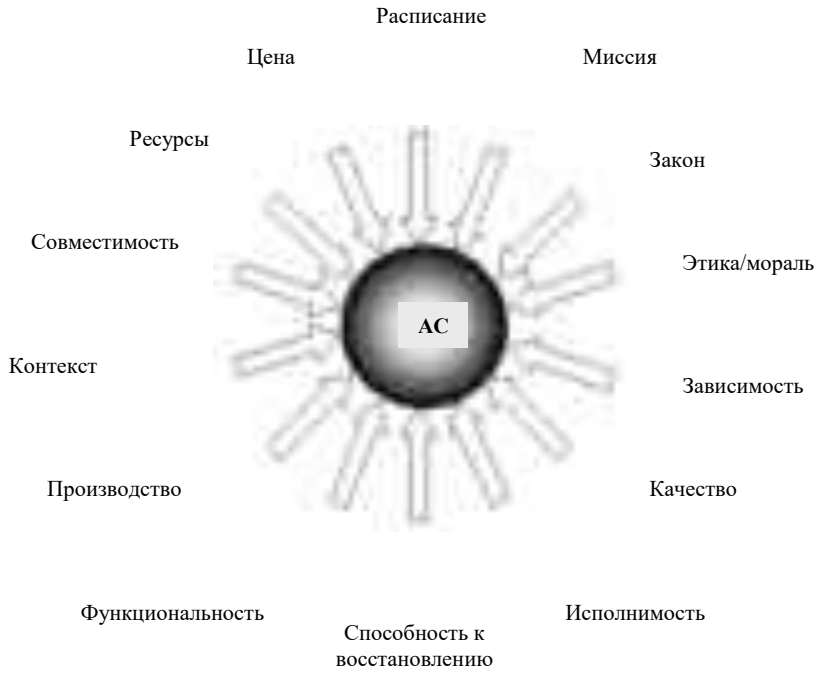


Рис.1.3. «Давление» на процесс разработки АС

В докладе отмечено, что разработчикам АС приходится преодолевать следующие «силовые давления», обусловленные факторами, раскрытыми выше в п.п.1–9:

1. **Давление на процесс** разработки со сторон:

Производства, включающего реализуемость АС (особенно если это новая разработка), управляемость (процесса и продукта) и устойчивость (к возможным воздействиям на части разрабатываемой системы).

Способности к восстановлению ((упругости, эластичности) охватывающей необходимость обеспечения таких характеристик качества, как *поддерживаемость (*supportability*), сопровождаемость (*maintainability*), адаптируемость (*adaptability*), масштабируемость (*scalability), *переносимость между платформами* (portability*), расширяемость (*extensibility*) и предрасположенность системы к изменениям(churn*) так, чтобы её можно было использовать в новых технологиях.

2. **Давление среды**, в составе которого важны:

- *Ресурсы*, включающие такие составляющие, как вычислительные мощности, электрическая энергия, физическое пространство и, возможно, другие персонально ограничивающие факторы.

- *Совместимость* с внешними элементами, недостаточная степень которой может привести к отклонениям от норм (процесса и технических стандартов; требований по комплексированию с другими системами).

- *Контекст* в рамках экономических , исторических и политических факторов, учёт которых носит обязательный характер.

3. **Давление со стороны будущего использования** АС, включающего:

Необходимость достижения запланированных функциональностей АС с позиций поведения системы и её практичности (usability).

Исполнимость (Performance) и характеристики качества с позиций других (а не только уже названных выше) нефункциональных требований к АС.

4. **Давление бизнеса**, включающее *Цену, Расписание (План) и Миссию* (особенно в формах долговременных деятельностных обязательств организации или предприятия перед партнёрами и/или клиентами) относят к числу наиболее важных давлений на разработчиков АС. Они связаны со временем, материалами и людьми, реальное наличие которых определяет возможность или невозможность любой конкретной разработки и её соответствие миссии.

1.4.2. Подходы к структурированию

Сложность ПО обусловлена не только названными выше факторами и давлениями на процесс его разработки и использования. Однако перечисленного уже достаточно для того, чтобы согласиться с необходимостью применения любых средств, позволяющих снизить сложность как конкретной АС, так и процесса её разработки.

В системной и программной инженерии накоплен определённый опыт, позволяющий повысить степень успешности в разработках АС. Наиболее традиционным способом снижения сложности является её структуризация, то есть разделение «сложности» на части, связанные в единое целое. В этом плане «сложность» может выступать в разных формах [12], например:

- в формах описания «сложного образования», что согласовано с метрикой сложности по Колмогорову;

- или в формах совокупности разнообразий состояний «сложного образования», выводящей на энтропийные меры сложности.

Обе приведённые формы пригодны, по крайней мере, для оценок в действиях по уменьшению сложности ПО:

- на уровне кода (исходного или исполняемого) существуют программные объекты, состоящие из десятков и сотен миллионов строк кода;

- ПО допускает автоматную интерпретацию, которая для больших программных объектов приводит к автоматным представлениям с практически бесконечным числом состояний (обойти которые, например, при тестировании, практически невозможно).

В системной инженерии системы принято делить на подсистемы и т. д. до уровня базовых единиц, используемых для структуризации подсистем, которые не содержат других подсистем. В программной инженерии деление кода на части часто проводят в формах «модуляризации» (modularity), использование которой приводит к структурам, элементами которой являются «модули». Для структуризации АС используют как подсистемы, так и модули. Достаточно часто конкретную АС (или её представление) разбивают на подсистемы, а структуризацию каждой из подсистем проводят на уровне модулей, которые понимаются и разрабатываются в соответствии с их определением в объектно-ориентированных языках программирования.

Существуют различные подходы и способы структуризации АС и их программной части [14, 31, 64]. С рядом типовых результатов структуризации связывают образцы архитектурных стилей, наиболее распространённые образцы которых приводится ниже в п. 3.4.

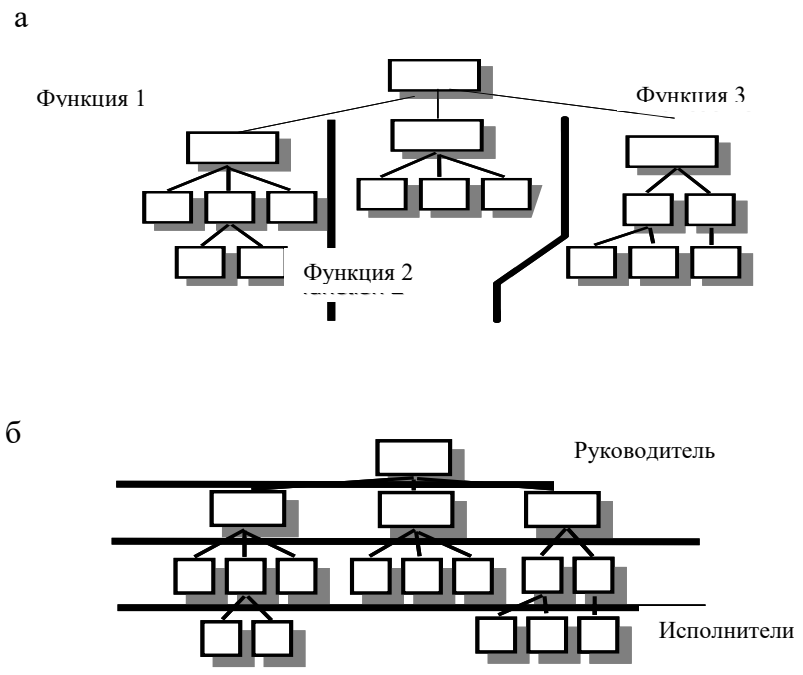


Рис.1.4. Деление на части (а – горизонтальное, б – вертикальное)

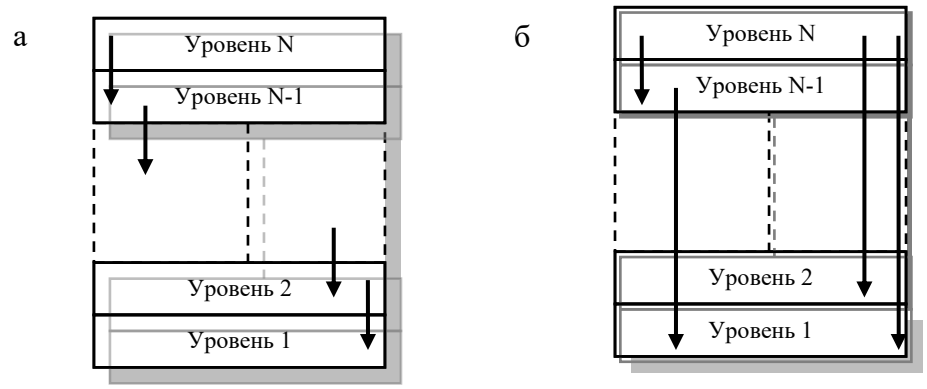


Рис. 1.5. Многослойные структуры (а – закрытого типа, б – открытого типа)

В закрытых многослойных структурах сообщения могут быть посланы только смежному нижнему уровню, в то время как в открытых – сообщения могут быть посланы любому нижнему уровню.

В любом из эскизов структуры разработчикам приходится использовать, а значит, по крайней мере, обобщённо раскрывать связи между элементами структуры. Для АС наиболее типичными вариантами связей являются связи по данным (достаточно часто выход данных одного элемента используется как вход другого), а также по управлению типа call (элемент вызывает активность другого элемента) или типа «событие» (элемент активизируется при определённых событиях, которые происходят в окружающей его среде, в том числе и из-за активность других элементов системы).

Разумеется, представленная структуризация является не единственным способом снижения сложности представления АС. С каждым из названных выше факторов, с каждым «давлением» на разработку связаны определённые подходы и способы. Наиболее общими приёмами для перехода от «сложного» к более «простому» являются:

- **абстрагирование** для отделения существенного для АС (или её части) от несущественного при запланированном или ситуативном взаимодействии со сложным (например, при принятии проектных решений) [12];

- **построение вида** (или совокупности видов) на АС с позиций определённого фактора или «давления» [14, 89];

- **разделение аспекта** (причины взаимодействия со сложным в АС) на составляющие и их распределение (например, метрик практичности) по частям АС [2].

Обобщённо раскроем сущность каждого из приёмов, используя гипотетическое представление АС в базе данных с объектно-ориентированной структуризацией БД ($ОБ_1(a_1, a_1,\ldots,a_i,\ldots, a_I),\ldots,ОБ_j(a_j, a_{j+1},\ldots,a_k,\ldots, a_K),\ldots, ОБ_1(a_n, a_{n+1},\ldots,a_m,\ldots, a_M)$) . Архитектура АС является представлением АС на уровне абстракции, для которого учитываются только существенные атрибуты, причём отражающие АС с позиций её целостности.

Если по таким атрибутам построить целостную проекцию для БД, то в результате будет образована новая (абстрагированная от БД) база данных $БД^A$, в которой:

- часть атрибутов из представления $ОБ_j(a_j, a_{j+1},\ldots,a_k,\ldots, a_K)$ будет исключена;

-некоторые объекты типа $ОБ_j(a_j, a_{j+1},\ldots,a_k,\ldots, a_K)$, возможно, в абстрактное представление АС не попадут;

- некоторые совокупности объектов типа $ОБ_j(a_j, a_{j+1},\ldots,a_k,\ldots, a_K)$ будут объединены в более «крупный» объект.

Такого рода приём можно нацелить на построение проекции АС с точки зрения определённого фактора или интереса к АС, в результате чего в $БД^A$ будут выделены проекции $БД^A_1$, $БД^A_2,\ldots$, $БД^A_r,\ldots$, $БД^A_R$, каждая из которых является целостным видом на АС, то есть её проекцией на определённую точку зрения. Именно эта идея вложена в стандарт IEEE-1471–2000.

Реальность разработок АС такова, что определённые существенные атрибуты (например, $a_i, a_j,\ldots,a_k$) образуют группы, члены которых распределены в группе объектов. То есть из элементов таких групп (обычно отражающих качество АС как целого или качество частей АС) нельзя образовать объект. Именно такая ситуация связана с разделением аспекта на составляющие, ответом на ко-

торую является разработка и использование аспектно-ориентированного проектирования [36] и аспектно-ориентированного программирования [2].

Подводя итог пункту, отметим, что для работы со **сложностью** в системной и программной инженерии накоплен полезный опыт, помогающий разделить сложные (обычно неявные представления систем) на связную совокупность простых представлений. Этот опыт приходится применять в разных состояниях существования АС, в том числе и в тех, когда АС существует как замысел или в виде концепции (или в виде архитектуры или в виде концептуального проекта). Для разработки архитектуры АС такой вид работ определяет её сущность.

1.4.3. Специфика структурирования АС

Для АС как систем, интенсивно использующих программное обеспечение, при структуризации их форм существования (таких как концепция, архитектура, концептуальный проект, …, физическая реализация, …, модифицированная версия) необходимо учитывать специфику АС как систем определённого класса, в первую очередь специфику структуризации ПО.

Практика разработок архитектур АС, представленная в [64], фиксирует целесообразность использования в задачах структуризации следующей специфики:

1. Позитивный опыт использования в ПО модульной структуризации (Module structure).

2. Сложившийся опыт использования программных единиц (компонентов), взаимодействие которых осуществляется (с помощью коннекторов) в компьютерных средах (Component-and-connector structures) .

3. Используемую на практике дополнительность «аппаратного и программного», приводящую к размещение программных структур в физическом пространстве (Allocation structures).

В основе модульной структуризации (рис.1.6) лежит следующее:

– декомпозиция (Decomposition): опирается на отношения между модулями и их связи в рамках иерархии, выделение модулей является стартовой точкой для проектирования АС, модули обычно ассоциируются с продуктами (или ар-

тефактами) процесса разработки, модульная структура согласована с «модифицируемостью» АС;

–использование (Uses) как специальный механизм, определённый в объектно-ориентированном программировании:

– разбиение на уровни (Layered);

– использование классов (Class or generalization), детализирующих модульные структуры с помощью отношений наследования и «экземпляр класса»

Рис.1.6. Специфика модульной структуризации

В основе структуризации «Component-and-Connector» лежит специфика поведения единиц структуры АС или её частей (рис.1.7):

– специфика процесса (Process) или взаимодействия процессов (синхронизация, коммуникация, отработка исключительных ситуаций);

– специфика параллельного исполнения (Concurrency);

– динамическое разделение данных (Shared Data, например, в рамках репозитария);

– специфика клиент-серверных отношений (Client-Server).

Рис.1.7. Структуризация динамики

В основе распределения составляющих АС в физическом пространстве и назначении им форм материализации (рис.1.8) лежит следующее:

1. **Размещение (Deployment):**

– показывает, как ПО назначаются аппаратные средства и средства коммуникации;

– представляет элементы ПО (обычно элементы процесса из вида «компоненты и коннекторы»), аппаратные единицы и коммуникативные связи;

– выражает отношение «allocated-to», показывающее, на какой физической базе элементы ПО размещены;

– предназначено для инженера, для того чтобы он мог рассуждать об исполнении кода, интеграции данных, доступности и защите;

– представляет особый интерес для распределённых параллельных систем.

2. Реализация (Implementation):

– показывает, как элементы ПО (обычно модули) распределены в файловых структурах в процессе разработки АС, интеграции её модулей и управляемого конфигурирования среды;

– представляет особый интерес для управления действиями разработчиков АС и развертывания процесса разработки.

3. Распределение работ (Work assignment):

- распределяет ответственность за реализацию и интеграцию модулей (частей) между членами команды разработчиков.

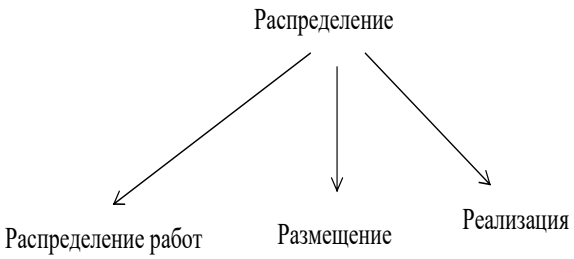


Рис.1.8. Структуризация распределения

Следует отметить, что: архитектурные структуры предоставляют различные перспективы системы: они не являются взаимоисключающими и часто дополняют друг друга; некоторая структура может быть «погружена» в другую; одна из структур может оказаться доминантной; чем сложнее система, тем более полезно представлять её рядом взаимодополняющих структур.

1.5. Место и роль архитектурных решений в разработке АС

1.5.1. Место архитектурных решений

Содержание определений архитектуры ПО явно и неявно указывает на место архитектурных решений и их роль в разработке АС. Место архитектурных решений и документов представим рядом рисунков, в основном заимствованных из публикаций, на основе которых составлено пособие.

В своей презентации доклада «Архитектура ПО» [10] G.Booch указывает на место разработки архитектуры (рис.1.9) в итерационном процессе типа RUP.

Рис.1.9. Итеративный процесс разработки

Подчёркивая особую важность Use-Case моделей в процессе разработки АС, И. Якобсон включает архитектуру в процесс, как показано на рис.1.10.

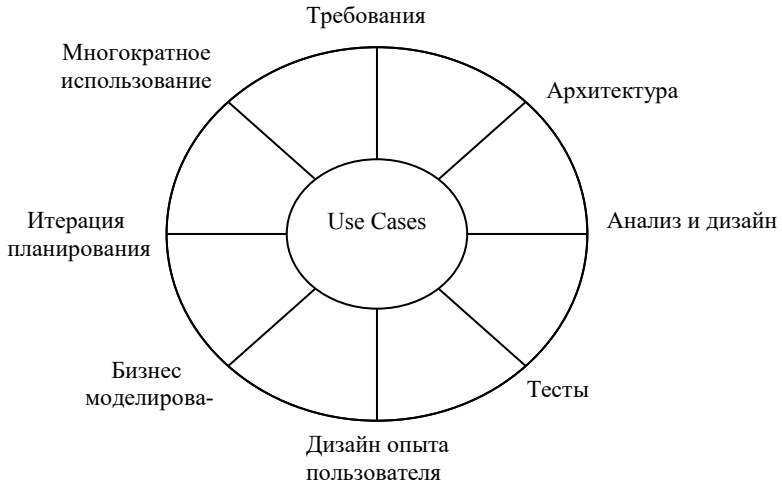


Рис.1.10. Use-case центрированная разработка

В современной практике разработки АС широко используется спирально-итеративная модель жизненного цикла [87], что приводит к необходимости возврата к вопросам разработки архитектуры (рис.1.11) с каждым циклом спирали.

Рис.1.11.Спиралевидный процесс разработки

В процессе разработки АС границы между этапами жизненного цикла и итерациями в рамках одного и того же этапа «размыты». Это особенно типично для этапов анализа требований (результатом которого является Система Требований – СТ) разработки архитектуры (результатом которого является Архитектурное Описание – АО). Более того, для практики (рис.1.11) типичны и возвраты от вопросов архитектуры к вопросам о требованиях, поскольку нормативы на АО являются очень важным источником требований.

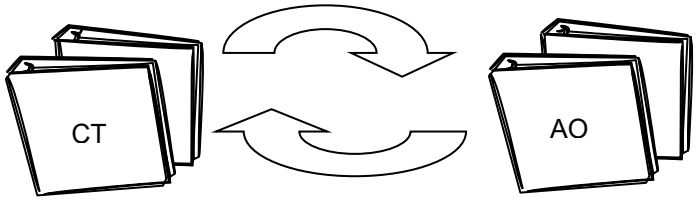


Рис. 1.12.Взаимодействие требований и архитектуры

Разумеется, каждый из этапов «анализа» и «архитектуры» приводит к определённой форме существования АС в рамках её жизненного цикла, что (с некоторыми деталями) показано на рис.1.13.

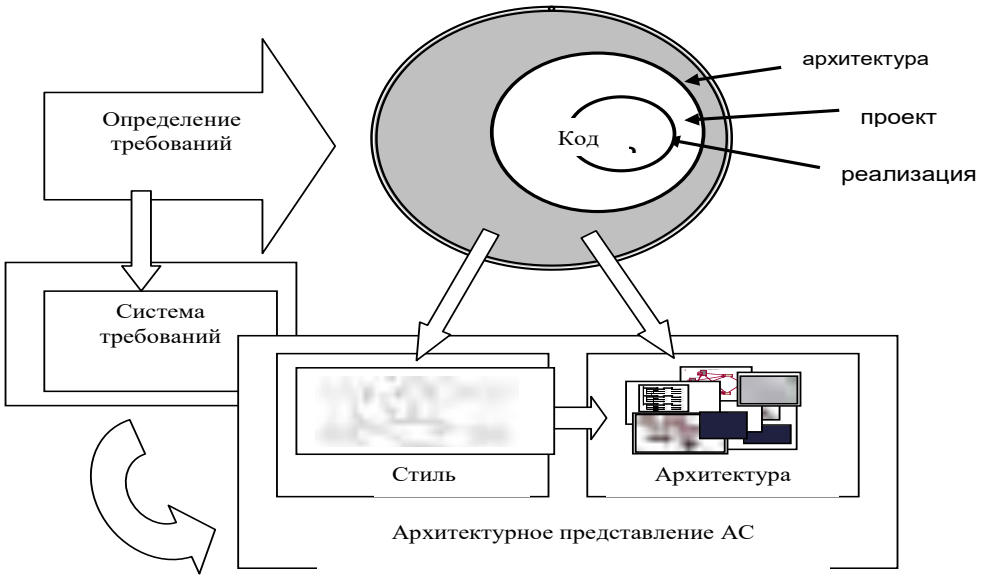


Рис.1.13. Архитектура как форма концептуального существования АС

На рисунке отражён тот факт, что формы существования АС на ранних этапах разработки являются концептуальными, то есть представляют собой связную совокупность текстовых (в том числе табличных) документов и графических моделей и диаграмм (очень часто использующих для своего представления язык UML).

1.5.2. Роль архитектурных решений

Явное построение и использование архитектуры в разработке АС не раз демонстрировало важные позитивные эффекты [29] . Обобщённое представление позитивных эффектов представлено на рис.1.14.

Рис.1.14. Обобщённое предназначение архитектуры

Более детально, архитектура:

- вносит вклад в извлечение треббований и формирование их системы;
- ясно показывает совокупность ранних проектных решений;
- предписывает организационную структуру АС (хорошо структурированные системы полны образцов);
- является общим архитектурным базисом для линеек программных продуктов;
- даёт возможность учитывать, согласовывать и предсказывать характеристики качества, в том числе и по результатам её исследования;
- даёт информацию о распределении работ и их календарных планах;
- даёт возможность для более точной оценки стоимости и расписания работ;
- снижает риски;

- является первой формой существования АС, которая может быть проверена (испытана) как целое;

- предоставляет возможность для переноса или повторного использования стилей и архитектурных каркасов;

- приносит выгоду в ограничении «словаря» альтернатив проекта или его частей;

- помогает в эволюционном прототипировании;

- оформляет концептуальную целостность АС и процесса её разработки;

- обслуживает понимание и взаимопонимание в индивидуальной и коллективной работе;

- есть средство выражения мыслей и рассуждений в коммуникации лиц, вовлечённых в разработку АС;

- предоставляет возможность рассуждать о потенциальных изменениях по ходу разработки АС и вносить вклад в управление такими изменениями;

- может быть базисом для изучения системы.

Приведённый перечень от разработки архитектуры впечатляет и, поэтому, трудно объяснить «Почему архитектуре до сих пор уделяется так мало внимания в разработке АС?». Разумеется, разработка архитектуры дорого стоит, но не дороже чем негативы от её отсутствия.

Именно по этой причине Rational Unified Process построен как архитектурно-центрированный процесс разработки АС [39].

1.6. Ретроспектива развития предметной области

Как уже отмечалось, «архитектура ПО» как определённая предметная область сложилась за последние 15 лет. Историю этих лет, а также предполагаемое будущее представим с помощью публикаций [52, 32, 43] и [62]. Многие считают, что первая из них положила начала архитектуре ПО как отдельной дисциплины в рамках программной инженерии. Вторая является публикацией, констатирующей состояние исследования и разработок по архитектуре приблизительно в середине пройденного пути. А две последних публикации 2006 года (одна из них [43] переведена на русский язык) были представлены в февральском номере IEEE Software с целью раскрыть историю вопроса.

1. Словосочетание «архитектура ПО» в свободном от разъяснений его употреблении или в форме рабочих определений (в различных исследованиях и публикациях) использовалось задолго до публикации, которая, как считают многие, положила начало «архитектуре ПО» (и «архитектуре АС») как отдельной дисциплине в рамках программной инженерии.

В статье была предложена знаменитая формула «{элементы, формы, объяснения} = программная архитектура» и её обобщённые выражения в терминах «архитектурных стилей». Раскрыта целесообразность заимствования из других предметных областей понятия «вида» и наполнения этого понятия содержанием, соответствующим задачам программной инженерии. Были намечены первоочередные шаги для создания предметной области «Архитектура». Эта публикация существенно повысила интерес к исследованиям и разработкам, раскрывающим различные аспекты архитектуры ПО, в том числе и её приложений к архитектурам АС различной степени интеграции, вплоть до архитектуры предприятий.

2. В 2000 году вышла публикация [32], фиксирующая состояние в предметной области «Архитектура» и определявшая «дорожную карту» для исследований и разработок в этой области.

После обобщённого представления истории программной инженерии по схеме рис.1.15 фиксируется современное состояние, в рамках которого выделяются (как успешные) три позиции: языки архитектурного описания, линейки продуктов и архитектурные образцы, в первую очередь образцы архитектурных стилей. Будущее связывается с направлениями в области сетевых приложений и с всеобъемлющем приложении ПО к различным предметным областям.

3. История и современное состояние предметной области «архитектура ПО и АС» раскрыто в публикации [43], которая начинается с представление разделов предметной области:

Архитектурный проект*:* как мы создаем архитектуру?

Анализ*:* как мы на основе архитектуры отвечаем на вопросы о свойствах конечного продукта?

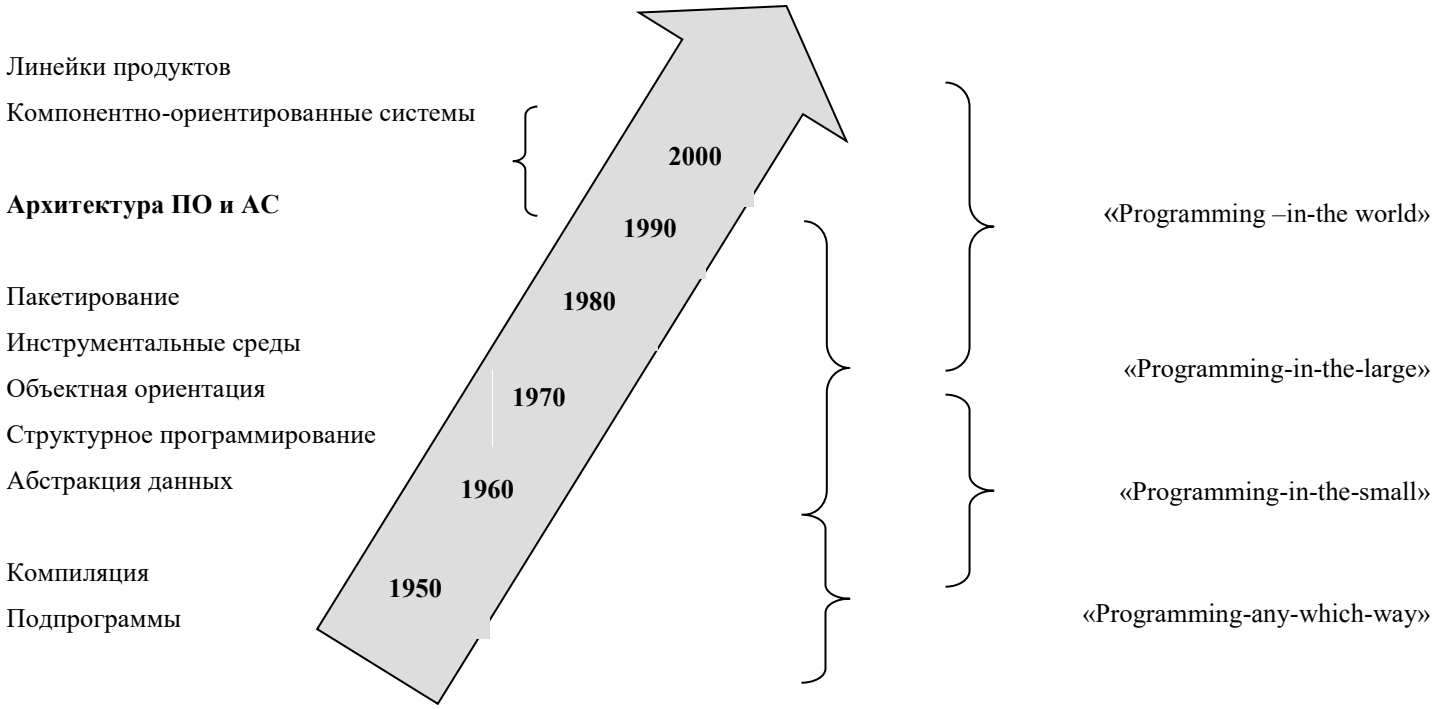


Рис.1.15. Дорожная карта программной инженерии

***Реализация**:* как мы создаем систему на базе архитектурного описания?

***Представление**:* как мы создаем надежные артефакты, чтобы «донести» архитектуру до людей и машин?

***Экономика**:* какие архитектурные проблемы влияют на бизнес-решения?

В истории становления предметной области «Архитектура» авторы выделяют этапы «до 1995 года», «1995-1998 г. г.», «1998-2005 г. г.» и «Сегодня», каждый из которых раскрывается с позиций основных достижений в теории и практике. Такая история обобщённо представлена на рис.1.16, содержащем в том числе публикации, используемые в пособии.

Отмечается, что концепция программной архитектуры как отдельной дисциплины сформировалась в период 1990–1992 годы, завершившийся основополагающей статьей Д. Перри и А. Уолфа [52]. В 1994 году вышла первая книга по программной архитектуре, написанная бывшими сотрудниками IBM Бернардом Виттом, Терри Бейкером и Эвереттом Мерритом [71].

В 1995-м начался настоящий бум программной архитектуры. События ускорились за счет многочисленных «вкладов» отраслевой и академической науки. Заметными явлениями стали метод анализа программной архитектуры SAAM (Software Architecture Analysis Method — первый из серии методов, предложенных SEI [6]), подходы на базе нескольких представлений (такие как модель представления «4+1» компании Rational [41] и четыре представления Siemens), а также специальные шаблоны для разработки программной архитектуры [9, 30]. Корпорации Siemens , Nokia , Philips , Nortel, Lockheed Martin, IBM и другие крупные разработчики программного обеспечения начали уделять внимание программной архитектуре.

В 1999 году состоялась первая конференция по программной архитектуре, были основаны рабочая группа IFIP Working Group 2.10 и институт Worldwide Institute of Software Architects. Появились и сформировались методы SAAM, BAPO и ATAM, а к уже имевшемуся общему стандарту архитектуры RM-ODP [88] добавился IEEE 1471 [89]. Одновременно в SEI продолжали выпускать книгу за книгой [5,18,19].

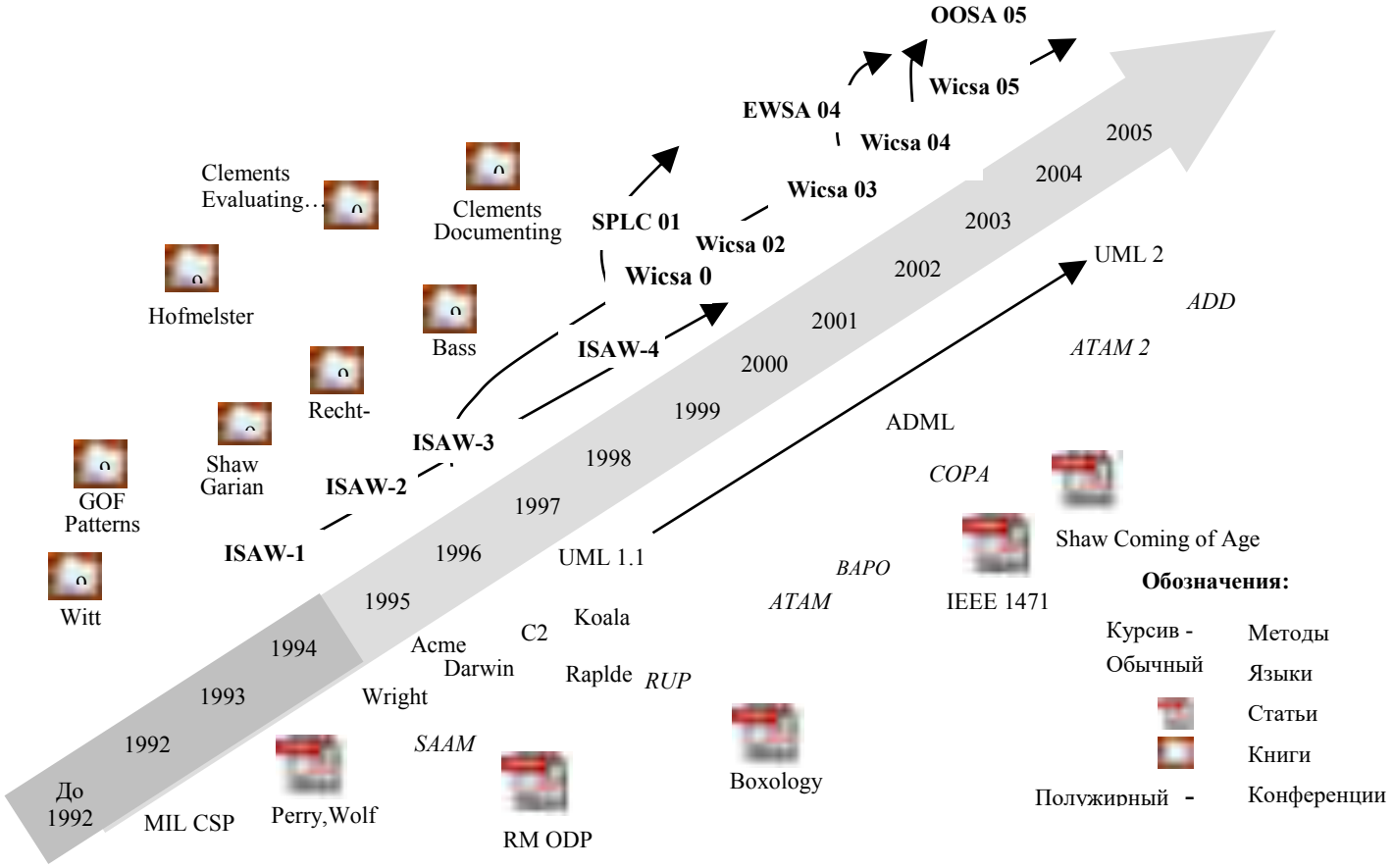


Рис.1.16. Ретроспектива проблемной области «Архитектура АС»

Современное состояние предметной области «Архитектура» авторы раскрывают с разных позиций, которые представлены и в пособии. Фиксируется, что сумма знаний о программной архитектуре изложена более чем в 25 монографиях и многочисленных научных статьях. Сформировалось активное сообщество специалистов, десятки университетов по всему миру преподают программную архитектуру, множество организаций предлагают курсы подготовки архитекторов (представлен обширный список публикаций, сайтов профессиональных сообществ, указателей на конференции по проблематике предметной области). Одним словом, дисциплина не только подтвердила своё право на существование, но и достигла достаточной научно-технической зрелости.

4. Ещё одна версия истории развития предметной области «Архитектура» представлена в публикации M. Show и [61]. Авторами выбран подход к вопросу с позиций увеличения степени зрелости этой области. За образец взята схема становления зрелости, которую часто повторяют новые технологии.

Такая схема обычно развёртывается на 15-летнем интервале и включает этапы:

1. Базовое исследование
2. Формулировка концепции
3. Развитие/расширение
4. Внутреннее расширение/исследование
5. Внешнее расширение/исследование
6. Популяризация

Становление зрелости проблемной области «Архитектура АС и ПО» проходило именно по такой схеме, причём так, что позволило авторам назвать то, что стоит за схемой становления, «Золотым веком Архитектуры».

Вопросы по первой главе

Q1. Какие системы относятся к классу «автоматизированных систем, интенсивно использующих программное обеспечение»?

Системы автоматизации проектирования.
Системы управления предприятиями.
Автоматизированные обучающие системы
Системы коллективного принятия решений.

Q2. Что представляет собой архитектура АС?

Концептуальную форму существования АС

Средство представления структуры АС

Средство выражения поведения АС

Q3. В каких интеллектуальных действиях при разработке АС используются архитектурные представления?

Рассуждения в актах принятия проектных решений.

Контроля в формах понимания за проектными решениями.

Согласования принятых решений.

Q4. Какие факторы давления на процесс разработки можно отнести к факторам среды?

Контекст

Исполнимость

Расписание

Способность к восстановлению

Q5. Какие факторы давления на процесс разработки можно отнести к факторам бизнеса?

Ресурсы

Закон

Цена

Качество

Q6. Какие факторы давления на процесс разработки можно отнести к факторам будущего использования?

Миссия

Зависимость

Функциональность

Совместимость

Q7. Чем характеризуется структуризация АС посредством многослойных структур закрытого типа?

Части одного уровня обмениваться между собой сообщениями и посылать их смежному верхнему уровню

Сообщения могут быть посланы любому нижнему уровню

Уровни закрыты друг от друга и действуют как автономные системы

Сообщения могут быть отправлены только смежному нижнему уровню

Q8. Что лежит в основе модульной структуризации?

Размещение, которое показывает как ПО назначаются аппаратные средства и средства коммуникации

Специфика параллельного исполнения

Декомпозиция

Q9. Что лежит в основе структуризации "Component-and-Connector"?

Динамическое разделение данных

Использование классов

Специфика клиент-серверных отношений

Q10. Что лежит в основе структуризации распределения?

Разбиение на уровни

Распределение работ

Специфика процесса или взаимодействия процессов

Q11. Какое место занимает разработка архитектуры в жизненном цикле АС?

Следует за этапом формирования системы требований.

Предшествует этапу детального проектирования.

Входит в состав концептуального проектирования.

Q12. Какой этап следует за Разработкой при спирально-итеративной модели жизненного цикла?

Анализ

Реализация

Тестирование

Q13. Какова роль архитектурных решений?

Вносит вклад в извлечение тре6бований и формирование их системы;

Предписывает организационную структуру АС

Помогает в эволюционном прототипировании;

Может быть базисом для изучения системы.

Q14. К каким позитивным эффектам приводит использование архитектуры?

Повышает степень успешности разработок АС

Снижает риски.

Оформляет концептуальную целостность АС и процесса её разработки.

Обслуживает понимание и взаимопонимание в индивидуальной и коллективной работе.

ГЛАВА ВТОРАЯ. АРХИТЕКТУРНЫЕ НОРМАТИВЫ

2.1. Архитектурные образцы

Как уже отмечалось в п. 1.3, архитектура является формой существования АС в виде её концептуального представления, регистрирующего и обслуживающего понятийный контроль за рассуждениями лиц, вовлечённых в процессы разработки и использования АС.

Процесс контроля на понимание можно организовать в различных версиях, в том числе и в версиях сравнения индивидуального и коллективного понимания с **образцами понимания**.

Образцы для контроля создают в различных областях деятельности, как для контроля действий, так и для контроля их результатов. Такие образцы, разумеется, зависят от того, что с их помощью контролируется. В тех случаях, когда контролируются естественно-профессиональные рассуждения, функции образцов понимания принято возлагать:

- на средства логической формализации рассуждений, использующей их перевод на подходящий искусственный язык и автоматический или автоматизированный контроль формул перевода;

- на результаты «перевода» рассуждений в образные конструкции (схемы, «картины»), выражающие целостное содержание сказанного в рассуждениях;

- на толкование рассуждений с помощью их построений в другой версии знакового выражения и/или с помощью примеров.

В первом варианте контроля рассуждений достаточно много сложностей, в первую очередь обусловленных разнообразием тематик, в рамках которых приходится рассуждать об АС, а значит, и разнообразием логических формальных средств, возможно и очень сложных. Использование единообразных формальных средств, например, языков архитектурного описания, по крайней мере пока, не находит широкого применения в практике разработок АС.

В практике разработок АС для контроля рассуждений в основном используются образные конструкции, в построениях которых используют языки визуального моделирования, причём, очень часто язык UML.

Третий вариант контроля в большей мере годится для оперативного контроля фрагментов рассуждений. Так, например, одной из проверок фрагмента рассуждений на правильность, является их проверка на корректное употребление понятий проекта АС. Примеры в проверках по третьему варианту выполняют функции целостных образцов, но частного характера.

Рассмотрим детали второго варианта контроля как наиболее распространённого на практике. Одной из принципиальных характеристик визуальной образной конструкции, обслуживающей процессы понимания, является её целостность, согласованная с определённой темой или набором родственных тем для рассуждений, сопровождающих разработку АС.

В искусственном интеллекте с типовыми темами связывают понятие фрейма (frame). Если типовая тема сопровождает определённую работу (work), то её логично называть «framework». Фреймы, а значит и frameworks, принято представлять семантическими структурами, состоящими из узлов и связей между ними. Узлы фрейма соответствуют определённым «понятиям», а связи – отношениям между «понятиями».

Концептуальные образования типа «framework» являются типовым нормативным средством (нормативными концептуальными схемами) для представления архитектур АС. Широкое применение нормативных архитектурных концептуальных схем (architecture frameworks) обусловлено следующими причинами:

- во-первых, любая architecture framework (как схема) является целостным концептуально-образным образцом, согласованным с рассуждениями, которые, на определённом уровне абстракции (на мета уровне описания архитектуры, которому соответствует нормативная «картина» архитектуры), применяют при построении и использовании архитектуры конкретной АС;

- во-вторых, любая architecture framework управляет построениями экземпляра архитектуры как средства, регистрирующего понимание и взаимопонимание, согласованное в коллективной работе (в оговоренных рамках проще договориться об общем понимании, то есть проще достичь консенсуса);

- в-третьих, architecture framework и построенная на её базе конкретная архитектура являются системой концептуально-образных образцов, с которыми следует сверять последующие (после построения архитектуры) рассуждения, разумеется, в рамках тем, вложенных в архитектуру.

Разработаны и применяются на практике различные схемы типа architecture framework. Это, чаще всего, обусловлено различиями предметных областей (например, разработки корпорации Microsoft существенно отличаются от АС, выполняемых по государственным заказам для военных), а для родственных предметных областей корпоративностью или традициями. Наиболее известные по литературе или по их практическому применению схемы типа architecture framework, рассматриваются ниже.

2.2. Стандарт IEEE-1471–2000 и его сущность

2.2.1. Основные понятия

В статье [52], которая относится к числу базовых фундаментальных публикаций, выделивших в программной инженерии «архитектуру ПО» как отдельную и важную проблемную область, была указана необходимость использования в архитектурных представлениях механизма «видов» на ПО, подобных «видам», применяемым в инженерной практике (например, в строительстве). Эта идея получила развитие, что привело к стандарту IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems [89] (Рекомендуемое для практики описание архитектуры систем, интенсивно использующих ПО).

В стандарте представлены обзор его области применимости (границы, цели, предполагаемые пользователи и соглашения по рекомендуемой практике), ссылки на родственные стандарты, концептуальный каркас (контекст архитектурного описания, заинтересованные в разработке АС лица, архитектурная деятельность в рамках жизненного цикла АС, использование архитектурных описаний), практика архитектурных описаний (документирование архитектуры, идентификация заинтересованных лиц и их интересов, селекция архитектурных

точек зрения, архитектурные виды, соответствие в совокупностях архитектурных видов, обоснования архитектур, образцы использования) и ряд приложений (библиография, замечания по терминологии, образцы точек зрения, отношения с другими стандартами).

Стандарт позиционирован его разработчиком (IEEE Architecture Planning Group) как «рекомендуемая практика». Он не является стандартом архитектуры, процесса архитектуры или метода разработки архитектуры, а предоставляет разработчикам АС рекомендации для описания архитектуры (Architectural Description, AD). Его рекомендации используют модальности «должен» (без обязательности) и «можно» (как вариант). Но рекомендации базируются на большом опыте успешных разработок АС. Рекомендации приняты мировыми лидерами индустрии ПО и АС.

Стандарт IEEE 1471 [89]:

- нацелен на его применение для архитектурного описания (АО) систем, интенсивно использующих программное обеспечение, в которых программная составляющая оказывает существенное влияние на проектирование, конструирование, развертывание и эволюцию системы как целого;

- предполагает, что организации и предприятия, решившие использовать стандарт, должны сами решать, в каком объёме и как он будет внедряться в разработки АС;

- дает широкие рамки интерпретации архитектуры, как средства, пригодного для процесса разработки АС;

- исходит из того, что описание архитектуры может быть согласовано широким кругом лиц, вовлечённых в разработку АС, в то время как система, проект, процесс и организация работ такой возможности обычно не предоставляют (из-за специфики разнородной профессиональной компетенции и языка);

- определяет концептуальную структуру (систему понятий) для описания архитектуры и его использования ;

- устанавливает термины и понятия для архитектурного мышления;

- устанавливает концептуальный каркас и словарь для рассуждений об архитектурной форме существования АС, в виде архитектурного описания;

- обеспечивает средства для рассуждений об архитектурном описании в контексте лиц, вовлечённых в процесс разработки АС, жизненного цикла АС и использования АО;

- служит как базис для развития АС на ранних этапах разработки, когда доступна только общая терминология об АС, которая ещё не материализована;

- вводит в системы требований для разработки АС специальный раздел «архитектурных требований»;

- идентифицирует и устанавливает тождество в описаниях архитектур и пропагандирует озвученное в архитектурной практике;

- открывает возможность для эволюции таких практик, как достигших достаточной инструментально-технологической зрелости;

- служит как базис для развития предметной области, в которой существует только общая терминология;

- вносит надежду разработчикам АС, что следование его рекомендациям внесёт в разработку такие позитивы деятельности как «быстрее», «лучше» и «дешевле».

2.2.2. Содержание стандарта

Перейдем к основе и деталям рекомендуемой практики, обслуживающей построение архитектурных описаний, документирующих архитектуры.

Основу рекомендуемой практики определяет концептуальный каркас (framework), представленный на рис.2.1 и раскрывающий то, что целесообразно включать в АО. Концептуальный каркас в стандарте оформлен в виде семантической сети, связывающей определёнными отношениями основные понятия архитектуры АС. В пояснениях к каркасу не используются спецификации для форматов описания и/или других средств, обеспечивающих построение АО. За отбор подходящих средств и их использование несёт ответственность архитектор.

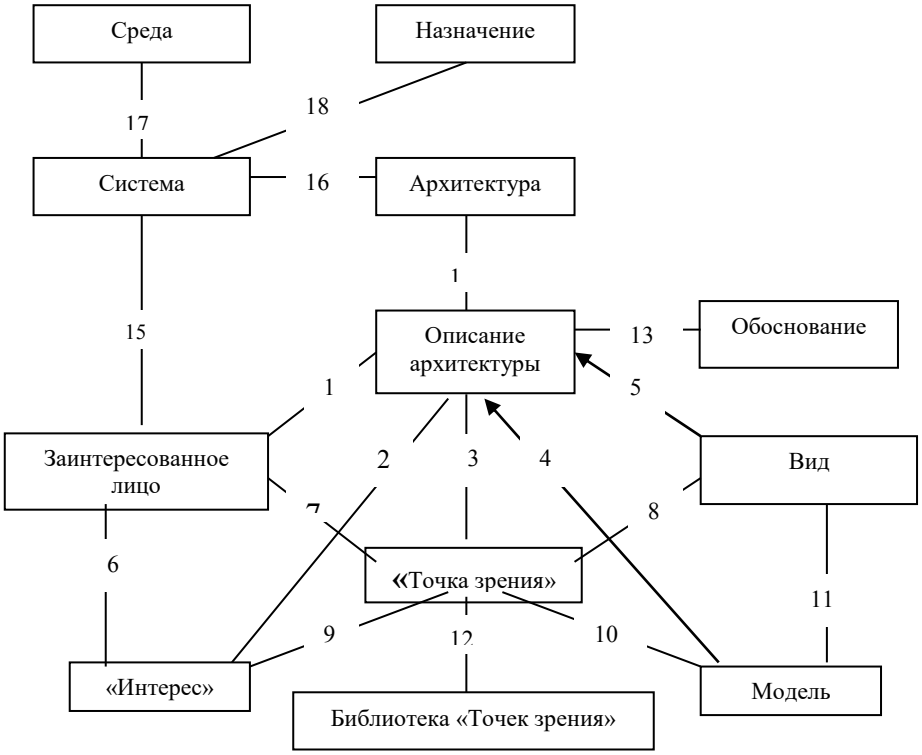


Рис.2.1.Стандартное описание архитектуры (отношения: 1 – различается, 2 – различается, 3 – выбирается, 4 – аггрегируется, 5 – состоит из видов, 6 – имеет, 7 – адресуется, 8 – реализуется, 9 – покрывает, 10 – устанавливает форму, 11 – состоит из моделей, 12 – содержит, 13 – убеждает, 14 – представляет, 15 – использует, 16 – имеет, 17 – рзмебщена, 18 – обеспечивает)

В рамках концептуального каркаса указаны понятия (и их связи), относящиеся к понятию «архитектурное описание» («заинтересованные лица», «интерес заинтересованных лиц», «вид», «точка зрения» и другие детали) и контексту его употребления (система, среда, миссия, архитектура). За каждым понятием стоит его (потенциальное или реальное) материальное воплощение определённой составляющей (или ряда составляющих) процесса и/или результата разработки АС. Факт отсутствия такого материального воплощения указывает на отклоне-

ние от схемы, которое (как показывает практика разработок АС) может привести к негативным последствиям разработки.

Отметим, что на рис.2.1 указаны связи между узлами, но не приведены (чтобы не загромождать рисунок) характеристики степени связи, то есть кратности «один к одному (1..1)», «один ко многим» (1..N) и др.. Степень связи легко установить, проведя оценочный анализ.

Выборочно представим содержание узлов схемы. Толкование ряда особо важных понятий каркаса (архитектурной концептуальной схемы АС) проведём с позиций тех требований, которые они выражают в системе требований к АС:

1. Различные значения понятия «Лица, заинтересованные в разработке АС. Stakeholders» вводятся для того, чтобы на множестве таких лиц определить типовые роли, исполнение которых осуществляют, в общем случае, группы лиц. Необходимо всегда помнить и принимать в расчёт, что архитектура разрабатывается для отмеченных лиц, архитектура должна отражать их требования и интересы, адекватное определение группы повышает шансы на успех разработки.

К типовым ролям относятся :

- лица (Acquirers), приобретающие систему;

- эксперты-консультанты (Assesors), проверяющие целесообразность приобретения;

- пропагандисты (Communicators), распространяющие сведения об этом через документы и обучение;

- разработчики (Developers), создающие систему;

- сопровождающие (Mainteners), внедряющие, сопровождающие и развивающие систему;

- снабженцы (Suppliers), поставляющие «комплектующие части»;

- пользователи (Users), обеспечивающие использование системы по назначению;

- администраторы (Administrators), обеспечивающие функционирование системы;

- тестировщики (Testers), проверяющие корректность работы системы.

В публикациях и конкретных инструментально-технологических системах разработки АС называют и используют не только названные роли, но и многие другие. Так, например, в RUP различается около 70 ролей stackeholders.

Определение группы и конкретных лиц выполнено правильно, если эти лица информированы (что позволяет им принимать хорошие решения), способны принять на себя обязательства и ответственность (даже если решения трудные) и авторитетные (что повышает степень доверия к их решениям).

2. За понятием «Интерес (Concern)» стоит то, что принято называть интересом лица, находящегося в определённой роли к процессу разработки АС, например, «интерес к тем функциям, которые исполнимы в АС, а значит к поведению АС», «интерес к защищённости АС по отношению к несанкционированному доступу» или «интерес к позитивным эффектам, достижимым при использовании АС». В общем случае конкретный «интерес» может выражать определённую заинтересованность группы лиц, исполняющих типовую роль, например, групп «разработчики» или «пользователи».

Выявить совокупность таких «интересов» и их значений, необходимых и достаточных для успешной разработки АС, является одной из первоочередных и важнейших задач концептуального проектирования АС, ответственного за «систему требований», «архитектуру» и «концептуальный проект».

3. Понятие «Точка зрения. Viewpoint» раскрывает позицию, с которой определённая группа лиц или группа групп, заинтересованных в разработке АС, желает, привыкла, готова или в состоянии представить АС как целое. С определённой «Точкой зрения» в архитектурном описании связывают, в общем случае, интегрированную совокупность родственных «Интересов», например,

4. Понятие «Вид. View» является родственным для «Видов», используемых в машиностроительном или строительном черчении. Примером совокупности видов, используемых в других приложениях, служит набор схем, представленных на рис.2.2.

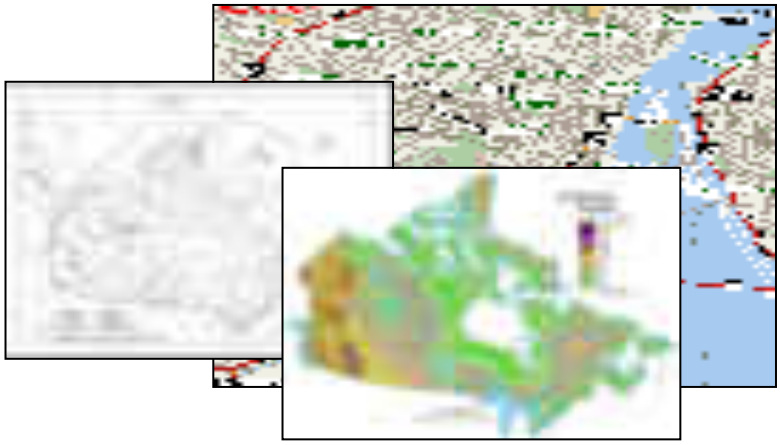


Рис.2.2. Набор картографических видов

Понятие «вид» в архитектурных описаниях АС соответствует «проекции АС» на определённый интерес или интересы. Для представления архитектурного вида используют, в общем случае, совокупность концептуальных моделей и документов, раскрывающих их содержание или дополняющих содержание, выраженное в моделях. Архитектурные виды АС принято визуализировать в форме, подобной визуализации, представленной на рис.2.3.

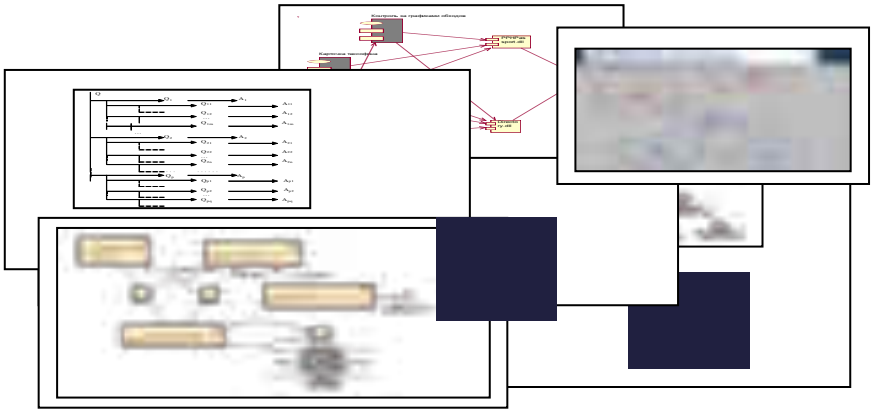


Рис.2.3. Визуализированный набор архитектурных видов

Виды в конкретной технологии разработки АС оформлены на определённом языке (например, на языке UML и/или других языках описания архитектуры, Architecture Description Language) и визуализированы в соответствии с определённым набором правил: каждый вид соответствует вполне определённой точке зрения (рис. 2.4); точка зрения является образцом для конструирования видов, причём точка зрения определяет правила не только для построения видов, но и для их использования.

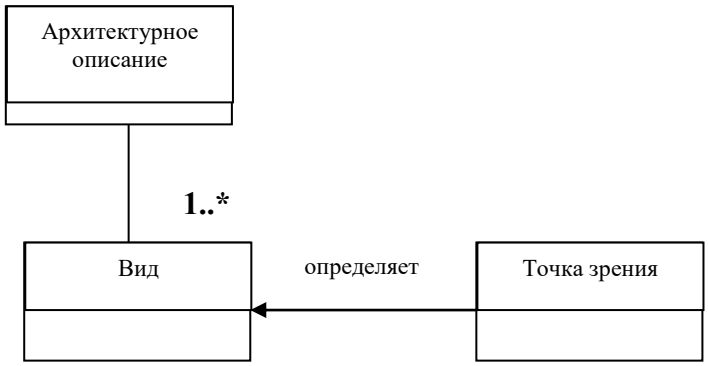


Рис.2.4.Фрагмент архитектурной семантической сети

Стандарт не фиксирует определённый набор видов, поскольку число разнородных АС и их приложений многообразно. В рамках вопроса об архитектурах АС не может быть окончательно решён вопрос об окончательном и полном наборе точек зрения и видов. В то же время точка зрения должна быть объявлена до её использования.

Виды в конкретной технологии разработки АС хорошо оформлены: каждый вид соответствует вполне определённой точке зрения; точка зрения является образцом для конструирования видов; точка зрения определяет правила не только для построения видов, но и для их использования.

Стандарт не фиксирует определённый набор видов, не может быть окончательно решён вопрос, откуда ещё могут прийти точки зрения, точка зрения должна быть объявлена до её использования.

Отношения между «точкой зрения», «интересами» {ci} «видами» «моделями» {mj} и документами {Dk}, обобщённо представленные на рис.2.5, завершают толкование основных понятий стандарта IEEE-1471–2000.

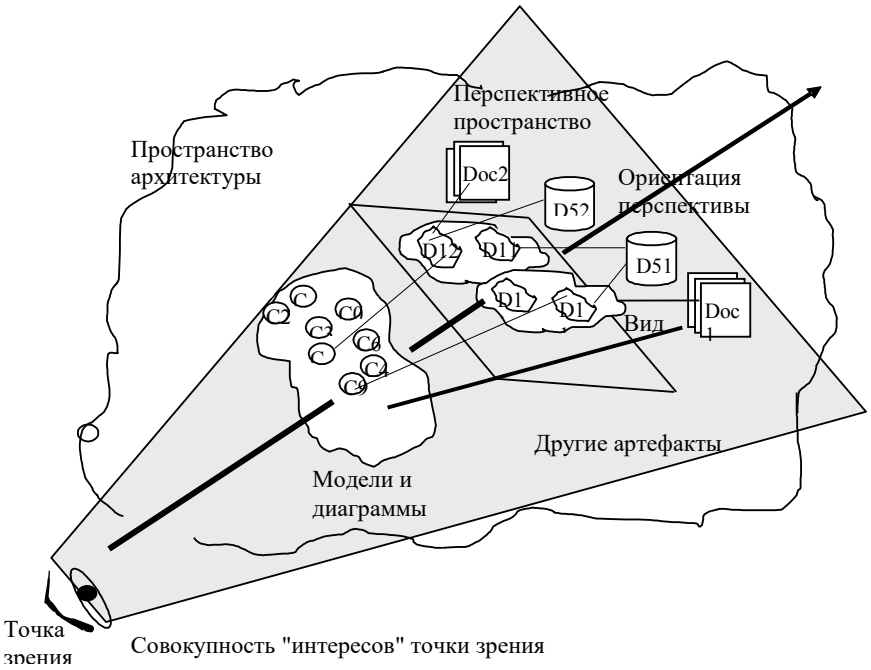


Рис.2.5. Образное представление отношений

Намерение использовать варианты употребления понятий стандарта IEEE-1471 в архитектурном описании предписывает определиться с их значениями, или, что то же самое, ввести архитектурные требования в систему требований АС и специфицировать эти требования. Следовательно, необходимо определиться:

- с «типовыми группами лиц», интересы которых следует или целесообразно учесть в разработке АС;

- с «интересами» типовых групп, провести их анализ и систематизировать;

- с «точками зрения» (возможно, выбрав подходящие в библиотеке «точек зрения»), на основании которых будет строиться архитектурное описание АС;

- с совокупностями «видов» на АС (возможно, выбрав подходящие в библиотеке «видов»), которые будут составлять архитектурное описание.

Решения, принятые относительно этих требований, определят самое важное для действий по проектированию архитектуры АС, в результате которого архитектурное описание наполнится деталями, в первую очередь подходящими концептуальными моделями и документами.

Для отмеченной работы с архитектурными требованиями полезны специальные табличные шаблоны (табличные образцы). Так, например, для объявления «точки зрения» можно использовать шаблон (таблица 2.1), указывающий на необходимость действий для заполнения его содержанием атрибутов.

Состав атрибутики и её значений для всех конструктов архитектурного описания, а не только для «точек зрения» должен быть таким, чтобы он позволял решать задачи на базе АО, в частности задачу выбора подходящих образцов для последующих (за этапом построения архитектуры) этапов жизненного цикла разработки АС.

Таблица 2.1.

Атрибут	**Значение**
Имя «точки зрения»	
Тип группы	
Интересы лиц	
Язык описания	
……	
Критерии целостности	
Критерии оценки	
Эвристики	
Образцы	
Руководства	

Стандарт IEEE 1471 отмечает необходимость использования архитектуры системы для решения таких задач, как следующие:

1. Анализ альтернативных проектов системы.
2. Планирование перепроектирования системы, внесения изменений в её организацию.
3. Общение по поводу системы между различными организациями, вовлеченными в её разработку, эксплуатацию, сопровождение, приобретающими систему или продающими ее.
4. Выработка критериев приёмки системы при её сдаче в эксплуатацию.
5. Разработка документации по её использованию и сопровождению, включая обучающие и маркетинговые материалы.
6. Проектирование и разработка отдельных элементов системы.
7. Сопровождение, эксплуатация, управление конфигурациями и внесение изменений и поправок.
8. Планирование бюджета и использования других ресурсов в проектах, связанных с разработкой, сопровождением или эксплуатацией системы.
9. Проведение обзоров, анализ и оценка качества системы.

2.2.3. Представления схемы IEEE-1471

Стандарт IEEE-1471 находит широкое применение на практике, что привело к его осмысливанию и переосмысливанию с различных позиций и применений. В таких действиях из семантического каркас стандарта либо извлекают часть сети, либо дополняют её или часть дополнительными фрагментами, либо строят родственные семантические сети.

На рис.2.6 представлено извлечение из семантической сети стандарта его наиболее существенной части, в которой раскрываются отношения между основными понятиями описания архитектуры. Фрагмент (рис.2.6) имеет более простой вид, удобный для практического использования в оперативных действиях при разработке архитектуры, для формирования документации на архитектуру и при изучении стандарта.

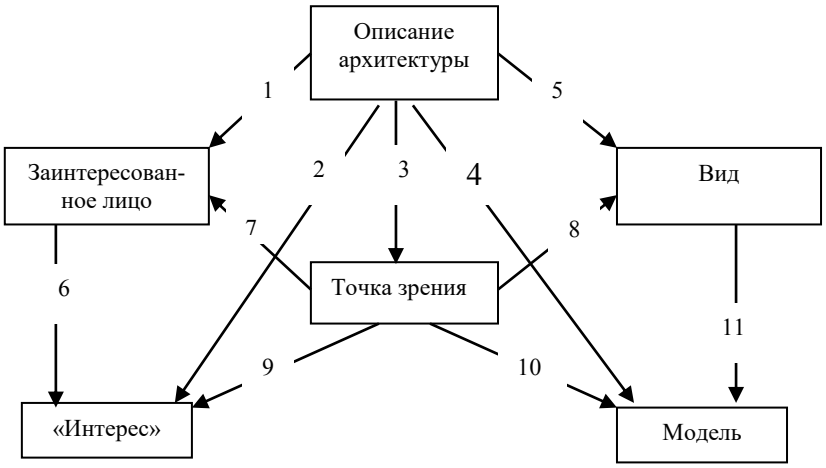


Рис.2.6. Фрагмент семантической сети IEEE-1471

В ряде публикаций и докладов G. Booch [10,14] связал со стандартом ряд мета моделей и рекомендовал их использовать в архитектурной практике. Одна из таких моделей представлена на рис.2.7. Число узлов и связей этой модели (по отношению к базовой концептуальной схеме IEEE-1471) расширено.

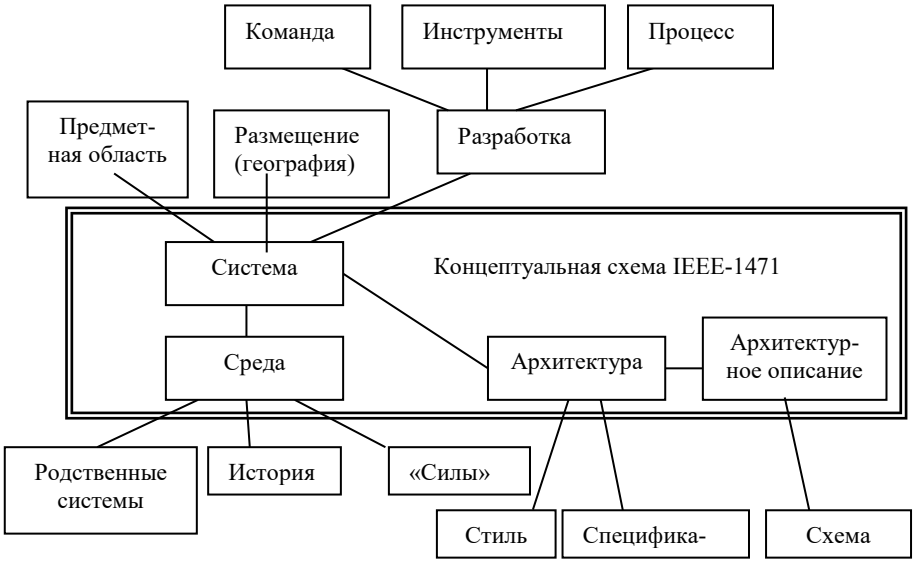


Рис.2.7.Расширенная семантическая сеть

Дополнительные узлы и связи, размещённые за рамками (двойная линия) схемы IEEE-1471, связаны с определёнными узлами схемы IEEE-1471. Добавлены:

- для узла «система» её отношение с «предметной областью», к которой она принадлежит, пространственное «размещение» системы и её «разработка»;

- узел «разработка» детализирован до «команды», проводящей разработку, «процесса» её деятельности и применяемых «инструментов»;

- в узел «архитектура» как составляющая добавлен «стиль» (в общем случае интегрированная совокупность стилей) и отражён тот факт, что архитектура имеет определённые «спецификации»;

- для «архитектурного описания» показано, что оно осуществляется по определённой «схеме» или, что то же самое, согласно определённому шаблону (template);

- по узлу «среда» отмечена целесообразность учёта «родственных систем», её «истории» и «силовых давлений», в том их смысле, который был раскрыт в п.1.3.

2.3. Архитектурные концептуальные схемы

2.3.1. Определение и ретроспектива

Как уже отмечалось выше, стандарт IEEE-1471является концептуальной схемой, по образцу которой рекомендуется описывать архитектуру АС. У этого стандарта были предшественники, а он сам использовался при построении других подобных конструктов. Таким образом, в практике архитектурных описаний АС разного типа сложился специфический класс «Архитектурных концептуальных схем» (Architecture Frameworks, AF) , принадлежность к которому конкретной «Концептуальной схемы» определяется по наличию у схемы вполне определённого набора признаков.

По сложившемуся пониманию конструкты AF являются обобщёнными прагматически руководствами:

- для описания архитектур, предоставляющими возможности не только для представления архитектур, но и для сопоставления описаний в совокупности архитектурных альтернатив;

- которые определяют системы видов и моделей, их целесообразно использовать при построении архитектуры, в том числе и для того, чтобы разделять, понимать и сравнивать архитектурную информацию;

- которые подсказывают, какими способностями архитектор/проектировщик должен овладеть, и как эти способности должны быть использованы в процессах построения архитектуры и её использования.

К числу особо важных составляющих AF относятся:

- «Виды», как средства для понимания, взаимопонимания и коммуникативного взаимодействия лиц, заинтересованных в разработке АС;

- «Методы», предоставляющие дисциплину, для того чтобы собирать и организовывать данные и конструкты «видов» способом , помогающим гарантировать целостность, точность и завершённость архитектурного представления АС;

- «Обучение/ опыт», что обеспечивает разумное и конструктивное применение методов и обслуживающих их инструментов.

2.3.2. Архитектурная концептуальная схема Дж. Захмана

Особое место среди архитектурных концептуальных образцов занимает концептуальная схема Дж. Захмана [35], представленная на рис. 2.8. Схема изначально нацеливалась на системное представление задач информатизации, в которых приходится учитывать специфику АС.

С момента своего создания (1987 год) схема Дж. Захмана постоянно усовершенствовалась, оставаясь определённого рода «вопросником» о существенных аспектах АС. Схема построена в форме табличной структуры, для формирования и использования которой необходимо построить систему ответов, каждый из которых заполняет определённую позицию таблицы. Ответ может быть как простым, так и сложным. Одной из наиболее распространённых форм ответов является визуализируемая концептуальная модель.

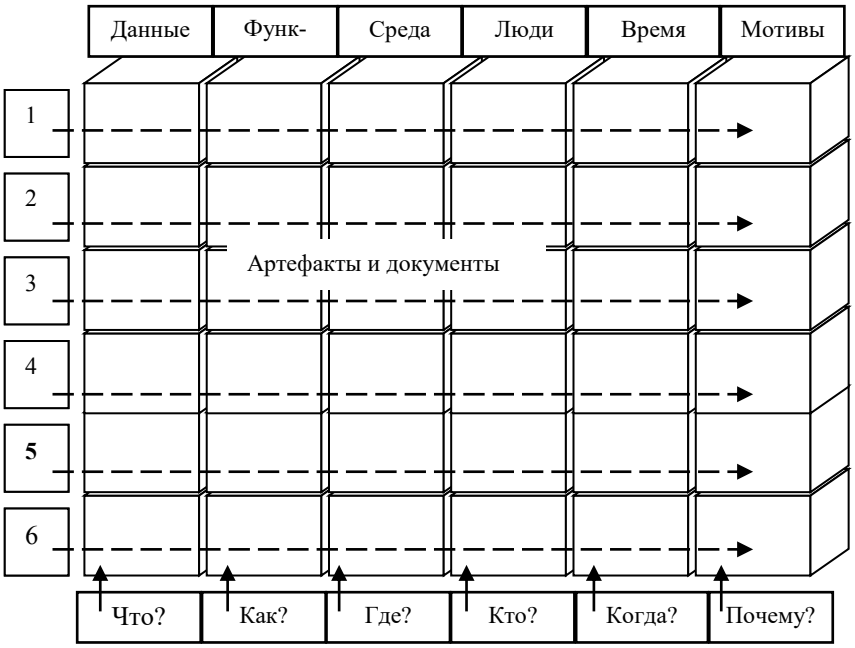


Рис.2.8. Архитектурная схема Дж. Захмана

То содержание, которое вкладывается в каждый ряд и каждый столбец схемы, представлено в таблицах 2.2 и 2.3 .

Таблица 2.2.

Ряд схемы	**Содержание**
1. Цели и границы (Вид планировщиков)	Определяет основные цели и границы, в рамках которых должны приниматься архитектурные решения
2. Модель предприятия, системы (Вид владельцев, ответственных за систему)	Определяет сущность предприятия (системы), включая структуры, процессы и организационные структуры
3. Модель базового понимания (информационного представления, Вид архитектора)	Определяет в более строгих терминах содержание ряда 2

4. Технологическая модель (Вид проектировщиков)	Определяет, как и какая технология должна быть использована для того, чтобы запросы третьего ряда были удовлетворены
5. Детализированное представление (Вид разработчика)	Определяет детали проектирования, применяемые языковые средства, структуры данных и другие средства
6. Функциональности системы (Вид пользователей)	Определяет, как должны работать системы в рамках предприятия (в контексте окружения)

Таблица 2.3.

Столбец	**Содержание**
1. Данные (Структура, Что?)	Фокусирует внимание на сущностях, объектах, компонентах и отношениях между ними
2. Функции (Активность, Как?)	Фокусирует внимание на той деятельности, которая осуществляется организацией и поддерживается автоматизированными системами
3. Сеть работ (Месторасположение, Где?)	Фокусирует внимание на географическом (физическом) распределении активностей
4. Люди (Кто?)	Фокусирует внимание на тех лицах, которые вовлечены в бизнес-процессы
5. Время (Когда?)	Фокусирует внимание на эффектах, связанных со временем (планирование, события)
6. Мотивации (Почему?)	Фокусирует внимание на целях, стратегиях и ограничениях, специфических для реализации системы

Использование этой схемы предполагает, что лица, отвечающие за архитектуру системы, в том числе и типа АС, заполнят «клетки» подходящим содержанием. Разумеется, содержание «клеток» включает спецификации (на уровне архитектуры) тех решений, которые по этой «клетке» приняты. В результате будет образована целостная архитектурная картина представляемой системы,

например АС, которая может быть использована в решении большинства из названных выше (п.2.3.1) задач.

Заметим, что Дж. Захман разрабатывал свою схему для предприятия, использующего любые информационные технологии, но схема находит широкое применение в теории и практике АС. Так, например, в работе [35] предложено развитие схемы Дж. Захмана до трёх измерений (рис.2.9), за счёт перехода к многослойной структуре системы моделей.

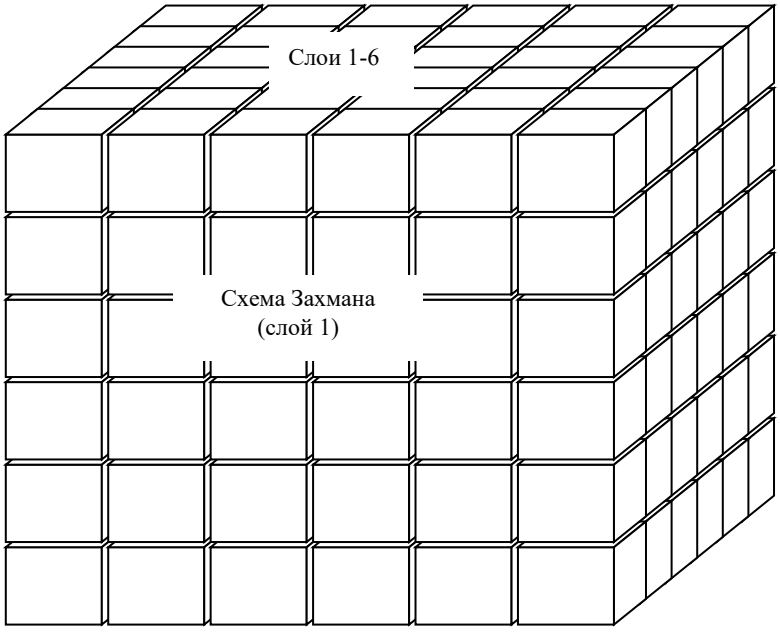


Рис.2.9. «Трёхмерная» схема Захмана

В трёхмерную структуру введены следующие дополнения, детализирующие содержание артефактов, которые приписываются «клеткам схемы Дж. Захмана»:

– слой разработки, раскрывающий «Что?» должно быть сделано для каждого артефакта, приписанного к «клетке» схемы Дж. Захмана;

– слой метаданных, включающий описание информации, которая должна быть собрана и «Как?» она должна быть организована;

– слой документирования, раскрывающий типовые формы документов, в которых (Где?) будет зафиксирована информация;

– слой ответственности, фиксирующий, «Кто?» из команды разработчиков будет исполнять требуемые работы;

– слой жизненного цикла, показывающий, «Где?» в процессе разработки будет создан соответствующий артефакт;

– слой целей, описывающий, «Почему?» следует включать артефакт в состав архитектурного описания.

Для предлагаемого расширения автором трехслойной модели разработаны детали для каждого слоя (и для каждой клетки слоя).

2.3.3. Архитектурная концептуальная схема DoDAF

Эволюция архитектурной концептуальной схемы DoDAF (Department of Defense Architecture Framework), созданной по заказу Департамента обороны США [77], учитывает всё новое, что появляется в теории и практике предметной области «Архитектура АС».

Архитектурное описание АС в DoDAF строится на базе модели данных ядра архитектуры (Core Architecture Data Model, CADM) и артефактов архитектуры. CADM представляет собой стандартизованную систему понятий для определения видов и их составляющих в базе данных.

Вид понимается традиционно и представляет определённый объем операций, систем и технических стандартов. Каждый вид является хорошо определённым множеством элементов данных в составе CADM. Архитектурные артефакты документируются с использованием языка UML.

Виды объединены в четыре группы. Первая группа называется «Все виды» (All Views) и содержит обзор, обобщения и интегрированный словарь по архитектуре DoDAF. Остальные группы – это «Операционные виды» (*Operational Views*), «Системные виды» (System Views) и «Технические виды» (*Technical Views*). Обобщённая картина видов представлена на рис.2.10.

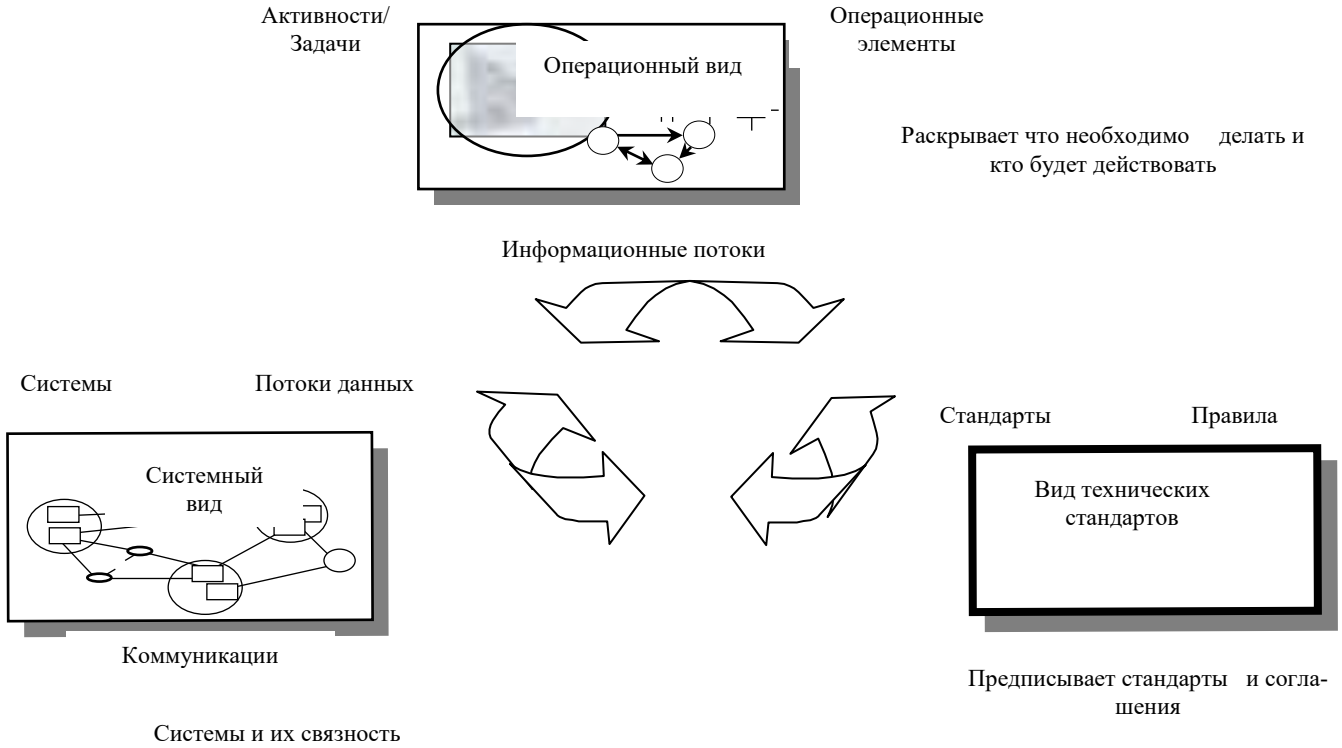


Рис.2.10. Архитектурная схема DoDAF (52 вида)

Операционные виды описывают специфику деятельности и операции, узлы операций, узлы соединений, информационный обмен, организационные отношения, правила операций, последовательности событий и логическую модель данных. Описания, содержащиеся в этой группе, распределены по 24 подвидам. (каждый подвид является видом по определению IEEE-1471).

Системные виды описывают систему и её компоненты, системные интерфейсы, коммуникативное взаимодействие, матрицу отношений на множестве систем и подсистем, функциональности, матрицу связности (traceability), эволюцию, использование и технологические предсказания. Группа состоит из подвидов.

Технические виды (группа из 12 подвидов) описывают текущий стандартный профиль и предсказания по его изменениям.

Архитектурная концептуальная схема DoDAF разработана для военных нужд и в этом плане она является предметно-зависимой. Следует отметить, что DoDAF предоставляет ограниченные возможности по моделировании конфигурации ПО и моделированию нефункциональных требований.

Ближайшим родственником архитектурной схемы DoDAF является схема MoDAF, разработанная европейскими союзниками США. В число основных потребностей, приводящих к отличиям схемы MoDAF от схемы DoDAF, входят следующие потребности:

1. Потребность решать и моделировать задачи приобретения разного рода комплектующих аппаратного и программного типов.
2. Потребность моделировать трансформационные программы и их взаимозависимости.
3. Потребность моделировать компетентность и другие виды способностей лиц, вовлечённых в разработку и использование АС.
4. Потребность моделировать необходимые для разработки АС кадровые ресурсы, подобно моделированию технических ресурсов.
5. Потребность объединять информационные представления в традиционные модели архитектуры, чтобы обеспечить необходимой информацией работу архитекторов предприятия.

Средства DoDAF согласованы с концептуальной схемой Захмана. Объём пересечений между схемами (отмеченный овалами) отражён на Рис.2.11. В версиях DoDAF указывается, что они согласованы со стандартом IEEE-1471.

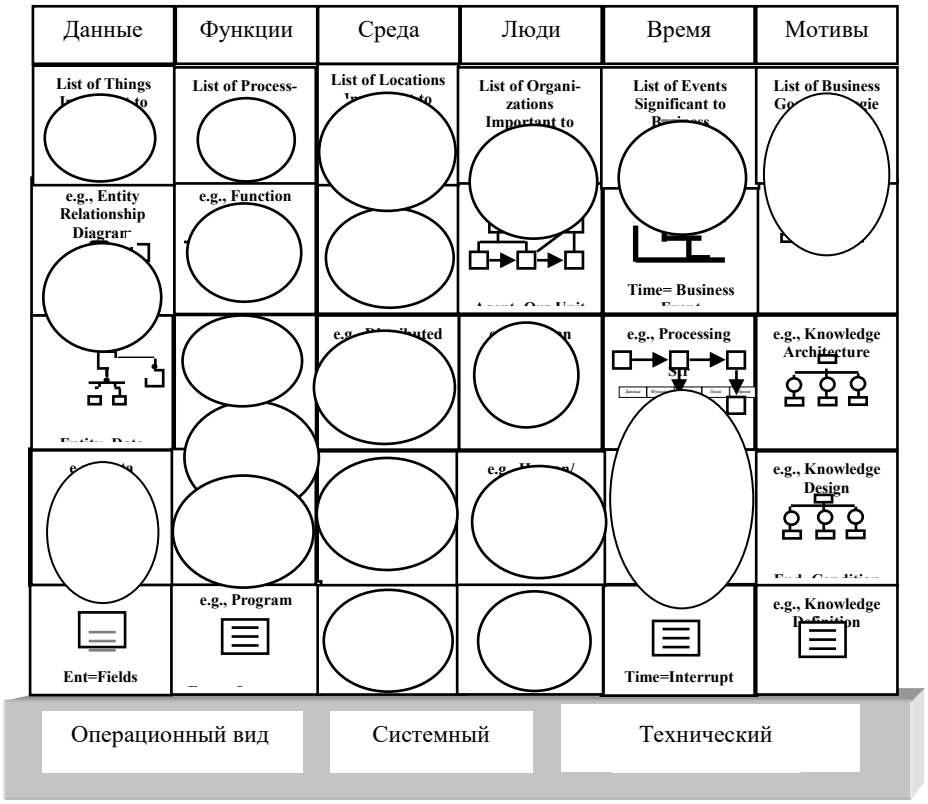


Рис.2.11. Проекция системы DoDAF на схему Дж. Захмана

2.3.4. Архитектурная концептуальная схема TOGAF

С момента её первого опубликования в 1995 году архитектурная концептуальная схема TOGAF (The Open Group Architecture Framework) развивается и используется. В настоящий момент времени доступна её версия TOGAF 8.1.1.

Сущность схемы TOGAF хорошо отражает жизненный цикл «Архитектурного описания АС» (рис.2.12), построенный в соответствии с рекомендациями этой схемы.

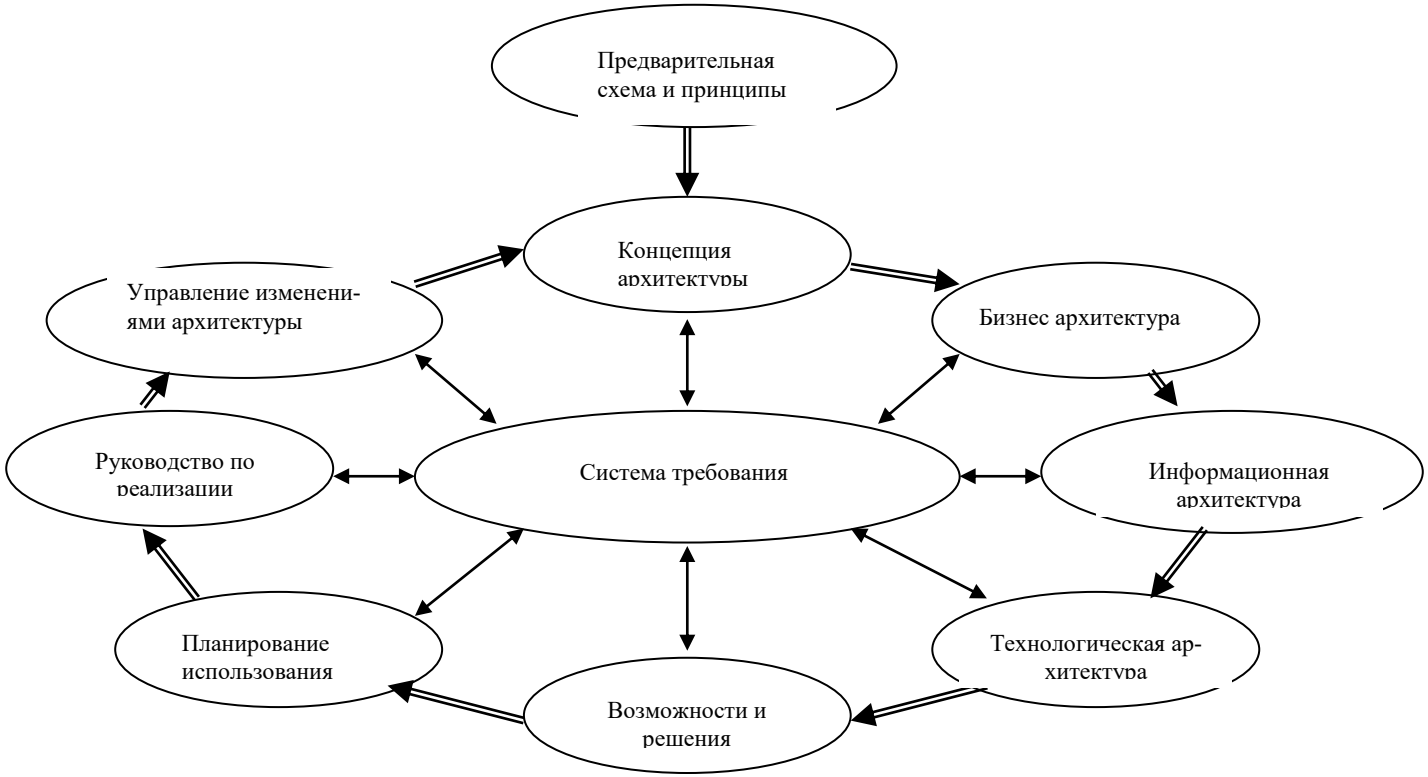


Рис.2.12. Жизненный цикл архитектуры АС по TOGAF

За каждым этапом (обозначены овалами) жизненного цикла стоит проверенная методика и руководство по использованию методики. Детальное описание концептуальной схемы TOGAF, включающее описание каждой из методик, представлено на сайте разработчиков TOGAF [80].

Применение TOGAF осуществляется как итеративный процесс. Итерации TOGAF (которые в руководствах называются циклами) обычно отличаются большей длительностью, чем итерации RUP, и объединяют несколько фаз реализации и обслуживания архитектуры предприятия. Это обусловлено тем, что область архитектуры предприятия шире области одного проекта RUP.

2.3.5.Архитектурная схема FEAF

История схемы FEAF (Federal Enterprise Architecture Framework) началась в 1998 году. Её специфика (рис.2.13) связана с её предназначением – разработка АС в рамках системы задач государственного масштаба для США. Подобные задачи в нашей стране намечено решать в рамках «Электронной России». Последняя версия FEAF доступна по адресу www.eaframeworks.com/**FEAF**.

Федеральная Архитектура – это концептуальная модель описания в координированной, структурированной форме деятельности федерального правительства и государственных организаций с функциональной точки зрения, вне зависимости от организационных структур, реализующих соответствующие функции, с целью улучшения их деятельности за счет использования информационных технологий.

По сути дела, это новый способ описания, анализа и улучшения деятельности государства и госорганизаций, а также расширения их возможностей по обслуживанию граждан. Федеральная Архитектура – это стратегический информационный актив, который определяет функции (бизнес) государственных организаций, информацию и технологии, необходимые для реализации функций, а также процессы преобразований, необходимые для внедрения новых информационных технологий в ответ на изменяющиеся бизнес-потребности. Отдельные государственные организации должны использовать эту общую модель для описания своих собственных архитектур.

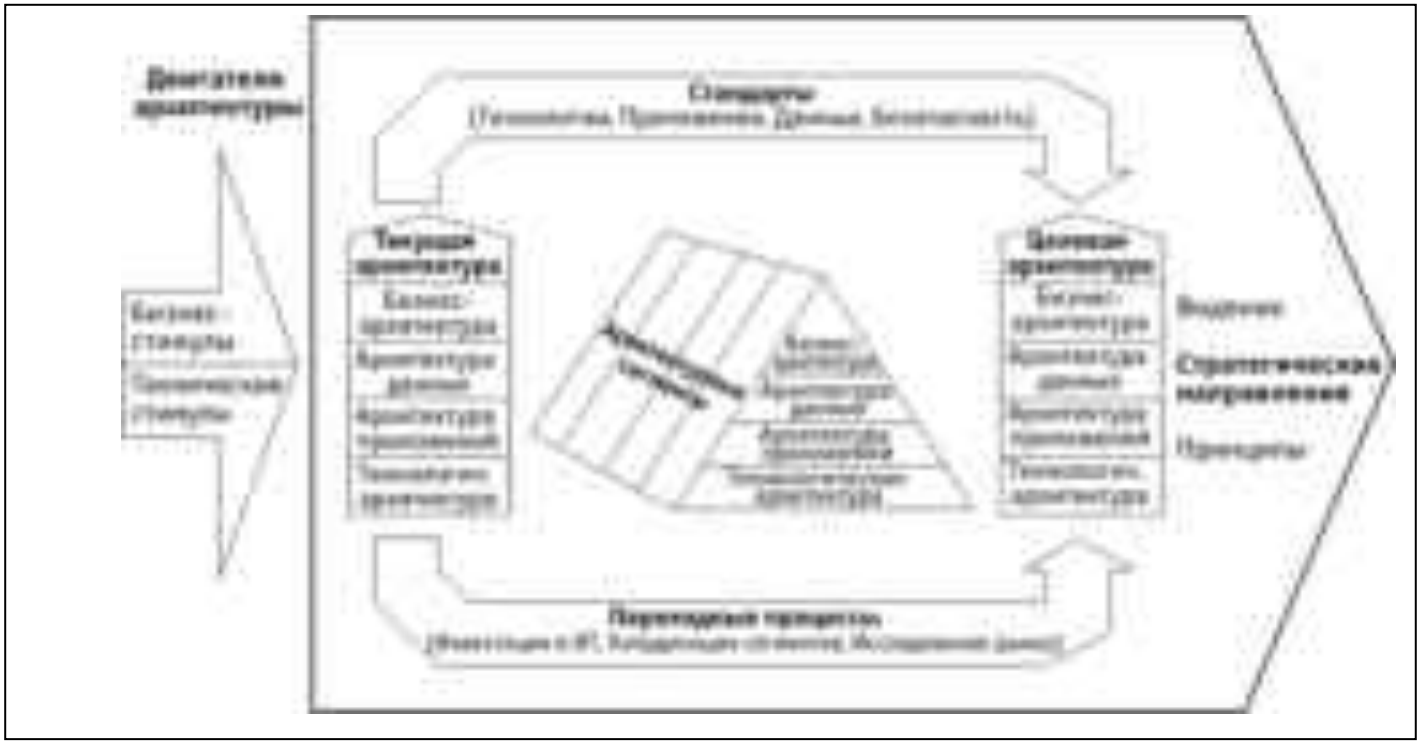


Рис.2.13. Архитектурная схема FEAF

2.4. Сравнительное сопоставление систем архитектурных видов

2.4.1. Проблема стандартной концептуальной схемы

Представленные выше архитектурные концептуальные схемы указывают на то, что организации и группы создают и поддерживают эволюционное развитие «собственных» понятийных схем. Одним из объяснений этого служит разнообразие предметных областей АС, каждая из которых имеет специфику, которую и стараются учесть в «собственной » концептуальной схеме. Ещё одной причиной, возможно, является состояние теории и практики архитектуры АС, которые ещё не достигли состояния, в котором можно было бы создать единую архитектурную концептуальную схему, детализирующую, например, стандарт IEEE-1471 до типологии видов, моделей и документов. К решению такой задачи можно продвинуться через сопоставление различных концептуальных схем и различных систем видов. Два примера таких сопоставлений приведены ниже.

2.4.2. Сопоставление систем видов

2.4.2.1. Архитектура «4+1»

Архитектура конкретной АС описывается множеством ее **представлений. В** архитектуре АС фиксируются главные решения проектирования структуры, показывающие, как приложение, вложенное в АС, разделено на **компоненты,** и как связаны эти компоненты для получения полезной **конфигурации.** Эти решения должны вытекать из **требований** функциональности и других факторов. С другой стороны, эти решения устанавливают дальнейшие **ограничения** на требования и на будущие проектные решения нижнего уровня.

При использовании подхода «4+1» разработчики АС начинают формирование архитектуры с типичного набора видов, представленного на рис.2.14.

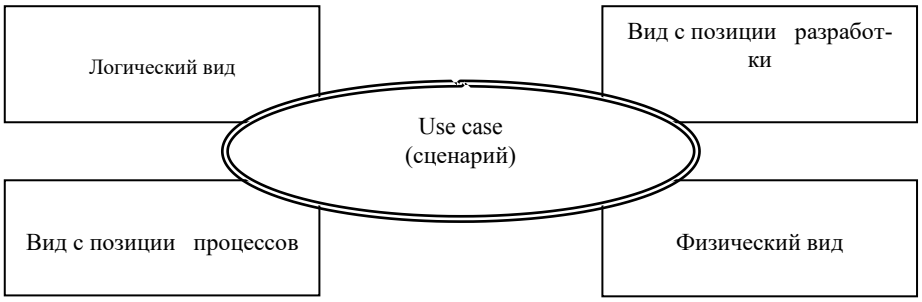


Рис. 2.14.Модель «4+1»

Набор видов включает:

1. Use case вид , который содержит прецеденты и сценарии, охватывающие архитектурно существенное поведение, классы или технические риски. Вид является подмножеством модели прецедентов.

2. Логический вид, который содержит наиболее важные классы проекта, их организацию в пакеты и подсистемы различных уровней. Здесь содержится уже некоторая реализация прецедента. Это представление является подмножеством модели проекта

3. Вид с позиции разработки, который содержит краткий обзор модели выполнения в терминах модулей пакетов и уровней. Здесь описывается также распределение пакетов и классов (из логического представления) в пакетах и модулях представления выполнения. Это представление является подмножеством модели выполнения.

4. Вид с позиций процесса, содержащий описание задач (процессов и нитей), их взаимодействия и конфигурации и распределения объектов и классов по задачам. Потребность этого представления возникает, только если система имеет существенную степень параллелизма. В Rational Unified Process 5.1 представление процесса – это подмножество модели проекта.

5. Физический вид, который содержит описание различных физических узлов для наиболее типичных конфигураций платформы, и расположение задач (из представления процесса) по физическим узлам. Потребность этого представления возникает, только если система распределена.

Дополнительные представления, например для представления интерфейса пользователя, представления защиты и представление данных не отвергаются, но ответственность за их связывание с базой «4+1» ложится на разработчиков АС.

Перечисленные выше представления могут изобразить проект системы в целом. Но архитектура интересуется только некоторыми определенными аспектами:

Структура модели – организационные элементы, например, иерархическое представление.

Существенные элементы – критические прецеденты, главные классы, общие механизмы и так далее, а не все элементы, представленные в модели.

Несколько **ключевых сценариев,** показывающих главный поток управления в системе.

Сервис, чтобы зафиксировать модульность, необязательные особенности, аспекты производственной линии.

*2.4.2.2. **Архитектурные решения SEI***

В Институте Программной Инженерии (SEI) разработан подход к формированию архитектуры, исходящий из того, что для конкретной АС, кроме базовых «точек зрения», может потребоваться дополнительный набор видов. А значит, архитектору должны быть предоставлены возможности создания необходимых ему типовых видов, в частности «box and line» средства.

Предлагается комплект средств выражения и методик для такой работы. Но главный акцент предлагается направить на работу с видами с позиций качества (рис.2.15). Значения характеристик качества должны оцениваться и, если значения не соответствуют требованиям, то в построении архитектуры долен происходить «возврат» к предшествующим шагам архитектурного моделирования для внесения коррекций в архитектуру или даже для принятия новых архитектурных решений.

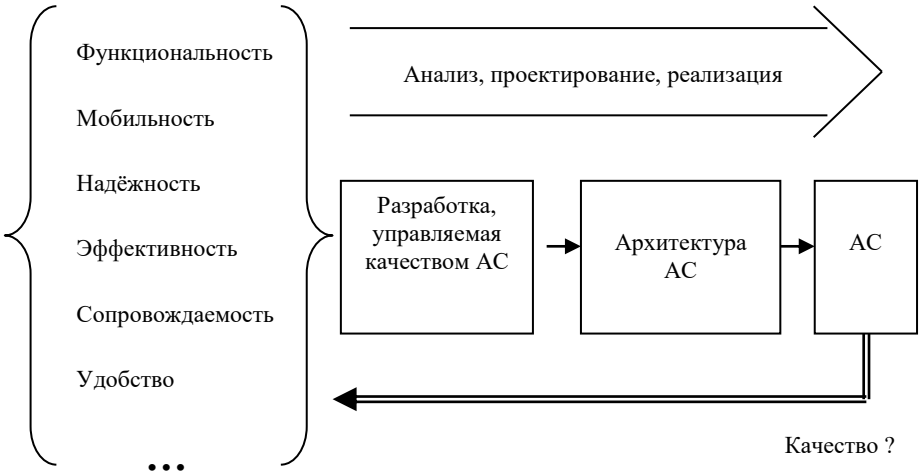


Рис.2.15. Архитектура, управляемая качеством АС

Архитектурные решения SEI согласованы со стандартом ИСО/МЭК 9126. Особое внимание рекомендуется уделять рассуждениям в группе разработчиков и документированию решений.

*2.4.2.3. **Архитектурные решения RM ODP***

Архитектурные решения RM-ODP [88] предоставляют пять точек зрения (рис.2.16) на систему:

– организационная точка зрения описывает постановку цели, области применения, способы и правила применения;

– информационная точка зрения описывает выражение (проявление) и семантику обрабатываемых системой данных, форматы и модели данных;

– компонентная (вычислительная) точка зрения описывает разделение приложения на функциональные модули и определяет их интерфейсы;

– инженерная точка зрения представляет распределение отдельных элементов системы по физическим ресурсам, а также их связи;

– технологическая точка зрения описывает технологии, используемые при реализации системы.

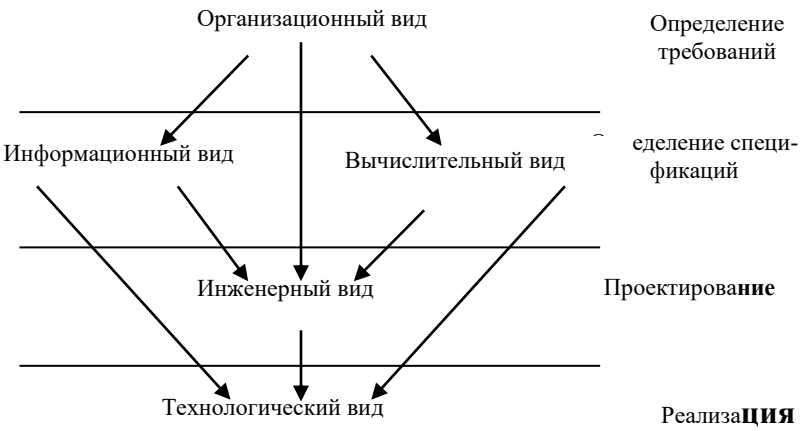


Рис.2.16.Архитектура RM ODP

При помощи этих пяти точек зрения могут быть описаны как существующие системы, так и разработаны новые АС и приложения. Стандарт содержит лингвистические средства для описания видов и систему правил для переходов между видами.

2.4.2.4. Архитектурные решения SIMENS

В архитектурных решениях Siemens (рис.2.17) использование «концептуального вида» в системе «видов» было предложено до создания языка UML, в котором необходимость визуального концептуального моделирования в разработках АС была переведена на уровень стандарта.

Система видов включала:

– концептуальный вид как описание архитектуры системы в понятиях, раскрывающих основные элементы АС и связи между ними;

– модульный вид, раскрывающий функциональную декомпозицию АС и её распределение по уровням детализации;

– вид исполнения, специфицирующий динамическую структуру АС;

– вид с позиций кодов, включающий кодовые компоненты и их библиотечную организацию в среде разработки.

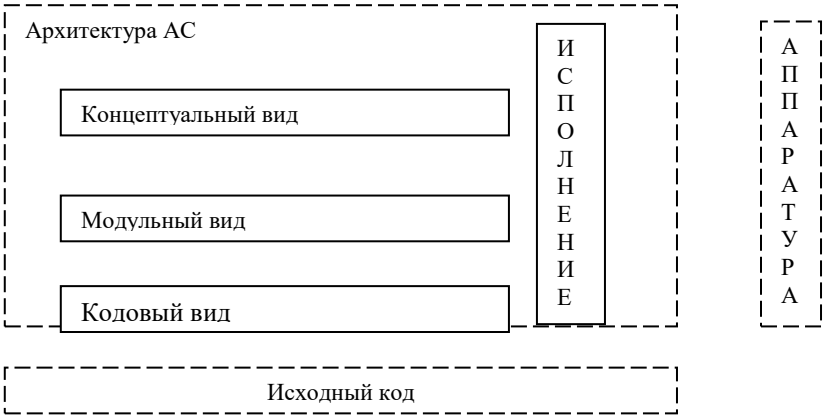


Рис.2.17.Архитектурные решения Siemens

*2.4.2.5.**Архитектурные решения ADS***

Подход «Спецификации рационального описания архитектуры (Rational ADS)» развивает подход «4+1» до его применений, включающих автоматизированные производственные системы, встроенные системы и другие системы, в которых может отсутствовать программное обеспечение. Вариант расширения представлен на рис.2.18.

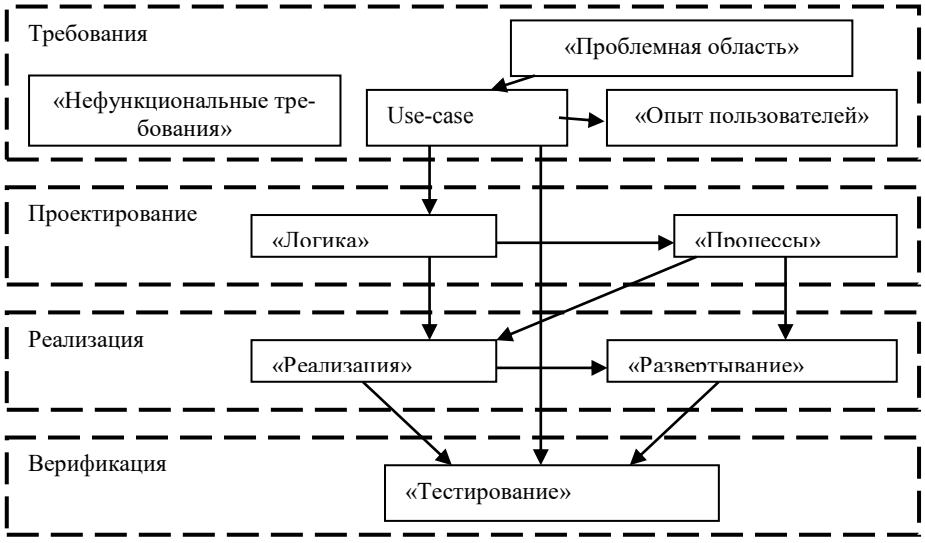


Рис.2.18. Архитектура SAD

В архитектуре виды распределены между четырьмя «точками зрения» (Требования, Проектирование, Реализация и Верификация). Содержание видов соответствует содержанию видов, рассмотренных выше, или понятно из названия. Архитектура нашла своё воплощение в инструментально-технологической среде RUP. Она согласована со стандартами ISO/MEK-12207 и IEEE-1471.

2.4.2.6. Сопоставление образцов архитектур

Перейдём к сопоставлению представленных архитектурных образцов, исходя из возможностей заимствования архитектурных решений для разработок АС. В сопоставлениях ограничимся «возможностью поддержки активности основных ролей, исполняемых членами группы разработчиков» и «наличием средств выражения основных «интересов». Результаты сопоставлений приведены в таблице 2.4 и таблице 2.5.

Таблица 2.4.

Образцы / Роли	**4+1**	**SEI**	**RM-ODP**	**Siemens**	**Rational ADS**
Количество точек зрения			5	4	4 / 9
Архитектор			0	4	3
Проектировщик			0	0	4
Кодировщик			5	0	3
Интегратор			0	0	1
Тестировщик			0	0	1
Специалист по качеству			0	0	0.5
Специалист по управлению			0	0	1
Пользователь			0	0	3
Специалист по руководствам			5	0	3

Таблица 2.5.

Образцы / Роли	«4+1»	SEI	RM-ODP	Siemens	Rational ADS
Количество точек зрения			5	4	4 / 9
Функциональность			3	2	3
Мобильность			1	2	2
Надёжность			2	1	2
Эффективность			1	2	3
Сопровождаемость			1	1	3
Удобство			0	0	2
Безопасность			0	0	1

Сопоставление указывает на то, что общая позиция по архитектурным представлениям АС ещё не устоялась. Системы архитектурных норм при разработках очередных АС, особенно такого типа, как КСА (Комплексы Средств Автоматизации), целесообразно пересматривать, опираясь на текущий опыт архитектурных решений в области программной инженерии.

Выбирая систему архитектурных норм для очередного проекта КСА, следует исходить из взаимодополнительности образцов, доказавших свою полезность на практике. Функции таких образцов способны выполнить образцы, представленные выше.

2.5 .Сопоставление концептуальных схем

Ещё одним примером сопоставления архитектурных концептуальных систем может служить работа [69], в которой (из-за разнородности систем) проводится сопоставление по трём фундаментальным позициям: «цели», «вход» и «выход-результат».

Если конкретная архитектурная система явно поддерживает содержательный элемент строки, то этот факт фиксируется символом «+». Если содержание строки явно не поддерживается или это не отмечается в документации, то в таблице используется символом «-». Частичная явная поддержка отмечается символами «+-». Степень поддержки в таблице не представлена.

Таблица 2.6.

Характеристики	ZF	4+1	FEAF	RM-OD+-	TOGAF	DoDAF
Цели						
Определение архитектуры и её понбимание	+-	+-	+	+	+	+
Процесс архитектурного моделирования	-	-	+	-	+	+
Поддержка эволюции архитектуры	-	-	+	+-	+	+
Анализ архитектуры	+	+	+	+	+	+
Архитектурные модели	+	+	+	+	+	+
Альтернативы проектирования	+-	+-	+-	+-	+-	+
Обоснование проектирования	+-	+-	+-	+	+	+-
Стандартизация	-	-	+-	+	+	+
База знаний архитектуры	-	-	+	+	+	+
Верифицируемость архитектуры	-	+-	-	+-	+	-
Входы						
Методики	+-	+-	+	+-	+	+
Включенность в технологии	-	-	+	+-	+	+
Функциональные требования	+	+	+	+	+	+
Информационная поддержка	+-	+-	+	+	+	+
Текущая архитектура	+-	+	+	+	+	+
Нефункциональные требования	+-	+	+-	+	+	+-
Результаты						
Модель предметной области	+	+-	+	+	+	+
Модель системы	+	+	+	+	+	+
Информационная модель	+	+	+	+	+	+
Вычислительная модель	+	+	+	+	+	+
Модель конфигурации ПО	-	+	-	+-	+	-
Модель процесса	+	+	+	+	+	+
Модель реализации	+-	+-	+-	+	+	+
Платформы	+	+-	+	+	+	+
Проектирование качества	+-	+	+-	+	+	+-
Преобразования в проектировании	-	-	+	-	+	+
Обоснования в проектировании	-	+-	-	+-	+-	+-

2.6. Примеры систем видов

В архитектурной практике находят применение и другие системы видов, содержащие виды, отличающиеся от представленных выше. Одной из таких систем, связанной со специфическим взглядом на качество АС является система видов (Таблица 2.7), предложенная в монографии Rozanski и P.Wood [73].

Таблица 2.7.

Вид	Содержание
Информационный	Фиксирует историю, преобразования, управление и распределение информации. Используются подходящие модели, отражающие статику и потоки данных. Цель анализа – ответить на существенные вопросы о содержании данных, структуре, принадлежности, умолчаниях, ссылках и перемещениях
Согласованности	Описывает согласованность структуры АС и карту функциональных компонентов, что позволяет ясно идентифицировать части системы , управляемо выполняющих согласованные действия. Это влечёт за собой создание моделей, демонстрирующих процессы и трассы их реализации с учётом коммуникативных отношений
Разработка	Раскрывает архитектуру, которая поддерживает процесс разработки АС. Указывает на определённые аспекты разработки, требующие включения в процесс разработки специалистов определённых квалификаций (например, тестирование, сопровождение)
Размещение	Описывает среду, в рамках которой система будет развёрнута, с учётом зависимостей системы от «реального времени» среды. Вид включает аппаратуру не только системы, но и её среды(например, сетевое окружение)
Операционный	Раскрывает как система инсталлируется, функционирует (используется), администрируется и поддерживается. В построении вида целесообразно использовать уровень задач

К этой системе видов в пособии будут дополнения, раскрывающие подходы авторов к учёту характеристик качества и построениям архитектуры.

Ещё один пример системы видов взят из публикаций [59], в которых предлагается целостный взгляд на архитектуру АС, получивший название SPAMMED. Система SPAMMED базируется на следующей системе видов:

1. Вид с позиции системы требований (Requirements over_View).

2. Вид с позиций сервисов (Service View).
3. Вид с позиций инфраструктуры ПО (Software Infrastructure View).
4. Вид процессов (Process View).
5. Вид предметной области АС (Business Domain View).
6. Вид размещения (Deployment View).
7. Вид среды разработки (Development Environment View).
8. Вид с позиций коммерческой и компьютерной безопасности (COMMSEC & COMPUSEC View).
9. Вид с позиций сохранности (Safety View).

Вопросы по второй главе

Q1. Чем по своей сути является стандарт IEEE-1471-2000?
Набор стандартизированных методов разработки
Ряд строго зафиксированных шаблонов построения АС
"Рекомендуемая практика" предоставляющая разработчикам АС рекомендации для описания архитектуры

Q2. Кто выбирает форматы спецификации концептуального каркаса и несёт за них ответственность?
Пользователь
Эксперт-консультант
Администратор
Архитектор

Q3. Чем являются "проекции АС" на определённый интерес или интересы в IEEE-1471-2000?
Архитектурными профилями
Видами
Точками зрения
Моделями АС

Q4. Что формирует точку зрения в IEEE-1471-2000?
Виды
Перспективы
Цели разработки
Интересы

Q5. Что определяет архитектурный вид?
Интересы
Язык описания архитектуры
Точка зрения
Цели разработки

Q6. Как понимаются виды в контексте "Архитектурных концептуальных схем"?

Средства для понимания, взаимопонимания и коммуникативного взаимодействия лиц, заинтересованных в разработке АС

Средства, помогающие гарантировать целостность, точность и завершённость архитектурного представления АС

Средства, обеспечивающие разумное и конструктивное применение методов и обслуживающих их инструментов

Q7. Что определяет вид планировщиков в архитектурной концептуальной схеме Дж. Захмана?

Определяет сущность предприятия (системы), включая структуры, процессы и организационные структуры

Определяет основные цели и границы, в рамках которых должны приниматься архитектурные решения

Определяет как технология должна быть использована для того, чтобы запросы третьего ряда были удовлетворены

Q8. Что определяет вид пользователей в архитектурной концептуальной схеме Дж. Захмана?

Определяет как должны работать системы в рамках предприятия (в контексте окружения)

Определяет основные цели и границы, в рамках которых должны приниматься архитектурные решения

Определяет как и какая технология должна быть использована для того, чтобы запросы третьего ряда были удовлетворены

Q10. Какой дополнительный слой в трёхмерной схеме Дж. Захмана показывает где в процессе разработки будет создан соответствующий артефакт?

Слой ответственности

Слой разработки

Слой метаданных

Q11. Какие виды в архитектурной концептуальной схемы DoDAF описывают текущий стандартный профиль и предсказания по его изменениям?

Системные виды

Операционные виды

Технические виды

Q12. Какие из предложенных видов относиться к архитектуре "4+1"?

Вид с позиции пользователя

Логический вид

Физический вид

ГЛАВА ТРЕТЬЯ. РАЗРАБОТКА АРХИТЕКТУРЫ

3.1. Архитектура как продукт разработки

Архитектура АС является продуктом определённой деятельности, что позволяет ввести для неё и использовать понятие «жизненного цикла архитектуры АС». Работы по созданию архитектуры конкретной АС начинаются в определённый момент процесса разработки АС, когда в достаточном количестве подобраны необходимые «строительные материалы», а также другие необходимые ресурсы и средства.

Построения архитектуры АС целесообразно доверять квалифицированным архитекторам, которые будут выполнять порученную им работу в рамках подходящих технологий с использованием «хороших» инструментов.

Архитектура АС является продуктом, создание которого осуществляется подобно тому, как создают АС. В результате разработки архитектуры появляется дополнительная автоматизированная система (обозначим её $АС^{А}$), применения которой (в процессах разработки и использования соответствующей АС) осуществляются в формах человеко-компьютерной деятельности.

Понимание $АС^{А}$ как автоматизированной системы вполне правомерно, поскольку $АС^{А}$ состоит из системы информационно-образных структур, которые по запросам визуализируются на мониторах рабочих мест в корпоративной сети. А значит, $АС^{А}$ относится к классу информационно-справочных систем. $АС^{А}$ является системой контроля (контроля рассуждений с помощью понимания), причём с богатым информационным содержанием и мощной системой операций.

Поэтому в разработках архитектуры АС следует использовать опыт разработок АС, включая и всё сказанное выше об архитектурах. То есть правомерно

различать, а значит строить и применять «архитектуру» АС^{А} или, что то же самое, «архитектуру» архитектуры АС.

Разумеется, у систем типа АС^{А} есть определённая специфика, которая находит своё выражение в специфике предметной области АС^{А}, а значит и в исходных принципах для деятельности архитекторов, определённых подходах к их работе, полезных образцах и инструментах.

Представленная точка зрения на архитектуру АС является авторской. Она является богатейшим источником аналогий и подсказок (особенно для понимания), что не раз будет использоваться в последующем тексте.

3.2. Архитектурные парадигмы

В публикациях по архитектуре ПО поднимаются вопросы о парадигмах ПО как «модели постановки проблемы и её решения», определяющей основные подходы к проблемам архитектуры.

Особую известность в этом направлении получила публикация [71], в которой раскрываются следующие технические парадигмы:

1. Объектно-ориентированная парадигма, в рамках которой используется Объектно-Ориентированный Подход (ООП) к структуризации АС на всех этапах её жизненного цикла.

2. Компонентно-ориентированная парадигма, исходящая из принципов сборочного проектирования, конструирования и производства АС, в основе которых лежат программные «компоненты» как единицы сборки.

3. Сервисно-ориентированная парадигма, применение которой базируется на идее массового сервисного обслуживания пользователей АС по их запросам.

Представим каждую из названных парадигм и отношения между ними детальнее. Начнём с отношений, поскольку они взаимодополнительны. Реальность такова, что в общем случае при разработке компонентов используют объектно-ориентированную структуризацию и реализацию, а при разработке сервисов используют компонентные сборки. На практике, особенно при разработке архитектуры АС, целесообразно переключаться между объектной, компонентной и сервисной картинами АС.

1. **Объектно-ориентированная парадигма**. В основе этой парадигмы лежат идеи, методы и средства Объектно-Ориентированного Анализа и Проектирования (ООАП). Наиболее последовательно эта парадигма раскрыта в методологии унифицированного процесса разработки [38] и реализована в работе [39] в комплексе инструментально-технологических средств Rational Unified Process (RUP). Если АС разработана в такой среде, то говорят, что она имеет Объектно-Ориентированную Архитектуру (ООА, Object-Oriented Architecture).

К числу основных характеристик ООА относятся:

- реализация основных принципов ООП, в первую очередь инкапсуляции, наследования и полиморфизма;

- классы объектов являются основными единицами моделирования, проектирования и реализации;

- объекты и их взаимодействие находятся в центре интересов ООА;

- интерфейсы реализуются как специальные классы объектов, обслуживающих взаимодействие объектов;

- удалённые объекты используются точно так же, как и локальные объекты.

Наиболее хорошим примером ООА является Common Object Request Broker Architecture (COBRA), разработанная группой OMG. COBRA определяет объектную модель, в соответствии с которой распределённые объекты должны создаваться, использоваться и управляться. COBRA также определяет Reference Architecture (как систему образцов и инструкций), раскрывающую «как достигается взаимодействие между распределёнными объектами». Для этих целей предлагается язык определения интерфейсов (Interface Definition Language, IDL). Объект клиента получает связь с объектом сервера, после чего он в состоянии активизировать его методы тем же самым образом, как и методы локальных объектов.

ООА может быть представлена различными совокупностями видов. Одной из наиболее распространённых таких совокупностей является система видов Ф. Кратчена «4+1 views» [41]. Для описания видов обычно используются средства языка UML.

2. **Компонентно-ориентированная парадигма.** Сложность современных АС и типичная для них форма разработки в распределённой среде коллективом

разработчиков привела к идее сборки АС из независимо разработанных, хорошо оттестированных и повторно-используемых (аппаратных и программных) компонентов (по образцу сборочного производства в других отраслях промышленности). Эта идея применима для любых этапов жизненного цикла АС, в том числе и для этапа разработки архитектуры АС.

Промышленное производство компонентов, а также платформы, подобные J2EE, CORBA Component Model и Microsoft .Net, перевели компонентно-ориентированную парадигму (и основанную на этой парадигме версию разработки Component-based software development (CBSD)) в практическое русло. Говорят, что системы, разработанные на основе CBSD, имеют Компонентно-Ориентированную Архитектуру (КОА,Component-Based Architecture, CBA).

К числу основных характеристик КОА (CBA) относятся:

- опора на компонентные модели (подобные CORBA Component Model);

- компоненты являются основными единицами моделирования, проектирования и реализации;

- интерфейсы и взаимодействие находятся в центре интересов разработки архитектуры АС;

- интерфейсы могут обслуживать различные компоненты (интерфейсы поддерживают их расширение и специализацию, в том числе и в формах наследования);

- компонентные модели требуют от компонентов поддержки «действий по самоанализу» так, чтобы их функциональности или свойства могли быть открыты и использованы «во время сборки» или в реальном времени;

- особое внимание уделяется образцам проектирования и принципам разделения интересов для определения роли компонента и его ответственности.

Компонентная модель также включает программную модель, раскрывающую как распределённые компоненты должны быть разработаны, собраны и размещены на физическом оборудовании. Кроме CORBA Component Model (CCM) на практике широко используются JavaBeans и Enterprise JavaBeans в рамках Java 2 Enterprise Edition (J2EE).

Для представления СВА могут быть использованы различные совокупности видов, например совокупность, использованная в стандарте для RM-ODP. Для описания видов часто используют средства UML.

3. **Сервисно-ориентированная парадигма**. Большой класс АС, особенно реализованных в виде WEB-приложений, разрабатывают как системы сервисов, к которым открыт оперативный доступ клиентов. Про систему такого рода говорят, что для её разработки использовалась Сервисно-Ориентированная Архитектура (СОА, Service-Based Architecture, SBA).

К числу основных характеристик СОА (SBA) относятся:

- свободное соединение, заключающееся в том что любой сервис может быть получен как ответ на запрос из любого (разрешённого) источника;

- для сервиса не требуется ничего запоминать от одного вызова до другого (состояния загружаются как часть данных сервиса из базы данных или поступают от запросчика сервиса);

- сервисы являются основными единицами моделирования, проектирования и реализации;

- определение, описание, открытие сервиса, протоколы доступа и аспекты качества являются центральными интересами в архитектурном проектировании АС;

- сервис «выставляет напоказ» свои возможности, включая функциональность, данные и характеристики качества сервисов через описание на специальных языках, таких как язык описания WEB-сервисов (Web Service Description Language, WSDL);

- сервис может быть независимо и динамически обнаружен (открыт) и использован;

- так как сервис не имеет состояний, то обмен данными между распределенными пользователями и провайдерами могут привести к существенным накладным расходам.

На рис. 3.1 приведены основные ключевые элементы, используемые в SBA. Сервис может быть реализован с использованием компонентно-ориентированных технологий или объектно-ориентированных технологий. Описание сервиса обычно регистрируется в базе сервисов в определённом

формате, например, в таком как Universal Description, Discovery and Integration (UDDI).

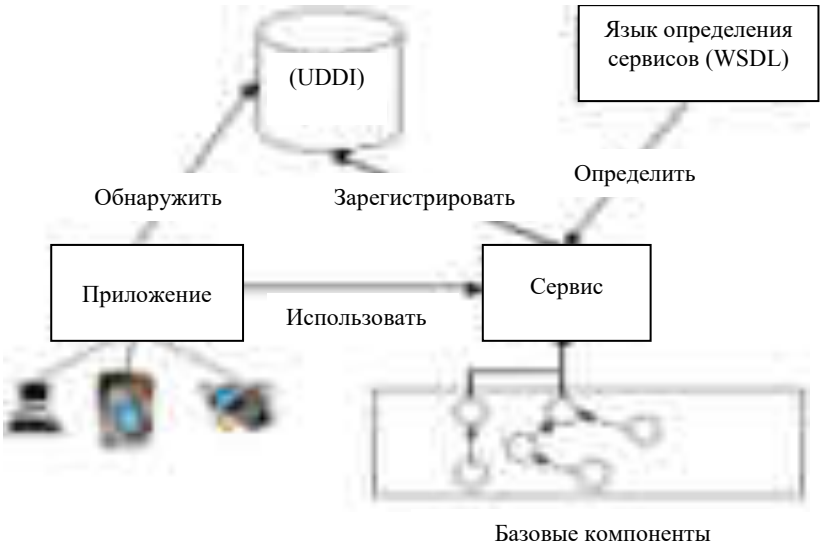


Рис. 3.1. Включение сервисов в приложение

С одной стороны SBA может быть представлена с помощью средств RM-ODP и UML, однако необходимо помнить, что эти средства не совсем адекватны задаче моделирования сервисов. Для описания сервисов более подходят специализированные языковые средства, такие как WSDL.

К месту отметить, что OOA, CBA и SBA предоставляют различные возможности и выгоды и могут быть использованы взаимодополнительно. С точки зрения функциональности сервис может быть реализован с помощью совокупности компонентов с учётом необходимых характеристик качества. В то же время функциональность компонента может быть структурирована в объектно-ориентированном виде и реализована с помощью объектно-ориентированного алгоритмического языка. Однако каждая из парадигм имеет специфику в представлении архитектуры АС. Эта специфика и другие характеристики парадигм в обобщённой форме представлены в таблице 3.1.

Таблица 3.1

Парадигма	OOA	CBA	SOA
Ключевые понятия	Класс, Интерфейс, Объект, ссылка	Компонент, Интерфейс, Контейнер, Сборка	Сервис, Определение сервиса, Регистрация/Обнаружение, Композиция сервисов,
Фокус	Идентификация и определение классов (объектов) и их взаимодействия	Определение компонентов и их интерфейсов, Паттерны (образцы) проектирования	Определение и описание сервисов и их интерфейсов и протоколов доступа
Ключевые особенности	Инкапсуляция операций и состояний, Ссылки между объектами и методы вызова	Промышленное производство компонентов и системных средств их связывания (middleware), Разделение компонентов, Разработка приложений в форме сборок компонентов	Слабое связывание, Самоопределение
Ключевые выгоды	Эффективность, Компактность, Зрелые технологии и языки	Низкая зависимость, Повторное использование, Общность, Легкая интеграция,	Масштабируемость, Расширяемость, Гибкость, Жизнеспособность, Лёгкость связывания, Комплексируемость
Ключевые аспекты	Зависимость, Сильная связность, Низкая гранулированность, Проблема изменений	Различия в middleware платформах, Необходимость поддержки различных стилей взаимодействия, Обслуживание	Качество услуг, Безопасность, Эффективность взаимодействия, Совместимость

Как результат сопоставления парадигм, в [71] предлагается следующий набор практических принципов для применения парадигм в разработке архитектуры АС:

1.Первый принцип нацеливает разработчиков на то, чтобы **Понять приложение** и **Очертить его границы**.

2.Второй принцип нацеливает на выявление **Желаемых атрибутов качества**, выделенный перечень которых открывает доступ к использованию подходящих парадигм по информации из таблицы 1.

3. Третий принцип указывает на необходимость первоочередного построения **Use Case** сценариев для интерфейсов, которые являются ключевыми поня-

тиями в каждой из представленных парадигм. Сценарии помогут прояснить роли, ответственность, характеристики взаимодействия и ожидаемые свойства базовых единиц разработки, а также помогут идентифицировать, какой из трёх типов единиц (объекты, компоненты, сервисы) является наиболее подходящим для разрабатываемой АС.

Применяя принципы в рамках определённой стратегии, необходимо помнить, что парадигмы открыты для их использования в дополнительной манере, разумеется, при необходимости.

Отметим, что совокупность парадигм, используемых в конкретной разработке, не исчерпывается вариантами ООА, CBA и SBA и открыта для модификаций и дополнений.

Отметим, что явное обращение к парадигмам архитектуры ПО в процессе разработки АС вносит в успешность разработки АС свой позитивный вклад.

3.3. Варианты архитектур

3.3.1. Основы архитектурных подходов

В архитектурной практике сложился ряд направлений, в каждом из которых используется определённый акцент на том, что, по мнению последователей направления (в определённых условиях), является центральным или базовым в разработках и использовании архитектуры. К числу таких направлений относятся:

1. объектно-ориентированная архитектура (**object-oriented architecture**) :
2. архитектура, базирующаяся на компонентах (**component-base-architecture**);
3. сервисно-ориентированная архитектура (**service-oriented architecture**);
4. архитектура, базирующаяся на событиях (**event-based architecture**);
5. архитектура, управляемая моделями (**model-driven architecture**);
6. архитектура, центрированная на данных (**data-centered architecture**);
7. архитектура, центрированная на людей (**people-centered architecture**).

Из перечисленных направлений три первых раскрыты выше в связи с вопросами об архитектурных парадигмах. Две последних понятны по названным акцентам, причём акцент на данных указывает на использование архитектурных стилей типа «репозитарий» и «доска сотрудничества». Поэтому представим только два направления с акцентами на события и модели.

3.3.2. Архитектура, ориентированная на события

В основе архитектур, основанных на событиях, лежит событийный архитектурный стиль, то есть приложения, составляющие систему, или их части активизируются при наступлении соответствующих событий. Что это за события и как они приходят в систему (извне или наступают внутри системы) – это вопросы специфики системы, которая, разумеется, должна найти отражение в архитектуре АС. Но главное в том, что процессы в АС управляются (полностью или частично) потоками событий.

К такому типу автоматизированных систем, например, относятся многоагентные системы различного назначения и системы массового обслуживания в распределённых сетях.

3.3.3. Архитектура, управляемая моделями

Архитектурное описание АС не относится к категории исполняемых артефактов. Такие описания не достигли, по крайней мере пока, формы описания, которую можно было бы автоматически оттранслировать в исполняемый код программной части АС. Но разработки исследователей и практиков в этом направлении проводятся интенсивно и уже привели к ряду положительных результатов. Одним из таких результатов является архитектура, управляемая моделями (Model-Driven Architecture, MDA).

Проект MDA был открыт в рамках Object Management Group (OMG), сопровождающей стандарты на версии языка UML [65]. Основная идея заключается в использовании по ходу разработки АС двух классов моделей – класса платформо-независимых моделей (Platform-Independent Models, PIM) и платформо-специфических моделей (Platform-Specific Models, PSM). Классы моде-

лей таковы, что для них разработаны средства преобразования элементов класса PIM в соответствующие им элементы класса SIM, а для моделей класса их преобразования – в программные коды (рис.3.2а). Модели родственны и, например (рис.3.2б), CORBA как специфическая модель PSM, обслуживающая сетевые приложения, для перехода на уровень операционной системы является платформо-независимой PIM и может, например, быть оттранслирована для ОС UNIX. В общем случае, при необходимости, возможны следующие варианты преобразований PIM→PIM, PSM→PSM, PIM→PSM, PSM→КОД, КОД→PSM, PSM→PIM.

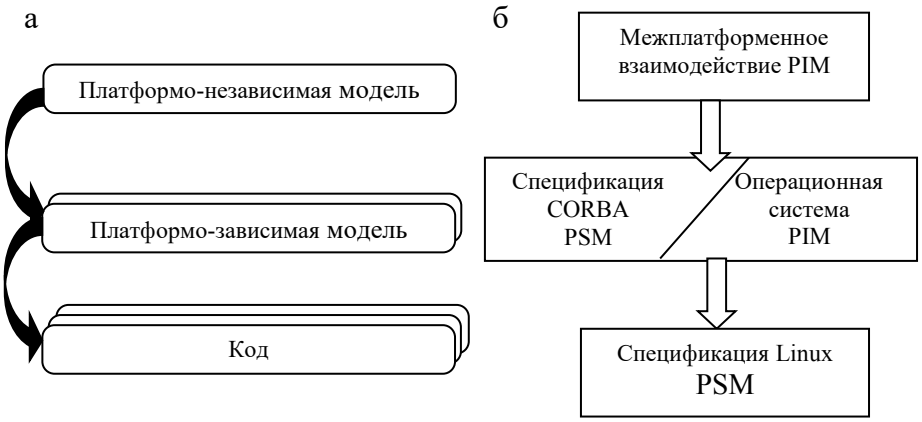


Рис.3.2. Преобразования моделей: а – последовательность преобразования моделей, б –межплатформенные преобразования

Кроме того, для классов моделей PIM и SIM существуют средства их формирования и редактирования в рамках языков со строгой семантикой, то есть языки моделирования являются специализированными формальными языками. Рекомендуемые базовые языки моделирования с указанием их места в MDA названы на рис.3.3. На этом же рисунке раскрыто ядро MDA и названы потенциальные применения подхода, а также отражены типовые функциональности, которые обычно приходится реализовывать в приложениях, используя MDA.

Рис.3.3. Приложения MDA

Завершая представление MDA, отметим, что формирование логики приложения на самом высоком уровне абстракции и автоматическая генерация всех элементов, связанных с платформой реализации приложения, способствуют сокращению сроков проектирования, тестирования и развертывания, полностью исключая затраты времени и сил на кодирование.

3.3.4. Архитектура, ориентированная на шаблоны

Одним из принципиальных подходов к архитектурным представлениям АС является ориентация на шаблоны (паттерны, patterns), что в архитектурах АС привело к направлению «Patten-Oriented Software Architecture», то есть к архитектурам, ориентированным на шаблоны. Следует отметить, что ориентация на образцы (шаблоны) используется в любых версиях построения архитектур. Когда же на «patterns» производится акцент, то имеют в виду направление, основы

которого заложены в работах К. Александра, адаптированные к специфике ПО «бандой четырёх» [30].

Сущность этого направления заключается в следующем понимании шаблона:

«Шаблон проектирования – это способ представления в общем виде как условия проектной задачи, так и правильных подходов к её решению»

Каждый полезный шаблон находит своё место в проекте в результате анализа контекста проектной задачи, что используется для обобщённого определения шаблона в виде

Имя_Шаблона (Контекст, Задача (Проблема), Решение, Результаты).

Ссылка на **Имя_Шаблона** указывает на задачу, её решение и последствия. С помощью словаря имён шаблонов можно вести обсуждение в коллективе, упоминать их в документации.

Для управляемого применения шаблона необходимо сформулировать **Задачу** и её **Контекст.** Также может быть включен перечень условий, при выполнении которых стоит применять шаблон.

С **Решением** связывается абстрактное описание задачи проектирования и того, как она может быть решена с помощью некоторого весьма обобщенного сочетания элементов (классов, объектов и других объектно-ориентированных средств).

Результаты – это следствия применения шаблона. Поскольку в объектно-ориентированном проектировании повторное использование является важным фактором, то к результатам следует отнести и влияние на гибкость, расширяемость и переносимость системы. Перечисление всех последствий помогает понять и оценить их роль.

Более детальное определение шаблона, получившее название GoF-формата приведено в таблице 3.2.

Таблица 3.2.

Имя шаблона	Имя в классификации [30]
Задача	Обобщённая постановка задачи
Также известен как …	Другие хорошо известные имена шаблона …
Мотивация	Сценарий, иллюстрирующий проблему
Применимость	Ситуации, в которых шаблон применим
Структура	Графическая презентация шаблона
Составляющие	Классы, объекты и отношения
Сотрудничество	Распределение ответственности
Последствия	К чему приведёт применение шаблона
Реализация	Как реализовать
Образец кода	Фрагменты кода
Известные реализации	Использовано в «системе»
Родственные шаблоны	Имена родственных шаблонов

Как уже отмечалось, шаблоны используются в любых версиях архитектур АС и их описаний. В то же время интерес к шаблонам и первые результаты были получены для объектно-ориентированной структуризации проектов АС, обеспечивающей структурные представления АС в терминах объектов и их классов.

Для объектно-ориентированной структуризации разработаны специальные библиотеки шаблонов, в основе которых лежат шаблоны, предложенные в [30] и представленные (их именами) в классификационной таблице 3.3.

Библиотеки шаблонов целесообразно строить так, чтобы они включали не только детальное представление каждого из шаблонов таблицы 3.3, но и их модификации, а также версии кодовых представлений на рабочих (для группы или для организации) языках программирования.

Библиотека шаблонов должна быть удобна для оперативного доступа и открыта для включения в её состав новых образцов, их модификаций и кодовых описаний.

Таблица 3.3.

Границы	Цель		
	Генерации	**Структур-ный**	**Поведения**
Класс	Factory Method	Adapter	Interpreter
Объект	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Momento Observer State Strategy Visitor

В иллюстративных целях представим один из шаблонов. Одним из наиболее распространённых и понятных шаблонов таблицы является шаблон Façade, сущность которого представлена на рис.3.3. Шаблон предназначен для «Предоставления единого интерфейса для набора различных интерфейсов в системе». Другими словами, шаблон Façade определяет интерфейс более высокого уровня, что упрощает работу с системой.

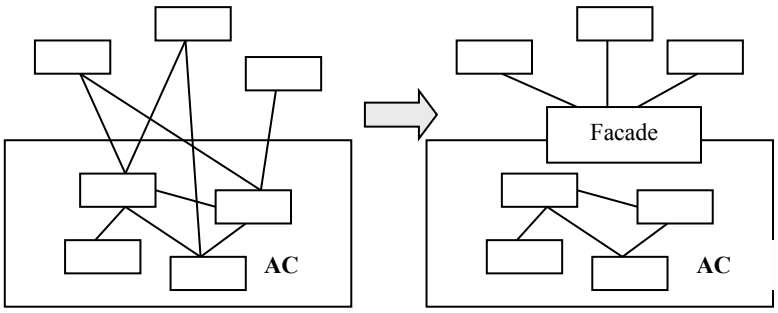


Рис. 3.3. Шаблон Façade: ▭ – элементы структуры

Ориентация на шаблоны приводит к следующей стратегии проектирования:

1. Отыскать шаблоны, присутствующие в проблемной области, и проанализировать полученный набор.

2. Для выделенного набора выполнить следующие действия:

- выбрать шаблон, который в наибольшей степени формирует контекст для всех остальных шаблонов;

- применить этот шаблон к самой высокоуровневой реализации проекта;

- выявить любые дополнительные шаблоны, которые могут появиться в полученной схеме, и добавить их к набору, который был получен в результате предыдущего анализа;

- повторно применить указанный процесс к оставшемуся набору шаблонов, выделенных в результате анализа.

3. Внести в проект все необходимые дополнительные детали.

Такая стратегия в большей мере относиться к «детальному проектированию» АС, чем к построениям её архитектуры. В то же время одна из важнейших позиций построения архитектуры АС связана с разработкой (пусть даже обобщённых) диаграмм классов. Богатым источником образцов для таких построений способны служить библиотеки объектно-ориентированных шаблонов.

Богатым русскоязычным источником учебного и практического материала по шаблонам проектирования является обзор О. Дубиной [84].

3.3.5. Архитектура, ориентированная на предметную область

Существует ряд предметных областей, в которых структуризация статики и динамики АС, используемых в этих областях, имеет устоявшиеся и проверенные на практике формы, приводящие к специфическим архитектурным видам на АС в виде блок-схем (boxes and lines). Например, предметная область «систем управления» с классической схемой управления, представленной на Рис.3.4.

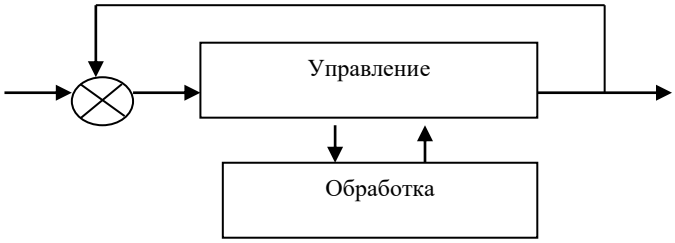


Рис.3.4. Структура системы управления

Ещё одним примером предметной области, устоявшиеся виды на АС в которой игнорировать неразумно, является область «систем массового обслуживания», в блок схемы которых обязательно включают очереди, а значит и обслуживание очередей. Одна из таких схем приведена на рис.3.5.

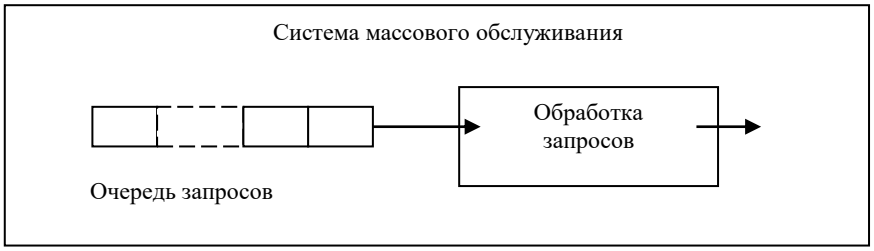


Рис.3.5. Система массового обслуживания

Разумеется, в архитектуру АС, разработанных для специфических предметных областей, следует включать не только традиционные блок-схемы, но и виды, о которых говорилось выше, например, виды RM ODP. Но из-за многократно повторяющихся разработок АС в специфических предметных областях накапливается опыт, в результате которого образуется типовая архитектура, которую целесообразно использовать как образец для подражания при разработках очередных АС.

Такие типовые архитектуры на английском языке называют «Reference Architecture» или «Domain Specific Software Architecture, DSSA». Первое название обычно используют для устоявшихся типовых архитектур в предметной обла-

сти «Архитектура АС». Например, к «Reference Architecture» относят систему архитектурных образцов в RUP. Второе название обычно применяют для специфических предметных областей, в том смысле, о котором говорилось выше.

Завершая пункт, отметим, что к классу специфических предметных областей относится область разработок и применения архитектур АС. Сущность специфики – концептуальное согласование пониманий лиц, вовлечённых в разработку АС, и оперативный контроль рассуждений и понимания в рамках жизненного цикла АС.

3.4. Архитектурные стили

3.4.1. Определение стиля

Как уже отмечено выше, в построениях архитектур АС важны исходные установки:

- в виде парадигмы или согласованной совокупности парадигм, определяющих «модели постановки проблемы и её решения»;

- подхода или согласованной совокупности подходов, выполняющих функции организующего начала для действий, в результате которых создаётся и используется архитектура АС.

Однако, кроме исходных установок, которые носят методологический характер и не привязаны к содержанию АС, лицам, участвующим в разработке и использовании архитектуры, необходимы образцы форм, которые можно было бы заполнять содержанием, раскрывающим конкретную разрабатываемую АС.

В первую очередь нужны образцы форм, которые бы помогли построить эскиз архитектуры АС, или, что то же самое, построить «архитектуру» архитектуры АС. В архитектурной практике функции «образцов форм для построения «архитектуры» архитектуры АС» принято возлагать на **архитектурные стили**.

Понятие архитектурного стиля вошло в архитектуру ПО одним из первых. Его место в разработке архитектуры было уже указано в [52].И все же детальное представление архитектурного стиля, классификация стилей с сопоставлениями

и рекомендациями по применению было проведено только в публикации M.Shaw и P.Clements [63], содержание которой детально раскрывается ниже.

Архитектурный стиль в [63] определялся как «совокупность правил проектирования, идентифицирующих виды компонентов и коннекторов, которые могут быть использованы для образования композиции систем или подсистем, вместе с локальными и глобальными ограничениями, в рамках которых производится построение архитектуры». Причём компоненты, включая их версию в виде вложенных (в систему) подсистем, могут отличаться по их вычислительной природе.

Тип компонентов может быть различным, в зависимости от их объединения в пакеты, например по виду их взаимодействия с другими компонентами. Функции компонентов могут выполнять программные модули, комплексы модулей, «клиенты», «серверы», базы данных, библиотеки, «уровни» [64].

Для того чтобы в каждом конкретном случае прояснить уровень абстракции, используемый в представлении компонентов, их взаимодействие выделяется и оформляется в виде соответствующих коннекторов (например, в виде вызовов подпрограмм, средств доступа к разделяемым данным, протоколов взаимодействия, потоков данных и т. д.). Определение коннекторов включает не только указание на средства взаимодействия, но и правила, управляющие таким взаимодействием.

Авторы [64] в публикации преследовали следующие цели:

- установить единый стандарт спецификации стилей, для того чтобы была возможность более точного и единообразного представления стилей архитекторами;

- обеспечить работу архитекторов систематическими средствами, поддерживающими работы по извлечению информации о стилях;

- находить различия между стилями, которые позволят адекватно выбирать более подходящие стили для определённых задач;

- установить последовательность этапов для выбора наиболее подходящего стиля для заданной проблемы.

3.4.2. «Архитектура» архитектуры

В публикациях по архитектурным стилям, в том числе в публикациях, раскрывающих использование архитектурных стилей, автору не встречалась интерпретация стилей как базовых форм для построения «архитектуры» архитектуры АС. В то же время такая интерпретация, во-первых, является богатым источником аналогий (из теории и практики архитектур АС), а, во-вторых, позволяет более адекватно определиться с назначением архитектурных стилей, а значит, и с их местом и ролью в процессах разработки АС.

Как уже отмечалось выше (п. 3.1), вполне правомерно подходить к созданию «архитектуры» $АС^{А}$ точно также, как к созданию архитектуры АС, учитывая специфику $АС^{А}$, в первую очередь специфику использования $АС^{А}$ как средства, обслуживающего контроль (в формах понимания) рассуждений в рамках жизненного цикла АС.

Следовательно, в рассуждениях по созданию $АС^{А}$ целесообразно выделять тему, содержание которой связано с ответами на самые существенные вопросы об $АС^{А}$, то есть об архитектуре АС. Функции основного из этих вопросов целесообразно возложить на следующий вопрос:

> **$Q^{А}$. Какая декларативно-процедурная структуризация будущей АС позволяет архитектору (или группе архитекторов) наиболее рационально обобщить и связать в единое целое требования лиц, заинтересованных в разработке АС?**

Выбор вопроса $Q^{А}$ на роль основного вопроса об «архитектуре» архитектуры АС объясняют следующие основания:

1. Основные проблемы современных АС лежат в создании и использовании их программного обеспечения, построение которого может быть осуществлено в различных декларативно-процедурных версиях, в каждой из которых используется специфическое разделение (как в компьютерной памяти, так и в физическом пространстве) на компоненты данных (декларативные компоненты) и исполняемые коды (процедурные компоненты).

2. Возможности согласования, обобщения и реализации требований лиц, заинтересованных в разработке АС, в существенной мере зависят от версии де-

кларативно-процедурной структуризации программного обеспечения АС. В наиболее подходящей версии декларативно-процедурной структуризации удаётся наиболее рационально интегрировать требования к АС. Выбор такой версии зависит от профессионализма архитектора(ов).

Наличие целостного декларативно-процедурного образца позволяет архитектору(ам) использовать такой образец для контроля (в формах понимания) собственных рассуждений, нацеленных на построение «архитектуры» для архитектуры АС. Кроме этого, конкретный декларативно-процедурный образец выполняет следующие важные функции:

- он позволяет осознанно и контролируемо проверить и обобщить систему требований к АС, построенную на предыдущем этапе «формирования системы требований»;

- образец используется как руководство для построения на его базе определённой альтернативы «архитектуры» $АС^{А}$ с приписанными ей характеристиками для последующего выбора на множестве альтернатив;

- образец выполняет функции ограничений для действий в процессах архитектурного моделирования АС.

Завершая пункт, отметим ещё и следующее:

- то, что принято относить к архитектурным стилям, является декларативно-процедурной структуризацией АС с позиций программного обеспечения;

- если это необходимо для построения «архитектуры» архитектуры АС, то можно объединять в единое целое группу декларативно-процедурных образцов (группу архитектурных стилей), подобно тому, как интегрируют архитектурные виды.

3.4.3. Классификация стилей

На рис.1.13, раскрывающем место архитектуры АС в процессе разработки АС, отражено место архитектурных стилей. Разработка или выбор и адаптация подходящего стиля проводятся на первых этапах построения архитектуры АС. Положенный в основу разработки АС архитектурный стиль входит в состав архитектуры как её «эскиз», как «архитектура» архитектуры АС.

Архитектурный стиль как эскиз раскрывает АС с позиций рациональной декларативно-процедурной структуризации её ПО, учитывающей специфики варианта реализации. К спецификам относятся специфика физического размещения данных и программ в компьютерных сетях, специфика реализации и передачи управления между частями программ (особенно в сетях), а также включение в процессы АС действий, выполняемых человеком.

Классификация архитектурных стилей, которая наиболее часто приводится в публикациях по архитектуре [63, 64], представлена на рис.3.6.

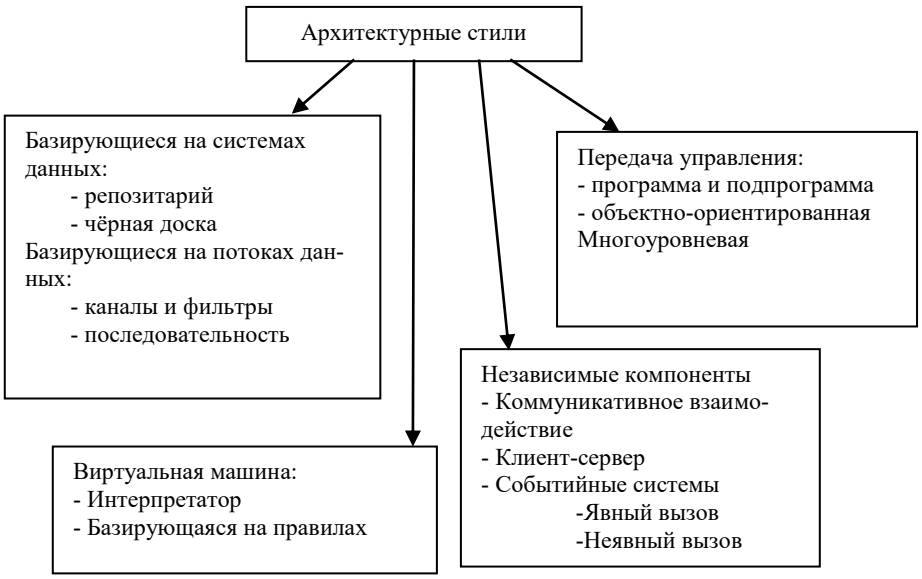


Рис.3.6. Классификация архитектурных стилей

Образцы стилей, раскрывающие содержательную сторону классификации и некоторые детали, приведены в следующем пункте. Заметим, что в определение любого стиля принято включать следующие характеристики :

- словарь элементов проектирования;

- типы компонентов и коннекторов;

- совокупность правил конфигурации, включающих правила:

- для топологических ограничений, определяющих композиции элементов (например, компонент может быть связан более чем с двумя другими компонентами);

- для семантической интерпретации (композиции элементов проектирования должны иметь хорошо понятное значение);

- возможности анализа систем, построенных на основе стиля.

3.4.3. Образцы стилей

Для того чтобы было проще ориентироваться с классификацией стилей, представим формы и обобщённое содержание для наиболее типичных архитектурных стилей.

1.Каналы и фильтры (Pipes and Filters).

Архитектурное описание строится как (или представляет собой) связная совокупность фильтров и каналов (рис.3.7), в которой:

фильтр – независимый блок, получающий на вход поток(и) данных и выдающий поток(и) данных;

возможна достаточно простая переконфигурация системы фильтров (соединение их с помощью pipes);

фильтры работают параллельно.

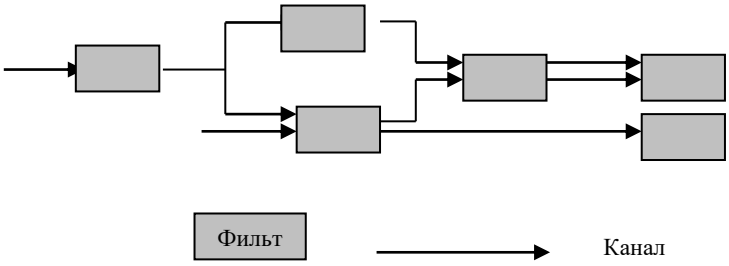


Рис.3.7. Стиль «каналы и фильтры»

2. Стиль, основанный на репозитарии (Repository-based)

В рамках стиля, базирующегося на репозитарии, общие данные разделяет определённая совокупность приложений, каждое из которых в своих обращениях к данным является независимым. Общие данные разделяются и в стиле, получившем название «blackboard».

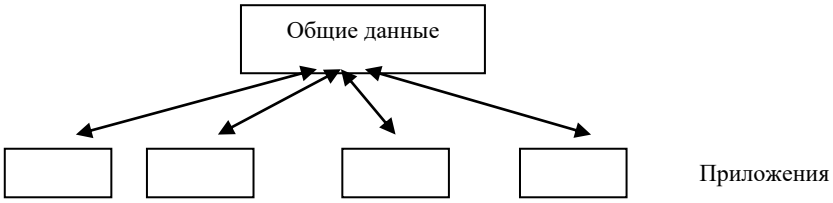


Рис. 3.8. Стиль «репозитарий»

3. Событийно-ориентированный (Event-based) стиль

Событийный стиль (рис.3.9) используется для композиции приложений, которые обращаются (регулярно или случайно) к родственному типу событий (возникающих регулярно или случайно), с которыми взаимодействует система.

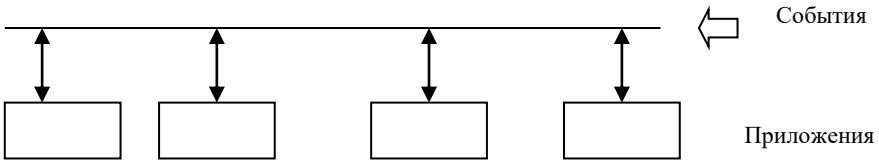


Рис. 3.9. Событийный стиль

4. Одноранговый (Pier-to-Pier) стиль

Одноранговый стиль (рис.3.9) применим в условиях, когда разрабатываемая система состоит из приложений (участников взаимодействия), каждое из которых связано с определённым количеством других. В каждом акте взаимодействия оно устанавливается и происходит только между двумя участниками

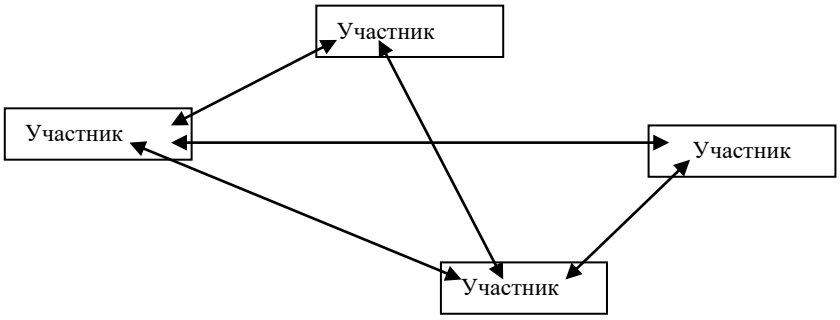


Рис. 3.10. Одноранговый стиль

5. Многоуровневый (Layered) стиль

5.1. Многослойный стиль

Как отмечалось выше, одним из эффективных подходов к разбиению системы на части является использование многослойных структур. Такой подход применим как к приложениям, развёрнутым на одном компьютере, так и для распределённых систем. Типовые разбиения на слои (для таких вариантов) близки по смыслу (рис. 3.11), но имеют различия в терминологиях, используемых для их описаний.

Рис.3.11. Многослойный стиль

5.2. Клиент-серверный стиль

Для распределённых систем типичен случай применения стиля «клиент-сервер», для реализации которого часть системы с определёнными функциональностями размещают на сервере, а другую часть – на клиентских местах пользователей. Два типовых варианта распределения функциональностей с указанием их обощённого содержания представлены на рис.3.12. Один из этих вариантов получил уточнение «тонкий» клиент, а второй – «толстый» клиент. Клиент-серверный стиль получил широкое применение как в корпоративных сетях без использования WEB-возможностей, так и в сетях, использующих такие возможности, в том числе и в сети Интернет.

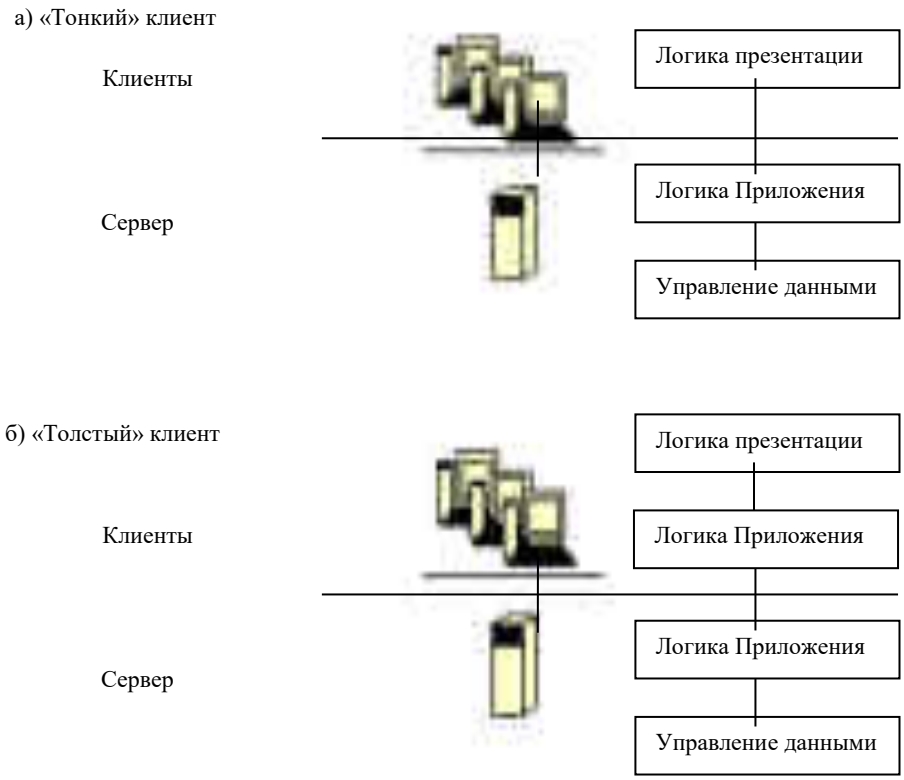


Рис. 3.12. Клиент-серверный стиль: 2 уровня

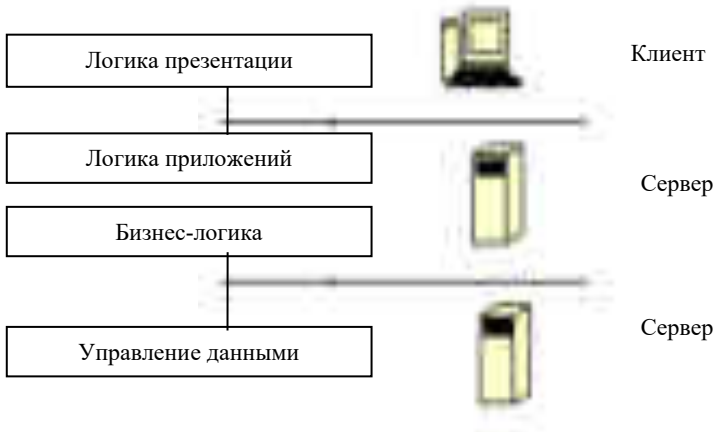


Рис. 3.13. Клиент-серверный стиль: 3 уровня

Представленные на рис.3.12 стили относятся к классу двухуровневых, но специфических двухуровневых, для которых используется не термин «Layer», а «Tier» . Это объясняется тем, что на практике получил широкое распространение архитектурный стиль, представленный на рис.3.13 и получивший название «3-tier». Этот стиль включает уровень бизнес-логики, который разделяет и координирует доступ ряда различных приложений к общим данным.

6. Брокерный (Broker) стиль

Брокерный стиль (рис.3.14) используется для организации доступа определённого количества клиентов к совокупности серверов. Разумеется, для организации такого взаимодействия требуется специальный программный сервис, получивший название «брокер».

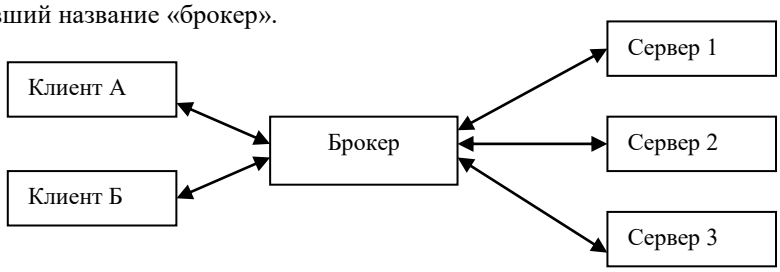


Рис. 3.14. Брокерный стиль

7. Стиль «Модель-представление-управление» (Model-View-Control, MVC)

Стиль MVC согласован с приложениями (рис.3.15), в которых основным видом работ является интерактивное взаимодействие пользователя с визуализированными моделями, представляющими определённые сущности.

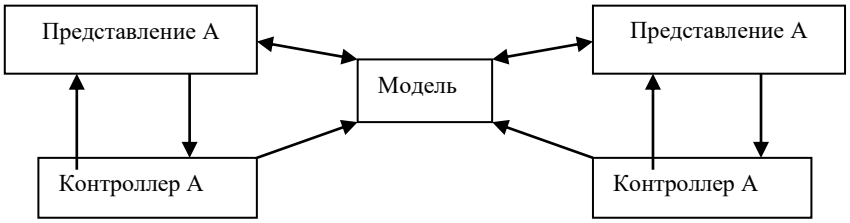


Рис.3.15. Стиль MVC

3.4.4. Сопоставление стилей

Каждый архитектурный стиль является образцом, выбор которого для подражания в конкретной архитектуре проводится на определённом множестве альтернатив. Для того чтобы выбор был возможен, архитектурные стили следует представить в виде, позволяющем выполнить такую работу. Любая попытка создать, например, автоматизированную методику выбора приведёт к «задаче многокритериального принятия решений в условиях неопределённостей».

Наиболее известной, среди используемых на практике, является формулировка отмеченной задачи и методика её решения в рамках **метода анализа иерархий**. Однако в архитектурной практике не распространено представление задачи выбора стиля как решения многокритериальной задачи в условиях неопределённостей. Этому мешает и тот факт, что в конкретной разработке АС могут быть использованы несколько стилей, объединённых во взаимодополнительный комплекс. Взаимодополнительность стилей является типичным архитектурным решением в разработках сложных АС.

Принято упрощать задачу выбора стилей до такого представления ситуации выбора, когда ситуация представлена некоторой совокупностью требований, а множество стилей – табличной структурой, в которой для каждого стиля отмечен тот набор требований, который он (в определённой степени) способен поддержать. Примером такой таблицы является результат сопоставления стилей, проведённый в [29] и пригодный для его использования в WEB-приложениях. Сопоставления архитектурных стилей проведены по характеристикам качества, что согласуется с подходами к разработке архитектуры с позиций качества.

Для практического использования таблицы сопоставлений следует из системы требований к АС выделить набор требований к качеству и использовать этот набор для выбора из таблицы подходящего стиля.

В учебном пособии таблица из работы [29] взята с сокращениями в виде Таблицы 3.3, которая содержит ссылки на стили, представленные выше. В работе [29] представлены детальные характеристики стилей и обоснования для каждой оценки, зафиксированной в Таблице 3.4.

Таблица 3.4

Архитектурный стиль	Исполнимость	Эффективность	Масштабируемость	Простота	Развиваемость	Расширяемость	Привычность	Конфигурируемость	Повторность использования	Визуализируемость	Мобильность	Надёжность
Каналы и фильтры	+ -			+	+	+		+	+			
Репозитарий	+ +		+									+
Клиент-серверный	-	-	+		+				+		+	
Многоуровневый		-	+		+				+		+	
Многоуровневый клиент-серверный	-		+ +	+	+ +				+		+	
Виртуальная машина		+	-	+ -		+	+			-	+	-
Одноранговый	-	-	+		+				+		+	
Мобильный	+	+ +		+ -		+ +	+	+		-	+	

3.4.5. Роли архитектурного стиля в разработке АС

Выше детально раскрыт механизм «точек зрения» и «видов» построении архитектурных описаний. В том случае, когда разработка АС осуществляется на базе композиции архитектурных стилей, каждый стиль выполняет функцию «вида» архитектуры АС. «Эскиз» архитектуры, «архитектура» архитектуры и «вид» на архитектуру – это основные варианты ролей «архитектурных стилей», которые они исполняют в процессе разработки АС.

Исполнение такой роли приводит к тому, что «архитектурный стиль» наследует, а вернее сказать «передает по наследству» архитектуре АС всё то, что отмечалось в п.1.5.2 для архитектуры. То есть, архитектурный стиль (текст приводится повторно специально):

- вносит вклад в извлечение требований и формирование их системы;

- ясно показывает совокупность ранних проектных решений;

- предписывает организационную структуру АС (хорошо структурированные системы полны образцов);

- является общим архитектурным базисом для линеек программных продуктов;

- даёт возможность учитывать, согласовывать и предсказывать характеристики качества, в том числе и по результатам её исследования;

- даёт информацию о распределении работ и их календарных планах;

- даёт возможность для более точной оценки стоимости и расписания работ;

- снижает риски;

- является первой формой существования (архитектуры) АС, которая может быть проверена (испытана) как целое;

- предоставляет возможность для переноса или повторного использования стилей и архитектурных каркасов;

- приносит выгоду в ограничении «словаря» альтернатив проекта или его частей;

- помогает в эволюционном прототипировании;

- оформляет концептуальную целостность АС и процесса её разработки;

- обслуживает понимание и взаимопонимание в индивидуальной и коллективной работе;

- есть средство выражения мыслей и рассуждений в коммуникации лиц, вовлечённых в разработку АС;

- предоставляет возможность рассуждать о потенциальных изменениях по ходу разработки АС и вносить вклад в управление такими изменениями;

- может быть базисом для изучения системы.

В отборе, анализе, композиции стилей и их применении важное значение имеют следующие вопросы, ответы на которые управляют действиями архитектора:

1. Какова система понятий, стоящая за определенным стилем?
2. Какие образцы стоят за стилем?
3. Какая вычислительная модель соответствует стилю?
4. Каковы существенные инварианты стиля?
5. Каковы примеры использования стиля?
6. Какие преимущества стиля?
7. Какие недостатки стиля?
8. Какова специализация стиля?

3.5. Архитектура и характеристики качества

3.5.1. Специфика требований к качеству АС

Выше не раз утверждалось, что на начальных этапах разработки АС принципиальная роль возлагается на архитектурные решения по обеспечению требований к АС, которые распределены по реализации АС и отражают определённые «интересы» групп лиц, заинтересованных в разработке АС и её использовании. К числу таких требований, в первую очередь, относятся характеристики качества АС, общепринятая система которых определена стандартом ИСО/ИЭК- 9126.

Необходимо отметить, что проявления характеристик качества при эксплуатации АС создают «слой» вокруг её базовых функциональностей [86], что образно отражено на рис.3.16. У такого слоя два проявления:

- во-первых, большинство характеристик качества воспринимаются пользователем АС со стороны его контактного взаимодействия с АС;

- во-вторых, характеристики качества формулируются, учитываются и оцениваются, начиная с ранних шагов разработки АС.

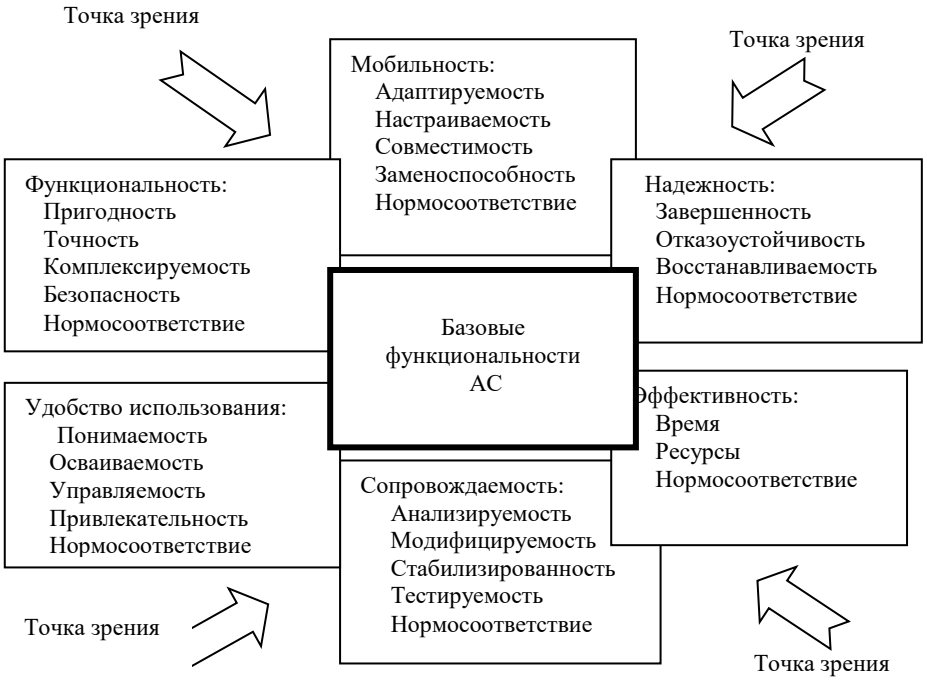


Рис.3.16. Система функциональностей АС

Реальные (достигнутые в процессе разработки) характеристики качества АС материализованы в её реализации, причем элементы материализации качества распределены среди базовых элементов АС или встроены в них. В объектную структуризацию АС «вшиты» (по аспектно-ориентированной терминологии) аспектные решения, обеспечивающие запланированные (с позиций качества) точки зрения на АС.

В процессе разработки и использования АС правомерны и практически полезны, например, «интерес» к АС с позиций «удобства использования» или «сопровождаемости», а также «интерес» с позиции «безопасности» АС. Практика «материализации интересов к качеству АС» показывает, что такие интересы не удаётся представить с помощью подходящего «вида» с точки зрения соответствующего качества. «Следы интереса» (метрики качества) приходится распределять по разным «видам», определённым в соответствии со стандартом IEEE-1471, и даже по разным моделям и документам «видов». «Интересы» кон-

кретного качеств пересекают другие интересы (относятся к типу crosscutting concerns), в том числе и интересы других качеств.

Всё это приводит к тому, что основной задачей построения архитектуры становится не её разработка с позиций функциональности, а разработка с позиций нефункциональных требований, то есть с позиций интегрального качества АС.

По своей сложности и объемам работ создание «слоя качества» часто сопоставимо с созданием системы средств, обеспечивающих реализацию базовых функциональностей АС. Включение «слоя качества» в состав АС существенно повышает её сложность. Однако это не может служить причиной для отказа от создания АС с запланированными и/или заданными требованиями по качеству.

Практика показывает, что обеспечение качества АС – не только запрос конкурентного рынка. Управляемая работа с требованиями качества в рамках АОП, включённая, например, в объектно-ориентированную технологию Rational Unified Process (RUP), способна дополнительно повысить успешность разработок АС [40].

3.5.2. Подход к построению архитектуры с позиций качества

Основной вклад в понимание и построение архитектуры с позиций качества внесли исследования и разработки института SEI Carnegie Mellon [6], в рамках которых разработку АС целесообразно проводить в соответствии со схемой, представленной на рис.3.17.

Схема указывает, что характеристики качества выполняют ключевую роль в построении архитектуры, которая, в свою очередь, управляет процессом разработки АС. Разумеется, функциональность должна быть отражена в архитектуре обязательно, но варианты её реализации должны быть сбалансированы с достижением требуемых характеристик качества, а проблемы такого баланса лежат в области качества.

Ещё одна проблема состоит в том, что баланс приходится осуществлять в условиях, когда АС отсутствует и о её характеристиках качества можно судить

только потенциально, рассуждая о взаимодействии с АС тех групп лиц, каждая из которых заинтересована в определённом качестве АС и его степени.

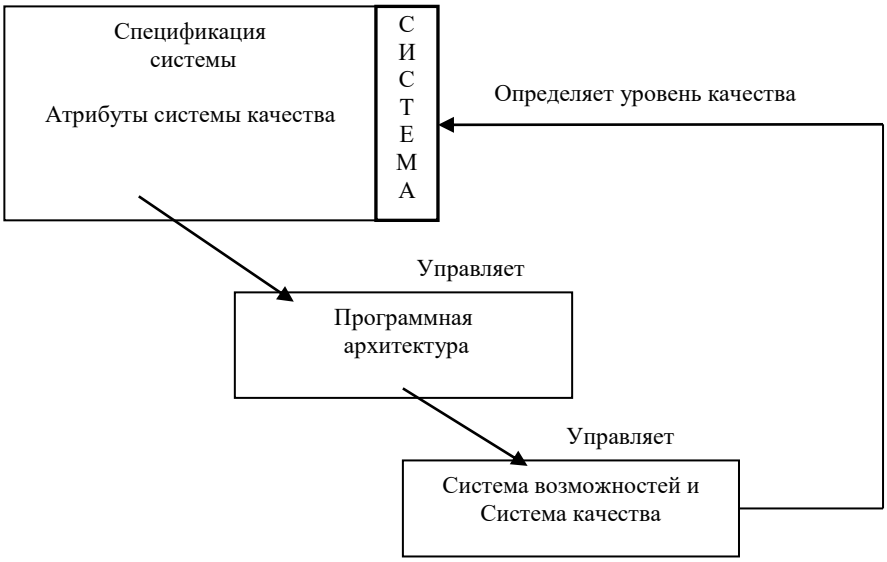


Рис.3.17. Управление качеством АС

Для таких рассуждений (а значит и для построения архитектуры) было предложено использовать сценарии, в каждом из которых отражена определённая «единица» реагирования АС на заданные условия. Была предложена, испытана и внедрена в практику следующая модель сценария:

1. Источник стимула: сущность (актор, человек, компьютер,…), порождающая стимулы.
2. Стимул: предполагаемое воздействие на АС.
3. Среда: условие или состояние, при котором воздействует стимул.
4. Артефакт: то, на что воздействует стимул.
5. Отклик: активность в ответ на стимул.
6. Мера отклика: требование, отражающее определённое проявление качества.

Такая модель сценария подобна схеме условного рефлекса и может быть интерпретирована как условный деятельностный рефлекс, то есть как вполне определённое реагирование актора на условия, которые сложились в опреде-

лённый момент времени и требуют от актора определённой реакции. В этом плане стимул выполняет функции обоснованной (внутренней для актора) причины, материализованной в реагировании, причём определённым образом.

В таблицах 3.5 и 3.6 с иллюстративными целями (для деталей понимания) приведены примеры сценариев и значений каждой из составных частей сценария.

Таблица 3.5

Часть сценария	**Возможное значение**
Источник	Внутренний или внешний для системы
Стимул	Ошибка, упущение, сбой
Артефакт	Процессоры системы, каналы коммуникации
Среда	Нормативное реагирование. Отклонения от запланированных режимов
Отклик	Система должна обнаруживать и демонстрировать определённые события, указание на необходимость «ремонта»
Мера реакции	Допустимое время реагирования, допустимое время на устранение «неполадок»

Таблица 3.6

Часть сценария	Возможное значение
Источник	Конечный пользователь, разработчик, системный администратор
Стимул	Желаемое добавление/модификация/ изменение функциональности, атрибута качества
Артефакт	Интерфейс, платформа, среда; система, взаимодействующая с разрабатываемой системой
Среда	Реальное время, время компиляции, время связывания, время разработки
Отклик	Локальные точки в архитектуре, которая должна быть изменена; модификации, которые не должны влиять на другие функции
Мера реакции	Цена в терминах усилий, финансовых средств и др.

Для разработки архитектуры необходимо извлечь из опыта, а также обнаружить, вообразить и зарегистрировать достаточное количество сценариев.

Этот массив данных следует обработать, сопоставляя значения соответствующих структурных элементов сценариев так, чтобы информация позволила решить основные задачи архитектуры, связанные с балансированием требований. Обработка нацелена на формирование групп сценариев (рис.3.18), позволяющих обнаружить конфликты и непротиворечивые дополнительности в ситеме требований к АС.

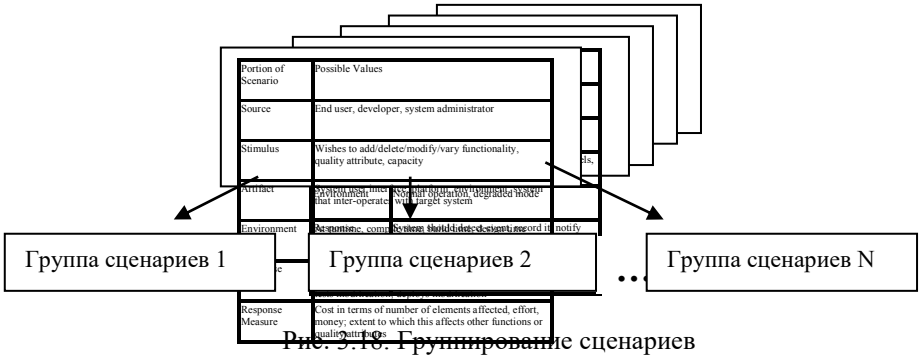


Рис. 3.18. Группирование сценариев

Формирование и обработка сценариев подготавливают последующие действия с этой информацией. Детали таких действий будут представлены в следующем пункте пособия, причём не только с позиций методов, разработанных специалистами SEI.

В настоящее время популярна позиция, предложенная [73], которая исходит из семантического расширения каркаса IEEE-1471, представленного на рис. 3.19.

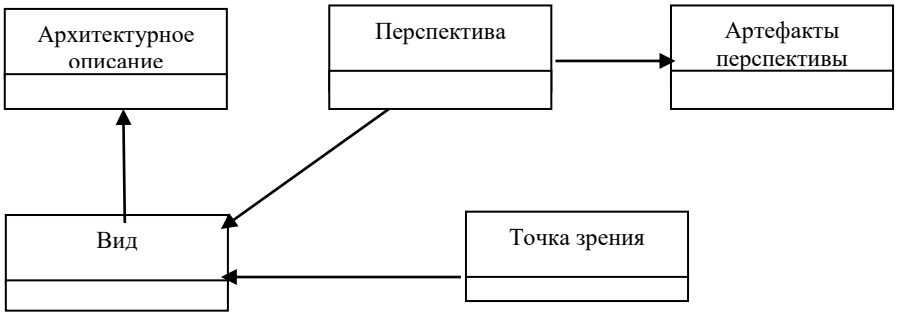


Рис.3.19.Семантическое расширение

В предложенном расширении с важными характеристиками качества связывается понятие «перспектива» со следующим содержанием: архитектурная пер-

спектива – это совокупность действий, схем проверки, тактик и руководств для управления процессом, гарантирующим то, что система будет обладать определённым множеством качественных свойств, согласованных с определёнными архитектурными видами.

«Перспективы» дополнительны к «видам» (рис.3.20), поскольку они позволяют ввести в архитектурное описание интересы, которые невозможно выразить с помощью подходящего вида. В предлагаемой авторами схеме различаются два класса перспектив – основной и дополнительный. Основной включает следующие характеристики качества: производительность, доступность, сопровождаемость, защищённость, размещённость . В дополнительный список перспектив включены

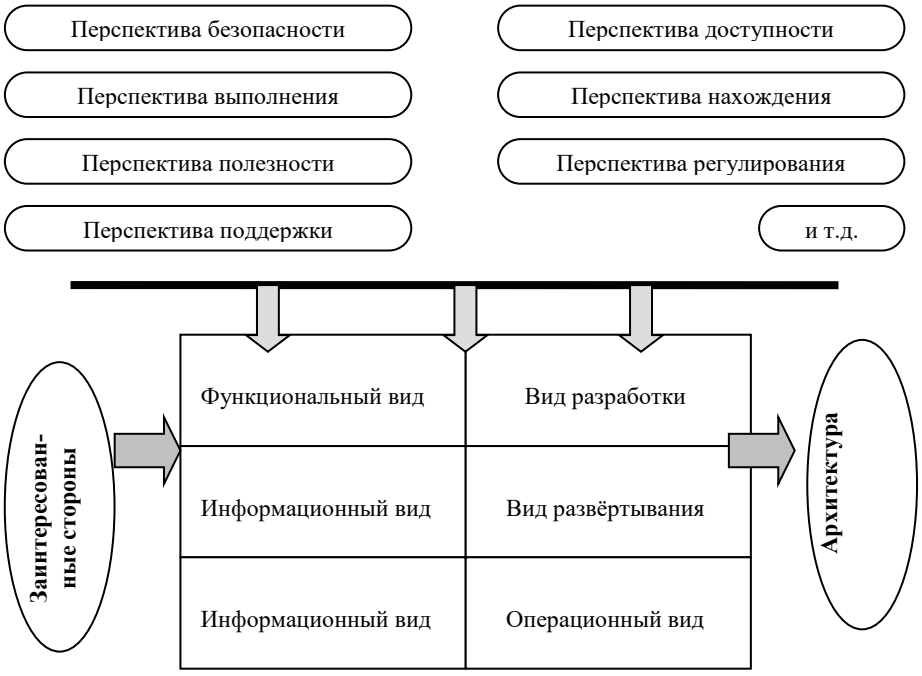


Рис. 3.20.Отношения между видами и перспективами

3.6. Архитектурное проектирование

3.6.1. Основы проектирования архитектур

Архитектура является одной из важнейших форм существования АС, которая до её разработки не существовала или существовала как реализация предшествующей версии или версий. А значит, на определённом этапе разработки АС с помощью определённого процесса и средств архитектура АС должна быть создана, возможно, в новой версии.

Как и любой продукт деятельности, архитектура создаётся для выполнения определённых позитивных функций, конкретнее, для обслуживания процесса разработки АС, а также её использования и сопровождения. Основные задачи, в работе с которыми от наличия АС достигаются определённые позитивы, были названы в п.1.5.2. Разумеется, позитивы достигаются не просто от наличия архитектуры АС, а от того насколько при построении АС были учтен факт и степень их достижения.

На практике широко используются архитектурно-центрированная разработка (Architecture-centric development) систем, интенсивно использующих ПО. Такой тип разработки включает:

- создание бизнес use-case для разрабатываемой системы;
- осознание (понимание) требований;
- создание или выбор архитектуры АС;
- документирование и обсуждение архитектуры;
- анализ и оценка архитектуры;
- материализация системы на базе архитектуры;
- оценка и подтверждение того, что материализация АС соответствует архитектуре.

Такая нагрузка на архитектуру предполагает, что задача разработки архитектуры АС должна быть адекватно (ожиданиям от создания архитектуры) сформулирована и решена. В результате такого решения будет построена концептуальная система «архитектура АС», которая будет выполнять функции «архитектурной модели» АС на последующих этапах её жизненного цикла.

Понимание архитектуры АС как архитектурной модели концептуального типа носит принципиальный характер, поскольку такое понимание предполагает, что «если архитектура АС в форме модели была создана, то архитектурная модель должна быть использована по запланированному назначению». Такая модель (как и любые другие модели) должна исследоваться, когда в этом возникает необходимость, для «получения ответов на появившиеся вопросы». Разумеется, архитектурная модель АС может ответить на вполне определённые типы вопросов, которые были потенциально учтены при создании модели.

Таким образом, для того чтобы архитектурную модель можно было использовать, её сначала нужно построить. Обобщённые детали построения архитектурной модели АС (или, что то же самое, архитектуры АС) представлены на рис.3.21.

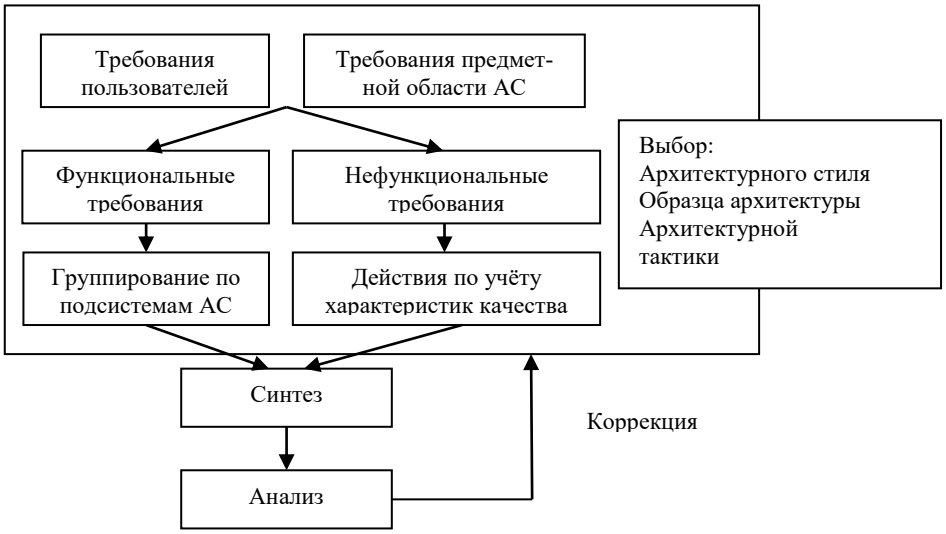


Рис.3.21. Построение архитектуры

Процесс создания «архитектуры АС» состоит из этапов, включающих этап её проектирования. Более точно, процесс создания архитектуры АС, встроенный в процесс разработки АС, носит спиралевидный итеративный характер, что приводит к возвратам к вопросам создания архитектуры АС от «витка спирали к витку».

Архитектурная модель начинает использоваться до её построения в целостном виде. Более того, в итеративном процессе разработки от очередной постро-

енной архитектурной модели переходят к построениям следующей версии модели. А значит, шаг за шагом архитектурная модель становится всё более и более богатой по информационному содержанию, что потенциально увеличивает позитивную отдачу от её использования. Основные позитивы от использования архитектуры (архитектурной модели) АС были представлены в пункте 1.5.2.

3.6.2. Языки архитектурных описаний

Разработка архитектуры АС завершается представлением её проекта, получившего название «Архитектурное Описание». Такой проект носит концептуальный характер и состоит из текстовых единиц (неформального и/или полуформального и/или формального типов), таблиц и графических объектов (в число которых часто включают UML-диаграммы, построенные с использованием формализованного языка UML). Следовательно, при формировании конкретного АО архитектор (или группа архитекторов) стоит перед вопросом «Какие фрагменты АО на каком языке лучше описать?»

За время становления предметной области «Архитектура» был предложен, испытан и внедрён в практику ряд языков описания архитектуры (Architecture Description Languages, ADL).

К числу таких языков относятся языки ACME [90], Rapide [94], Wright [95], Aesop [91], C2 SADL [93] и другие, представленные на рис.3.22.

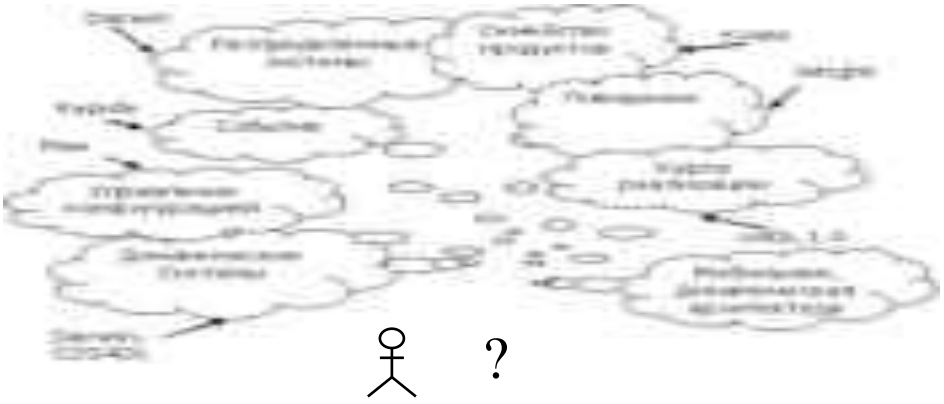


Рис.3.22.Проблема выбора языка архитектурного описания

В семантике и выразительных возможностях языков архитектурного описания много общего. По этой причине для демонстрации на рис.3.23 представлен фрагмент описания архитектуры для клиент-серверной композиции.

```
System simple_cs  = {
Component client  = {Port send request}
Component server = {Port receive-request}
Connector rpc   = { Roles {caller, callee}}
Attachments : {client.send-request to rpc.caller;
                    Server.receive-request to rpc.callee}}
```

Рис.3.23. Фрагмент описания архитектуры

В применениях языков архитектурного описания различаю как позитивы, так и негативы:

1. Позитивы:

позволяют описывать архитектурные структуры формально;

нацелены на их использование как человеком, так и машиной;

поддерживают описание на более высоком уровне абстракции, чем это было возможно до описания;

позволяют осуществлять анализ архитектуры с позиций соответствия, неопределённостей и реализуемости;

нацелены на автоматическое порождение определённых состояний или форм ПО.

2.Негативы:

- отсутствуют общепринятые соглашения о том, что ADLs должны представлять;

- представления на таких языках трудны для машинного разбора и не поддерживаются коммерческими продуктами;

- большая часть языков создана с академическими, а не коммерческими целями.

В настоящее время широкое использование для построения архитектурных описаний находит язык UML. Детальный анализ языков и причин их трудного внедрения в практику представлен в [75].

3.6.3. Методы проектирования

В практике создания архитектур АС накоплен достаточный опыт проектирования архитектуры, оформленный в виде методов. Так, например, с каждой из приведённых выше концептуальных архитектурных схем связана определённая система построения и использования архитектуры АС. Легко доступны через Интернет, например, системы методов DoDAF [77], TOGAF [80] и FEAF [78]. Для небольших проектов интересна и полезна система методов SPAMMED [59].

В то же время особого внимания, из-за их пионерского вклада в теорию и практику архитектурного моделирования АС, заслуживают методы, созданные в Институте программной инженерии Carnegie Mellon [6,17-20]. По этой причине раскроем эти методы в деталях.

Учёными и специалистами SEI создана система методов разработки архитектуры, обслуживающих действия архитектора или архитекторов по основным этапам жизненного цикла. К числу этих методов относятся: Architecture Tradeoff Analysis Method (ATAM), Cost-Benefit Analysis Method (CBAM), Quality Attribute Workshop (QAW), Active Reviews for Intermediate Designs (ARID), Attribute-Driven Design (ADD). Место использования методов в рамках жизненного цикла отражено в таблице.3.7.

Таблица 3.7

Уровень жизненного цикла	**QAW**	**ADD**	**ATAM**	**CBAM**	**ARID**
Бизнес необходимости и принуждения	Ввод	Ввод	Ввод	Ввод	
Требования	Ввод Вывод	Ввод	Ввод Вывод	Ввод Вывод	
План архитектуры		Вывод	Ввод Вывод	Ввод Вывод	Ввод
Детальный план					Ввод Вывод
Реализация					
Тестирование					
Развёртывание					
Поддержка				Ввод Вывод	

Разумеется, совокупность методов, разработанных в SEI, не перекрывает те виды активностей, которые приходится использовать при построении архитектуры. Место этих методов в составе совокупности активностей отражают таблицы 3.8 и 3.9.

Таблица 3.8

Стадии жизненного цикла	**Архитектурно-центрированная разработка**
Деловое моделирование	Создание системы документов, фиксирующих бизнес - цели: среда, возможности, обоснования и ограничения, презентация образца
Требования	Извлечение, проверка, систематизация и документирование требований (на основе сценариев, раскрывающих атрибутику качества)
Архитектурное проектирование	Проектирование архитектуры с использованием ADD; Документирование архитек5турны с использованием совокупности видов; Анализ архитектуры с использованием ATAM, ARID и CBAM
Детальное проектирование	Проверки с использованием ARID.
Реализация	
Тестирование	
Развёртывание	
Сопровождение	Использование ADD и CBAM.

Таблица 3.9

Метод	Роль	Строка описания	Детали трудового процесса	Продукты
QAW	Системный аналитик	Требования	Понимание необходимостей организатора	Бизнес выбор Добавочная спецификация
ADD	Архитектор ПО	Анализ и замысел	Определение кандидатов архитектур Исполнение архитектурного синтеза	Архитектурные документы ПО
ATAM/CBAM	Технический обозреватель	Анализ и замысел	Усовершенствование архитектуры	Обзор записей Архитектурные документы ПО
ARID	Технический обозреватель	Анализ и замысел	Усовершенствование архитектуры Анализ режимов работы	Обзор записей

Представим сущность этих методов, используя их имена в виде аббревиатуры на английском языке. Первым с позиций его включения в общий поток работ по разработке АС является метод QAW. Его назначение связано с формированием подсистемы требований (как части общей системы требований), на основе которых будет проводиться разработка архитектурной модели. Метод «стоит» на границе между этапами «Анализ требований» и «Проектирование архитектуры». Его применение позволяет внести в общую систему требований вклад (в виде подсистемы этой системы), который будет использован при проектировании архитектуры АС.

Метод QAW базируется на коллективной работе отобранной группы лиц и включает следующую совокупность действий:

1. Презентация группе метода QAW с позиций его сущности и методик.
2. Представление группе архитектурного плана.
3. Выбор и идентификация архитектурных драйверов (характеристик качества).
4. Выявление и формирование множества сценариев.
5. Отбор сводного списка сценариев.
6. Определение приоритетов в списке сценариев.
7. Коррекция списка и его систематизация.

Роль входной информации для такой работы выполняют бизнес use-case и документ «Концепция (Vision)» как архитектурный план. Результатами реализации метода являются откорректированный бизнес use-case и библиотека сценариев. Основная работы при исполнении метода осуществляется в виде рассуждений разной формы.

Следующим, с позиций последовательности применения методов SEI, идёт метод проектирования ADD, использование которого базируется на следующей циклической совокупности действий:

1. Отобрать модуль для декомпозиции.

2. Усовершенствовать представление модуля в соответствии со следующими шагами:

2.1. Выбрать архитектурные драйверы.

2.2. Выбрать архитектурные образцы, удовлетворяющие драйверам.

2.3. Проиллюстрировать модуль примерами и распределить функциональности на основе use-case. Представить модуль с помощью подходящей системы «видов».

2.4. Определить интерфейсы с дочерними модулями

2.5. Проверить и откорректировать use-case и сценарии, выражающие качества, и сформировать на их базе ограничения для дочерних модулей.

3. Повторить шаги для следующего модуля.

Роль входной информации для метода выполняют «концепция» (как ограничения), use-case модель (как источник функциональных и нефункциональных требований) и библиотека сценариев. Выходом служит комплект документов, раскрывающих декомпозицию, распараллеливание и вид, фиксирующий «размещение». И при исполнении этого метода основным видом работ являются рассуждения разной формы.

Методы QAW и ADD представлены обобщённо по публикации [6] авторов методов, которая раскрывает возможность их включения в потоки работ RUP. Главным при исполнении методов является их опора на сценарии, при работе с которыми необходимо придерживаться следующих рекомендаций:

1. При отборе для модуля (или компонента) подходящих сценариев из библиотеки сценариев необходимо отдавать предпочтение тем, которые наиболее существенны и выполняются чаще других.

2. Исходя из определённого архитектурного стиля, проводится определение места модуля в процессе исполнения каждого отобранного сценария и дополнительных (с позиции сценария) модулей, обеспечивающих целостное исполнение сценария.

3. Особо внимание уделяется интерфейсам и обменам сообщениям между модулями для каждого сценария.

3.6.4. Подходы к оцениванию архитектуры

Для анализа характеристик архитектуры АС и оценки её пригодности, а также для сравнительного анализа альтернативного набора архитектур в SEI был разработан ряд методов [18], из которых первым был метод анализа *архитектуры ПО* (Software Architecture Analysis Method, SAAM), идея которого представлена на рис.3.24.

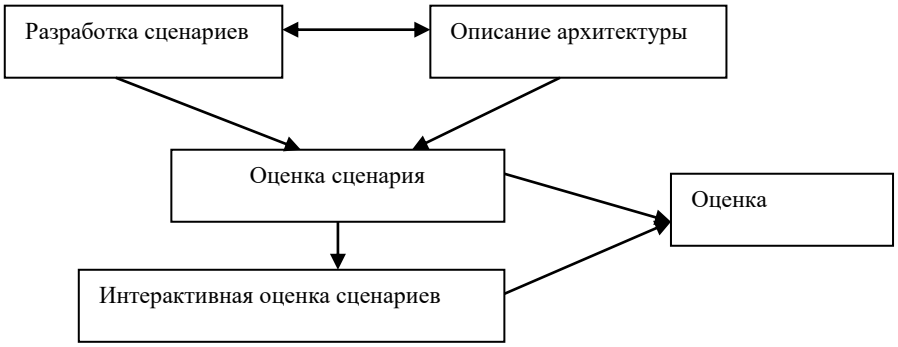


Рис. 3.24.Схема анализа архитектуры

В основе использования метода SAAM лежит работа со сценариями, в содержании которых отражается необходимая для построения архитектуры и её оценивания система функциональных и нефункциональных требований к АС.

Применение метода предполагает выполнение следующих шагов:

1. Определить архитектуру (или несколько сравниваемых архитектур). Описание архитектуры должно быть нормативным для используемой технологии разработки и, возможно, подготовленным для целей анализа и используемых методов оценки.

2. Для оцениваемой архитектуры отобрать из библиотеки сценариев (если она есть) и из других источников представительный набор сценариев. Чем представительней и адекватней набор сценариев, тем выше будет качество анализа. Со сценариями можно связать ожидаемые риски.

3. Провести классификацию сценариев с позиций их потенциальной реализуемости в рамках архитектуры (в том числе и с учётом возможных изменений архитектуры).

4. Для каждого сценария с учётом их классификации провести оценку степени реализуемости в рамках оцениваемой архитектуры. Определиться с необходимыми изменениями архитектуры с достаточной степенью детализации.

5. Выявить взаимодействие сценариев с учётом смысловых связей между взаимодействующими сценариями. Степень связности несёт информацию о потенциальной декомпозиции структуры АС, зарегистрированной в описании архитектуры.

6. Оценить архитектуру в целом (или сравнить несколько заданных архитектур). Для этого целесообразно использовать подходящие методы оценки, учитывающие важности сценариев и степень их поддержки архитектурой.

Кроме метода SAAM, в SEI для оценки архитектур были разработаны, испытаны и внедрены в практику методы ATAM, CBAM и ARID [6].

В основе метода ATAM лежит следующая совокупность шагов:

1. Презентация группе метода ATAM с позиций его сущности и методик.

2. Представление группе совокупности бизнес-драйверов.

3. Представление группе архитектуры.

4. Идентификация архитектурных подходов.

5. Формирование дерева качества.

6. Анализ архитектурных подходов.

7. Мозговой штурм и приписывание приоритетов сценариям.

8. Анализ архитектурных подходов.

9. Представление результата.

Входными данными для метода являются бизнес use-case, документация на архитектуру и содержимое библиотеки сценариев. В результате исполнения ATAM формируются архитектурные документы, включающие результаты анализа и предложения по коррекции. Основным видом работ при исполнении метода являются рассуждений (индивидуальные и коллективные).

В основе метода CBAM лежит следующая совокупность шагов:

1. Проверить и сопоставить сценарии.

2. Усовершенствовать сценарии.

3. Приписать сценариям приоритеты.

4. Определиться с межсценарными характеристиками качества и построить их дерево.

5. Разработать архитектурные стратегии и определить уровни реакций атрибутов качества.

6. Определить выгоды от ожидаемых значений характеристик качества.

7. Вычислить общую выгоду, полученную от архитектурной стратегии.

8. Выбрать архитектурные стратегии на основе возврата от инвестиций.

9. Найти интуитивные подтверждения полученных результатов.

Входными данными для метода являются бизнес use-case, документация на архитектуру и содержимое библиотеки сценариев. В результате исполнения CBAM формируются архитектурные документы, включающие результаты оценок выгод, и предложения по развитию (обогащению) библиотеки сценариев. Основным видом работ при исполнении метода являются рассуждений (индивидуальные и коллективные).

В основе метода ARID лежит следующая совокупность шагов:

1. Определиться с группой рецензентов.

2. Подготовить справку по проектированию.

3. Подготовить группу родственных сценариев.

4. Подготовить другие необходимые материалы.

5. Провести презентацию (в группе рецензентов) сущности и деталей ARID.

6. Провести презентацию проекта.

7. Провести мозговой штурм и приписать приоритеты сценариям.

8. Применить сценарии.

9. Провести обобщение.

Функции входа в ARID выполняют документы архитектурного описания, в которых представлены элементы проектирования доступных сервисов, и выборки из библиотеки сценариев. Применения метода оформляется в виде отчёта

по ревизии. Основным видом работ при исполнении метода являются рассуждений (индивидуальные и коллективные).

На практике применяют и другие методы проектирования архитектуры [25, 59, 80], но в любом случае исполнение этих методов базируется на рассуждениях групп специально отобранных лиц и оформлении результатов их работы в виде определённой совокупности документов.

3.6.5. Документирование архитектурных решений

В построении архитектуры АС рекомендуется (целесообразно и полезно) использовать стандарт IEEE-1471,опираясь и на другие стандарты и нормативы. Результат таких построений должен быть оформлен документально. В состав документов целесообразно включать не только окончательный результат, но и промежуточные результаты, то есть документирование должно сопровождать разработку архитектуры на всех этапах её жизненного цикла. Хорошим руководством по документированию архитектуры способна служить монография [17].

Вопрос о подходящей системе документов, фиксирующих процесс и результата разработки архитектуры, должен решаться с учётом специфики организации, заказавшей разработку АС или выполняющей заказ. Однозначный ответ о требуемом наборе документов отсутствует. Существуют только конкретные инструментально-технологические среды и практика авторитетных организаций (например, Microsoft, IBM, Siemens).

Разработка документации на архитектуру является работой, в процессе исполнения которой приходится решать определённые задачи. Сущность задач необходимо обязательно привязывать к тем нагрузкам, которые возлагаются на «документацию архитектурного описания». К основным применениям документации архитектуры относят:

1. Использование документации для целей обучения, в первую очередь тех лиц, которые будут воплощать архитектуру в реализацию АС. К числу важных групп лиц, которым способна помочь документация на архитектуру, относятся пользователи АС и лица, ответственные за сопровождение АС.

2. Использование документации на архитектуру в виде средства коммуникации между лицами, вовлечёнными в разработку АС на всех этапах жизненного цикла, особенно на ранних этапах.

3. Использование документации для целей анализа состояния разработки и принимаемых решений. И в этой задаче документы по архитектуре полезны для их применения на всех этапах жизненного цикла, особенно на этапах сопровождения системы и создания её новых версий.

В каждой конкретной разработке совокупность задач следует сформулировать и решать с учётом специфики предметной области и используемых инструментально технологических средств. Только в этом случае будет достигнут баланс между требованиями, предъявляемыми к документам на архитектуру АС.

В основе системы документов на архитектуру АС лежат документы (рис.3.25), фиксирующие:

- «виды», вложенные в архитектуру;
- информацию, раскрывающую, как «виды» интегрируются в единое целое.

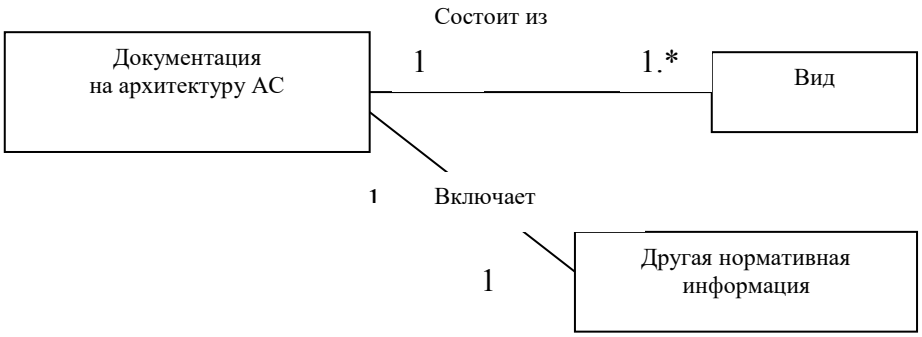


Рис. 3.25. Схема документирования

Виды являются средством, вводящим в документацию базовую структуру не только АС как целого, но и на уровне вида. В практике документирования зарекомендовала себя структура документов на вид, представленная на рис.3.26
.

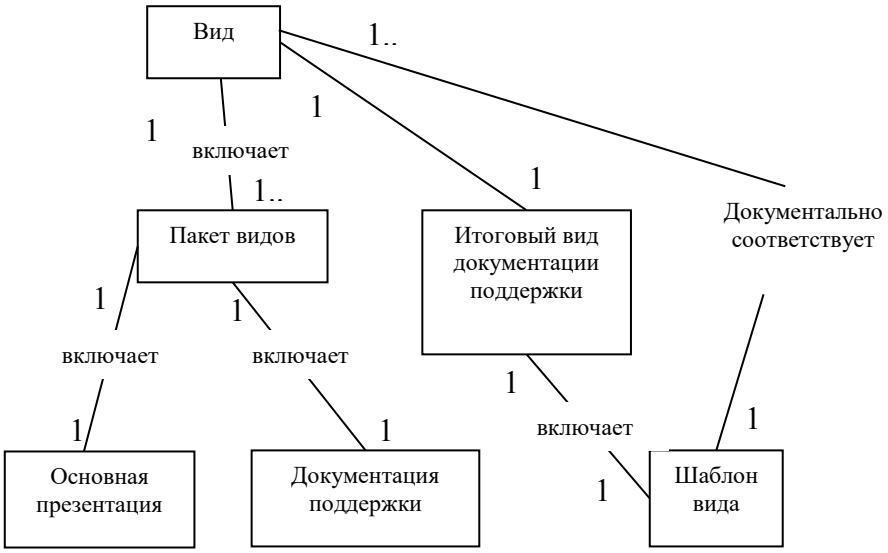


Рис.3.26. Структура документов

Структура документов на вид включает следующее:

- в общем случае (из-за большого количества моделей и документов) документальное представление вида можно разложить по пакетам;

- содержимое каждого пакета может включать графические представления (например, моделей) и сопровождающую (поддерживающую) документацию;

- за рамками пакетов документируется их интеграция и другая поддерживающая информация, для создания целостного документального представления вида;

- если имеется шаблон (template) для документирования вида, то все отмеченные позиции заполняются в соответствии с шаблоном;

- включение документов и их частей, не предусмотренных шаблоном, если есть основания внести изменения в шаблон.

Документы на каждый вид и другие документы должны быть включены в архитектурное представление АС так, чтобы структура общего описания и доступ к частям этого описания были удобны для решения задач, возложенных на

систему документов на архитектуру. Для подобных структур документов также разрабатывают и используют шаблоны.

3.6.6. Рассуждения в разработке и использовании архитектуры

В пособии не раз отмечалось, что в основе построений и использований архитектуры АС лежат рассуждения лиц, заинтересованных в разработке АС. Такие рассуждения в одних определённых условиях выполняются одним лицом, а в других определённых условиях выполняются коллективно (например, мозговой штурм в формировании или анализе системы сценариев).

Одним из средств инициирующих и направляющих рассуждения в архитектурном моделировании являются образцы (например, образцы архитектурных стилей, семантическая сеть IEEE-1471, образцы архитектурных описаний). Каждый образец неявно «задаёт вопросы» о составляющих, из которых он состоит, их свойствах и отношениях между составляющими. На такие вопросы необходимо ответить для того, чтобы попытаться адаптировать образец к исследуемому случаю, возможно, внеся в него изменения.

Для работы с определённым образцом можно использовать различные варианты рассуждений. От того, как организовывать и осуществлять подобные рассуждения, существенно зависит результат работы архитектора(ов).

К сожалению, в практике разработок АС, использующей богатейшие коллекции образцов, в том числе и для архитектурного моделирования, практически отсутствуют образцы для работы с рассуждениями. К числу немногих результатов в области «автоматизированной поддержки рассуждений в процессе разработки АС» относится ряд результатов, полученных в SEI [5], в которых явно называются «рассуждения, reasoning» и предлагаются схемы их реализации. Интерес к рассуждениям в SEI появился около пяти лет назад и связан, в основном, с их базовыми идеями по характеристикам качества АС при построениях и оценках архитектурных моделей.

3.6.7.Аспектно-ориентированный подход к структуризации и интеграции архитектуры АС

Для практики разработок современных АС типичны случаи, когда «интересы», необходимые для учёта, должны оставлять следы, материализующие факт их учёта в совокупности мест, распределённых по физической реализации АС. К такого рода интересам относятся, например, интересы, связанные с требованиями к АС по качеству (раскрыто в п.3.5.1).

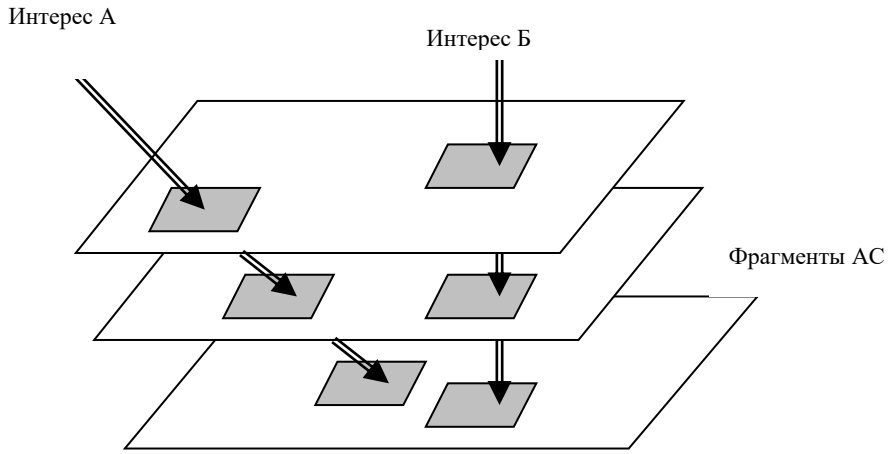


Рис.3.27. Пересечение «аспектов»

Материализованные «следы интересов» в их концептуальном представлении принято называть «аспектами». Реальность «интересов», приводящих к их представлению в виде распределённой совокупности «материализованных аспектов» такова, что они пересекаются (рис.3.27):

- либо разделяя некоторую совокупность фрагментов ПО;
- либо оставляя «следы» одного «интереса» на следах другого «интереса».

Для включения кодов следов в состав фрагментов ПО разработаны специальные средства программирования, получившие название средства «аспектно-ориентированного программирования» [2]. Но для того, чтобы знать, в какие фрагменты ПО и в какие места фрагментов «следы» включить, следует в процессе проектирования АС использовать средства «аспектно-ориентированного

проектирования». Именно такое распределение аспектов по коду программных составляющих АС составляет сущность «аспектно-ориентированного проектирования» [2]. Аспектно-ориентированное проектирование считают очередным шагом в эволюции подходов к проектированию, который следует за «объектно-ориентированном анализом и проектированием» и дополняет его.

Основные проектные решения, относящиеся к задачам аспектно-ориентированного проектирования АС, принимаются при построении архитектуры АС.

Вопросы по третьей главе

Q1. Какая из архитектурных парадигм является наиболее низкоуровневой?

Компонентно-ориентированная парадигма.
Объектно-ориентированная парадигма.
Сервисно-ориентированная парадигма.

Q2. В каком стиле общие данные разделяет определённая совокупность приложений?

Стиль, основанный на репозитарии.
Событийно-ориентированный стиль.
Одноранговый стиль.
Мнгоуровневый стиль.

Q3. Какой стиль используется для композиции приложений, которые обращаются к родственному типу событий?

Стиль, основанный на репозитарии.
Событийно-ориентированный стиль.
Одноранговый стиль.
Мнгоуровневый стиль.

Q4. Чем характеризуется клиент-серверный стиль тонкого клиента?

На сервере компоненты управления данными и логика представления.
На сервере компоненты управления данными и логика приложений.
На сервере компоненты логики приложений и представления.
На сервере компонент управления данными.

Q5. Чем характеризуется клиент-серверный стиль толстого клиента?

На клиенте компоненты управления данными и логика приложений.
На клиенте компоненты логики приложений.
На клиенте компонент логики представления.
На клиенте компоненты логики приложений и представления.

Q6. Чем характеризуется трёхуровневый серверный стиль?
Включает уровень базовых объектов.
Включает уровень интерфейса.
Включает уровень представления.
Включает уровень бизнес-логики.

Q7. Какая отличительная черта у брокерного стиля?
Организован доступ клиентов к совокупности серверов.
Организован доступ ряда различных приложений к общим данным.
Бизнес-логика вынесена в отдельный уровень.
Приложения напрямую обращаются к серверу.

Q8 К какому стилю можно отнести архитектуру WWW?
Тонкий клиент.
Толстый клиент.
Брокерный стиль.
Одноранговый стиль.

Q9. Какова функция уровня бизнес-логики?
Разделяет доступ клиентов к общим приложениям.
Представляет данные от разных серверов в унифицированном для системы виде.
Разделяет и координирует доступ ряда различных приложений к общим данным.
Организует доступ определённого количества клиентов к совокупности серверов.

Q10. Какие языки относятся к языкам описания архитектуры?
Язык UML
Язык ADL.
SDL.
Язык C++.

Q11. Что включает в себя вид с позиции разработки в архитектуре "4+1"?
Прецеденты и сценарии, охватывающие архитектурно существенное поведение, классы или технические риски.
Описание задач (процессов и нитей), их взаимодействия и конфигурации и распределения объектов и классов по задачам.
Краткий обзор модели выполнения в терминах модулей пакетов и уровней.

ГЛАВА 4. СОВЕРШЕНСТВОВАНИЕ НОРМАТИВНОГО АРХИТЕКТУРНОГО ПРЕДСТАВЛЕНИЯ СИСТЕМ

4.1. Основы совершенствования стандарта IEEE-1471

Текст предшествующих трёх глав идентичен пособию, опубликованному автором около 10 лет назад **[**91**]** где в списке литературы использовались два источника [42] и [77], представляющих прошлое, настоящее и будущее предметной области «Архитектурное моделирование систем с программным обеспечением». В этой главе пособия раскрывается как на самом деле происходило совершенствование архитектурного моделирования (реализация «будущего») за прошедшие десять лет на примере адаптации стандарта **IEEE 1471** к международной нормативной базе **ISO/IEK**.

Началось с того, что в 2007 году Международная организация по стандартизации (ISO) опубликовала стандарт **ISO / IEC 42010: 2007**, ***Systems and software engineering — Recommended practice for architectural description of software-intensive systems***. (*Системы и программная инженерия — Рекомендуемая практика архитектурного описания систем, интенсивно использующих программное обеспечение*). Текст этого стандарта **ISO / IEC 42010: 2007** был идентичен стандарту **IEEE 1471: 2000** и был взят за основу для совместной ревизии ISO и IEEE.

Основные направления, цели и составляющие ревизии представлены в [92] и в этой их версии рассматривались на международной конференции **WICSA'2007** (**Working IIIE/IFIP Conference on Software architecture**).

Было отмечено, что стандарт **IEEE-1471** выполнил несколько своих первоначальных целей:

1. Установить систему отсчета терминов и понятий для архитектурного описания.
2. Кодифицировать лучшие практики для архитектурного описания систем, интенсивно использующих программное обеспечения».
3. А также служить в качестве основы эволюции мышления в предметной области «**Архитектуры программное обеспечения**».

Среди основных целей усовершенствования стандарта были названы:

1. Расширение сферы применения приложений стандарта от программных систем до **общей архитектуры с**истем (включая архитектуру предприятия);
2. Согласование с процессами жизненного цикла **ISO** (**ISO 15288**) и разработки программного обеспечения (**ISO 12207**);
3. А также создание общего словарь для согласования терминов и концепций с другими проектами архитектуры **ISO**, включая **RM-OD**P (**ISO 10746**) и **GERAM** (**ISO 15704**).

В результате четырёхлетней ревизии и согласований был принят и в настоящее время активно используется стандарт **ISO/IEC/IEEE 42010:2011**, ***Systems and software engineering — Architecture description (Системная и программная инженерия — Описание архитектуры)***, который доступен по ссылке [93]. Стандарт русифицирован и включён в российскую нормативную базу как **ГОСТ Р 57100—2016 "Системная и программная инженерия. Описание архитектуры"** [94].

Содержание стандарта раскроем следующим образом. Его разработка будет интерпретироваться как **проект нормативной системы** с архитектурой, представленной **совокупностью видов**, каждый из которых фокусируется на определённой особенности стандарта или их совокупности. Такое решение выводит на возможность использование терминологии и моделей стандарта **IEEE 147**1, представленных в предыдущих главах пособия.

Начнём с обобщённо-онтологического вида, приведённого на рисунке 4.1, для понимания которого приведём следующие комментарии:

1. **Систем**у, всегда следует рассматривать в **контексте её ок**ружения (среды) с учётом **интересов** (concerns) **заинтересованных сторон** (stakeholders).

Рис. 4.1. Архитектурное моделирование в разработке систем

2. Каждая система имеет **архитектуру**.
3. Архитектура описывается с помощью **описания архитектуры (architectural description, AD).**
4. **Заинтересованные стороны** используют **AD**, в первую очередь, для **понимания** архитектуры и, посредством этого, **понимания** системы.
5. **AD** создается с использованием **архитектурных каркасов** (architecture frameworks, **AF**) и **языков описания архитектуры** (architecture description languages, обозначено выше как **AD**L)
6. AD включает в себя следующее:
   - Спецификации **заинтересованных сторон**, вовлеченных в разработку и эксплуатацию системы и, в первую очередь, их **интересы**, которые учтены;
   - **Точки зрения** на архитектуру и соответствующие им **виды**;
   - Использованные **модели** выбранных **архитектурных типов**;
   - **Обоснование архитектуры** и **архитектурные решения**, соответствующие обоснованиям;
   - **Правила соответствия** и **экземпляры** таких правил.
7. Архитектурные **точки зрения** обслуживают построение видов и их овеществление в системе.
8. Архитектурные **виды** строятся на базе подходящих **архитектурных моделей**.
9. **Виды** порождаются и регулируются на основе **точек зрения**.
10. Построение **моделей** управляются на основе **типов моделей**.
11. **Обоснование архитектуры** оправдывает **архитектурные решения**.
12. **Правила соответствия** определяют достижение **соответствия**.

Таким образом стандарт определяет совокупность требований к структуре и содержанию следующих артефактов:

• Архитектурное описания системы;

• Выбранные для описания архитектурные каркасы;

• Языки описания архитектуры;

• Выбранные для **AD точки зрения** на архитектуру.

И всё это в контексте следующего определения архитектуры:

> ***Архитектура - это фундаментальные концепции или свойства системы в ее среде, воплощенные в ее элементах, отношениях и принципах ее проектирования и эволюции***

Частично повторяясь с уже сказанным выше, отметим ряд деталей:

1. В онтологии стандарта различают **элементы AD**, к числу которых относятся составляющие набора учитываемых интересов, модели их источников (каждого из различных **заинтересованных сторон**), представления точек зрения, видов, типы моделей, архитектурные решения, обоснования и другие.
2. К числу **заинтересованных сторон** относятся как определённые лица (например, проектировщики, пользователи, владельцы, поставщики и другие лица), а также (возможно) другие системы, **интересы** (требования) которых приходится учитывать в создании АД.
3. С каждым **интересом** к системе связано определённое отношение определённой **заинтересованной стороны** или группы сторон, например заинтересованность в том, чтобы система обладала определённой характеристикой качества.
4. Каждая **точка зрения** устанавливает **соглашения** для построения, интерпретации и использования в архитектурном представлении материализуемого воплощения соответствующего **интереса**. В общем случае, определённый **интерес** может исходить от ряда точек зрения. Соглашения обслуживаются лингвистическим средствами (например, **ADL**), обозначениями, типами моделей, правилами проектирования, методами моделирования и другими операциями над архитектурными элементами и их композициями.
5. **Вид** выражает архитектурную составляющую, представляющую определённый **интерес** или их совокупность согласованно с соответствующей **точкой зрения**.
6. **Тип модели** определяет типовой вариант block-and-line схемы изображающей **вид** или его составную часть.
7. **Архитектурная модель** является результатом спецификации соответствующего **типа модели** в определённых условиях, например, определённая UML-диаграмма.

8. **Обоснование** обычно включает объяснение, обоснование или аргументацию в отношении принимаемых архитектурных решений основу для принятия решения, альтернативы и компромиссы, возможные последствия решения и ссылки на источники дополнительной информации.
9. **Решение** определённый результат выбора, оказывающий воздействие на архитектуры системы или воплощённый в элементах архитектуры или их композициях.
10. **Правило с**оответствия определяющее согласование совокупности видов и **соответствие** как реализация правила или их группы для совокупности видов.
11. **Архитектурный карк**ас (например, каркас Захмана, **TOGAF** или **RM-ODP**), интегрирующий в единое целое определённую совокупность видов. Включает в себя соглашения, принципы и практики для описания архитектур, установленных в определенной области применения и / или сообществе **заинтересованных сторон**.
12. **Язык архитектурного опис**ания (например, Rapide, Райт, SysML или ArchiMate), обслуживающий формирование AD и его составляющих.

Архитектурное описание для конкретной системы создается архитектором или их группой, на которых в коллективе разработчиков лежит принципиальная ответственность. В существующих технологиях разработки систем с программным обеспечением такой фронт работ связывают с ролью «архитектор», для исполнения которой в технологию встраивается необходимый инструментарий.

4.2. Системная ориентация архитектурного моделирования

Принципиальное отличие стандарта **ИСО/ИЕК 42010** от предшественников в том, что он указывает на необходимость конструктивного использования системного подхода, в соответствии с которым:

1. Любую систему **S** как совокупность компонент и связей между ними следует рассматривать в контексте её связей с окружающей её среды.
2. С системной **точки зрения**, любая компонента системы **S** может рассматриваться как подсистема (тоже как система) в согласованной с ней окружающей среде.

Более того, с каждой системой связан её жизненный цикл по ходу которого происходит её становление и существование в определённых состояниях до момента времени, когда существование системы прекратится.

Именно такое понимание системы конструктивно детализировано в стандартах **ISO 15288** и **ISO 12207**, ориентация на которые рекомендована в стандарте **ИСО/ИЕК 42010**. Но такая рекомендация не категорична и разрешает лицам, использующим стандарт, вкладывать его нормы в различные жизненные циклы, различные модели процессов и различные методы архитектуры.

В то же время, в любом случае применения стандарта к некоторой системе S, её окружение (компоненты окружения со связями между ними) на интервалах жизненного цикла следует конструктивно представлять и специфицировать. Такая работа лежит в сфере ответственности архитектора (или группы архитекторов) системы S.

С позиций архитектуры, конструктивное представление среды системы в обязательном порядке должно включать:

1. Модели того, что стоит за объектами, процессами и факторами, оказывающими воздействие на систему по ходу её жизненного цикла.
2. Модели тех лиц и их групп, которые проявляют заинтересованность в системе на этапах её жизненного цикла.

А значит составляющие среды, связанные с воздействиями и заинтересованностями, следует выявлять, моделировать с учётом связей между ними и учитывать в процессах архитектурного моделирования. Такой вид активности отражает концептуальная модель, приведённая на рисунке 4.2, на котором составляющие среды воздействующие на жизненный цикл системы, представлены как «сущности». В реальности «сущность» может быть субъектом или их группой, объектом или их совокупностью, не инкапсулируемым в объект образованием (полем ил подобием ему), системой, процессом или их совокупностью.

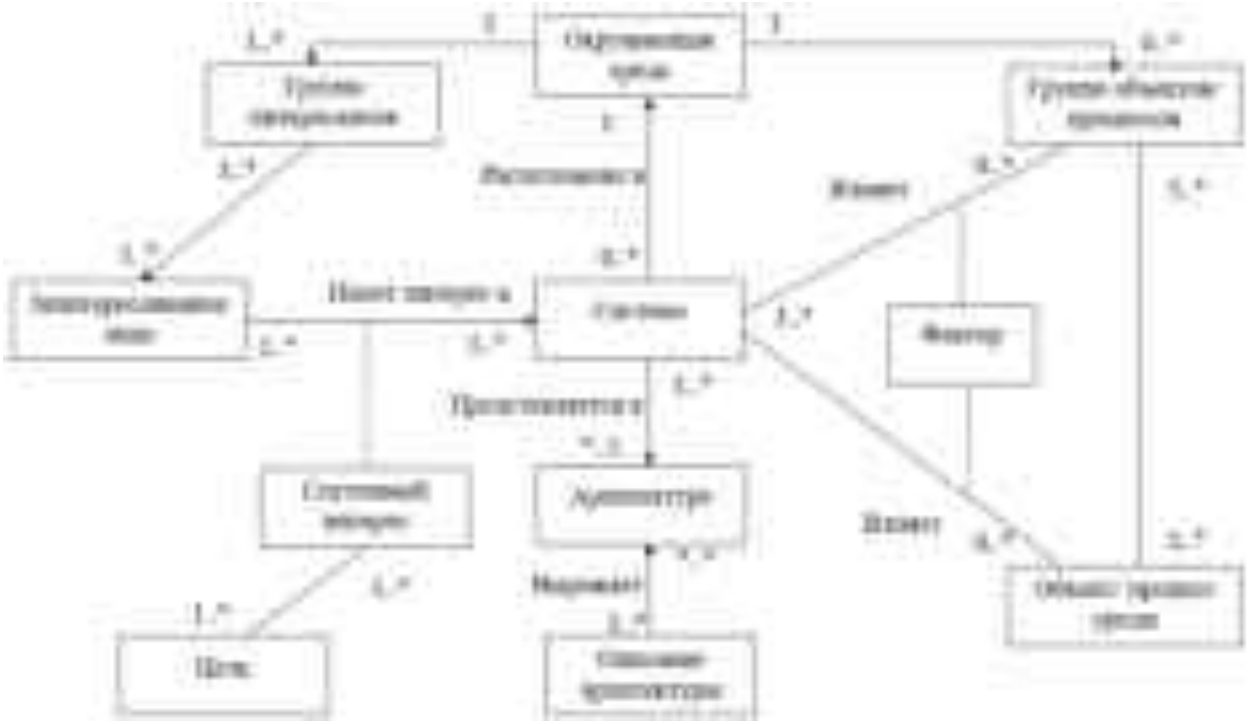


Рис. 4.2. Детали среды архитектурного моделирования

На рисунке 4.2 также показано, что с каждым прямым или опосредованным влиянием на архитектуру определённой составляющей среды связано определённое обстоятельство, которое следует представлять и учитывать, как соответствующий фактор. Ряд таких факторов был представлен в п. 1.4, раскрывающем сложность процесса разработки **АС** и «силовое давление» на этот процесс.

Отметим, что в архитектурном моделировании приходится оценивать широкий спектр контекстуальных факторов, прежде чем принимать решение о наиболее подходящем процессе разработки и его результатах. Практически в любом проекте по ходу его разработки возникают непредсказуемые заранее ситуационные обстоятельства, что является одним из важнейших источником причин уникальности проектов.

Очень полезный обзор ситуативных факторов приведён в публикации [95], в которой исследовано около 400 ситуационных факторов, аналитическая обработка которых привела авторов к справочнику, содержащему 8 классификаций 44 факторов, выводящих на 170 подчинённых подфакторов. Все единица справочника содержательно определены.

Такое влияние среды на разработку в её рамках системы подсказывает целесообразность создания полезных видов на эту сущность, удовлетворяющих рекомендациям стандарта **ISO/IEC/IEEE 42010:2011**. Полезный анализ такого применения стандарта приведён в публикации [96], в которой её авторы предложили различать в среде (в контексте разработки систем) её составляющие, представленные на рисунке 4.3.

Версия контекстной структуризации построена исходя из того, что разрабатываемые приложения будут работать в постоянно меняющихся условиях с **точки зрения** их технических, эксплуатационных и социально-экономических характеристик и должны учитывать этические, социальные, юридические, безопасность и экономические проблемы.

Однако, обычно программные системы концептуализируются и моделируются на основе функциональных и нефункциональных требований, в то время как внешняя среда исполнения, зависимости и контекст неизвестны, воспринимаются как должное или предполагаются или ведут себя определенным образом, что может привести к необоснованным предположения

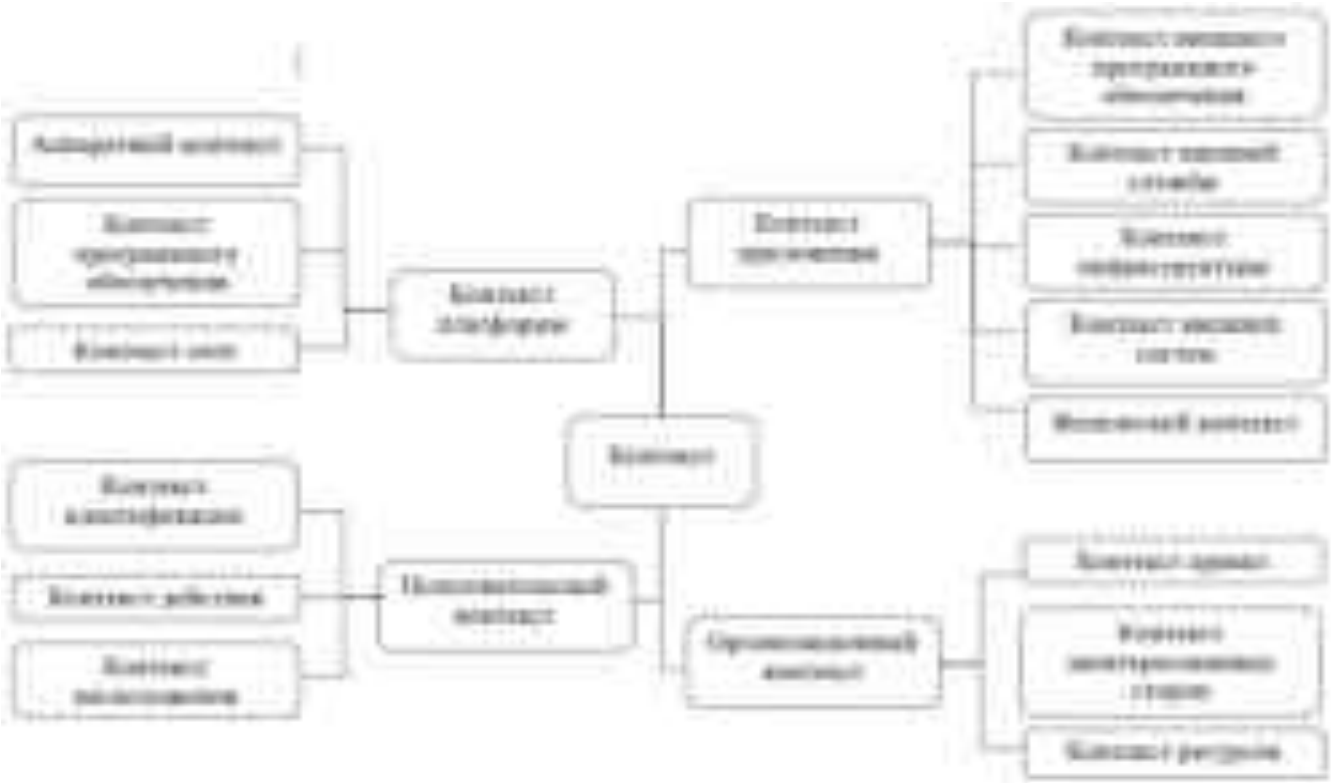


Рис.4.3. Типовые составляющие окружающей среды

Содержание составляющих схемы обобщённо понятно по их названиям. Разумеется в разработке конкретной системы для этих составляющих придётся проводить анализ их содержания в тех деталях, которые придётся учитывать в процессе проектирования. Авторы публикации [96] считают, что в такой работе следует ориентироваться на следующие вопросы:

Какие составляющие находятся в диапазоне управления системой (системная область), а какие нет (область контекста)?

Где граница между системой и ее контекстом и какие взаимодействия между системой и ее контекстом пересекают эту границу?

Кто является пользователями системы; каковы их типы, роли и характеристики; и как и где они получают доступ и используют систему?

Какие внешние службы и / или приложения имеют отношение к системе, включая их свойства и поставщиков?

Какова ожидаемая или желаемая среда технического исполнения, в которой система будет работать?

Какие **заинтересованные стороны**, включая организации и их ресурсы, влияют на систему и каким образом?

Какое влияние оказывает система на организации и **заинтересованные стороны**?

Ещё одной версией учета составляющих среды и их представления является явный учёт пространства проектирования (design space, **S**), документирование его проявлений, а также построение и использование полезных моделей S приводят к позитивам как для процесса проектирования **АС**, так и его результатов.

В наиболее общем плане под пространством проектирования понимают открытое динамическое образование, состоящее из связных объектов разной природы, ограничивающих активность проектировщиков, вовлечённых в согласованную работу. Такое понимание **S** предполагает явное или опосредованное взаимодействие проектировщиков с его составляющими. Более того, оно подсказывает целесообразность отображения **S** на его концептуальное представление - концептуальное пространство проектирования [97].

.

4.3. Заинтересованные лица и их интересы

Разнообразие **заинтересованных сторон** и их **интересов** создает богатство окружающей среды разрабатываемой системы, что, в свою очередь, определяет сложности, в условиях которых архитекторы обязаны осуществить сбалансированный учёт **интересов** и их адекватную материализацию в архитектурных решениях конкретного проекта.

А значит выявление **заинтересованных сторон** и понимание их **интересов** является основой успешной архитектуры. Такая работа должна найти свое явное выражение в её результате, например, в виде представленном (для примера) на рисунке 4.4, где отражено, что отобранные для учёта **интересы** должны быть согласованно систематизированы. По сути дела, за отбором и учётом **интересов** стоит формирование совокупности требований к системе, определяющих ту архитектуру, которую и будет иметь система после её разработки. А значит, эта совокупность определяет класс архитектурных требований к системе.

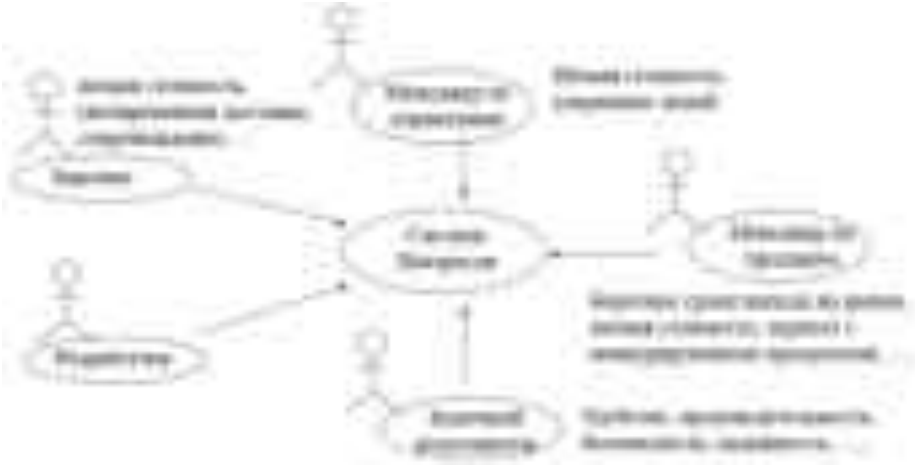


Рис. 4.4. Системазизация интересов

Следует отметить, что для выявления **заинтересованных сторон** и их **интересов** целесообразно использовать подходящие информационные средства (например, справочники) и инструменты, помогающие архитектуру и повышающие результативность его действий.

Обое место среди таких информационных средств занимают перечни типовых **интересов**, напрмер, следующий их список, выбранный из стадарта

ИСО/ИЕК 42010 и из публикации [98]: приемлемость, доступность, отчетность, точность, адаптивность, администрирование, доступность, гибкость, уверенность, аудитодоступность, аутентификация, автономность, доступность, резервное копирование, бизнес-цели, бизнес-стратегии, сертификация, совместимость, полнота , сложность, соответствие нормативным требованиям, концептуальная целостность, параллельность, конфиденциальность, конфигурируемость, согласованность, контроль, правильность, стоимость, доверие, опыт работы с клиентами, настраиваемость, доступность данных, целостность данных, конфиденциальность данных, надежность, развертывание, аварийное восстановление, документация, долговечность, легкость обучения, простота использования, экономичность, эффективность, защита окружающей среды, обработка ошибок, расширяемость, отказоустойчивость, осуществимость, гибкость, функциональность, общность, внедрение взаимозаменяемость, интернационализация, интероперабельность, обучаемость, правовые, лицензирование, локализуемость, логистика, ремонтопригодность, управляемость, мобильность, изменяемость, модульность, мониторинг, топология сети, открытость, оперативность, эксплуатационные расходы, оптимизация, организация, производительность, постоянство, совместимость с платформой, переносимость, предсказуемость, цена, конфиденциальность, качество обслуживания, возможность восстановления, соответствие нормативным требованиям, надежность, повторяемость, отчетность, воспроизводимость, время отклика, отзывчивость, повторное использование, надежность, безопасность, масштабируемость, расписание, удобство обслуживания, простота, стабильность, изменение состояния, поддержка, живучесть, устойчивость, технологические ограничения, тестируемость, пропускная способность, своевременность, надежность, понятность, удобство использования, универсальность, управляемость.

Отметим, что ориентируясь на типовые **интересы**, следует помнить, что за материальным воплощением каждого **интереса** в проекте стоит определённая цена, а значит, выявляя **интересы**, релевантные проекту, следует учитывать вклад их цены в общую стоимость проекта.

Типовые **интересы** принято структурировать. Наиболее обобщённо, виды **интересов** представлены на рисунке 4.5, где в отдельный вид выделены мотивы, цели, намерения и стремления [99].

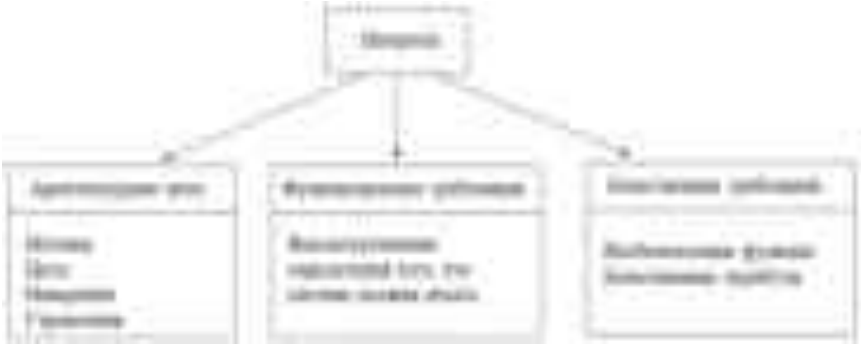


Рис. 4.5. Разновидности интересов

Для типовых **интересов** и их подмножеств, отобранных для разработки конкретного проекта, используют не только их имена и виды, а также их определения (спецификации), включающие область значений для каждого атрибута каждого определения. В формулировке определений **интересов** принято ориентироваться на концептуальную модель их окружения, представленную на рисунке 4.6.

Рис. 4.6. Концептуальная модель «интереса»

Отметим, что для многих **интересов** существуют их определения. Так, например, для **интересов**, связанныъ с качеством, полезны стандарты **ISO9126: 2001** и его усовершенствование **ISO 25010:2010**.

Определения (спецификации) **интересов** должны в обязательном порядке учитывать их согласованную объективацию (материальное воплощение) в реализованном проекте. Более того, по ходу проекта следует целессообразно отслеживать состояния объективации каждого **интереса** и согласованности их объективации.

Обобщая содержание пункта, отметим, что в разработке архитектурного описания, принципиальными видами работ являются:

1. Выявление **интересов** лиц, прямо или опосредованно имеющих отношения к проекту.
2. Согласованная спецификация отобранных **интересов** и, тем самым, формирование системы **интересов**.
3. Мониторинг состояния объективации каждого из **интер**есов (степень достижения запланированнх значений материализации) и их системы по ходу реализации проекта.

4.4. Концептуальная модель

Центральное место в стандарте ИСО/ИЕК 42010 занимает вид, модель которого (рисунок 4.7) расширяет концептуальный каркас стандарта IEEE1471, представленный во второй главе пособия на рисунке 2.1. Как и для предыдущего стандарта, по ходу применения каркаса его составляющие следует наполнить содержанием, соответствующим разрабатываемому проекту. В процессе спецификации схемы, представленной на рисунке 4.7, основное внимание архитектора направлено на формирование «архитектурного описания», в котором центральное место занимают конструкты типов «вид моделей» и «**точка зрения**».

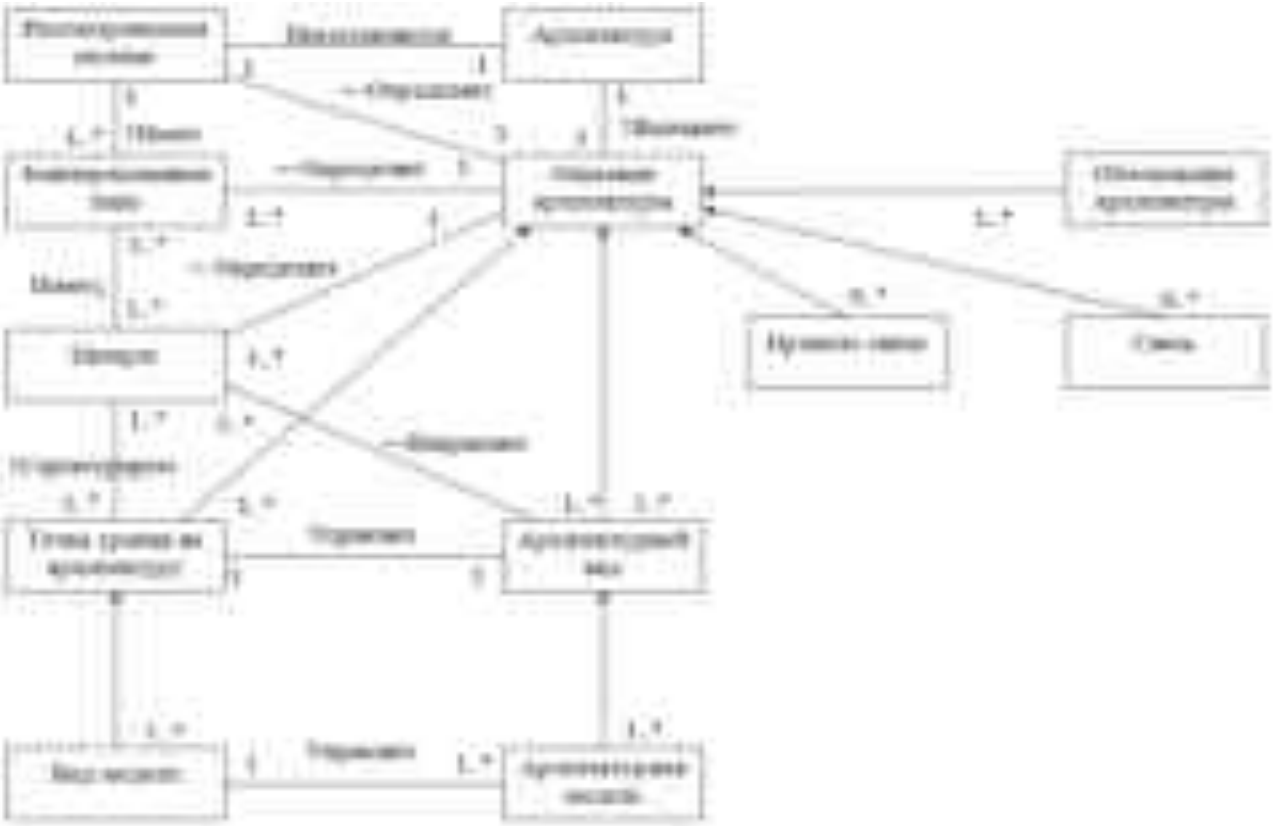


Рис. 4.8. Концептуальная модель стандарта

При создании каждого вида, архитектору приходится, учитывая его физическую и концептуально-алгоритмическую реализуемость, выполнять следующие работы:

1. Согласованно и обоснованно выбирать и/или строить составляющие его модели так, чтобы их композиция составляла системное образование в форме, способствующей пониманию структуры и содержания вида.
2. Порождая вид, обеспечивать его согласованность с остальными видами архитектурного описания.
3. При обнаружении несогласованностей, вводить необходимые изменения в уже построенные виды, используя подходящие механизмы управления изменениями.

Несложно заметить, что за обобщённо названными работами стоит целесообразность использования лучших практик, освоенных в постоянно расширяющейся предметной области «Архитектурного моделирования систем». Например, в известнейшей технологии Rational Unified Process [100], специальный набор лучших практик включён в состав инструментального сопровождения роли «Архитектор» [101].

Необходимость использования архитектором (или лицом, исполняющим роль архитектора, или их группой) расширяемого набора лучших практик приводит к их оценкам с позиции профессиональной зрелости осуществляемого архитектурного моделирования. Для таких оценок принято использовать так называемые модели профессиональной зрелости (Maturity Models) [102].

С позиций профессионально зрелого архитектурного моделирования и оценок его реализации следует различать:

1. Профессиональную зрелость построения архитектурного описания разрабатываемой системы.
2. Профессиональную зрелость овеществления архитектурного описания в реализации системы.

Именно по этой причине, архитектурное описание (изменяясь, когда в этом возникает необходимость) востребовано на всех этапах жизненного цикла соответствующей системы.

Полезная версия модели профессионально зрелого построения **AD** приведена в публикации 2, в которой специфика уровней зрелости отражена в их названиях.

Рис. 4.8. Уровни профессиональной зрелости

Раскроем детальнее содержание уровней.

На «Диаграммном» уровне зрелости для описания архитектуры используются средства графики, например, средства псевдографики Microsoft PowerPoint. Использование block-and-line схем облегчает понимание архитектуры. Основным недостатком таких схем является их семантическая бедность, из-за чего содержание диаграмм может быть неверно истолковано, что может приводить в проекте к семантическим ошибкам.

Использование средств уровня «**Моделирование**» нацелено на уменьшение риска неправильного толкования за счёт формального описания архитектуры. Именно такая нагрузка возложена на языки типа **ADL**. Другим видом формального описания является использование мета моделирования. Таким образом, семантика определяется в метамодели, которая выражает, как структурированы действительные экземпляры модели. В реальной практике архитектурного моделирования, наиболее популярной метамоделью для описания программных систем является система метамоделей **Unified Modeling Language**.

На «**Документированном**» уровне в дополнение к документации уровня моделирования, в которой основное внимание уделяется описанию архитектуры, соответственно структуре системы, документирование раскрывает **проект-**

ные решения. Это означает, что в документации необходимо указывать использование шаблонов проектирования и их связь с элементами архитектуры. Этот уровень зрелости требует указания использования шаблонов проектирования, но не требует объяснений использования соответствующего шаблона проектирования.

Уровень «**Трансированный**» раскрывает **причинно-следственные связи проектных решений** с **требованиями** к разрабатываемой системе. Кроме того, при изменении требований зависимые проектные решения могут быть легко **идентифицированы** и надлежащим образом изменены. Основным преимуществом уровня 4 является сокращение использования несоответствующих шаблонов проектирования и **управление изменениями**.

Пятый уровень зрелости требует раскрыть **обоснования проектных решений**, исходя из **причин** и учитывая **альтернативы**. Такое обоснование решений способствует их пониманию лицами, которые не участвуют в разработке системы, но должны ее поддерживать. На этом уровне раскрываются **зависимости между решениями** и проводятся **обоснования** их **согласованности**.

Включение в процессы архитектурного моделирования интерпретации его процессов и результата с позиций профессиональной зрелости приводит:

1. К конструктивному представлению (на каждом уровне) «точек зрения» для соответствующих «видов» соответствующих **интересов**.
2. К возможности оценивания каждого **интереса** учитываемого в **AD** как лингвистической переменной с нечёткими значениями, привязанными к шкале уровней зрелости.

По первой из этих позиций напомним какое содержание вносится в понятие «**точка зрения**». Это набор образцов, шаблонов и соглашений для построения одного типа «архитектурных видов». В нем определяются **заинтересованные стороны**, чьи проблемы отражены в **точке зрения**, а также руководства, принципы и шаблонные модели для построения соответствующих видов.

Таким образом за определённым употреблением понятия «**точка зрения**» стоит построение соответствующего вида, возможно по шаблону из ресурсов «**точки зрения**», причём в диаграмматической форме с дополнениями, выполняющими интегрально роль совокупности требований к системе, реализация

которых приведёт к овеществлению соответствующего **интереса** с запланированной в архитектурном описании оценкой профессиональной зрелости.

В приведённом толковании «**точки зрения**» специально выделены два этапа действий – формирование совокупности требований к воплощению определённого **интереса** (например, характеристики качества) в разрабатываемую систему и материализация этих требований в разработанной системе.

Представленная на рисунке 4.8 структура уровней и их содержание относятся к оценкам первого этапа. Сохранит или нет (вложенная в **AD**) оценка зрелости **интереса** своё значение в разработанной **AC** зависит от уровня профессиональной зрелости «лучших практик», материализующих требования к воплощению **интереса** на этапе реализации, то есть на втором этапе.

По этой причине с каждым **интересом**, выбранным для воплощения в **AC**, целесообразно связывать две оценки его интерпретации как лингвистической переменной – первую оценку для представления **интереса** в **AD**, а вторую оценку для материализации в разработанной **AC**.

Отметим, что использование интерпретации **интересов** как лингвистических переменных с нечёткими значениями открывает возможность для вычислений оценок совокупностей **интересов**, учитываемых в **AD**, а также **AD** как целого. Более того, появляется возможность отслеживания значений оценок для текущего состояния процесса разработки.

4.4. Специфика архитектурных решений

В предыдущем пункте был раскрыт ряд деталей, связанных с действиями архитектора или их группы, нацеленными на формирование и использование «архитектурных видов». В результате таких действий каждый **интерес**, выбранный для его учёта в разрабатываемой **AC**, получает определённое диаграммматическое представление, дополненное деталями, соответствующими тому, каким образом **интерес** будет овеществлён в реализованной **AC**. А это означает, что каждый «архитектурный вид» - это форма выражения определённого требования или требований, которые должны быть материализованы в **AC**

Чуть выше было отмечено, что четвертый уровень профессиональной зрелости построений **AD** исходит из того, что в **AD** регистрируются связи с требо-

ваниями к АС , которые были выявлены за рамками архитектурного моделирования, в первую очередь с теми требованиями, на основании которых архитектор приступил к построениям **AD**, пытаясь их (согласовав) интегрировать в целостности в формах, способствующих пониманию.

Каждую из таких целостностей, то есть определённый «вид» *V(t*) архитектор начинает формировать, в общем случае, в рамках следующей импликации

составляющие которой имеют следующее содержание:

1. Для вида V(t), который решено вложить в **АС** известно намерение – его реализация в **АС** должна овеществить соответствующий **интерес**, представленный (семантически нечетко) лингвистической переменной. На этот факт в импликации у составляющей V(t) указывает знак вопроса «?».
2. Формирование вида начнётся и будет осуществляться в условиях U(t), в которых архитектор обязан учитывать сложившуюся до начала его работы совокупность требованиям, но он, если это полезно для проекта, может вносить в них изменения, а также расширять эту совокупность, в том числе за счёт требований, интегрируемых в V(t). На такой вид неопределённостей, с которыми сталкивается архитектор в совей работе, в составляющей U(t) указывает два знака вопроса «??».
3. Ещё больше неопределённостей, которые придётся преодолеть архитектору в той работе W(t), которую ему придётся выполнить, специфицируя импликацию для последующей материализации вида V(t). По этой причине W(t) в импликации помечена тремя знаками вопроса «???».

Специфика работы архитектора, обусловленная достаточной свободой в поиске ответов на вопросы, обусловленные отмеченными неопределённостями, определяет специфику архитектурных решений и их включение в нормативы стандарта **ISO 25010:2010**. В частности, в этот стандарт включена концептуальная модель, представленная на рисунке 4.9 и фокусирующая внимание на архитектурном решении.

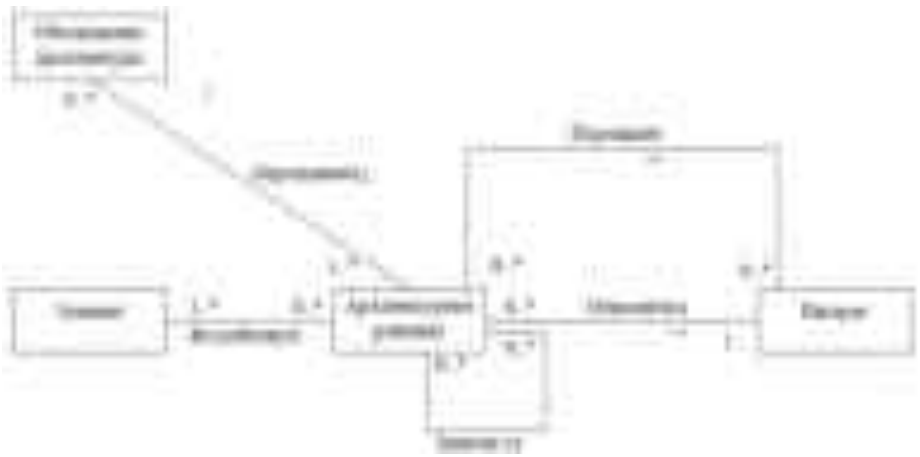


Рис. 4.9. Контекст архитектурного решения

Особой составляющей схемы архитектурного решения является то, что обозначено как «элемент». В наиболее общем плане «элемент» - это любая конструкция в описании архитектуры (**заинтересованная сторона**, **интерес**, **точка зрения**, вид модели, архитектурная модель, архитектурное решение, обоснование и другие составляющие **AD**). Важным является то, что многое включается в **AD** в результате решения, которое должно быть обосновано.

А значит, «архитектурное решение» можно и целесообразно рассматривать с позиций полезных **точек зрения**. В этом плане отметим исследования «архитектурных решений», представленные в публикации [103], авторы которой конструктивно определили и проверили на практике следующий **точки зрения**:

- «Детали принятия решений»;
- «Связь с требованиями»;
- «Хронология принятия решений»;
- «Участие **заинтересованных сторон** решения».

Исследования и спецификации предложенных точек зрения проведены в [103] с использованием следующего набора вопросов:

Q1 Какие решения были приняты?

Q2 Каков текущий набор соответствующих решений?

Q3 Каково обоснование решения D?

Q4 Какие **интересы** C_i имеют отношения к решению D?

Q5 Какие силы воздействуют / повлияли на каждое решение?

Q6 Какие решения затрагивают **интерес** Q?

Q7 Какие решения имеют находятся в конфликте с **интереса** C?

Q8 Какие решения затребованы в решении D?

Q9 Какие решения противоречат решению D?

Q10 Какие решения зависят от решения D?

Q11 Какие решения связаны с решением D?

Q12 Какие решения влияют на решение D (или элемент E архитектуры)?

Q13. На какие решения допускают изменение?

Q14 Какие решения будут затронуты при интеграции набора решения S?

Q15 Как применить набор решений из другого проекта?

Q16 Какие **заинтересованные стороны** затронуты решением D?

Q17 Какие решения затрагивают **заинтересованные стороны** S?

Q18 Какие **заинтересованные стороны** были вовлечены в решение D?

Q19 На какие решения влияют **заинтересованные стороны** S?

Q20 Каков порядок принятия в совокупности решений?

Q21 Какие решения были изменены с момента времени T?

Q22 Какие решения стали устаревшими после изменения CH?

Q23 Какие решения D или подпункты решения SG могут быть повторно использованы в других проекты?

В ответах на эти вопросы, которые были распределены между названными **точками зрений**, они были конструктивно специфицированы, и, что особо важно, авторами [103] был построен концептуальный каркас, представленный на рисунке 4.10.

Рис.4.10. Интегрированный вид на архитектурное решение

.

Интегрированный вид подчёркивает, что, в общем случае, архитектурное решение носит итерационный характер, обусловленный необходимостью согласовать совокупность решений в непротиворечивое целое.

Отметим, что в следующей за [103] публикации [104] авторы расширили набор точек зрения на архитектурное решение, добавив в него **точку зрения** «Силовое воздействия на архитектурное решение» (Force Decision Viewpoint), модель которой приведена на рисунке 4.11.

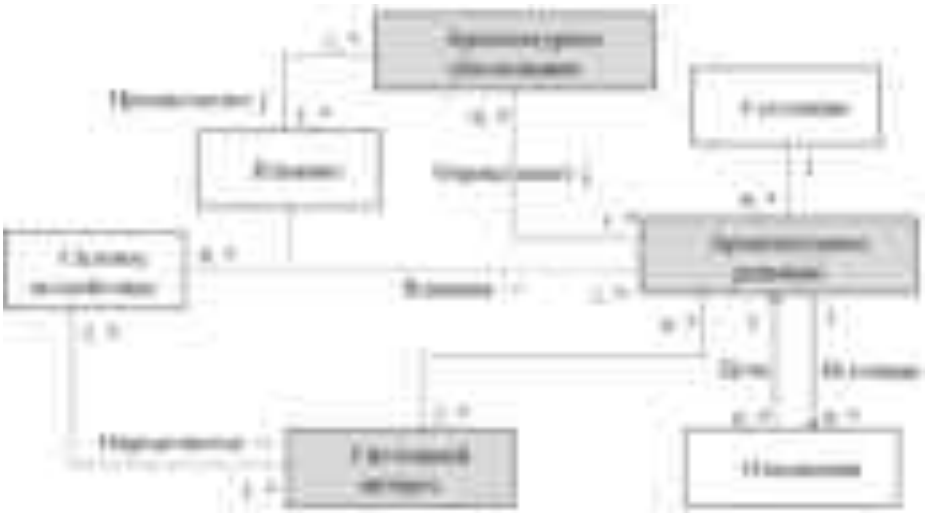


Рис.4.11. Силовое воздействие на архитектурное решение

Представления, использующие **точку зрения** «Силовое воздействия на архитектурное решение», четко определяют отношения между архитектурными решениями и силами, которые влияли на архитектора при принятии решений При наличии множества альтернатив. Термин «сила» берется из множества образцов, которые использует силы для разработки описания проблемы, которая должна быть решена с помощью подходящего шаблона. «Силу» можно определять, как «любой аспект проблемы, который следует учитывать при ее решении». [105]. Аналогичным образом, при рассмотрении архитектурных решений сила - это любой аспект архитектурной проблемы, возникающей в системе или ее окружении (операционная, развивающая, деловая, организационная, политическая, экономическая, правовая, нормативная, экологическая, социальная и т. Д.), учитывается при выборе среди доступных альтернатив решения. Напомним,

что силовая интерпретация воздействий разработку **АС** рассматривалась выше в п. 1.4 пособия.

Силы возникают из многих источников:

- Чаще всего из требований, а также из ограничений, принципов архитектуры и других «намерений», налагаемых на систему;
- Личных предпочтения или опыта архитектора(ов) и команды разработчиков;
- Бизнес-цели, в частности таких как ограничения по времени на разработку проекта, ограничений на финансирование;
- Стратегической ориентации на конкретные технологии.

Перед принятием решений архитектор рассматривает и оценивает все «силы», имеющие значение в контексте разрабатываемой системы. Это может быть хорошей практикой для формирования и использования списка типовых «сил», специфичных для предметной области **АС**.

Различные силы могут быть ортогональны друг другу, они могут поддерживать, противодействовать или противоречить друг другу. Поэтому архитектор должен балансировать силы для принятия наилучших возможных решений.

Так что, конструктивная спецификация **точки зрения** «Силовое воздействия на архитектурное решение» открывает возможность её использование по важному направлению согласования архитектурных решений.

Завершая пункт, отметим, что именно практика согласованного принятия архитектурных решений вывела на аспектно-ориентированное распределение **интересов** по архитектурным видам. Другими словами, в общем случае, конкретный **интерес**, например определённая характеристика качества не инкапсулируется в одном «архитектурном виде» (раскрывалось выше в п. 1.4). То есть, определённое качество распределяется по некоторой совокупности видов, в которых они могут «пересекаться» с другими характеристиками качества [106].

Вопросы по четвёртой главе

Q1. К какому типу интересов относится то, что называют «конфигурируемость»?

Архитектурные цели.

Качественные требования.

Функциональные требования.

Q2. Чем отличаются такие составляющие архитектурного описания системы как «точка зрения» и «архитектурный вид»?

Средства «точки зрения предназначены для построения «архитектурного вида».

Средства «архитектурного вида» предназначены для построения «точки зрения».

Не отличаются.

Q3. Какой формальный язык не используют при построении архитектурного описания?

SysML.

C#.

Rapid.

Q4. Что из ниже перечисленного может выполнять роль «интереса»?

Удобство пользователя.

Намерение.

Производительность.

Q5. Может ли интерес распределятся по совокупности видов?

Да, может.

Нет.

В особых случаях.

Q6. Для какого класса систем можно применять стандарт ISO/IEC/IEEE 42010:2011?

Информационные системы.

Системы теплоэнергетики.

Автоматизированные системы.

Q7. Что из ниже перечисленного оказывает воздействие на архитектуру системы?

Операционная среда.

Нормативные ограничения.

Экологическая обстановка применения системы

Q8 Спецификация каких составляющих AD остается за рамками стандарта IEEE-1471?

Правила соответствия.
Архитектурные решения.
Виды моделей.

Q9. На каком этапе разработки АС применим стандарт ISO/IEC/IEEE 42010:2011?
Концептуальный этап
Этап сопровождения.
Этап тестирования.

Q10. Какие языки относятся к языкам описания архитектуры?
Язык UML
Язык ADL.
SDL.
Язык C++.

Q11. Содержит ли стандарт ISO/IEC/IEEE 42010:2011 требования по аспектно-ориентированному распределению интересов по архитектурным видам?
Да, содержит.
Содержит рекомендации, а не требования.
Нет, не содержит.

Заключение

Во втором издании учебного пособия раскрывается специфика предметной области «Архитектура автоматизированных систем» по зарубежным (в основном) и российским источникам, включающим стандарты, монографии, статьи и отчёты, представленным в Интернете. Учебный материал пособия распределён по четырём главам. Содержание трёх первых глав идентично содержанию первого издания, а четвёртая глава содержит информацию о совершенствовании нормативной базы архитектурного моделирования за последнее десятилетие.

За время прошедшее после первого издания основная суть архитектурного моделирования не изменилась, а именно, в архитектурном моделировании должно осуществляться формирование архитектурного описания на основе «визуальных моделей», интегрированных в систему «архитектурных видов» исходя из совокупности «точек зрения» «заинтересованных сторон», у каждой из которых имеются определённые «интересы» и их (эти интересы) необходимо объективировать (овеществить) в разрабатываемой системе.

В четвертой главе, расширяющей первое издание пособия, особое внимание уделяется нормативному совершенствованию стандарта IEEE-1471, которое закреплено в стандарте **ISO/IEC/IEEE 42010:2011** и его русифицированной версии **ГОСТ Р 57100—2016. С достаточной детальностью** раскрыты причины: расширения ответственности нормативов архитектурного моделирования **(от систем, интенсивно использующих программное обеспечение до систем любого типа**); введения дополнительных структур в концептуальную модель (фреймворк) стандарта IEEE-1471; дополнения новой версии концептуальной модели уточнениями в виде совокупности подчинённых моделей; уточнения терминологии. Ещё одним расширением материалов первого издания является включение Приложения, которое содержит шаблон архитектурного описания системы.

Особое внимание уделяется вопросам качества АС, языкам описания архитектур и методам их проектирования, а также вопросам оценки и документирования результатов архитектурного моделирования. Обобщённо представляются

идеи аспектно-ориентированного анализа и проектирования АС. Каждая из глав заканчивается списком контрольных вопросов, на каждый из которых приведён ряд потенциальных ответов.

Список использованных источников

[1] Allen R. and D. Garlan. A Formal Approach to Software Architectures. // Proceeding, IFIP 92, Elsevier, 1992, pp. 134–141.

[2] Aspectj home page. Xerox PARC, USA. http://aspectj.org/.

[3] Architecture and Architecture Modeling Techniques. hpttp://www.agiledata.org/ essays/enterpriseArchitectureTechniques.html

[4] Agile Architectural Modeling. http://www.agilemodeling.com/essays/ agileArchitecture.htm

[5] Bass L. et al. Reasoning Frameworks. (CMU/SEI-2005-TR-007), Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[6] Bass L. et al. Software Architecture in Practice. Addison Wesley, Boston, MA, USA, 2003.

[7] Bittner K.and I. Spence. Use-Case modeling, Addison-Wesley, 2002.

[8] Building Core Ontologies. White paper of the DELOS Working Group on Ontologies Harmonization, 2003.

[9] Buschmann E et al. Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, 1996.

[10] Booch G. Software Architecture: The Next Step. // Proceeding, 1st European Workshop Software Architecture (EWSA 04), Springer, 2004.

[11] Booch G. The Limits of Software. http://www.booch.com/ architecture/ blog.jsp? part=Papers

[12] Booch G. The Complexity of Programming Models. http://www.booch.com/ architecture/ blog.jsp? part=Papers

[13] Booch G. Collaborative Development Environments. http://www.booch.com/ architecture / blog.jsp?part=Papers

[14] Booch G. Quantitative Observation and Theoretical Construction in Software Architecture. http://www.booch.com/ architecture/blog.jsp?part=Papers

[15] Charette R.N. Why software falls. IEEE Spectrum,vol 42,#9, 2005, pp. 36-43.

[16] Clarke S. and R. J. Walker. Composition patterns: An approach to designing reusable aspects. // Proceeding, InternationalConferenceonSoftwareEngineering, 2001, pp. 5–14.

[17] Clements P. et al., Documenting Software Architectures: Views and Beyond, Addison-Wesley, 2002.

[18] P. Clements P., R. Kazman and M. Klein, Evaluating Software Architecture. Addison-Wesley, 2002.

[19] Clements P. and L. Northrop. Software Product Lines: Practice and Patterns. Addison-Wesley, 2002.

[20] P.Clements et al. A practical method for documenting software architectures. http: //www-2.cs.cmu.edu/afs/cs/project/ able/ftp/icse03-dsa/submitted.pdf

[21] Clements P. et al. Tutorial F3: Documenting Software Architectures: Views and Beyond.// Proceeding, International Conference on Software Engineering, 2003.

[22] Cockburn A. Agile Software Development, Addison-Wesley, 2002,

[23] Dikel D.M., D. Kane and J.R. Wilson. Software Architecture: Organizational Principles and Patterns.— Prentice Hall.— 2001.

[24] Eeles P. Layering strategies. http://www/ibm.com/developerworks/rational/ library/4699.html

[25] Eeles P. Capturing Architectural Requirements. http://www/ibm.com/developerworks/ rational/library/4706.html

[26] Eeles P. What is a software architecture? http://www/ibm.com/developerworks/ rational/library/feb06/eeles

[27] Eeles P. Characteristics of a software architect. http://www/ibm.com/developerworks/ rational/library/mar06/eeles

[28] Eeles P. The benefits of software architecting. http://www/ibm.com/developerworks/ rational/library/may06/eeles

[29] Fielding R. T. Architectural styles and the design of network-based software architecture.// Dissertation, university of California, Irvin, 2000. www.ics.uci.edu/~fielding/ pubs / dissertation/fielding_cv_2000.htm

[30] Gamma, E., R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[31] Garlan D. and M. Shaw. An Introduction to Software Architecture. Advances in Software Engineering and Knowledge Engineering, vol. 1, World Scientific, 1993, pp. 1–39.

[32] Garlan D. Software architecture: a Roadmap The Future of Software Engineering. A. Finkekstein (Ed), ACM Press, 2000

[34] Hofmeister C., R. Nord and D. Soni, Applied Software Architecture. Addison-Wesley, 1999.

[35] Jacobson I., K. Palmkvist and S. Dyrhage, Systems of Interconnected Systems. // Report on Object-Oriented Analysis and Design (ROAD), May-June 1995, vol. 2, no. 1.

[36] Jucan G. A 3D Software Architecture Framework www.opendatasys.com/ 3D_Frmwk.html

[37] Katera M. and S. Katz. Architectural views of aspects. // Proceedin, International Conference on Aspect-oriented Software Development,2003, pp. 1–10.

[38] Kazman R. et al. An Architectural Analysis Case Study: Internet Information Systems

[39] Kazman R. et al. SAAM: A Method for Analyzing the Properties of Software Architectures // Proceeding, 16th Int'l Conf. Software Eng. (ICSE 94), IEEE CS Press, 1994, pp. 81–90.

[40] Kroll P. The Spirit of the RUP. The Rational Edge, 2001.

[41] Kroll P. and Ph. Kruchten. The Rational Unified Process Made Easy: A Practitioners Guide to the RUP. Addison-Wesley, 2003.

[42] Kruchten Ph. The 4+1 View Model of Software Architecture. IEEE Software, vol. 12, no. 6, 1995, pp. 42–50.

[43] Kruchten Ph. The Rational Unified Process-An Introduction. Addison-Wesley, 1998.

[44] Kruchten Ph., Henk Obbink, Judith Stafford. The Past, Present, and Future for Software Architecture. IEEE Software, IEEE Computer Society, 2006

[45] Leffingwell D. and D. Widrig, Management Software Requirements: A Unified Approach. Addison-Wesley, 1999.

[46] Luders F. Architectural Styles in Component-Based Software Engineering. www.idt.mdh.se/kurser/phd/CBSE/reports/Final_reports/report_05.pdf

[47] May N. A survey of software architecture viewpoint models. // Proceeding, The Sixth Australasian Workshop on Software and System Architectures (AWSA

2005), Swinburne University of Technology, Melbourne, Australia., 2005, pp. 13–24.

[48] Monin B. Practical Software Architecture For the Enterprise. www.safe-house.org/ SH-Book/ safe-house-book.html

[49] Norris D. Communicating Complex Architectures with UML and the Rational ADS. // Proceeding, IBM Rational Software Development User Conference, 2004

[50] Obbink H. et al. Report on Software Architecture Review and Assessment (SARA). V1.0, Feb. 2002; www. philippe.kruchten.com/architecture/SARAv1.pdf.

[51] Obbink H. et al. COPA: A Component-Oriented Platform Architecting Method for Families of Software-Intensive Electronic Products (Tutorial). // . Proceeding, 1st Software Product Line Conf. (SPLC1), 2000.

[52] Ommering R. et al. The Koala Component Model for Consumer Electronics. IEEE Trans. Computers, 2000, vol. 33, no. 3.

[53] Perry D.E. and A.L. Wolf. Foundations for the Study of Software Architecture. ACM SIGSOFT Software Eng. Notes, Oct. 1992, pp. 40–52.

[54] Putman J. Architecting with RM-ODP. Prentice Hall, 2000.

[55] Ran A. ARES Conceptual Framework for Software Architecture. Software Architecture for Product Families: Principles and Practice, M, Addison-Wesley, 2000.

[56] Rechtin E. Systems Architecting: Creating and Building Complex Systems. Prentice Hall, 1991.

[57] Rechtin E. and M. Maier, The Art of Systems Architecting. CRC Books, 1997.

[58] W.E. Royce, W. Royce, Software Architecture: Integrating Process and Technology. TRW Quest., 1991, vol. 14, no. 1.

[59] Rotem-Gal-Oz A. Service Oriented Architecture. http://www.rgoarchitects.com/blog/ default.aspx

[60] Rotem-Gal-Oz A. Software Architecture. http://www.rgoarchitects.com/blog/ default.aspx

[61] Selic B. The Pragmatics of Model-Driven Development. IEEE Software, 2004, vol. 20,

[62] Shaw M. and P. Clements, The Golden Age of Software Architecture: A Comprehensive Survey, tech. report CMU-ISRI-06-101, Inst. for Software Research Int'l, Carnegie Mellon Univ., Feb. 2006.

[63] Shaw M. The Coming-of-Age of Software Architecture Research. // Proeeding,. 23rd Int'l Conf. Software Eng., IEEE CS Press, 2001, pp. 656–664.

[64] Shaw M. and P. Clements. A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems. // Proceeding, COMPSAC 97: Int'l Computer Software and Applications Conf., IEEE CS Press, 1997, pp. 6–13.

[65] Shaw M. and D. Garlan, Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, 1996.

[66] Soley R. Model-Driven Architecture. Object Management Group, 2000.

[67] Sommerville I.Software Engineering. Addison Wesley, Boston, MA, USA, 6th edition, 2000.

[68] Soni D. et al. Software architecture in industrial applications.// Ptroceedi8ng, International Conference on Software Engineering, pages 196–207, 1995.

[69] Sosnin P. Question-Answer Processor for Cooperative Work in Human-Computer Environment. // Proceeding, the 2 International IEEE conference Intelligent System 2004 p.452-456/.

[70] Tang A, J. Han and P.Chen, A Comparative Analysis of Architecture Frameworks // Technical Report: SUTIT-TR2004.01,CeCSES Centre Report: SUT.CeCSES-TR001/ 2004.

[71] The Standish group, Charting the Seas of Information Technology-Chaos, The Standish Group International, 1994.

[72] Wang G. and C. K. Fung Architecture Paradigms and Their Influences and Impacts on Component-Based Software Systems // Preceeding, 37 Hawaii International Conference on Systems Sciences, 2004, 10pp.

[73] Witt F., Baker and E. Merritt, Software Architecture and Design: Principles, Models and Methods. Van Nostrand Reinhold, 1994.

[74] Wood E. Using Architectural Perspectives. http://www.eoinwoods.info/index.php?page=articles

[75] Wood E. Software Architecture: Stakeholders, Viewpoints, Perspectives. http://www. eoinwoods .info/index.php?page=articles

[76] Wood E. The Past, Present and Future of Software Architecture. http://www. eoinwoods .info/ index.php?page=articles

[77] Wood E. Architectural Evaluation for Fun and Profit! http://www.eoinwoods.info/ index.php? page=articles

[78] Ссылки:

[79] DoD Architecture Framework and Software Architecture Workshop Report – March 2003 http://www.ichnet.org/DODAF%20SEI%20report.pdf

[80] FEAF: FEA Frameworks // www.eaframeworks.com/FEAF/index.html

[81] MODAF: Ministry of Defence Architecture Framework// www.telelogic.com/standards/modaf.cfm

[82] TOGAF: The Open Group Architectural Framework// www.togaf.org/

[83] G. Booch http://www.booch.com/ architecture

[84] I. Jacobson http://www. Ivarjacobson.com

[85] Дубина О. Обзор паттернов проектирования . http://www.citforum.ru /SE/ project/pattern

[86] Аналитический отчет. Анализ международного опыта стандартизации архитектуры программного обеспечения государственных информационных систем . http://projects.economy.gov.ru/pms/public/ PublicWorkProducts.aspx?projectid=695269fc-eaeb-474c-b0e3-99b905fa17d1

Стандарты:

[87] International Standard ISO/IEC 9126 – 1:2001 Software engineering - - Product quality // www.iso.org/iso/en

[88] International Standard ISO/IEC 12207 – Software Life Cycle Process// www.abelia/com/docs/12207cpt.pdf

[89] ISO/IEC 10746:1995, Reference Model of Open Distributed Processing (RM-ODP). ITU Rec. X901, 1995.

[90] IEEE--**1471**. Recommended Practice for Architectural Description of Software-Intensive Systems. Institute of Electrical and Electronics Engineers, Sept. 2000. IEEE Std 1471-2000.

Список использованных источников (Глава 4)

[91] Соснин, П. И. Архитектурное моделирование автоматизированных систем: учебное пособие / П. И. Соснин. – Ульяновск : УлГТУ, 2007. – 146 с.

[92] Updating IEEE 1471: Architecture frameworks and other topics with David Emery. Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008). preprint, slides

[93] ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description, pp. 1-46, Dec. 2011

[94] ГОСТ Р 57100-2016/ISO/IEC/IEEE 42010:2011 Системная и программная инженерия. Описание архитектуры, http://docs.cntd.ru/document/1200139542

[95] Clarke P. and O'Connor R.V. The situational factors that affect the software development process: Towards a comprehensive reference framework, Journal of Information Software and Technology, № 54(5), pp. 433-447, 2012

[96] Bedjeti A., Lago P., Lewis G. A., Boer R. D. D. and Hilliard R., Modeling Context with an Architecture Viewpoint, 2017 IEEE International Conference on Software Architecture (ICSA), pp. 117-120, 2017.

[97] M. A. Boden M. A. Conceptual Spaces, P. Meusburger et al., Eds., Milieus of Creativity, Knowledge and Space 2, Springer Science + Business Media B.V., pp. 235-243, 2009

[98] Hilliard R. Lessons from the fundamental unity of architecting In Software Engineering in the Systems Context, editors: Ivar Jacobson and Harold 'Bud' Lawson, 2015

[99] J. Schenkhuizen. Consistent Inconsistency Management: a Concern-Driven Approachhttps://dspace.library.uu.nl/bitstream/handle/1874/334223/thesisv1_digitaal.pdf

[100] Полис Г., Огастин Л., Мадхар Д. Разработка программных проектов: на основе Rational Unified Process (RUP). – М.: ООО «Бином-Пресс», 2009

[101] Кватрани Т., Палистрант Д. Визуальное моделирование с помощью IBM Rational Software Architect и UML. Пер. с англ. – М.: КУДИЦ-ПРЕСС. – 2007

[102] Rathfelder C. and Groenda H. Towards an Architecture Maintainability Maturity Model (Softwaretechnik-Trends vol 28 4) pp 3-7, 2008

[103] Van Heescha U., Avgerioua P. and R. Hilliard. A documentation framework for architecture decisions. The Journal of Systems & Software, **85**(4), pp. 795-820. 2012

[104] Van Heesch U, Avgeriou P. and Hilliard R. 2012. Forces on Architecture Decisions - A Viewpoint. In Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA-ECSA '12). IEEE Computer Society, Washington, DC, USA, 101-110)

[105] G. Mustapic, A. Wall, C. Norstrom, I. Crnkovic, K. Sandstrom,J. Froberg, and J. Andersson, "Real world influences on software architecture-interviews with industrial system experts," in Fourth Working IEEE/IFIP Conference on Software Architecture, pp. 101–111, 2004.

[106] Hilliard R. Using aspects in architectural description,h LNCS, volume 4765, pp-. 139-154, 2007.

ПРИЛОЖЕНИЕ

ШАБЛОН ОПИСАНИЯ АРХИТЕКТУРЫ СИСТЕМЫ В СООТВЕТСТВИИ СО СТАНДАРТОМ ISO/IEC/IEEE 42010

Использование шаблона

Это шаблон (руководство), предназначенный для документирования архитектурного описания (**AD**) в соответствии со стандартом ISO/IEC/IEEE 42010: 2011. Оригинальная версии шаблона разработана Р. Хиллардом [6].

Шаблон предоставляет схему для **AD** и определяет набор «слотов» или информационных элементов, которые будут разработаны архитектором (или группой архитекторов), использующего (-их) шаблон для создания **AD**. Каждый слот идентифицируется заголовком, за которым следует краткое описание его предполагаемого содержания и руководства для разработки этого содержания.

В шаблоне используются следующие метки:

* «**Musts**» - это предметы, которые должны присутствовать, чтобы соответствовать стандарту. Musts помечены следующим образом.

Δ «**Shoulds**» - это элементы, рекомендованные для присутствия, но не требуемые Стандартом.

Необязательные элементы отмечены следующим: (**необязательно)**.

Руководство, которое определяет, объясняет или иным образом усиливает требуемые элементы или термины, используемые в нем, выглядит следующим образом.

Последняя оригинальная версия шаблона доступна по адресу: http://www.iso-architecture.org/42010/templates/.

1. Введение

1.1 Идентифицирующая информация

*** Назовите описание архитектуры** <Имя архитектуры> **или, если необходимо, приведите имя** <Системы интересов> (<System-of-interests>), для которой создано описание архитектуры

Согласно ISO / IEC / IEEE 42010, система (или <Система интересов>) – это общее имя для любого типа их разнообразных версий, включая антропогенные системы, программные продукты и услуги, а также «системы, интенсивно использующие программное обеспечение» (Software Intensive Systems, SIS) включая «отдельные приложения, системы в традиционном смысл, подсистемы, си-

стемы систем, производственные линии, семейства продуктов, целые предприятия и другие представляющие интерес агрегаты».

Примечание. В пособии выделены автоматизированные системы (АС), которые правомерно рассматривать как подкласс SIS.

1.2 Дополнительная информация

*** Приведите дополнительную информацию, определенную проектом и / или организацией**.

Детали идентифицирующих и дополнительных информационных элементов не определяются Стандартом. В большинстве организаций или проектов введены и применяются собственные требования к документированию такой информации.

В состав идентифицирующей и дополнительной информации принято включать: дату выдачи и статус; авторы, рецензенты, утверждающие и/или согласующие документ или его составляющие; история изменений; резюме; объем; контекст; словарь терминов; информация по управлению версиями; информацию по управления конфигурацией и ссылки.

1.3 Другая информация (необязательно), включаемая в AD

Хотя архитектурные виды и модели являются основной формой организации в описании архитектуры, **AD** также может содержать информацию, не являющуюся частью любого вида или модели.

Примеры типов информации о архитектуре, которые не могут быть частью любого архитектурного вида:

- обзор системы или архитектуры;
- руководство для читателей по **AD**;
- результаты оценки архитектуры;
- обоснование ключевых решений;
- просмотр и соответствие моделей.

1.3.1 Обзор (необязательно)

Предоставьте обзор описываемой архитектуры, ее основные моменты и краткое изложение разрабатываемой системы.

Обзор может включать разделы для спецификации цели, области и контекста архитектуры.

Предоставьте обзор остальной части документируемого AD как руководство для его читателей.

1.3.2 Оценка архитектуры

*** Включите результаты любых оценок архитектуры, которые были документированы.**

1.3.3 Обоснование ключевых решений

*** Описание архитектуры должно включать обоснование для каждого решения, которое считается ключевым решением**

2. Заинтересованные стороны и их интересы

В этом разделе шаблона содержатся информационные элементы для заинтересованных сторон по архитектуре, проблемы заинтересованных сторон в отношении этой архитектуры и отслеживание интересов заинтересованных сторон.

2.1 Заинтересованные стороны

*** Определить и описать заинтересованные стороны для архитектуры.**

*** При подготовке АД следует учитывать следующие заинтересованные стороны:**

- Приобретающая сторона, или покупатель (acquirer)
- Заказчик, или клиент (customer)
- Разработчик (developer)
- Сопровождающая сторона (maintainer)
- Оператор системы (operator);
- Владелец (owner)
- Производитель (producer)
- Поставщик (supplier)
- Производитель (producer)
- Сопровождающая сторона (maintainer)
- Ликвидатор (disposer)
- Пользователь (user)

Примечание. Идентифицируя заинтересованные стороны следует иметь ввиду, что такую роль могут выполнять организации, системы (входящие в окружающую среду разрабатываемой системы) и другие сущности.

2.2 Интересы

*** Определите интересы, которые считаются фундаментальными для архитектуры системы.**

*** Рассмотрите другие интересы и включите их в AD, например, исходя из следующих вопросов**:

Какова цель (и) системы, представляющей интерес?

1 За счёт чего архитектура способствует достижению целей?
2 Насколько возможно построить и развернуть систему, представляющую интерес?
3 Каковы потенциальные риски и последствия системы, представляющей интерес для ее участников на протяжении всего жизненного цикла?
4 Как поддерживать и развивать систему интересов?

2.3 Прослеживаемость заинтересованных сторон

*** Свяжите каждый указанный интерес с идентифицированными заинтересованными сторонами, например, с помощью таблицы следующего вида.**

	Заинтересованная. сторона 1	Заинтересованная. сторона 2	Заинтересованная. сторона 3
Интерес 1	+		+
Интерес 2	+	+	
Интерес 3		*	+

3. Точки зрения +

AD содержит несколько видов архитектуры; каждое представление придерживается конвенций точки зрения архитектуры. В этой главе описываются требования к документированию точек обзора для **AD**.

*** Включите спецификации для каждой точки зрения архитектуры, используемой в этом AD.**

Точки зрения должны быть выбраны для **AD** таким образом, чтобы каждый идентифицированный интерес объективировался по крайней мере одной точкой зрения.

*** Укажите обоснование для каждой используемой точки зрения.**

Обоснование может включать обсуждение с точки зрения его заинтересованных сторон, озабоченности, сформулированные точкой зрения, актуальность ее модельных видов и соглашений о моделях.

Каждая точка зрения, используемая в **AD**, должна быть указана в соответствии с положениями ISO / IEC / IEEE 42010, 7.

Примечание. AD содержит один или несколько видов архитектуры, для формирования каждого из которых определена и используется точка зрения. В **AD** не требуется упорядочить взгляды или точки зрения. Лицам, взаимодействующим с **AD**, необходимо будет обратиться к спецификациям точки обзора, чтобы понять предмет представления, его обозначения, модели и используемые условные обозначения моделирования. Для упорядочения набора видов (Vi) и

согласованных с ними точек зрения (VPi), архитектор может использовать следующие возможные варианты:

1 Точки обзора, сначала: VPi, за которым следуют представления: Vi;
2 Чередующиеся представления с их точками зрения: Vi, VPi, Vj, VPj,

3.1 Наименование точки зрения

***Укажите имя точки зрения.**

Если существуют синонимы или другие общие наименования, которые известны для этой точки зрения, следует записать их.

3.2 Обзор для точки зрения

***Предоставьте абстрактный или краткий обзор точки зрения.**

***Опишите ключевые функции точки обзора**.

3.3 Интересы и заинтересованные стороны

Архитекторы, которые пытаются подобрать подходящую точку зрения, часто используют выявленные интересы и обычно учитываемые типы заинтересованных сторон. Поэтому важно (и это требуется Стандартом) документировать интересы и заинтересованные стороны, для которых предназначена точка зрения.

Может оказаться полезным зарегистрировать виды источников, для которых точка зрения *не* является *приемлемой.* Формулирование противоположных интересов может оказаться хорошим противодействием для определенных чрезмерно используемых моделей и нотаций.

3.3.1 Интересы

***Опишите каждый интерес**.

Интерес может быть очень общим (например, «надежность») или достаточно конкретной (например, «система должна поддерживать сетевую латентность»).

При её применении в AD, точка зрения становится «контрактом» между архитектором и заинтересованными сторонами, что соответствующие ей интересы будут учтены и специфицированы в виде, создаваемом с помощью этой точки зрения.

Примечание. Может быть полезно выразить интерес в форме вопросов, например:

1 Как система управляет сбоями?
2 Какие услуги предоставляет система?

Стандарт ISO / IEC / IEEE 42010, 5.3 содержит список возможных интересов, которые необходимо учитывать при создании описания архитектуры. Они могут быть рассмотрены (при заполнении шаблона для их соответствия документируемой точке зрения) в ответах на следующие вопросы:

1 Какова цель (и) учитываемой совокупности интересов?

2 Насколько пригодна документируемая архитектура для достижения целей совокупности интересов?

3 Насколько возможно построить и объективировать учитываемые интересы?

4 Каковы потенциальные риски и последствия учета объективируемых интересов для соответствующих заинтересованных сторон на протяжении всего жизненного цикла АС?

5 Как поддерживать и развивать учитываемую совокупность интересов

3.3.2 Типичные заинтересованные стороны

**** Сформируйте список учитываемых типов заинтересованных сторон.***

Напоминание. Каждая точка зрения затрагивает интересы определённых заинтересованных сторон, например, присутствующих в следующем списке:

- Приобретающая сторона, или покупатель (acquirer)
- Заказчик, или клиент (customer)
- Разработчик (developer)
- Сопровождающая сторона (maintainer)
- Оператор системы (operator);
- Владелец (owner)
- Производитель (producer)
- Поставщик (supplier)
- Производитель (producer)
- Сопровождающая сторона (maintainer)
- Ликвидатор (disposer)
- Пользователь (user)

Примечание. После того, как точка зрения выбрана для использования и применена в описании архитектуры, это описание архитектуры требуется задокументировать ассоциацией фактических заинтересованных сторон системы с интересами, структурированными с помощью каждой точкой зрения

3.3.4 «Анти-интерес»

Может оказаться целесообразным зарегистрировать те типы интересов, для которых эта точка зрения не подходит или не особенно полезна.

3.4 Виды моделей+

***Определите каждый тип модели, используемый в точке зрения.**

В стандарте каждый вид архитектуры состоит из нескольких архитектурных моделей. Каждая модель управляется типом модели (например, метамоделью), который устанавливает обозначения, условные обозначения и правила для моделей такого типа

***Повторите следующий раздел для каждого вида модели.**

3.5 Наименование вид модели

*** Определите вид модели.**

3.5.1 Соглашения для вида модели

*** Опишите соглашения для моделей такого вида.**

Соглашения включают языки, обозначения, методы моделирования, аналитические методы и другие операции. Это ключевые ресурсы моделирования, которые «тип модели» предоставляет архитекторам и определяет словари для построения моделей такого типа и, следовательно, раскрывает как эти модели интерпретируются и используются.

Целесообразно разделить эти соглашения на:

- часть относящуюся к языку (с точки зрения метамодели или спецификации обозначений, которые будут использоваться в документировании);
- и на часть процесса, в которой следует описать способы моделирования, используемые для создания моделей и практик, которые могут использоваться на моделях (к ним относятся операции над моделями этого типа.

Настоящий стандарт не определяет какой-либо один стиль для документирования видов моделей. Вид модели может быть зарегистрирован многими способами, включая:

I. задание метамодели, которая определяет его основные конструкции;
II. обеспечение шаблона модели для заполнения пользователями;
III. через языковое определение или с помощью ссылки к существующему языку моделирования;
IV. некоторую комбинацию этих методов.

Руководство для методов I)-III) представлено ниже.

I) Вид модели: метамодель

Метамодель представляет собой элементы описания архитектуры, которые включают в себя словарь вида моделей. Существуют различные способы представления метамодели. Метамодель следует представлять как:

- сущности (объекты): Каковы главные типы элементов, которые присутствуют в моделях этого вида?
- атрибуты: Какие свойства реализуют сущностные (объектовые) процессы в моделях этого вида?
- отношения: Какие отношения определены среди сущностей (объектов) в моделях этого вида?
- ограничения: Какие виды ограничений существуют для сущностей (объектов), атрибутов и/или отношений в моделях этого вида?

Сущности (объекты), атрибуты, отношения и ограничения - это любые «элементы» описания архитектуры

Примечание - Когда точка зрения определяет множественные виды моделей, полезно найти единственную точку зрения метамодели, унифицирующую определения видов моделей. Кроме того, часто бывает полезным использовать единственную метамодель, чтобы выразить множественные, связанные точки зрения (например такой, когда определяется структура архитектуры).

II) Вид модели: шаблон

Обеспечивается шаблон или форма, определяющие формат и/или содержание моделей этого вида моделей.

III) Вид модели: языки

Определяется существующая нотация или язык модели так, чтобы они могли использоваться для моделей этого вида. Описывается, если это необходимо, их синтаксис, семантика, поддерживающие инструментарии.

3.5.2 Операции с видом модели (необязательно)

***Укажите операции, определенные на моделях такого типа.**

3.5.3 Правила соответствия для вида модели

*** Документируйте любые правила соответствия, связанные с типом модели.**

3.6 Операции с видами и моделями

Операции определяют методы, применяемые к видам и их моделям. Операции могут быть разделены на категории:

- *Методы создания* - это средства, с помощью которых представления подготовлены с использованием этой точки зрения. Они могут быть представлены в форме руководства процесса (как начать, что делать в дальнейшем); руководства для рабочих продуктов (шаблоны для представлений этого типа); эвристики, стилей, образцов или других выражений;
- *Интерпретирующие методы* - это средства, с помощью которых представления становятся понятными заинтересованным сторонам системы и читателям;
- *Методы анализа* - используются для того, чтобы проверять, рассуждать, преобразовывать, прогнозировать, применять и оценивать архитектурные результаты из конкретного представления;
- *Методы проектирования и реализации* - используются для того, чтобы реализовывать или конструировать системы, применяя информацию из конкретного представления.

3.7 Правила связи

Документируются любые правила связи, определенные конкретной точкой зрения или ее видами моделей. Обычно эти правила будут "пересекающейся моделью" или "пересекающимся представлением", так как ограничения *в пределах* вида моделей будут определены как часть соглашений этого вида моделей.

3.8 Примеры

Этот раздел содержит примеры.

3.9 Примечания

Любая дополнительная информация, в которой пользователи этой точки зрения могут нуждаться или находят ее полезной.

3.10 Источники

Определяются источники конкретной точки зрения, если таковые имеются, включая автора, историю, литературные ссылки, предшествующие наработки.

4 Архитектурные виды+

Большая часть материала в AD представлена через его виды архитектуры. Каждая точка зрения следует конвенциям ее руководящей точки зрения. Представление состоит из архитектурных моделей.

*** Включите представление архитектуры для каждой точки зрения, выбранной в пункте 3.**

***Повторите и выполните следующий раздел для каждого вида архитектуры в AD.**

4.1 Вид: наименование

*** Назовите вид**

*** Укажите любую идентифицирующую и дополнительную информацию об этом виде.**

Скорее всего подробности этой информации будут указаны организацией и / или в проекте.

Архитектурные виды имеют свою собственную идентифицирующую и дополнительную информацию, отличную от **AD**, потому что они могут разрабатываться и развиваться отдельно в течение всего жизненного цикла проекта.

*** Определите точку зрения, регулирующую эту точку зрения, из числа тех, которые указаны в пункте 3**.

4.1.1 Модели +

Архитектурный вид состоит из одной или нескольких моделей архитектуры.

*** Обеспечьте одну или несколько моделей архитектуры, придерживающихся руководящей точки зрения**.

Модели должны учитывать все проблемы, созданные в соответствии с руководящей точкой зрения, и охватывать всю систему с этой точки зрения.

***Повторите раздел ниже для каждой модели**.

4.1.2 <Название модели>

* Каждая модель архитектуры должна включать идентификацию версии, указанную организацией и / или проектом.

* Каждая модель архитектуры должна идентифицировать свой тип управляющей модели и придерживаться конвенций этого типа модели из п. 3.5. См. ISO / IEC / IEEE 42010, 5.4.

Архитектурная модель может быть частью более одного вида архитектуры. Это позволяет обмениваться деталями и решать различные, но связанные с этим проблемы без избыточности. Другие виды использования нескольких моделей: аспектно-ориентированный стиль описания архитектуры: модели архитектуры, разделенные между представлениями архитектуры, могут использоваться для выражения архитектурных перспектив [6] и текстур архитектуры [5]. Архитектурные модели могут использоваться в качестве контейнеров для применения шаблонов архитектуры или стилей архитектуры для выражения принципиальных схем (таких как уровни, трехуровневый, одноранговый, модель-view-controller) в представлениях архитектуры.

4.1.3 Известные проблемы с представлением

*** Документируйте любые несоответствия между представлениями и их концепциями**.

Каждое представление архитектуры должно соответствовать соглашениям с точки зрения архитектуры управления.

Известные проблемы могут включать в себя: несоответствия, подлежащие заполнению пункты, открытые или нерешенные проблемы, исключения и отклонения от конвенций, установленных точкой зрения. Открытые вопросы могут привести к принятию решений. Исключения и отклонения могут быть задокументированы как результаты решения и обоснование.

5 Согласованность и соответствия

В этой главе описываются требования к согласованности, регистрация известных несоответствий в **AD**, а также использование и документирование соответствий и правил переписки.

5.1 Известные несоответствия

*** Запишите любые известные несоответствия в AD.**

Несмотря на то, что согласованные АД, очевидно, должны быть предпочтительными, иногда невозможно или непрактично разрешать все несоответствия по причинам времени, усилий или недостаточной информации.

Δ Описание архитектуры должно включать анализ согласованности его архитектурных моделей и их представлений.

5.2 Соответствие в AD

* Определите каждое соответствие в **AD** и ее участвующих элементах **AD**.

*Определите правила соответствия, регулирующие построения **AD**.

Корреспонденты используются для выражения, записи, обеспечения соблюдения и анализа согласованности между моделями, представлениями и другими элементами **AD** в описании архитектуры, между **AD** или между **AD** и другими формами документации.

Элементы **AD** включают экземпляры заинтересованных сторон, проблемы, точки зрения и взгляды, модели и модели, решения и обоснования. Конструкции, вводимые по точкам зрения и типам моделей, также являются элементами **AD**.

Соответствия - это n-арные математические отношения. Соответствия могут быть представлены через таблицы, через ссылки или через другие формы ассоциации (например, в UML).

5.3. Правила соответствия

*** Определите каждое правило соответствия, применяемое к AD.**

Правила соответствия могут быть введены **AD**, одной из его точек зрения, или из используемого языка архитектуры или архитектуры.

*** Для каждого идентифицированного правила соответствия записывайте, соблюдает ли правило (выполняется) или иным образом регистрирует все известные нарушения**.

6 Архитектурные решения и обоснование

Стандарт не требует принятия архитектурных решений. В этом разделе описываются рекомендации («shoulds», помечено Δ) для их записи.

Δ Предоставить доказательства рассмотрения альтернатив и обоснование сделанных выборов.

Δ Решения архитектуры записи, которые считаются ключевыми для архитектуры системы.

В число областей, которые следует учитывать при выборе ключевых решений, относятся следующие:

- затрагивающие ключевые заинтересованные стороны или многие заинтересованные стороны
- существенное значение для планирования и управления проектами
- дорогостоящий контроль или внедрение
- высокая чувствительность к изменениям или дорогостоящие изменения

- с участием сложных или неочевидных рассуждений
- относящихся к архитектурно значимым требованиям
- требующие значительных затрат времени или усилий на
- в результате капитальных затрат или косвенных затрат

Δ При принятии решений о регистрации следует учитывать следующие информационные элементы:

- уникальный идентификатор решения
- утверждение решения
- соответствия или связанные проблемы, к которым он относится
- владелец решения
- соответствия или связи с затронутыми элементами **AD**
- обоснование, связанное с решением
- силы и ограничения на решение
- допущения, влияющие на решение
- рассмотренные альтернативы и их потенциальные последствия

Δ См. [106] и ссылки там для различных подходов к документированию решений, совместимых со Стандартом.

Обозначения и сокращения

АО – Архитектурное Описание
АОАП – Аспектно-Ориентированный Анализ и Проектирование
АС – Автоматизированная Система
БД – База Данных
ИИ – Искусственный Интеллект

КСА – Комплекс Средств Автоматизации
ОБ – Объект
ООАП – Объектно-Ориентированный Анализ и Проектирование
ООП – Объектно-Ориентированный Подход
ПО – Программное Обеспечение
СТ – Система Требований

AD – Architecture Description
ADD – Attribute-Driven Design
AF – Architecture Framework
ARID – Active Reviews for Intermediate Designs
ATAM – Architecture Tradeoff Analysis Method
CAD – Computer Aided Design
CAE – Computer Aided Engineering
CAM – Computer Aided Machinery
CBA – Component Based Architecture
CBAM – Cost-Benefit Analysis Method
CBSD – Component Based Software Development
DoDAF – Department of Defense Architecture Framework
ECCAI – European Coordinating Committee for Artificial Intelligence
IEEE – Institute of Electrical and Electronic Engineers
IDL – Interface Definition Language
FEAF – Federal Enterprise Architecture Framework
MDA – Model Driven Architecture
MoDAF – Minister of Defense Architecture Framework
MVC – Model-View-Controller
PIM - Platform-Independent Models
QAW – Quality Attribute Workshop
RUP – Rational Unified Process
RM-ODP – Reference of Open Distributed Processing
SBA – Service Based Architecture
SEI – Software Engineering Institute
SAAM – Software Architecture Analysis Method
TOGAF – The Open Group of Architecture Framework
UML – Unified Modeling Language
WSDL – Web Service Definition Language

Методическое пособие по дисциплине «Б1.В.05 Архитектурное моделирование в проектировании АС» доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/130183

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ

Декан факультета информационных систем и технологий

_____ Святов К.В.

«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Дисциплина (модуль) Экспериментальные исследования в проектировании интеллектуальных систем

наименование дисциплины (модуля)

Уровень образования магистратура

(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр

Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Негода В. Н. Экспериментальное исследование в проектировании интеллектуальных систем. Электронные учебно-методические материалы по выполнению расчетно-графической работы. – Ульяновск: УлГТУ. 2022.

Введение

Методические материалы предназначены для студентов магистратуры кафедры ВТ УлГТУ, обучающихся по образовательной программе «Искусственный интеллект в автоматизации проектирования» направления «Информатика и вычислительная техника». Излагается методика решения задач по организации экспериментальных исследований программных и программно-аппаратных средств автоматизированных систем, в том числе компонентов, реализующих методы искусственного интеллекта. Приводятся исходные тексты и описания таких программных прототипов поддержки экспериментальных исследований, которые могут быть использованы в расчетно-графической работе (РГР).

РГР по дисциплине «Экспериментальное исследование в проектировании интеллектуальных систем» нацелена на следующее:

- приобретение умений и навыков решения задач экспериментальных исследований при создании средств автоматизации проектирования автоматизированных систем, в том числе с использованием методов искусственного интеллекта;
- формирование материалов для разделов магистерской выпускной работы, в том числе:
  - раздела, в котором через эксперименты с прототипами оценивается актуальность магистерских исследований;
  - раздела, в котором разрабатываются аналитические модели для создания средств автоматизации экспериментальных исследований;
  - раздела, в котором через эксперименты с программными компонентами выбираются инструменты разработки и включаемые в проект сторонние средства автоматизации;
  - раздела, в котором через эксперименты с прототипами выбираются проектные решения;
  - раздела, в котором через эксперименты строятся функциональные зависимости для расчетов в проектировании, т.е. эмпирические формулы и модели;
  - раздела, в котором через эксперименты оцениваются позитивные эффекты от созданных в работе средств автоматизации и доказывается факт достижения целей работы.

.

Структуро-образующие основы РГР

Наименование РГР

Наиболее общее наименование работы, в которой создаются материалы для магистерской диссертации, целесообразно начинать с названия создаваемых средств и завершать словосочетанием «экспериментальное исследование». Например:

- «Сегментация изображений при распознавании дорожных знаков: экспериментальное исследование»;
- «Автоматизация управления учебно-исследовательскими проектами: экспериментальное исследование»;
- «Программное моделирование машин Голдберга: экспериментальное исследование».

В случае частных исследований наименование должно представлять содержание экспериментов. Например:

- «Оценка скорости сходимости алгоритма обучения в задаче выделения контуров изображений»;
- «Оценка параметров модуля прогнозирования стоимости ценной бумаги»;
- «Оценка точности фасетной классификации проектных спецификаций в технической документации автоматизированных систем»;
- «Анализ сходимости генетического алгоритма генерации программы управления мобильным роботом».
- «Оценка погрешностей алгоритма фильтрации нечетких временных рядов»

Цели работы

Цели работы зависят от содержания магистерских исследований и прежде всего. Наиболее важными целями являются:

- оценка эффективности создаваемых в рамках диссертации средств автоматизации в части повышения производительности труда пользователей; в этом случае обычно строятся модели двух процессов, одни из которых не использует созданные средства, а другой – использует; для каждой модели проводятся серии экспериментов и оценивается ареал позитивного эффекта и характер изменения его величины в зависимости от значимых факторов;
- экспериментирование с целью построения зависимостей критериальных параметров, характеризующих внутренние свойства создаваемых средств автоматизации, от значимых факторов; критериальными параметрами

могут быть время реакции, пропускная/нагрузочная способность, релевантность ответов поисковым запросам, точность, затраты памяти, устойчивость, сходимость, и т.п.

- экспериментирование с целью выбора проектных решений из нескольких альтернатив;
- экспериментирование с целью нахождения оптимальных конфигурационных параметров создаваемых средств автоматизации.

Примерный порядок работы и содержание отчета

Порядок работы приводится ниже для случая, когда преследуются две группы целей экспериментирования:

- оценка зависимости степени повышения производительности труда пользователей создаваемых средств автоматизации от значимых факторов, в том числе выявления ареала положительного эффекта;
- оценка зависимости внутренних свойств создаваемых средств автоматизации от значимых факторов либо с самоцелью анализа этих зависимостей, либо с целями выбора проектных решений, либо с целями нахождения оптимальных конфигурационных параметров.

При этом предполагается по 3 эксперимента для каждой группы целей. Каждый эксперимент решает одну задачу экспериментирования. Число задач экспериментирования может быть не равным 3 – перечень задач согласовывается с преподавателем.

Прототип оглавления представлен ниже.

1. Формулировка задач экспериментирования

1.1.1. Общее описание создаваемых средств автоматизации

1.1.2. Организация экспериментов по оценке степени повышения производительности труда благодаря создаваемым средствам автоматизации

1.1.3. Формулировка целей экспериментирования

1.1.4. Анализ критериальных параметров, характеризующих производительность труда и степень ее увеличения за счет применения создаваемых средств автоматизации

1.1.5. Анализ факторов, влияющих на значения критериальных параметров
число факторов, вовлекаемых в анализ должно быть больше числа факторов, зависимости от которых должен выявить эксперимент; нужно обосновать выбор факторов

1.1.6. Формулировка задач экспериментирования
каждая задача должна быть нацелена на выявление зависимости значений конкретного критериального параметра от значимых факторов

1.2. Организация экспериментов по оценке свойств создаваемых средств автоматизации, либо выбору проектных решений, либо нахождению оптимальных конфигурационных параметров.
1.2.1. Формулировка целей экспериментирования
1.2.2. Анализ критериальных параметров, характеризующих свойства создаваемых средств автоматизации
1.2.3. Анализ факторов, влияющих на значения критериальных параметров
1.2.4. Формулировка задач экспериментирования
2. Планирование и организация экспериментов по оценке степени повышения производительности труда благодаря созданию средств автоматизации
2.1. Разработка моделей двух сопоставляемых процессов
2.1.1. Модель исходного автоматизируемого процесса
2.1.2. Модель результирующего процесса, имеющего место после автоматизации
2.2. Разработка планов экспериментов
2.2.1. Эксперимент 1 «*Название эксперимента*»
a. Выявление границ значений факторов и их обоснование
b. Формирование множеств значений факторов
c. Формулировка требований к механизмам наблюдения за результатами экспериментов
2.2.2. Эксперимент 2 «*Название эксперимента*»
a. Выявление границ значение факторов и их обоснование
b. Формирование множеств значений факторов
c. Формулировка требований к механизмам наблюдения за результатами экспериментов
2.2.3. Эксперимент 3 «*Название эксперимента*»
a. Выявление границ значение факторов и их обоснование
b. Формирование множеств значений факторов
c. Формулировка требований к механизмам наблюдения за результатами экспериментов
2.3. Разработка средств автоматизации проведения экспериментов
2.3.1. Анализ информационных процессов, имеющих место при проведении экспериментов.
2.3.2. Разработка структурно-функциональной организации средств автоматизации проведения экспериментов
2.3.3. Разработка структур данных и алгоритмов автоматизации проведения экспериментов
2.3.4. Реализация средств автоматизации проведения экспериментов
2.4. Проведение экспериментов и формирование отчета
2.4.1. Эксперимент 1
2.4.2. Эксперимент 2
2.4.3. Эксперимент 3

2.4.4. Анализ результатов экспериментирования
3. Планирование и организация экспериментов по оценке внутренних свойств создаваемых средств автоматизации вне сравнения с иными существующими средствами
3.1. Разработка модели процесса,
3.2. Разработка планов экспериментов
3.2.1. Эксперимент 1 «*Название эксперимента*»
d. Выявление границ значений факторов и их обоснование
e. Формирование множеств значений факторов
f. Формулировка требований к механизмам наблюдения за результатами экспериментов
3.2.2. Эксперимент 2 «*Название эксперимента*»
d. Выявление границ значений факторов и их обоснование
e. Формирование множеств значений факторов
f. Формулировка требований к механизмам наблюдения за результатами экспериментов
3.2.3. Эксперимент 3 «*Название эксперимента*»
d. Выявление границ значений факторов и их обоснование
e. Формирование множеств значений факторов
f. Формулировка требований к механизмам наблюдения за результатами экспериментов
3.3. Разработка средств автоматизации проведения экспериментов
3.3.1. Анализ информационных процессов, имеющих место при проведении экспериментов.
3.3.2. Разработка структурно-функциональной организации средств автоматизации проведения экспериментов
3.3.3. Разработка структур данных и алгоритмов автоматизации проведения экспериментов
3.3.4. Реализация средств автоматизации проведения экспериментов
3.4. Проведение экспериментов и формирование отчета
3.4.1. Эксперимент 1
3.4.2. Эксперимент 2
3.4.3. Эксперимент 3
3.4.4. Анализ результатов экспериментирования

Прототипы базовых функций поддержки экспериментальных исследований

Приводимые ниже прототипы охватывают, с одной стороны, наиболее широко используемую функциональность, а, с другой стороны, такую функциональность, которая слабо представлена в доступной научно-технической литературе. Все прототипы нацелены на быстрое вхождение студента в процесс исследований, вовлечение преподавательского исходного кода и программно-технических решений в разработки студента, порождение

расширений спектра факторов на основе наблюдения за результатами измерений. Возможные варианты расширений для описываемых ниже прототипов приводятся для выбора возможной темы научно-исследовательской работы студента в рамках экспериментальной части своей магистерской диссертации.

Базовый класс протоколирования, организации серий измерений и обработки результатов

Класс реализован на языке C++ и обеспечивает исполнение базовых функций накопления результатов экспериментов в векторе данных типа double, фильтрации результатов путем отбрасывания заданного числа наименьших и наибольших значений, вычисления среднего, минимального, максимального, среднеквадратичного отклонения в натуральных значениях и в процентах от среднего. Исходный текст является самодокументированным, поэтому не требует отдельного описания. Все методы возвращают адрес объекта описываемого класса, что позволяет композировать методы в фазе вызова на основе технологии «цепочка вызовов».

```
// Log.h – поддержка сохранения протокольных записей в векторе данных
// типа double, фильтрации путем отбрасывания заданного числа наименьших
// и наибольших значений, вычисление параметров описательной статистики
// организации серии измерений и распечатки результатов
#pragma once
#include <iostream>
#include <vector>
#include <sstream>
#include <numeric>
#include <cmath>
#include <intrin.h>
#include <algorithm>
#include <thread>
#include <omp.h>
#include <iomanip>
#include <map>

using namespace std;

// Код упорядочивания: естественный, вначале минимальные, вначале максимальные
enum { O_NATURAL, O_MIN, O_MAX};

/* Параметры конфигурирования формата распечатки и фильтрации
   PREC_VAL - число знаков после запятой в распечатки сохраняемых в протоколе значений
   PREC_AVG - число знаков после запятой в распечатке
   FILTR_MIN, FILTR_MAX - число отбрасываемых наименьших и наибольших перед подсчетом среднего и СКО
*/
enum Config {CONFIG_FIRST = 0, PREC_VAL = 0, PREC_AVG,
```

```
            FILTR_MIN, FILTR_MAX, CONFIG_SIZE} ;

// Получение информации о процессоре через команду функцию
// __cpuid(unsigned *) info, funcCode),
// которая выполняет машинную команду CPUID, предварительно загружая в EAX
// код функции funcCode.
// CPUID возвращает 4 четырехбайтных кода: 0:EAX, 1:EBX, 2:ECX, 3:EDX,
// а функция __cpuid перемещает эти слова по адресу info
void cpuInfo() {
   // int regs[4]; // 0:EAX, 1:EBX, 2:ECX, 3:EDX
   char nameCPU[80]; // имя процессора
   // Получение имени процессора
   // Коды 0x80000002..0x80000004 позволяют получить полное имя CPU по 16 байтов
   __cpuid((int *)nameCPU, 0x80000002);
   __cpuid((int *)nameCPU + 4, 0x80000003);
   __cpuid((int *)nameCPU + 8, 0x80000004);
   int cores = thread::hardware_concurrency();

   cout << endl << nameCPU << "\tПотоков: " << cores << endl << endl;
}
// Класс протоколирования, обработки, вывода, организации серий измерений
class Log {
public:
   double dmin = 1.0E10, dmax = 0.; // минимальный и максимальный элемент
   double avg = 0., sqdev = 0., sqdev_perc = 0; // среднее, СКО, СКО%
   vector <double> arr; // протокол результатов измерения
   int conf[CONFIG_SIZE] = {0, 0, 0, 0};

   // Динамически формируемый текст для вставки в консольный вывод
   ostringstream msgfiltr;

   Log() {}

   // Обработка результатов протоколирования
   Log & calc() {
      vector<double> vect = arr;
      msgfiltr.str("");
      if (conf[FILTR_MIN] > 0) { // удаление наименьших
         sort(vect.begin(), vect.end());
         vect.erase(vect.begin(), vect.begin() + conf[FILTR_MIN]);
         msgfiltr << "\nУдалено " << conf[FILTR_MIN] << " наименьших";
      }
      if (conf[FILTR_MAX] > 0) { // удаление наибольших
         sort(vect.begin(), vect.end(), [](double a, double b) {
            return a > b; });
         vect.erase(vect.begin(), vect.begin() + conf[FILTR_MAX]);
         msgfiltr << "  Удалено " << conf[FILTR_MAX] << " наибольших";
      }
      // подсчет среднего
      avg = accumulate(vect.begin(), vect.end(), 0.0,
              [&](double x, double y)
```

```
            {return x + y / vect.size(); });
   // подсчет СКО
   sqdev = sqrt(accumulate(vect.begin(), vect.end(), 0.,
               [&](double sq, double v)
                  {double q = avg - v; return sq + q * q; })
          / vect.size());
   // подсчет СКО%
   sqdev_perc = 100 * sqdev / avg;
   // нахождение минимального и максимального
   dmin = *min_element(vect.begin(), vect.end());
   dmax = *max_element(vect.begin(), vect.end());
   return *this;
} // calc

// Конфигурирование путем передачи пар формата
// {индекс_параметра, значение_параметра}, где индексы определены
// в спецификации enum Config
Log& config(map< int, int > param) {
   for (auto it = param.begin(); it != param.end(); it++) {
      int n = (*it).first;
      if (n >= CONFIG_FIRST && n < CONFIG_SIZE)
         conf[n] = (*it).second;
   }
   return *this;
} // config

// Серия измерений, каждое из которых выполняется функцией meter,
// возвращающей значение измеренного параметра.
// При clear == true протокол сначала очищается, иначе
// результаты серии измерений дописываются в уже существующий протокол
Log& series(bool clear, int count, double (meter)()) {
   if (clear) arr.clear();
   for (int n = 0; n < count; n++) {
      arr.push_back(meter());
   }
   return *this;
} // series

// Добавление группы результатов измерения, например из файла протокола,
// полученного другим приложением
Log& set(vector <double> source) {
   for (double v : source)
      arr.push_back(v);
   return *this;
} // set

// Вывод в консоль описательной статистики результатов измерений
Log & stat(double scale, string unit) {
   if(dmin > dmax && arr.size() > 0) {
      cout << "\nПопытка вывести статистику без выполнения calc()" << endl;
   }
```

```
      cout.setf(ios::fixed);
      cout << "Статистика " << arr.size() << " измерений(" << unit << ")"
           << endl << msgfiltr.str() << endl
           << setprecision(conf[PREC_VAL]) << "Минимум: " << scale * dmin
           << "  Максимум: " << scale * dmax
           << setprecision(conf[PREC_AVG]) << "  Среднее: " << scale * avg
           << "  СКО: " << scale * sqdev << "  СКО%: " << sqdev_perc << endl;
      return *this;
   } // stat

   // Вывод с масштабом scale первых len в одном из трех порядков:
   // O_NORM - естественный, O_MIN - минимальные, O_MAX - максимальные
   Log & print(int ord, double scale, unsigned len) {
      // Работа ведется над копией, которая упорядочивается согласно ord
      vector <double> vect = arr;
      string head, suffix = "";
      if (conf[FILTR_MIN] + conf[FILTR_MAX] > 0)
         suffix = " до фильтрации";
      int nsetw = int(ceil(log10(dmax*scale))) + conf[PREC_VAL] + 3;
      switch (ord) {
         case O_NATURAL:
            head = "Первые значения" + suffix + ": ";
            break;
         case O_MIN:
            sort(vect.begin(), vect.end());
            head = "Наименьшие" + suffix + ": ";
            break;
         case O_MAX:
            sort(vect.begin(), vect.end(),
                 [](double a, double b) { return a > b; });
            head = "Наибольшие" + suffix + ": ";
            break;
      }
      cout.setf(ios::fixed);
      cout << head << setprecision(conf[PREC_VAL]) << endl;
      for (unsigned n = 0; n < min(len, unsigned(vect.size())); n++)
         cout << setw(nsetw) << scale * vect[n];
      cout << endl;
      return *this;
   } // print
}; // class Log
```

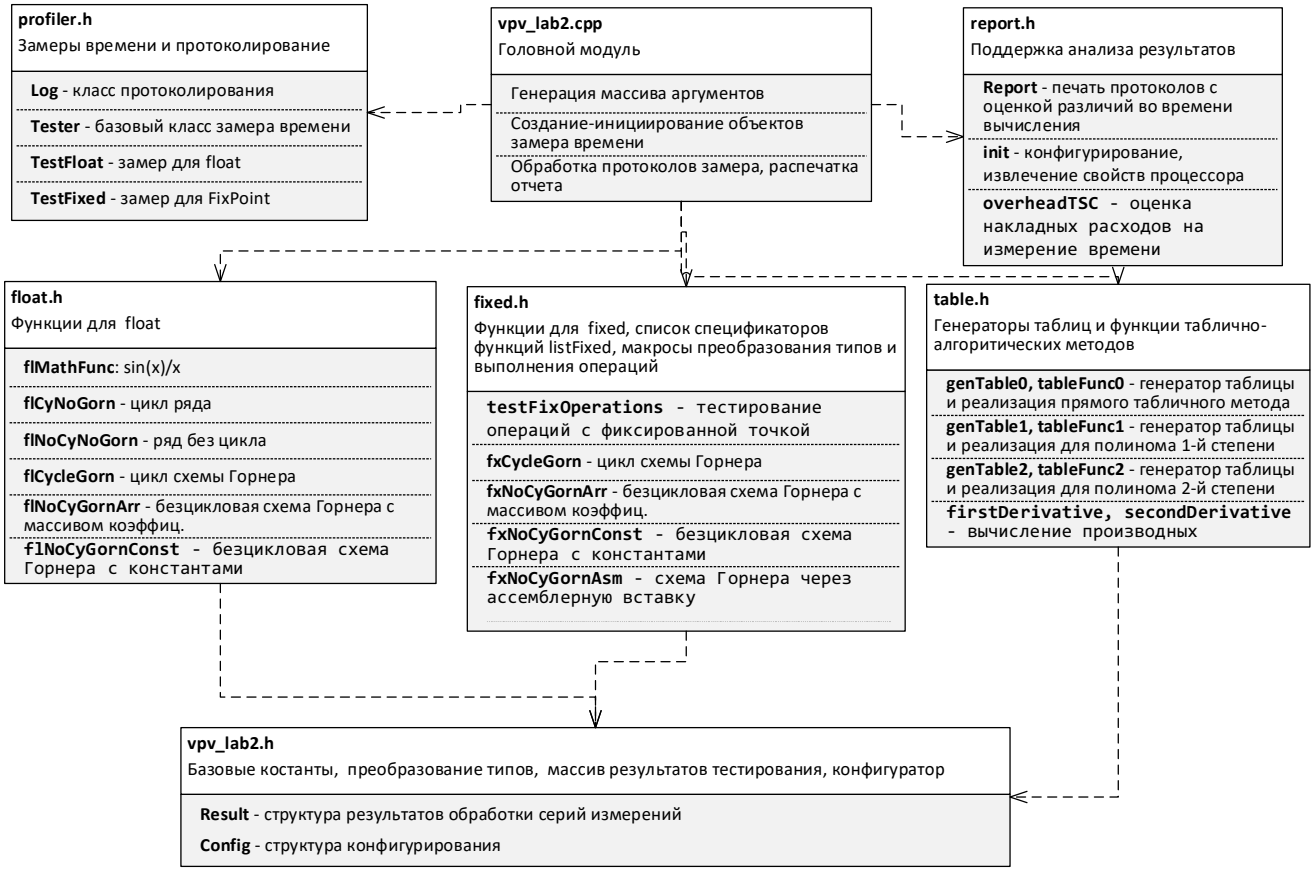
Ниже приводятся пример использования указанного класса.

```
// Организация серии испытаний, в которой задействована
// функция измерения затрат времени clockIntervalUsingQPC
cout << "\n\nОценка повторяемости 1000 измерений clock-интервала "
      << "через счетчик QPC без фильтрации\n";
scale = MCS_IN_SEC; // Значения будем выводить в микросекундах
int nPasses = 5; // 5 проходов для оценки повторяемости
// Конфигурирование
log.config({
```

```
    {PREC_AVG, 2}, // Для среднего и СКО задаем точность 2 знака
    {FILTR_MIN,0},{FILTR_MAX, 0} }); // нет фильтрации
for (int n = 1; n <= nPasses; n++) {
    cout << "\nСерия " << n << endl << endl;
    log.series(true, 1000, clockIntervalUsingQPC)
       .calc().stat(scale,
           "Число mcs в clock-интервале через QPC (без фильтрации)")
       .print(O_NATURAL, scale, 10)
       .print(O_MIN, scale, 10).print(O_MAX, scale, 10);;
} // for
```

Прототип сравнительного исследования быстродействия нескольких реализаций элементарных функций

В данном эксперименте используется один фактор – вариант реализации элементарной функции. Диаграмма классов прототипа представлена ниже.



Значения фактора разбиты на три группы:

- Реализации рядом Тейлора с использованием вещественных чисел типа float - описаны в модуле float.h;
- Реализации рядом Тейлора с использованием чисел с фиксированной точкой - описаны в модуле fixed.h;
- Реализации на основе табличного и таблично-алгоритмического метода – описаны в модуле table.h.

Серию измерений выполняют класс протоколирования Log и классы замера времени Tester, TesterFloat, TesterFixed, описанные в файле profiler.h. Поддержку анализа результатов эксперимента обеспечивает класс Report, описанный в файле report.h.

Использование всех перечисленных методов достаточно наглядно иллюстрирует исходный код функции main:

```
int main(int argc, char * argv[]) {
  init(argc, argv, config); // Инициализация конфигурационных данных
  // Создание массива объектов тестирования
  vector <Tester * > arr = {
    // Создание объектов, реализующих ряд Тейлора с плавающей точкой
    new TestFloat("flMathFunc","Float - библиотечная реализация sin(x)/x",
                  flMathFunc, config),
    new TestFloat("flCyNoGorn","Float - цикл формулы ряда", flCyNoGorn,
config),
    new TestFloat("flNoCyNoGorn", "Float - безцикловая формула ряда",
                  flNoCyNoGorn, config),
    new TestFloat("flCycleGorn", "Float - цикл схемы Горнера", flCycleGorn,
                  config),
    new TestFloat("flNoCyGornArr",
                  "Float - безцикловая схема Горнера(массив коэффициентов)",
                  flNoCyGornArr, config),
    new TestFloat("flNoCyGornConst",
                  "Float - безцикловая схема Горнера(константы)",
                  flNoCyGornConst, config)
    // Создание объектов, реализующих ряд Тейлора с фиксированной точкой
    new TestFixed("fxCycleGorn", "Fixed - цикл схемы Горнера",
                  fxCycleGorn,   config),
    new TestFixed("fxNoCyGornArr",
                  "Fixed - безцикловая схема Горнера(массив коэффициентов)",
                  fxNoCyGornArr, config),
    new TestFixed("fxNoCyGornConst",
                  "Fixed - безцикловая реализация схемы Горнера (константы)",
                  fxNoCyGornConst, config),
    new TestFixed("fxNoCyGornAsm",
                  "Fixed - безцикловая реализация схемы Горнера (asm-вставка)",
                  fxNoCyGornAsm, config)
    // Создание объектов, реализующих табличные методы
    new TestFloat("tableFunc0", "Прямой табличный метод", tableFunc0, config),
    new TestFloat("tableFunc1",
                  "Таблично-алгоритмический метод - полином 1 степени",
                  tableFunc1, config),
    new TestFloat("tableFunc2",
                  "Таблично-алгоритмический метод - полином 2 степени",
                  tableFunc2, config)
  };

  // Функциональное тестирование всех реализаций, чтобы обеспечить
  // замеры времени корректных процедур
  cout << endl << "Верификация..." << endl;
  // Сначала проверка правильности макросов преобразования между Fixed и float
  if (!testFixOperations(config))
    return 1;
  // Затем верификация функций, подвергаемых профилированию
  for (Tester * test : arr)
     test->verify();
```

```
  // Выполнение замеров времени
  cout << endl << "Замеры времени ..." << endl;
  // Выполнение config.pass проходов измерений
  for(int n = 0; n < config.pass; n++) {
     cout << endl << "Проход " << (n + 1) << endl;
     if (config.lenPrintLog > 0)
       cout << "Первые " << config.lenPrintLog
            << " результатов замеров времени:" << endl;
     // Циклическое выполнение «измерение-обработка-выгрузка в отчет»
     for (Tester * test : arr)
       test->timeSpent();
     report.calc();
     cout <<endl << "Итоги:" << endl;
     report.print();
     report.log.clear();
  }
  return 0;
} // main
```

Отчет, сформированный в первом проходе, выводится в консоль и в случае реализации функции sin(x)/x с погрешностью не больше 2^{-19} имеет следующий вид:

```
  ------------------------------------------------------------------------
  Функция             Рейтинг   Среднее       СКО      СКО%   Минимум  Максимум
  ------------------------------------------------------------------------
  flMathFunc              2.2    132.16      7.61      5.76       123       990
  flCyNoGorn            19.73   1180.95     51.10      4.33      1137      5181
  flNoCyNoGorn          11.03    659.93     14.52      2.20       636      1488
  flCycleGorn            1.62     97.04      4.04      4.17        81       192
  flNoCyGornArr          1.10     65.84      7.69     11.68        54       123
  flNoCyGornConst        1.10     66.02      7.61     11.53        57       126
  fxCycleGorn            2.50    149.68      7.15      4.78       135       432
  fxNoCyGornArr          2.13    127.27      7.61      5.98       114       513
  fxNoCyGornConst        2.08    124.76      7.75      6.21       114       438
  fxNoCyGornAsm          1.00     59.85      7.71     12.87        51       354
  tableFunc0             5.00    299.20     80.15     26.79        33      1497
  tableFunc1             1.42     84.69     61.18     72.24        54       387
  tableFunc2             1.23     73.71     24.84     33.70        60       333
  ------------------------------------------------------------------------
```

Параметр «Рейтинг» характеризует отношение средних затрат разных реализаций функций к среднему времени самого быстрого метода. Самый быстрый имеет рейтинг, равный 1. Это обеспечивает сочетание достаточно наглядного представления и с максимальной достоверностью без использования графики

Как мы видим, реализация на основе стандартной библиотеки в 2.2 раза медленнее. Традиционная «наивная» цикловая реализация оказалась в 19.73 раза медленнее.

Разворачивание цикла «наивной» реализации ускорило в 1.79 раза (19.73 / 11.03), а разворачивание цикла схемы Горнера для float ускорило в 1.49 раза и

для FixPoint – примерно на 20% в случае C-процедуры и в 2.5 раза в случае написания бесцикловой реализации на ассемблере.

Использование ассемблера всего на 10% ускоряет бесцикловую реализацию на основе схемы Горнера.

Наибольшее удивление вызывают результаты табличных методов, особенно метод tableFunc0, при реализации которого значение функции просто извлекается из таблицы без каких-либо дополнительных операций обработки извлеченного значения. Естественно объяснить этот факт влиянием кэш-промахов. Для проверки этой гипотезы разместим результаты первого и третьего проходов рядом:

Функция	Рейтинг	Среднее	СКО	СКО%	Минимум	Максимум
Из итогов первого прохода						
fxNoCyGornAsm	1.00	59.85	7.71	12.87	51	354
tableFunc0	5.00	299.20	80.15	26.79	33	1497
tableFunc1	1.42	84.69	61.18	72.24	54	387
tableFunc2	1.23	73.71	24.84	33.70	60	333
Из итогов третьего прохода						
fxNoCyGornAsm	1.00	60.90	7.61	12.50	51	120
tableFunc0	1.32	80.64	40.68	50.45	36	294
tableFunc1	1.08	65.77	10.16	15.45	54	276
tableFunc2	1.17	70.97	8.05	11.35	60	135

Прогрев кэша к третьему проходу существенно уменьшает среднее время реализации табличных методов. При этом степень полинома не становится определяющей время вычисления – полином нулевой степени (метод tableFunc0) дает менее быстрое решение, чем полиномы 1-й и 2-й степени (tableFunc1 и tableFunc2 соответственно. Это порождает потребность во включении параметров кэша и его использования в перечень факторов исследования. В то же время, различие результатов разных проходов наблюдается при конкретном случае распределения потока значений аргументов, поступающего на различные методы реализации, для конкретной функции и конкретном ограничении на погрешность. Это порождает еще несколько кандидатов на расширение спектра факторов за счет включения в него параметров распределения, максимально-допустимой погрешности, реализуемой функции.

Свойства прототипа, эксперименты с которым порождают много направлений экспериментальных исследований, являются основой как для активного повторного использования функциональности прототипа, так и для формулировок индивидуальных заданий, имеющих потенциал порождения результатов, уместных для публикации магистерских исследований в научно-технической литературе.

Фрагмент прототипа трехфакторного эксперимента с многопоточными реализациями численного интегрирования

В данном прототипе все реализации строятся на методе средних прямоугольников и задействованы три фактора: вариант программной реализации, число потоков, гранулярность, представляемая числом прямоугольников в интервале интегрирования. Три фактора предопределили размещение результатов в куб результатов, в котором размещено *methods* матриц размером *treads * granularity*, где *methods* – число методов реализации, threads – число потоков, *granularity* – число значений гранулярности из последовательности (100, 1000, 10000, 100000, …). Базовые структуры данных, обеспечивающие внутреннее представление куба, имеют вид:

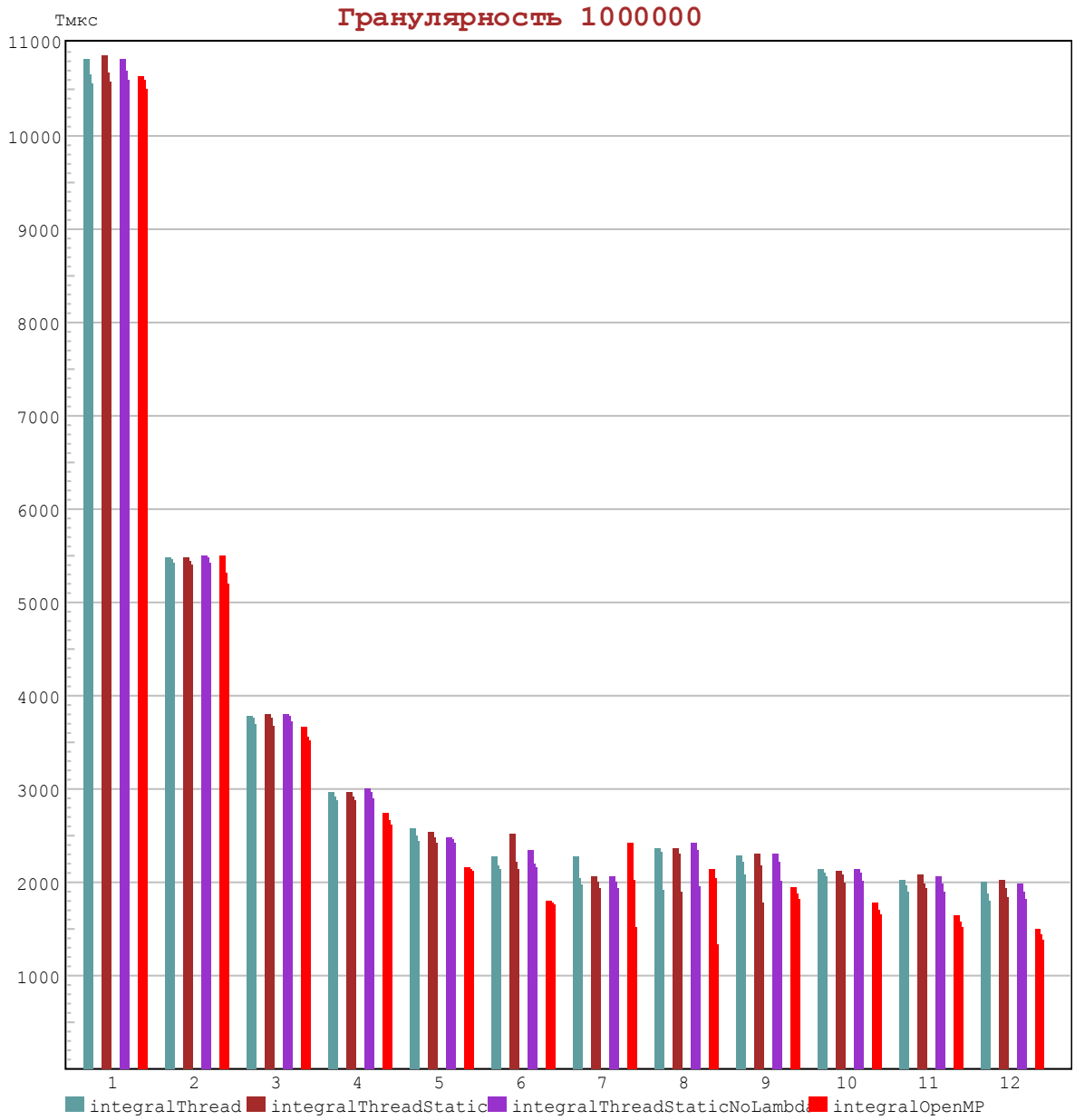
```
struct ResultOfSeries {
  // Число потоков и гранулярность, при которых проводилась серия измерений
  int threads, granularity;
  // минимум и максимум затрат времени (до фильтрации) в секундах
  double min, max;
  double avg, dev;      // среднее и СКО
  double error; // ошибка вычисления
  ResultOfSeries() {}
};

// Результат измерения для функции с параллельностью
struct ResultOfFunction {
  ResultOfFunction() {}
  // Текстовое имя функции - способа распараллеливания
  string name;
  // Матрица thread * granularity результатов обработки серий измерений
  ResultOfSeries matrix[MAX_THREAD][COUNT_GRANULARITY];
};
```

В исследование вовлечены 3 метода распараллеливания, использующие многопоточность на основе библиотеки класса std::thread C++, и один метод на основе OpenMP. Генератор отчетов обеспечивает фильтрацию результатов путем удаления заданного числа наименьших и наибольших значений, а также формирование таблиц всех основных параметров статистической обработки данных по серии измерений заданного размера. В перечень параметров входят: среднее, минимальное и максимальное значение затрат времени, СКО в натуральном и в процентном выражении, абсолютная ошибка численного интегрирования. Кроме того, генерируются svg-гистограммы для различных методов, уровней параллельности и гранулярности. Необходимость встраивания функций генерации гистограмм в средства поддержки экспериментальных исследований вызвана стремлением представить графическую информацию в более компактном виде, нежели это делают традиционные генераторы гистограмм. Это обеспечивается представлением минимального, среднего и максимального значения одной полосой, со ступенчатой верхней частью.

Ниже представлена гистограмма для случая, когда гранулярность соответствует 1000000, число потоков лежит в интервале 1-12, а измерения выполнялись в компьютере с CPU Ryzen 7 3700x. Несмотря на наличие в этом CPU 16 параллельно работающих вычислительных конвейера команд, ожидаемая гиперболическая зависимость времени от числа потоков наблюдается только при увеличении числа потоков только до 6. При этом у реализации через OpenMP близость к гиперболической зависимости времени от числа потоков более заметна, нежели у методов на основе класса std::thread C++.

Код прототипа построен таким образом, что собственно функции обработки данных, специфицированные в модуле integral.h, по своим определениям изолированы от значительно более емкого по объему кода поддержки проведения измерительных экспериментов и обработки его результатов. Тем

самым, прототип позволяет существенно сэкономить время исполнения индивидуального задания, в котором фигурируют те же самые факторы, но решаемые задачи распараллеливаемыми программами совсем иные.



Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.01.01 Технологии обработки и анализа больших массивов данных

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Рекомендовано научно-методической комиссией факультета информационных систем и технологий в качестве практикума.

Вертешев Антон Сергеевич, к.э.н., доцент

Методические указания по выполнению курсового проекта по дисциплине: «Технологии обработки и анализа больших массивов данных» Вертешев А.С.. – Ульяновск : УлГТУ, 2021.

Предназначены студентам, обучающимся по направлению магистратуры: 09.04.01 «Информатика и вычислительная техника»

Предоставлены задания, рекомендации и требования по выполнению курсового проекта, разработанные в соответствии с рабочей программой дисциплины.

СОДЕРЖАНИЕ

1. Задание по курсовому проекту

Учебным планом предусмотрен курсовой проект по данной дисциплине. Наполнение курсового проекта будут составлять, выполненные студентами 8 практических заданий, объединенных одной целью разработать систему хранения и обработки больших данных с помощью изученных технологий.

2. Этапы, которые нужно пройти для написания курсового проекта (представлены в таблице 1)

Этапы выполнения Табл.1

Номер этапа	Задание
1	Знакомство с технологией Hadoop MapReduce Включить в отчет теоретическую информацию о данных технологиях, провести сравнительный анализ данных технологий.
2	Использование реляционных хранилищ данных для big data (PostgreSql) Выполнить практические задания 1-7. Включить в отчет результаты по выполнению заданий.
3,4	Применение NoSql хранилищ данных для big data (Neo4J,CouchDB, Redis, Apache Cassandra) Включить в отчет теоретическую информацию о данных технологиях, провести сравнительный анализ данных технологий
5	Использование облачных вычислений при помощи Apache Spark Выполнить практические задания 8-10. Включить в отчет результаты по выполнению заданий.
6.	Спроектировать собственный вариант системы хранения и обработки больших данных, с помощью выбранных инструментов и технологий.

3. Промежуточная аттестация (представлена в таблице 3)

Промежуточная аттестация будет поделена на несколько частей, каждая часть соответствует, этапам задания и будет оцениваться преподавателем:

Шкала оценивания с учетом текущего контроля работы обучающегося в семестре Табл.2

Части и соответствующие им этапы выполнения	Процент выполнения	Оценка
Этапы 1 Знакомство с технологией Hadoop MapReduce; Этапы (1)	10%	Да/Нет
Этап 2 Использование реляционных хранилищ данных для big data (PostgreSql)	20%	Да/Нет

Этапы 3 Применение NoSql хранилищ данных для big data (Neo4J,CouchDB, Redis, Apache Cassandra)	10%	Да/Нет
Этапы 4 Использование облачных вычислений при помощи Apache Spark	10%	Да/Нет
Этап 5 Система хранения и обработки больших данных, реализованная студентом с помощью выбранных инструментов и технологий	50%	Да/Нет
Итого:	100%	

4. Результатом работы должна быть система хранения и обработки больших данных, разработанная с помощью, выбранных студентом технологий.

По данным результата выставляется зачет с оценкой в соответствии со шкалой оценивания приведенной в таблице 3.

Шкала оценивания с учетом срока сдачи Таблица 3

Критерии оценивания	Оценка
Студент правильно выполнил этапы работы и разработал систему обработки и хранения больших данных, показав в полном объеме знания и умения, допустив минимум ошибок	Отлично
Студент правильно выполнил этапы работы и разработал систему обработки и хранения больших данных, показав знания и умения, допустив некоторые ошибки	Хорошо
Студент правильно выполнил этапы работы, сделав 50% задания, допустив множественные ошибки	Удовлетворительно
Студент не выполнил этапы работы	Неудовлетворительно

Список используемой литературы

1. Макшанов А. В. Большие данные. Big Data : учебник для вузов / А. В. Макшанов, А. Е. Журавлев, Л. Н. Тындыкарь. СанктПетербург : Лань, 2021. 188 с.

2. Методы и модели исследования сложных систем и обработки больших данных : монография / И. Ю. Парамонов, В. А. Смагин, Н. Е. Косых, А. Д. Хомоненко ; под редакцией В. А. Смагина, А. Д. Хомоненко. — СанктПетербург : Лань, 2020. — 236 с. : ил. — (Учебники для вузов. Специальная литература)

3. Макшанов А. В. Современные технологии интеллектуального анализа данных : учебное пособие для СПО / А. В. Макшанов, А. Е. Журавлев, Л. Н. Тындыкарь. — СанктПетербург : Лань, 2020. — 228 с.

4. Анализ данных : учебник для вузов / В. С. Мхитарян [и др.] ; под редакцией В. С. Мхитаряна. — Москва : Издательство Юрайт, 2021. — 490 с. — (Высшее образование). — ISBN 978-5-534-00616-2. — Текст : электронный // Образовательная платформа Юрайт

5. Парфенов, Ю. П. Постреляционные хранилища данных : учебное пособие для вузов / Ю. П. Парфенов ; под научной редакцией Н. В. Папуловской. — Москва : Издательство Юрайт, 2021. — 121 с. — (Высшее образование). — ISBN 978-5-534-09837-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL:

6. Миркин, Б. Г. Введение в анализ данных : учебник и практикум / Б. Г. Миркин. — Москва : Издательство Юрайт, 2020. — 174 с. — (Высшее образование). — ISBN 978-5-9916-5009-0. — Текст : электронный // Образовательная платформа Юрайт [сайт].

7. Крутиков, В.Н. Анализ данных : учебное пособие / В.Н. Крутиков, В.В. Мешечкин ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Кемеровский государственный университет». - Кемерово : Кемеровский государственный университет, 2014. - 138 с. : ил. - Библиогр. в кн. - ISBN 978-5-8353-1770-7 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=278426

8. Жуковский, О.И. Информационные технологии и анализ данных : учебное пособие / О.И. Жуковский ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2014. - 130 с. : схем., ил. - Библиогр.: с. 126. - ISBN 978-5-4332-0158-3 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=480500

9. Волк В.К. Базы данных : учебное пособие. Ч.1. Проектирование и программирование / В.К. Волк ; Министерство науки и высшего образования Российской Федерации, Курганский государственный университет ; [науч. ред. В.А. Симахин]. - Курган : Издательство Курганского государственного университета, 2018.

10. Волк В.К. Базы данных : учебное пособие. Ч.2. Администрирование / В.К. Волк ; Министерство образования и науки Российской Федерации, Курганский государственный университет ; [науч. ред. В.А. Симахин]. - Курган : Издательство Курганского государственного университета, 2018. - 127, [1] с. - Библиогр.: с. 127. - ISBN 978-5-4217-0440-9.

Методические указания по выполнению курсового проекта
по дисциплине:
«Технологии обработки и анализа больших массивов данных»

Автор
Вертешев Антон Сергеевич
УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.01.02 Создание приложений искусственного интеллекта на языке python

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Методическое пособие по дисциплине «Б1.В.ДВ.01.02 Создание приложений искусственного интеллекта на языке python» доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/179915

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):

Б1.В.ДВ.02.01 Теоретические основы САПР

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Методическое пособие по дисциплине «Б1.В.ДВ.02.01 Теоретические основы САПР» доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/168620

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.02.02 Методы управления знаниями и принятием решений

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Бондарева И.О.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Дисциплина (модуль): Методы управления знаниями и принятием решений
наименование дисциплины (модуля)

Уровень образования: магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация: Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Методические рекомендации составлены

на кафедре	Вычислительная техника
факультета	Информационных систем и технологий
в соответствии с учебным планом по направлению подготовки (специальности)	09.04.01 Информатика и вычислительная техника
профиль (программа / специализация)	Искусственный интеллект в автоматизации проектирования

Составитель: к.т.н., доц. Бондарева Ирина Олеговна

Тематика и задания самостоятельной работы

Раздел 1. Базы знаний. Базы опыта. Базы правил. Базы прецедентов

Задание 1.1 – Решение задач.

Требования к выполнению данного задания:

По материалам лекций и списка рекомендованной литературы необходимо построить математическое представление структуры БЗ.

Порядок выполнения задания:

Необходимо заранее изучить материалы лекций и списка рекомендованной литературы.

Для выбранной предметной области (соответствующей теме магистерской ВКР) разработать математическое представление БЗ, состоящей не менее чем из 60 элементов.

Математическое представление должно соответствовать множеству:

(M_1, M_2, M_3, I_1, I_2, I_3), где

M_1 – база глубинных знаний, M_2 – база фактов, M_3 – база метазнаний, I_1 – интерфейсы между M_1 и M_2, I_2 – интерфейсы между M_2 и M_3, I_3 – интерфейсы между M_1 и M_3.

Более подробное описание Вы найдете в материалах лекций и списка рекомендованной литературы.

Форма контроля – решение задач.

Требования к оформлению задания:

Задание должно быть оформлено в форме электронного документа в формате doc. Шрифт 12 пт.

Рекомендуемые источники [9].

Задание 1.2 – Решение задач по теме «Методы поиска решений в базе прецедентов. Метод ближайшего соседа (NearestNeighbor — NN)».

Требования к выполнению данного задания:

По материалам лекций и списка рекомендованной литературы необходимо решить задачи.

Порядок выполнения задания:

Необходимо заранее изучить материалы лекций и списка рекомендованной литературы. Решить предложенные преподавателем задачи.

Форма контроля – решение задач.

Требования к оформлению задания:

Задание должно быть оформлено письменно в тетради либо в виде электронного документа MS Excel или MS Word.

Рекомендуемые источники [1-4, 6].

Задание 1.3 – Решение задач по теме «Методы поиска решений в базе прецедентов. Метод ближайшего соседа (NearestNeighbor — NN)».

Требования к выполнению данного задания:

По материалам лекций и списка рекомендованной литературы необходимо решить задачи.

Порядок выполнения задания:

Необходимо заранее изучить материалы лекций и списка рекомендованной литературы. Решить предложенные преподавателем задачи.

Форма контроля – решение задач.

Требования к оформлению задания:

Задание должно быть оформлено письменно в тетради либо в виде электронного документа MS Excel или MS Word.

Рекомендуемые источники [1-4, 6].

Задание 1.4. – Подготовка к отчету по практической работе №1. Когнитивное представление базы знаний с использованием инструментального средства онтологического проектирования OntoStudio. Подготовка ответов на контрольные вопросы.

Требования к выполнению данного задания:

Подготовить ответы на контрольные вопросы к практической работе.

Контрольные вопросы к практической работе №1:

1. Охарактеризуйте различные интерпретации понятия «онтология».
2. Как представляется модель онтологии?
3. Что такое модель расширенной онтологии? Охарактеризуйте ее компоненты.
4. Какие этапы построения онтологии предусмотрены стандартом IDEF5?
5. Каково назначение онтологии верхнего уровня? Приведете примеры таких онтологий.
6. Каково назначений онтологии предметного уровня? Приведете примеры таких онтологий.
7. Перечислите основные возможности редактора онтологий OntoStudio.

Порядок выполнения задания:

Необходимо заранее изучить методические рекомендации по выполнению практической работы, а также материалы лекций. Обратить внимание на цель занятия, на основные вопросы для подготовки к занятию, на контрольные вопросы самопроверки после практической работы, на содержание темы занятия.

Форма контроля– отчет по практическим работам.

Требования к оформлению задания:

Устно составить конспект по практической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы.

Рекомендуемые источники [1-11].

Задание 1.5. – Выполнение индивидуального задания к практической работе №1.

Требования к выполнению данного задания:

Представить выполненное индивидуальное задание к практической работе №1. Когнитивное представление базы знаний с использованием инструментального средства онтологического проектирования OntoStudio.

Порядок выполнения задания:

1. Для выбранной предметной области (соответствующей теме магистерской ВКР) выделить не менее 30 понятий (концептов).
2. Дать определения этим понятиям (то есть составить тезаурус).
3. На множестве понятий ввести отношения и функции интерпретации для построения онтологии по предметной области. Построить онтологию, используя инструментальное средство онтологического проектирования OntoStudio.
4. Осуществить поиск информации по разработанной предметной онтологии.

Форма контроля – Отчет о выполнении индивидуального задания к практической работе №1.

Требования к оформлению задания:

Устно составить конспект выполнения индивидуального задания кпрактической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы, выводы по результатам.

Рекомендуемые источники [1-11].

Раздел 2. Интеллектуальный поиск знаний

Задание 2.1. – Подготовка к отчету по практической работе №2.Инжиниринг знаний в системе PROTÉGÉ. Подготовка ответов на контрольные вопросы.

Требования к выполнению данного задания:

Подготовить ответы на контрольные вопросы к практической работе.

Контрольные вопросы к практической работе №2:

1. Как представить онтологию в Protégé?
2. Объясните назначение редактора классов для представления знаний.
3. В какие форматы можно передать онтологию, созданную в Protégé?
4. Какие основные задачи можно решать с помощью Protégé?
5. Назовите и охарактеризуйте основные функциональные возможности Protégé.
6. Какие действия необходимо последовательно выполнить припостроении онтологии в системе Protégé?
7. Что такое слоты, какие атрибуты слотов имеются в Protégé?
8. Какие типы классов реализованы в Protégé?
9. Есть ли возможность изменения ранее введенных исходных данных, добавления в онтологию значений новых, ранее не известных параметров?

Порядок выполнения задания:

Необходимо заранее изучить методические рекомендации по выполнению практической работы, а также материалы лекций. Обратить внимание на цель занятия, на основные вопросы для подготовки к занятию, на контрольные вопросы самопроверки после практической работы, на содержание темы занятия.

Форма контроля – отчет по практическим работам.

Требования к оформлению задания:

Устно составить конспект по практической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы.

Рекомендуемые источники [1-11].

Задание 2.2. – Выполнение индивидуального задания к практической работе №2.

Требования к выполнению данного задания:

Представить выполненное индивидуальное задание к практической работе №2. Инжиниринг знаний в системе PROTÉGÉ

Порядок выполнения задания:

1. Для выбранной предметной области (соответствующей теме магистерской ВКР) выделить не менее 30 понятий (концептов).

2. Дать определения этим понятиям (то есть составить тезаурус).

3. На множестве понятий ввести отношения и функции интерпретации для построения онтологии по предметной области. Построить онтологию, используя инструментальное средство онтологического проектирования PROTÉGÉ.

4. Осуществить поиск информации по разработанной предметной онтологии.

Форма контроля – Отчет о выполнении индивидуального задания к практической работе №2.

Требования к оформлению задания:

Устно составить конспект выполнения индивидуального задания кпрактической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы, выводы по результатам.

Рекомендуемые источники [1-11].

Задание 2.3. – Подготовка к отчету по практической работе №3. Инжиниринг знаний в системе FluentEditor. Подготовка ответов на контрольные вопросы.

Требования к выполнению данного задания:

1. Подготовить ответы на контрольные вопросы к практической работе.
2. Контрольные вопросы к практической работе №3:
3. Что такое онтология?
4. Что является результатом онтологического анализа?
5. Цель создания онтологии?
6. Что является главным отличием FluentEditor от других редакторов онтологий?
7. Что такое контролируемые языки и для чего они необходимы?
8. Какой встроенный механизм использует FluentEditor для отслеживания грамматики онтологии наCNL?
9. Каким образом осуществляется создание классов, подклассов, экземпляров в FluentEditor?
10. Какие существуют правила написания имени класса, подкласса, экземпляра?
11. Для чего используется функция Reasoner?
12. Какие логические выражения используются для создания онтологий в FluentEditor?

Порядок выполнения задания:

Необходимо заранее изучить методические рекомендации по выполнению практической работы, а также материалы лекций. Обратить внимание на цель занятия, на основные вопросы для подготовки к занятию, на контрольные вопросы самопроверки после практической работы, на содержание темы занятия.

Форма контроля – отчет по практическим работам.

Требования к оформлению задания:

Устно составить конспект по практической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы.

Рекомендуемые источники [1-11].

Задание 2.4. – Выполнение индивидуального задания к практической работе №3.

Требования к выполнению данного задания:

Представить выполненное индивидуальное задание к практической работе №1. Онтологический инжиниринг знаний в системе FluentEditor

Порядок выполнения задания:

1. Для выбранной предметной области (соответствующей теме магистерской ВКР) выделить не менее 30 понятий (концептов).
2. Дать определения этим понятиям (то есть составить тезаурус).
3. На множестве понятий ввести отношения и функции интерпретации для построения онтологии по предметной области. Построить онтологию, используя инструментальное средство онтологического проектирования FluentEditor.
4. Осуществить поиск информации по разработанной предметной онтологии.

Форма контроля – Отчет о выполнении индивидуального задания к практической работе №3.

Требования к оформлению задания:

Устно составить конспект выполнения индивидуального задания кпрактической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы, выводы по результатам.

Рекомендуемые источники [1-11].

Раздел 3. Процесс принятия решений. Методы оценки альтернатив

Задание 3.1. – Подготовка к отчету по практической работе №4. Поддержка принятия решений на основе построения моделей в системе WiMi. Подготовка ответов на контрольные вопросы.

Требования к выполнению данного задания:

Подготовить ответы на контрольные вопросы к практической работе.

Контрольные вопросы к практической работе №4:

1. Каковы основные функции программного продукта WiMi?
2. В чем заключается Технология runtime?
3. Области применения программного продукта WiMi.

Порядок выполнения задания:

Необходимо заранее изучить методические рекомендации по выполнению практической работы, а также материалы лекций. Обратить внимание на цель занятия, на основные вопросы для подготовки к занятию, на контрольные вопросы самопроверки после практической работы, на содержание темы занятия.

Форма контроля – отчет по практическим работам.

Требования к оформлению задания:

Устно составить конспект по практической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы.

Рекомендуемые источники [1-4].

Задание 3.2. – Выполнение индивидуального задания к практической работе №1.

Требования к выполнению данного задания:

Представить выполненное индивидуальное задание к практической работе №4. Поддержка принятия решений на основе построения моделей в системе WiMi

Порядок выполнения задания:

1. Осуществить все операции, описанные в методических указаниях по выполнению практической работы №4 для выбранной предметной области, соответствующей теме магистерской ВКР.
2. Создать и редактировать качественные модели ситуаций/предметных областей в Wi!Mi.
3. Провести структурный анализ моделей, получить логический вывод решения и объяснить его в виде последовательности выполненных действий в Wi!Mi.

Форма контроля – Отчет о выполнении индивидуального задания к практической работе №4.

Требования к оформлению задания:

Устно составить конспект выполнения индивидуального задания к практической работе, включив в него название работы, краткое изложение теоретической части, описание хода выполнения работы, выводы по результатам.

Рекомендуемые источники [1-11].

Список литературы

1. Павлов, С. И. Системы искусственного интеллекта: учебное пособие / С. И. Павлов. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2011. – Ч. 2. – 194 с. – Режим доступа: URL: https://biblioclub.ru/index.php?page=book&id=208939
2. Павлов, С. И. Системы искусственного интеллекта: учебное пособие / С. И. Павлов. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2011. – Ч. 1. – 175 с. – Режим доступа: URL: https://biblioclub.ru/index.php?page=book&id=208933
3. Интеллектуальные информационные системы и технологии: учебное пособие / З. Ю. Ю. Громов, О. Г. Иванова, В. В. Алексеев и др.; Тамбовский государственный технический университет. – Тамбов: Тамбовский государственный технический университет (ТГТУ), 2013. – 244 с.: ил. – Режим доступа: URL: https://biblioclub.ru/index.php?page=book&id=277713
4. Моделирование систем: Подходы и методы: учебное пособие / В.Н. Волкова, Г.В. Горелова, В.Н. Козлов и др.; Министерство образования и науки Российской Федерации, Санкт-Петербургский государственный политехнический университет. - СПб.: Издательство Политехнического университета, 2013. - 568 с.: схем., ил., табл. - Режим доступа: URL: http://biblioclub.ru/index.php?page=book&id=362986 .
5. Система формирования знаний в среде Интернет: монография / В.И. Аверченков, А.В. Заболеева-Зотова, Ю.М. Казаков и др. - 3-е изд., стер. - Москва: Флинта, 2016. - 181 с. - Библиогр. в кн. - ISBN 978-5-9765-1266-5. Режим доступа: URL: URL: http://biblioclub.ru/index.php?page=book&id=93354
6. Тельнов, Ю.Ф. Инжиниринг предприятия и управление бизнес-процессами. Методология и технология: учебное пособие / Ю.Ф. Тельнов, И.Г. Фёдоров. - Москва: ЮНИТИ-ДАНА, 2015. - 207 с.: ил. - (Серия «Magister»). - Библ. в кн. - ISBN 978-5-238-02622-0; [Электронный ресурс]. Режим доступа: URL: URL: http://biblioclub.ru/index.php?page=book&id=447146
7. Коробова, И. Л. Принятие решений в системах, основанных на знаниях: учебное пособие / И. Л. Коробова, Г. В. Артемов; Тамбовский государственный технический университет. – Тамбов: Тамбовский государственный технический университет (ТГТУ), 2012. – 81 с.: ил. – Режим доступа: URL: https://biblioclub.ru/index.php?page=book&id=277800
8. Соснин П.И. Управление знаниями и опытом в проектной организации: учебное пособие / Соснин П.И., Маклаев В.А., Перцев А.А.. —

Ульяновск: Ульяновский государственный технический университет, 2018. — 215 с. — ISBN 978-5-9795-1869-5. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: https://www.iprbookshop.ru/106126.html

9. Башмаков А. И., Башмаков И. А. Интеллектуальные информационные технологии: Учеб. пособие. –М.: Изд-во МГТУ им. Н. Э. Баумана, 2005. – 304 с.

10. Варшавский П. Р., Куриленко И. Е., Михайлов И. С. Программное обеспечение интеллектуальных систем: учебное пособие / – М.: Издательский дом МЭИ, 2011. – 64 с.

11. Куриленко И. Е. Современные методологии разработки программных средств: учебное пособие / – М.: Издательский дом МЭИ, 2011. – 112 с.

Ресурсы информационно-телекоммуникационной сети «Интернет», необходимые для освоения дисциплины (модуля)

1. Электронно-библиотечная система издательства «Лань» http://e.lanbook.com/
2. Электронная библиотека по всем отраслям знаний — Режим доступа: www.iprbookshop.ru
3. Электронная библиотека по всем отраслям знаний — Режим доступа: http://biblioclub.ru
4. Научная электронная библиотека http://elibrary.ru/defaultx.asp
5. Информация о системе PROTÉGÉ https://protege.stanford.edu/products.php
6. Информация о системе FluentEditor https://www.fluentd.org

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.03.01 Интеллектуальное управление мобильными роботами

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Декан факультета информационных систем и технологий
_____ Святов К.В.
«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Дисциплина (модуль) Интеллектуальное управление мобильными роботами
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Лабораторная работа №1

Советуем устанавливать ROS melodic на linux ubuntu 18.04. Данное сочетание ПО работает корректно и не должно вызвать каких-то трудностей в процессе выполнения лабораторных работ.

Установка ROS

Для корректной установке ROS необходимо последовательно выполнять следующие шаги:

Шаг 1

Подготовительная настройка репозитория убунту.

Шаг 2

Настройка компьютера для приема ПО с ros.org:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $ (lsb_release -sc) main"> /etc/apt/sources.list.d/ros-latest.list'
```

Шаг 3

Настройка ключей:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Если возникают проблемы с подключением к серверу, необходимо заменить ссылку в предыдущей команде

на `hkp://pgp.mit.edu:80` или `hkp://keyserver.ubuntu.com:80`

Или использовать альтернативную команду:

```
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654' | sudo apt-key add -
```

Шаг 4

Убедитесь, что все обновления загружены:

```
sudo apt update
```

Установите один из наборов ROS:

Полный набор (рекомендован):ROS, rqt, rviz, различные библиотеки для 2D/3D симуляции и роботов:

```
sudo apt install ros-melodic-desktop-full
```
Неполный набор: ROS, rqt, rviz и различные библиотеки для роботов:

```
sudo apt install ros-melodic-desktop
```
Для поиска доступных пакетов можно использовать команду:

```
apt search ros-melodic
```
Для установки конкретного пакета необходимо использовать команду:

```
sudo apt install ros-melodic-PACKAGE
```
Данная команда понадобится для установки некоторых компонентов в следующих лабораторных работах.

Шаг 5

Добавление настроек параметров ROS в каждый сеанс bash:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Шаг 6

Установка инструментов:

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator
python-wstool build-essential
```
Инициализация rosdep:

```
sudo rosdep init
rosdep update
```
Установка ROS завершена.

Установка и настройка catkin

На данный момент catkin уже должен быть у вас установлен, если нет, то необходимо его установить:

```
sudo apt-get install ros-melodic-catkin
```
Создание catkin workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```
Настройка catkin окончена.

Создание пакета для лабораторной работы

Теперь в вашей домашней папке находится папка catkin_ws. Внутри нее находится папка src. Тут будут находиться ваши пакеты.

Создание пакета:

```
cd ~/catkin_ws/src
catkin_create_pkg ИМЯ_ПАКЕТА geometry_msgs rospy
```
Далее в папке пакета создаем папку launch. В ней будут храниться launch файлы.

В папке src вашего пакета создаем python-файл со скриптом для вашей черепахи.

Собираем пакет:

```
cd ~/catkin_ws
catkin_make
```

Разработка алгоритма движения черепахи

На текущем шаге необходимо написать скрипт, который позволит вашей черепахе перемещаться по заданной траектории.

По ссылке вы сможете найти программу, в которой реализовано движение черепахи. А также там можно найти launch файл. Он позволит вам запустить все нужные узлы программы одной командой.

Считывание текущих координат черепахи

Узнать текущие координаты черепахи можно путем подписки на ноду с позицией черепахи:

```
pose_subscriber = rospy.Subscriber('/turtle1/pose', Pose, poseCallback)
```
Где:

Pose - тип сообщения, который необходимо импортировать:

```
from turtlesim.msg import Pose
```
poseCallback - функция, которая будет вызвана для обработки сообщения.

Пример данной функции:

```
def poseCallback(pose_msg):
    curx = pose_msg.x
    cury = pose_msg.y
    yaw = pose_msg.theta
```
После написания скрипта для движения черепахи необходимо только запустить ее.

Запуск приложения

Способ 1: без launch файла

Обратите внимание, что каждая команда запускается в новой консоли.

Запуск ros:

```
roscore
```
Запуск симуляции черепахи:

```
rosrun turtlesim turtlesim_node
```
Запуск ноды управления черепахой с клавиатуры:

```
rosrun turtlesim turtle_teleop_key
```
Запуск скрипта движения:

```
rosrun НАЗВАНИЕ_ВАШЕГО_ПАКЕТА НАЗВАНИЕ_СКРИПТА.py
```

Способ 2: с launch файлом

Данной командой будут запущены все ноды, прописанные в launch файле:

```
roslaunch НАЗВАНИЕ_ВАШЕГО_ПАКЕТА НАЗВАНИЕ_LAUNCH_ФАЙЛА.launch
```
Также в таких файлах можно задавать параметры для нод.

Полезные ссылки

Установка ros - http://wiki.ros.org/melodic/Installation/Ubuntu

Туториалы по catkin - http://wiki.ros.org/catkin/Tutorials

Лабораторная работа №2

Установка пакетов gazebo

```
sudo apt-get install ros-melodic-gazebo-pkgs
```
Если по какой-то причине у вас не установлено gazebo, то установить его можно комадной:

```
curl -sSL http://get.gazebosim.org | sh
```

Решение часто возникаевых ошибок

rosdep: command not found

Решение:

```
sudo apt-get install python-pip
sudo pip install -U rosdep
sudo rosdep init
rosdep update
```

Failed to create the dwa_local_planner/DWAPlannerROS planner

Решение:

```
sudo apt-get install ros-melodic-dwa-local-planner
```

Создание своего мира в gazebo

Запуск gazebo:

```
gazebo
```
После запуска у вас откроется пустой мир. Стены и прочие элементы можно создавать с помощью вкладки insert.

Если у вас не отображаются стандартные модели, то необходимо добавить путь до них "/home/ИМЯ/.gazebo/models".

Далее необходимо просто выбирать объект из стандартных моделей и устанавливать его. В меню инструментов на верхней панели можно найти инструменты для перемещения, вращения и изменения размеров.

После создания мира его необходимо сохранить.

File -> Save World As.

Установка turtlebot3

```
cd ~/catkin_ws/src/
git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone -b kinetic-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
cd ~/catkin_ws && catkin_make
```

Построение карты

На этом этапе необходимо написать launch файл, который будет запускать ваш мир, робота нужной модели, ноду управления роботом с клавиш и slam алгоритм.

Далее необходимо с помощью клавиш пройти роботом всю карту.

Карта сохраняется командой:

```
rosrun map_server map_saver -f ~/map
```

Перемещение с помощью карты

В примерах находятся наброски скрипта, позволяющего роботу перемещаться к указанным координатам.

Его необходимо доработать.

Далее необоходимо написать launch файл, который запускает симуляцию мира и робота, ваш скрипт и навигацию робота.

Запуск launch-файла описан в методических указаниях к лабораторной работе №1/

Полезные ссылки

Документация turtlebot3
- http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Документация gazebo - http://gazebosim.org/

Лабораторная работа №3

Подготовка

Шаг 1

Сначала необходимо написать launch-файл для запуска генерации мира, робота и управления роботом с помощью клавиш.

Тем самым уже будет реализовано произвольное движение робота с помощью клавиш.

Шаг 2

Далее любым способом необходимо добавить движение робота по траектории из лабораторной работы №1. Считывание координат робота также происходит через подписку на ноду.

Написание скрипта

Публикация сообщений роботу осуществляется с помощью создания публикатора и дальнейшей публикации сообщений типа Twist.

Создание публикатора:

```
velocity_publisher = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
```
Получение данных с лидара осуществляется с помощью подписки на ноду:

```
laser_subscriber = rospy.Subscriber('/scan', LaserScan, las_callback)
```
Где LaserScan - тип сообщения, который необходимо импортировать:

```
from sensor_msgs.msg import LaserScan
```
las-callback - функция, которая будет обрабатывать данные с лидара.

Пример функции:

```
def las_callback(msg)
    right = msg.ranges[270]
```

Далее необходимо написать функцию ПИД регулятора, которая будет обрабатывать расстояние до стены и возвращать какое-то управляющее воздействие на робота.

Различные графики можно выводить с помощью библиотеки matplotlib.

Полезные ссылки

Лекция по ПИД регуляторам
- https://github.com/ulstu/robotics/tree/master/lectures/lec%2005.%20PID%20regulation

Лабораторная работа №4

Подготовка мира gazebo

За основу берётся мир AutoRace. Запустить его можно с помощью следующей команды:

```
roslaunch turtlebot3_gazebo turtlebot3_autorace.launch
```
С помощью редактора моделей создаётся фигура требуемой формы. Нажав правой кнопкой можно указать материал модели. Ссылка на список материалов и цветов находится в секции "Полезные ссылки" .

Функция вставки позволяет добавить созданную ранее фигуру в мир. После добавления достаточного количества моделей разных цветов следует сохранить мир в удобное место.

Данный мир предполагает использование вариации робота burger. Чтобы его добавить, используется следующие параметры в файле запуска:

```
  <param name="robot_description" command="$(find xacro)/xacro $(find
turtlebot3_description)/urdf/turtlebot3_burger_for_autorace.urdf.xacro" />
  <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf" args="-urdf -
model turtlebot3_burger -x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -
param robot_description" />
```

Получение изображения дороги

Получение несжатого изображение из камеры:

```
self.image_subscriber = rospy.Subscriber("camera/image", Image,
self.image_callback)
```
Преобразование изображения из ROS Image в формат для работы с OpenCV:

```
def image_callback(self, image):
    self.bridge = cv_bridge.CvBridge()
    cv_image = self.bridge.imgmsg_to_cv2(image, "bgr8")
```

Проективное преобразование изображение с целью выделения дороги:

```
def image_projection(image):
    height, width, _ = image.shape

    pts_src = numpy.array([[
        [width / 2 - width / 6, height / 2 + height / 13],  # Верхняя левая
точка
        [width / 2 + width / 6, height / 2 + height / 13],  # Верхняя правая
точка
        [width - width / 5, height],                        # Нижняя правая
точка
        [width / 5, height],                                # Нижняя левая
точка
    ]], numpy.int32)
    pts_dst = numpy.array([[200, 0], [800, 0], [800, 600], [200, 600]])

    h, _ = cv2.findHomography(pts_src, pts_dst)

    return cv2.warpPerspective(image, h, (1000, 600))
```

Здесь из исходного изображения вырезается область дороги, представляемая трапецией, и преобразуется в проективное изображение.

Трапеция имеет несколько параметров:

- Увеличение параметров width / 6 уменьшает длину верхнего основания
- Увеличение параметров width / 5 увеличивает длину основания
- Увеличение параметров height / 13 увеличивает длину высоты

Пример преобразования из реальной жизни:

Обнаружение дороги

Способ 1

Из изображения дороги выделяются по цвету левая и правая линии:

```
def select_left_line(image):
    lower_yellow = numpy.array([22, 93, 0])
    upper_yellow = numpy.array([45, 255, 255])

    return cv2.inRange(image, lower_yellow, upper_yellow)

def select_right_line(image):
    lower_white = numpy.array([0, 0, 180])
    upper_white = numpy.array([25, 36, 255])

    return cv2.inRange(image, lower_white, upper_white)
```

После чего найти дорогу можно либо используя преобразования Хафа, либо с помощью использования метода скользящего окна для построения полинома второй степени.

Для построения полинома одной из линий сначала необходимо построить гистограмму:

```
histogram = numpy.sum(image, axis=0)
```

После чего находится позиция текущего окна:

```
if left_or_right == 'left':
    lane_base = numpy.argmax(histogram[:center])
elif left_or_right == 'right':
    lane_base = numpy.argmax(histogram[center:]) + center
```

Гистограмма разделяется на определённое количество окон и для каждого окна находится наилучшая линия:

```
# Нахождение границ окна по x и y
win_y_low = image.shape[0] - (window + 1) * window_height
win_y_high = image.shape[0] - window * window_height
win_x_low = x_current - window_width
win_x_high = x_current + window_width

# Нахожденение ненулевых пикселе в окне
good_lane_indices = ((nonzero_y >= win_y_low) & (nonzero_y < win_y_high) &
                     (nonzero_x >= win_x_low) & (nonzero_x <
win_x_high)).nonzero()[0]
lane_indices.append(good_lane_indices)

# Удовлетворяет ли длина линии минимальной
if len(good_lane_indices) > pixel_threshold:
    x_current = numpy.int(numpy.mean(nonzero_x[good_lane_indices]))
```

Из полученных индексов линии можно построить полином:

```
lane_indices = numpy.concatenate(lane_indices)
x = nonzero_x[lane_indices]
y = nonzero_y[lane_indices]

if len(x) > 0 and len(y) > 0:
    lane_fit = numpy.polyfit(y, x, 2)
else:
    return
```

```
plot_y = numpy.linspace(0, image.shape[0] - 1, image.shape[0])

return lane_fit[0] * plot_y ** 2 + lane_fit[1] * plot_y + lane_fit[2]
```
Найдя полином для обоих линий можем найти центральную линию:

```
left_line_pts = numpy.array([numpy.transpose(numpy.vstack([left, plot_y]))])
right_line_pts =
numpy.array([numpy.flipud(numpy.transpose(numpy.vstack([right, plot_y])))])

center = numpy.mean([left, right], axis=0)
pts = numpy.hstack((left_line_pts, right_line_pts))
center_line = numpy.array([numpy.transpose(numpy.vstack([center, plot_y]))])
cv2.fillPoly(image, numpy.int_([pts]), (0, 255, 0))
```

Способ 2

Также можно обнаружить дорогу и посчитать расстояния до линий.

Для этого сначала необходимо наложить нужную маску. (Пример есть в первом способе) А далее выделить контуры с помощью:

```
_, cnts, _ = cv2.findContours(mask_yellow, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```
Где cnts - лист с точками контура.

Далее каждый контур можно вписать в прямоугольник минимальной площадью и получить его точки.

```
 for cnt in cnts:
     rect = cv2.minAreaRect(cnt)
     box = cv2.boxPoints(rect)
     box = np.int0(box)
     A, B, C, D = box
```
В A,B,C,D будут находиться координаты соответствующих точек.

Далее можно определять то, что это линия по ее пропорциям и считать расстояние до нее от центра изображения, зная его ширину.

Обнаружение объектов

Подготовка изображения

Для обнаружения объектов нужного цвета требуется перевести изображение в формат hsv:

```
hsv_target = cv2.cvtColor(source_image, cv2.COLOR_BGR2HSV)
```
После чего следует подобрать hsv-границы для обнаружения требуемого цвета с помощью сайта из секции "Полезные ссылки". Задав границы, можно выделить объекты лишь нужного цвета.

Пример для выделения объектов фиолетового цвета:

```
def select_target_objects(image):
    lower_purple = numpy.array([140, 10, 0])
    upper_purple = numpy.array([160, 255, 255])
    return cv2.inRange(image, lower_purple, upper_purple)
```

Для более точного обнаружения объектов желательно размыть полученное изображение и избавить его от шумов:

```
target_mask = cv2.blur(target_mask, (9, 9), 3)
target_mask = cv2.erode(target_mask, None, iterations=2)
target_mask = cv2.dilate(target_mask, None, iterations=2)
```

Нахождение объектов

Обнаружение шаров можно выполнить с помощью команды:

```
detected_circles = cv2.HoughCircles(target_mask, cv2.HOUGH_GRADIENT, 0.9, 50,
                                    param1=100, param2=55, minRadius=0,
maxRadius=500)
```

В остальных случаях придётся использовать поиск контуров:

```
edged = cv2.Canny(target_mask, 75, 200)
contours = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

Отслеживание объектов

Для отслеживания объектов следует хранить два словаря: содержащий центры найденных объектов и содержащий количество кадров, пройденных с потери объекта из области видимости.

При каждом вызове функции обнаружения объектов осуществляется проверка найденных ранее объектов. Если были найдены шары, сохраняются координаты их центров.

```
for (x, y, r) in numpy.array(on_screen_objects[0, :]):
    on_screen_centroids.append((x, y))
```

Если в данный момент нет отслеживаемых объектов, то они добавляются в список отслеживаемых:

```
for centroid in on_screen_centroids:
    self.add_object(centroid)
```

Иначе находятся расстояния от найденных шаров до отслеживаемых:

```
distances = dist.cdist(all_centroids, on_screen_centroids)
min_rows = distances.min(axis=1).argsort()
min_columns = distances.argmin(axis=1)[min_rows]
```

Для каждого полученного минимального расстояния обновляется координата центра и сбрасывается счётчик, отвечающий за количество кадров, с момента, когда объект пропал из области видимости:

```
for (row, column) in zip(min_rows, min_columns):
    if row in used_rows or column in used_columns:
        continue
```

```
    object_id = all_ids[row]
    self.on_screen[object_id] = on_screen_centroids[column]
    self.disappeared[object_id] = 0

    used_rows.add(row)
    used_columns.add(column)
```

Если количество отслеживаемых объектов больше или равно, чем количество найденных, то для каждого неиспользованного расстояния увеличивается счётчик пропажи. Если он превысит необходимое количество, то объект удаляется из отслеживаемых. Если же количество найденных объектов оказалось больше, чем отслеживаемых, то все они добавляются в список отслеживаемых:

```
if distances.shape[0] >= distances.shape[1]:
    for row in unused_rows:
        object_id = all_ids[row]
        self.disappeared[object_id] += 1

        if self.disappeared[object_id] > self.delay:
            self.remove_object(object_id)
else:
    for column in unused_columns:
        self.add_object(on_screen_centroids[column])
```

Полезные ссылки

Список доступных материалов в gazebo
- http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials

Сайт для подбора hsv-цветов - https://alloyui.com/examples/color-picker/hsv.html

Обнаружение дороги - https://medium.com/@mithi/advanced-lane-finding-using-computer-vision-techniques-7f3230b6c6f2

Отслеживание объектов - https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

Создание и импорт моделей из Blender 2.9 в Gazebo 11

Загрузка Blender

Blender – проект 3д редактора для растровой графики с открытым исходным кодом. Он доступен бесплатно для загрузки с официального сайта blender для всех основных ОС.

- **Windows 7, 8, 10** – доступен установщик в формате msi.

- **Linux (Debian и Ubuntu)** – установка в виде deb пакета. Также установка может производиться из стандартных репозиториев с помощью утилиты apt или менеджеров пакетов aptitude и synaptic. Однако, в данном случае, вы получите более старую версию программы.

- **Mac OS** – установка из образа данных диска в формате dmg.

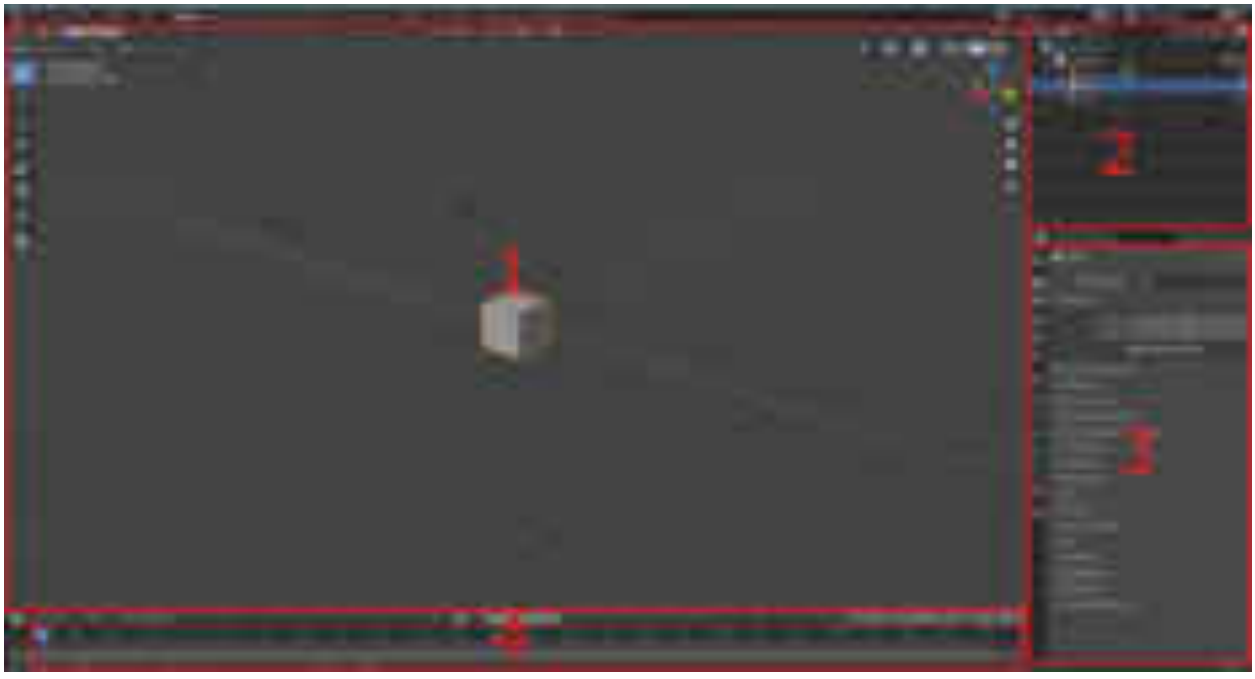
- **Прочие системы** – доступна сборка из исходников.

Сам процесс установки является тривиальным и в данной статье рассматриваться не будет.

Внимание! В конце установки при первом запуске программы будет предложен выбор настроек выделения объектов, вращения камеры и перемещения. Выбираете более удобный для вас вариант. В данном руководстве используется пресет настроек Blender 27x.

Основы работы в Blender 2.9

Интерфейс программы



1. ViewPort – Окно 3д просмотра сцены.

2. Outliner – Дерево проекта. Все объекты в текущем проекте, объединенные в иерархические группы.
3. Properties – Свойства сцены, объекта, мира и тд.
4. TimeLine – Окно с временной шкалой для создания анимации.

Данные компоненты интерфейса могут масштабироваться и взаимозаменяться в зависимости от текущих задач и предпочтений конкретного пользователя.

Режимы работы

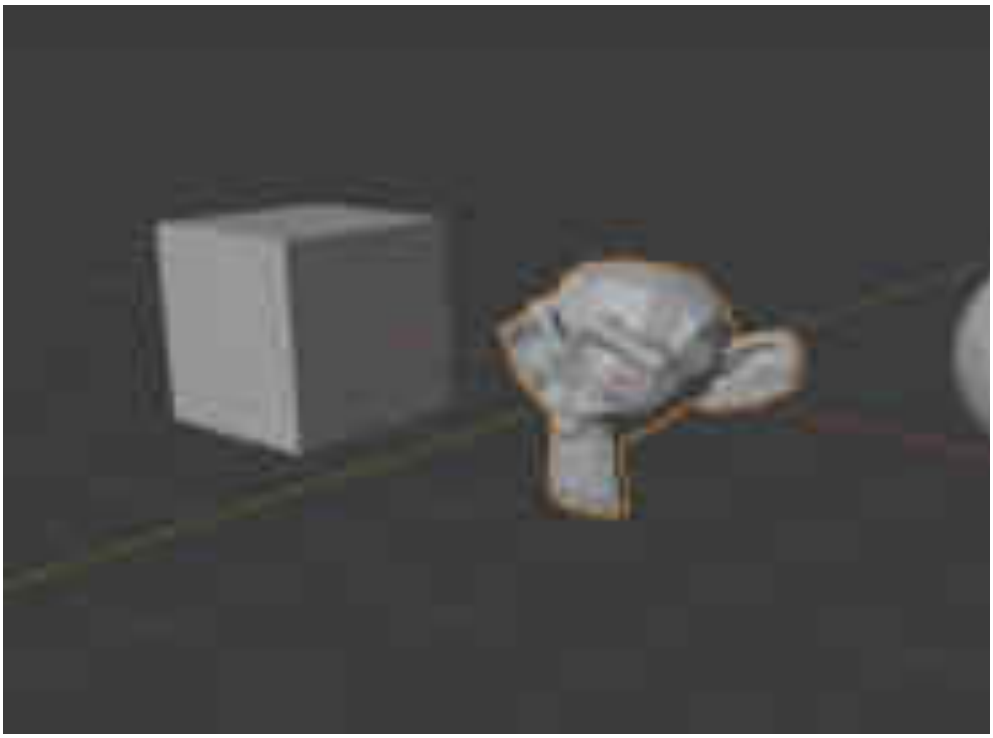
Два основных режима работы в blender, в которых пользователи проводят большую часть времени это объектный режим и режим редактирования. Их переключение осуществляется через клавишу **Tab**.

- Объектный режим служит для перемещения, масштабирования, вращения объектов по пространству сцены.
- Режим редактирования используется для изменения геометрии объектов и экструдирования (выдавливания) моделей.

Основы работы в объектном режиме

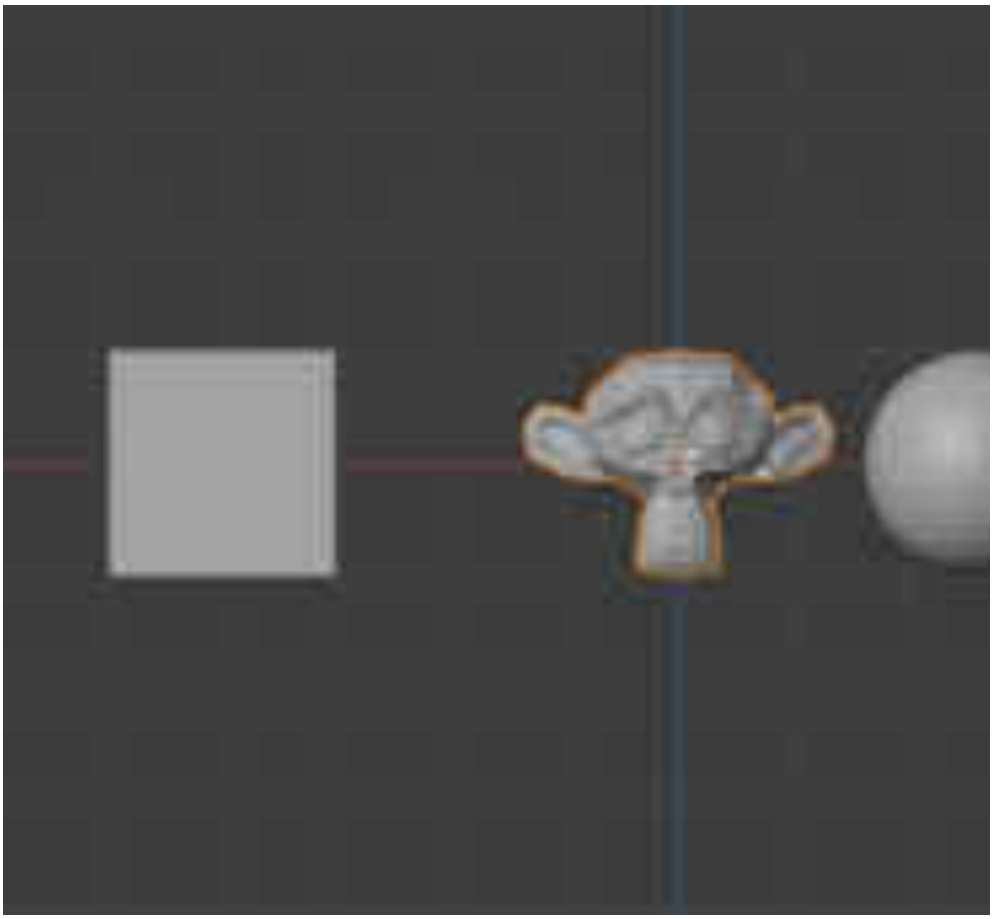
Выбор вида

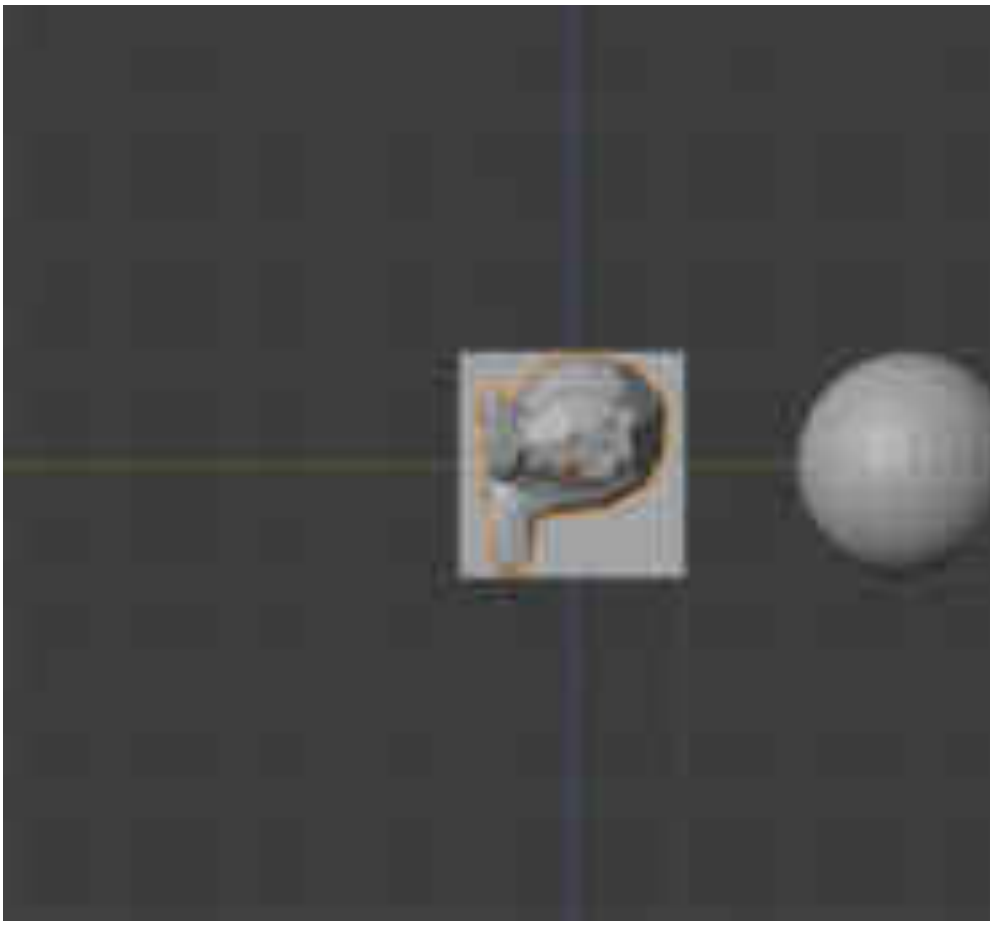
Выбор вида осуществляется с помощью цифровых клавиш клавиатуры на блоке **numpad**.

Вид	**Отображение**
Изначальный вид сцены	

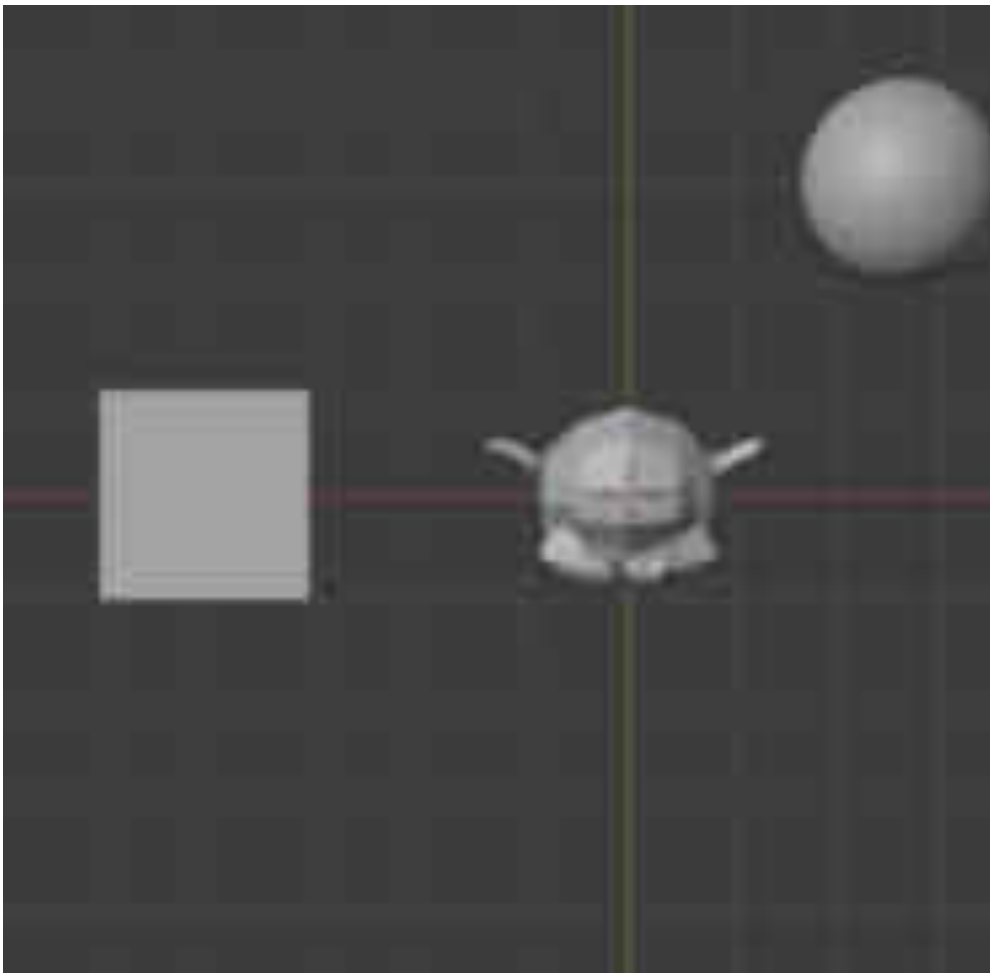
Вид **Отображение**

1 – вид спереди



Вид	Отображение
3 – вид сбоку	

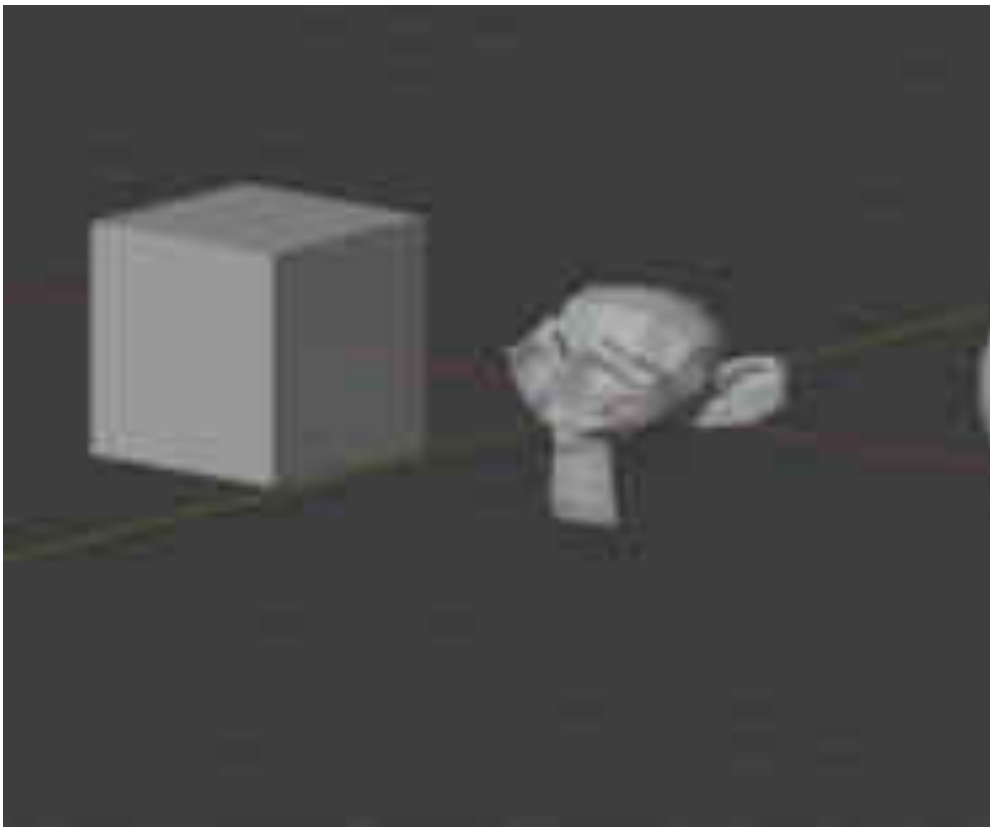
Вид **Отображение**



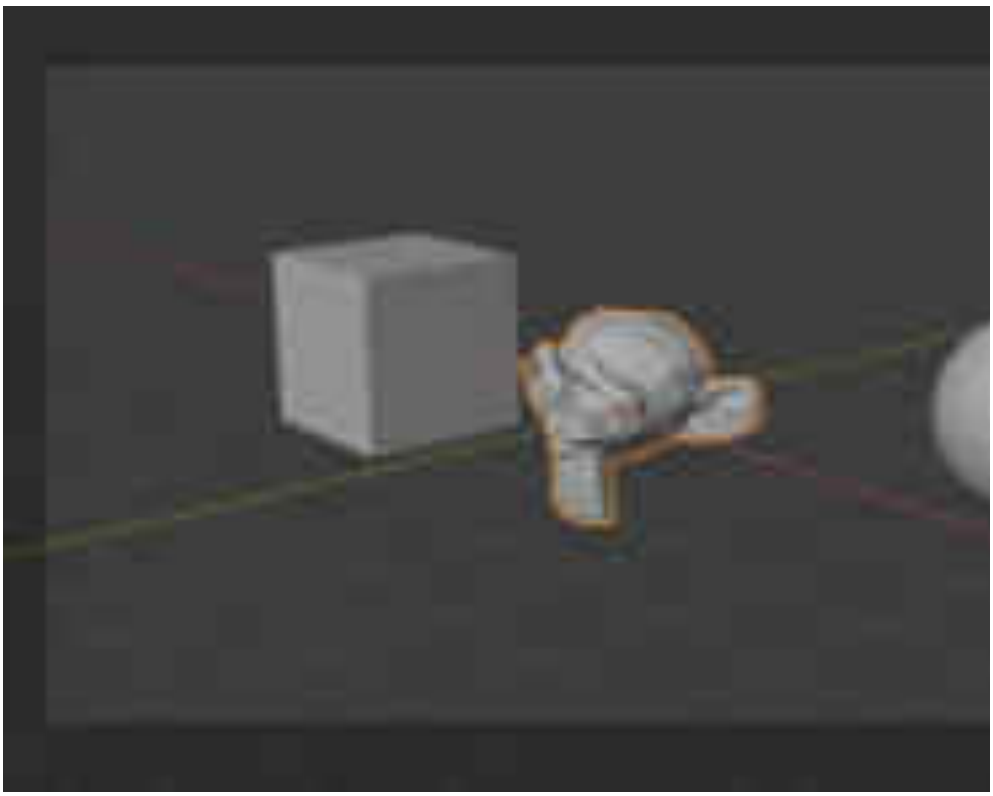
7 – вид сверху

Вид **Отображение**

5 – смена ортографической и перспективной проекции

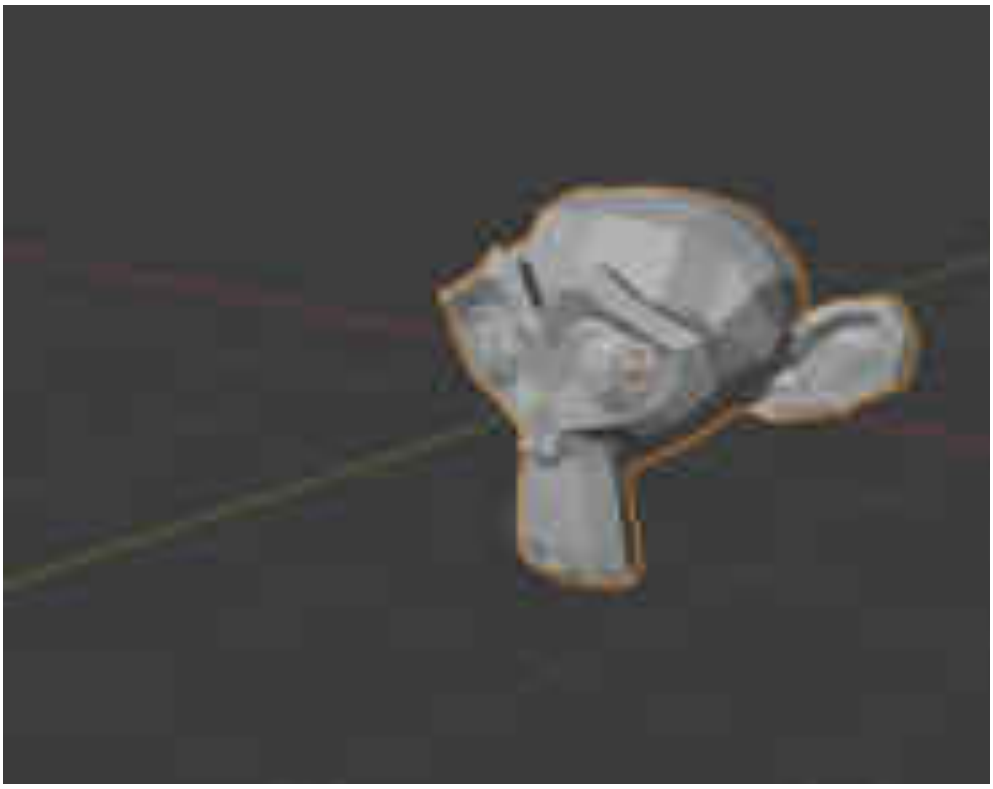


0 – вид с камеры



Вид **Отображение**

/ – локальный режим.Данный режим скрывает все объекты кроме выделенного, это может быть полезным при редактировании. Повторное нажатие вернет Вас в глобальный режим.



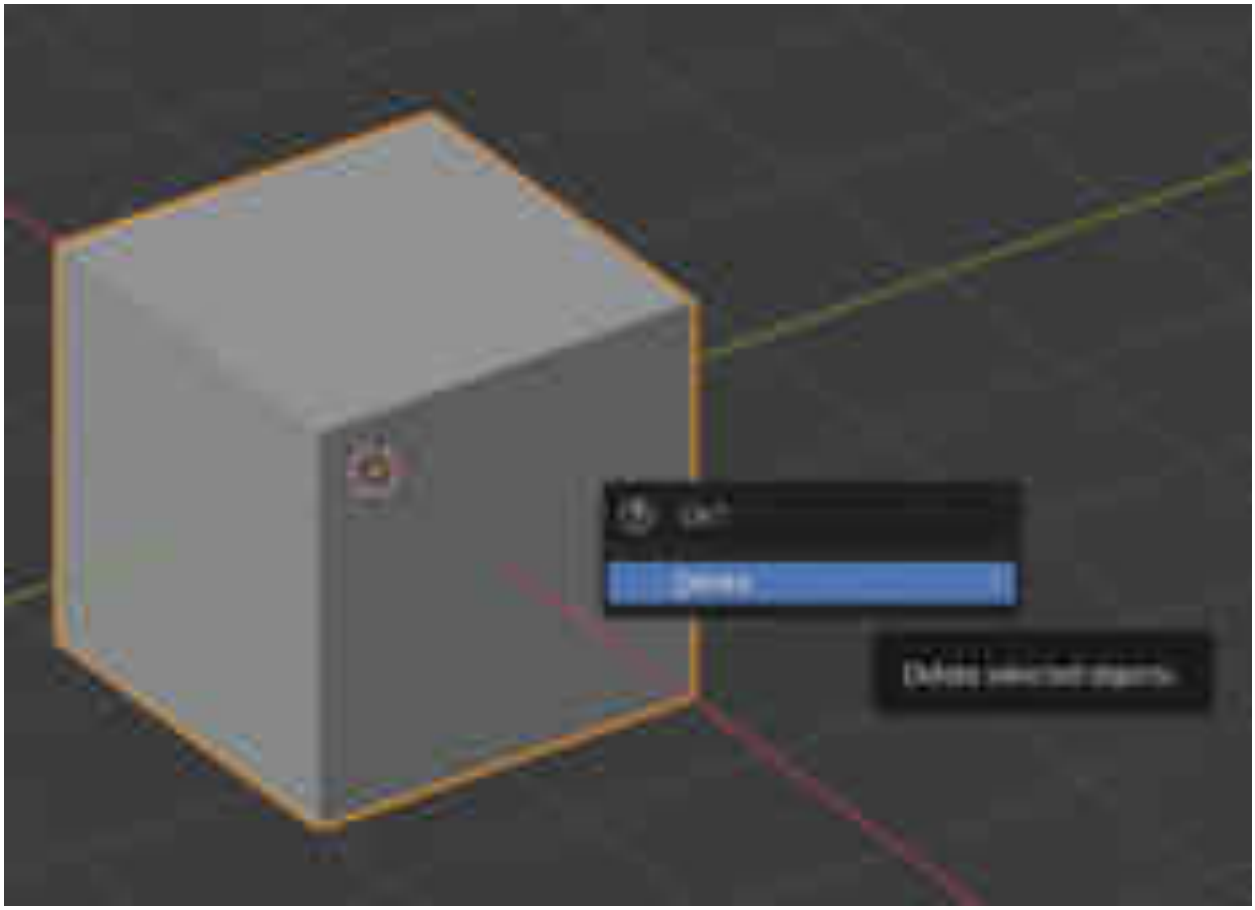
Добавление и удаление объектов

Blender изначально располагает набором геометрических примитивов для построения на их основе сложных объектов путем редактирования и объединения.

Для добавления объекта необходимо поместить курсор в пространство ViewPort и нажать **shift+A**. Для построения простых сцен Вам будет достаточно раздела *Mesh*. Он содержит все необходимые геометрические примитивы.

После создания объекта в правом нижнем углу появится окно свойств объекта.

Для удаления объектов выделите нужные объекты и нажмите X.



Выделение объектов

Для выделения объекта, грани, вершины или ребра применяется ПКМ или ЛКМ в зависимости от настроек, что были выбраны при установке. Чтобы выделить все объекты в сцене – нажмите A. Повторное нажатие снимает выделение со всех объектов.

Специальные режимы выделения

Если требуется быстро выделить несколько объектов или область объекта в режиме редактирования, то можно применить следующие режимы:

- Выделение рамкой – клавиша **B**.
- Выделение круговой области – клавиша **C**. Вращение колеса мышки изменяет размер области.

Перемещение, вращение, масштабирование

Для всех действий можно ограничить их применение по одной или двум осям.

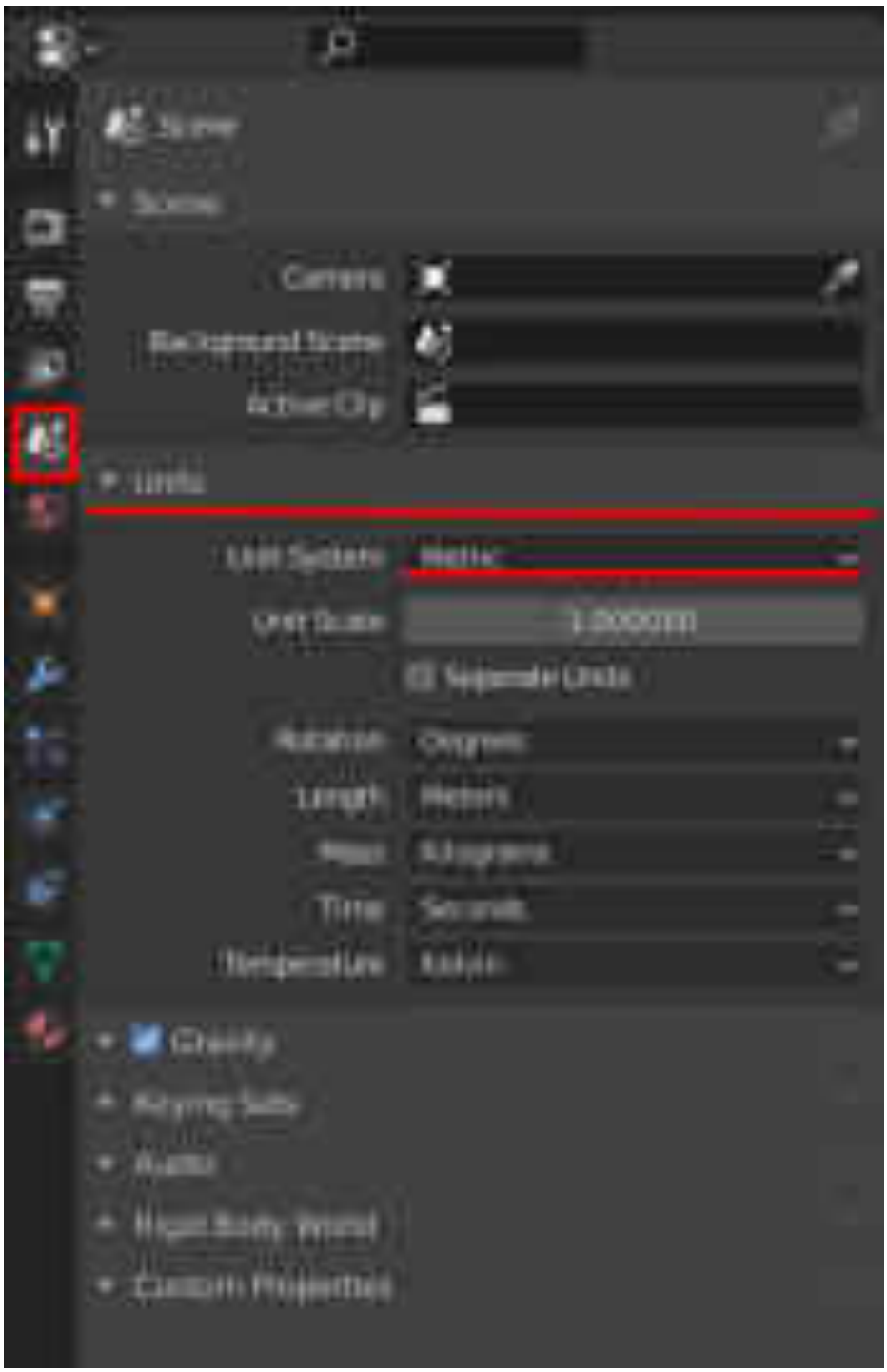
Для того чтобы ограничить выбор одной осью нажмите клавишу x, y, z после нажатия клавиши действия. Чтобы оставить лишь две оси и работать с ними, исключите 3ью ось нажатием shift+[x, y, z]

Перемещение	**Вращение**	**Масштабирование**
G	**R**	**S**

Основы моделирования

Предварительная настройка сцены

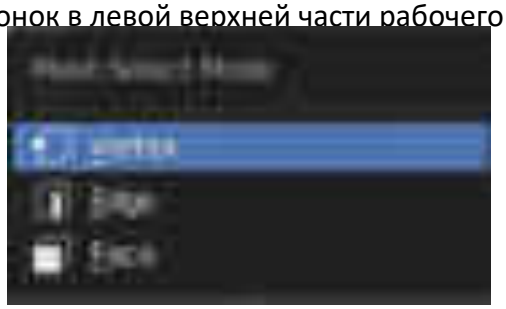
Зайдите на вкладку **Scene** в окне свойств и установите метрическую систему мер. Это упростит дальнейшее создание модели и ее экспорт в Gazebo



Режимы выделения

При редактировании объекта мы можем работать в одном из трех способов выделения текущего меша:

- вершины
- ребра

- грани Для их смены используете меню иконок в левой верхней части рабочего экрана или комбинацию клавиш **ctrl+tab** s 

Для выделения нескольких граней, ребер или вершин удерживаете shift. Удерживание Alt позволяет выделить непрерывную линию данных элементов. Например, вершины, расположенные по кругу.

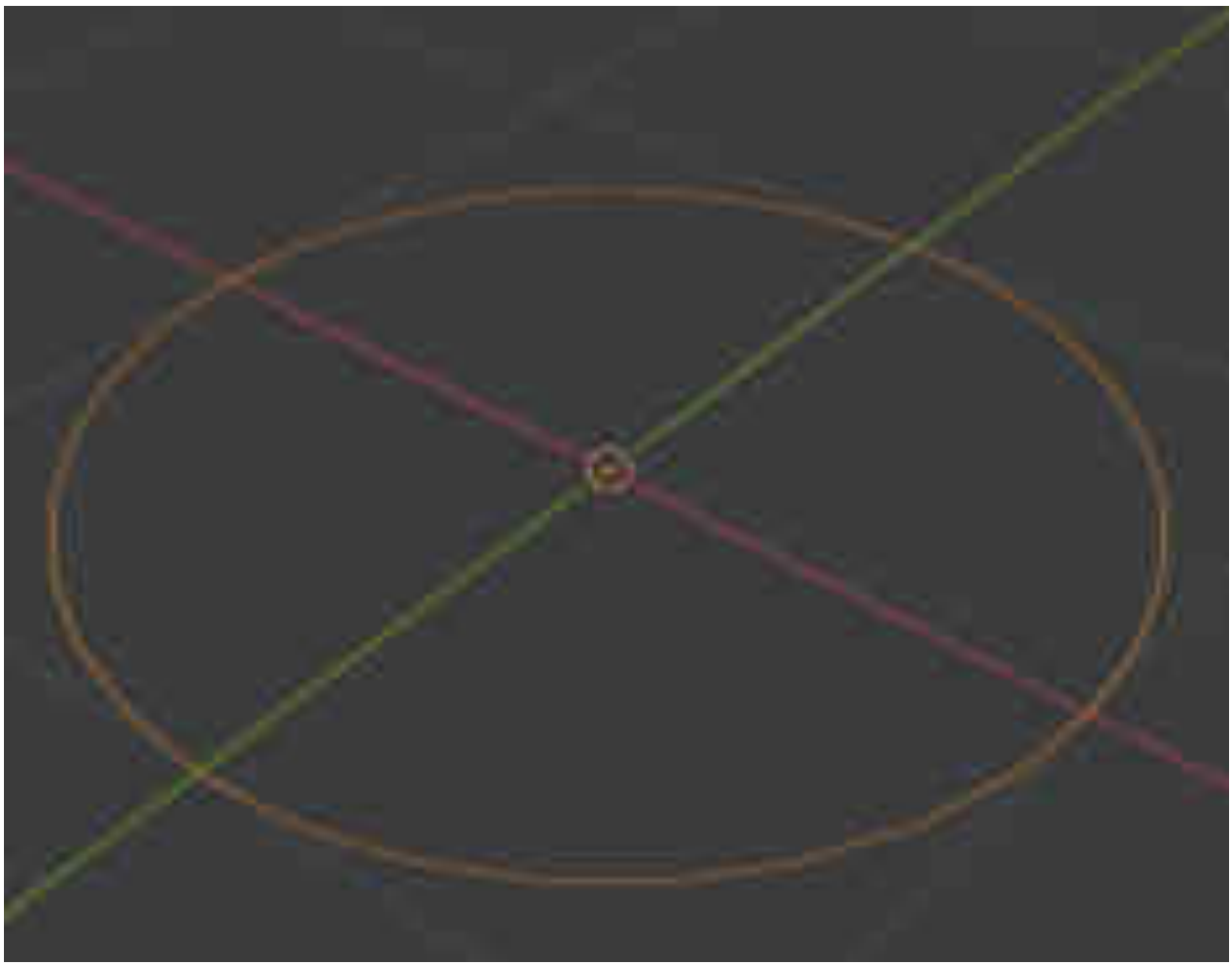
Экструдирование

Основой создания новых объектов является – экструдирование. Для его выделите ребра или грани, требующие экструдирования, и нажмите **E**. Автоматически экструдирование производится в направлении вектора нормали текущей грани или сумме векторов граней. Однако, к нему также можно применять ограничения по осям, описанное в разделе «Перемещение, вращение, масштабирование». Для того, чтобы выдавить грань на заданное расстояние нажмите E, выберите требуемую ось и введите расстояние на клавиатуре.

Пример экструдирования

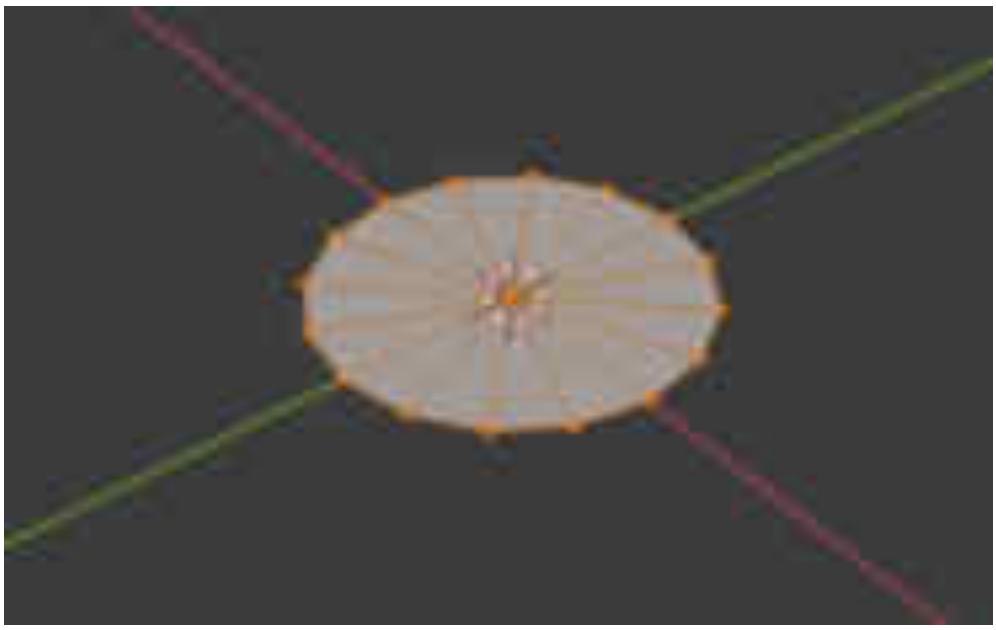
Создание столба, экструдированием круга по оси Z на 5 метров вверх

1. Добавляем круг

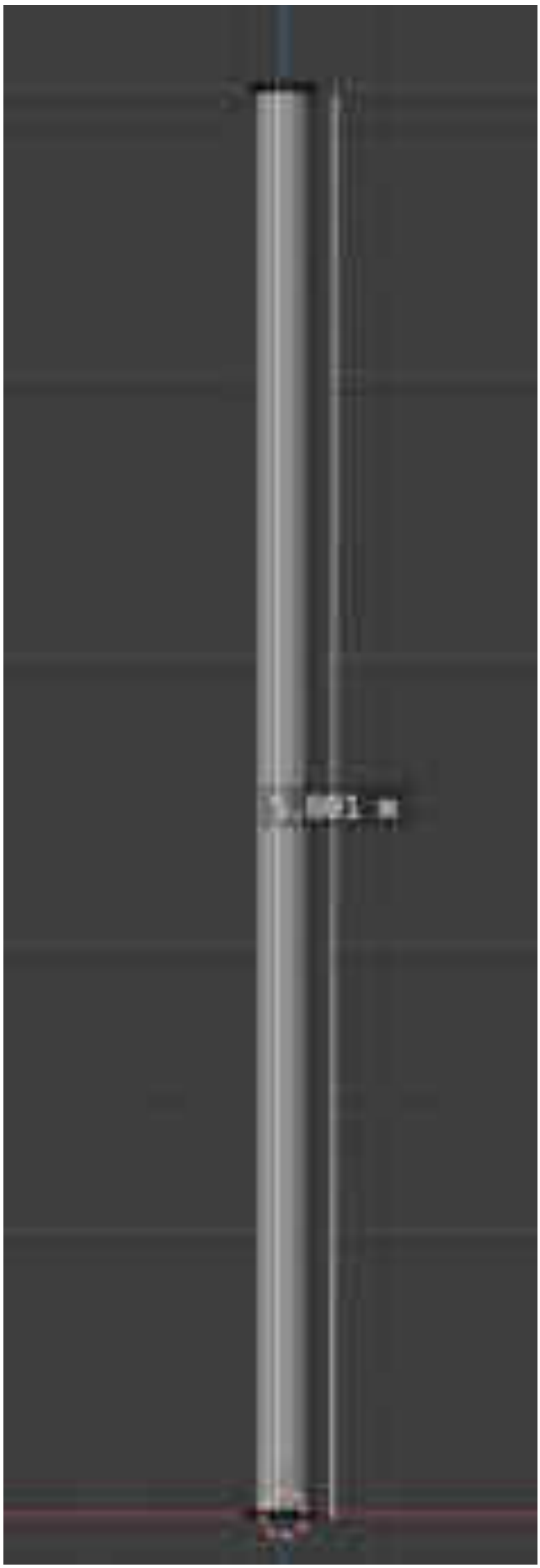


2. В окне свойств понизим количество вершин до 16, уменьшим радиус до 10см и установим тип заполнения Triangle Fan

3. Переходим в режим редактирования (**tab**). Выделяем все грани **A**



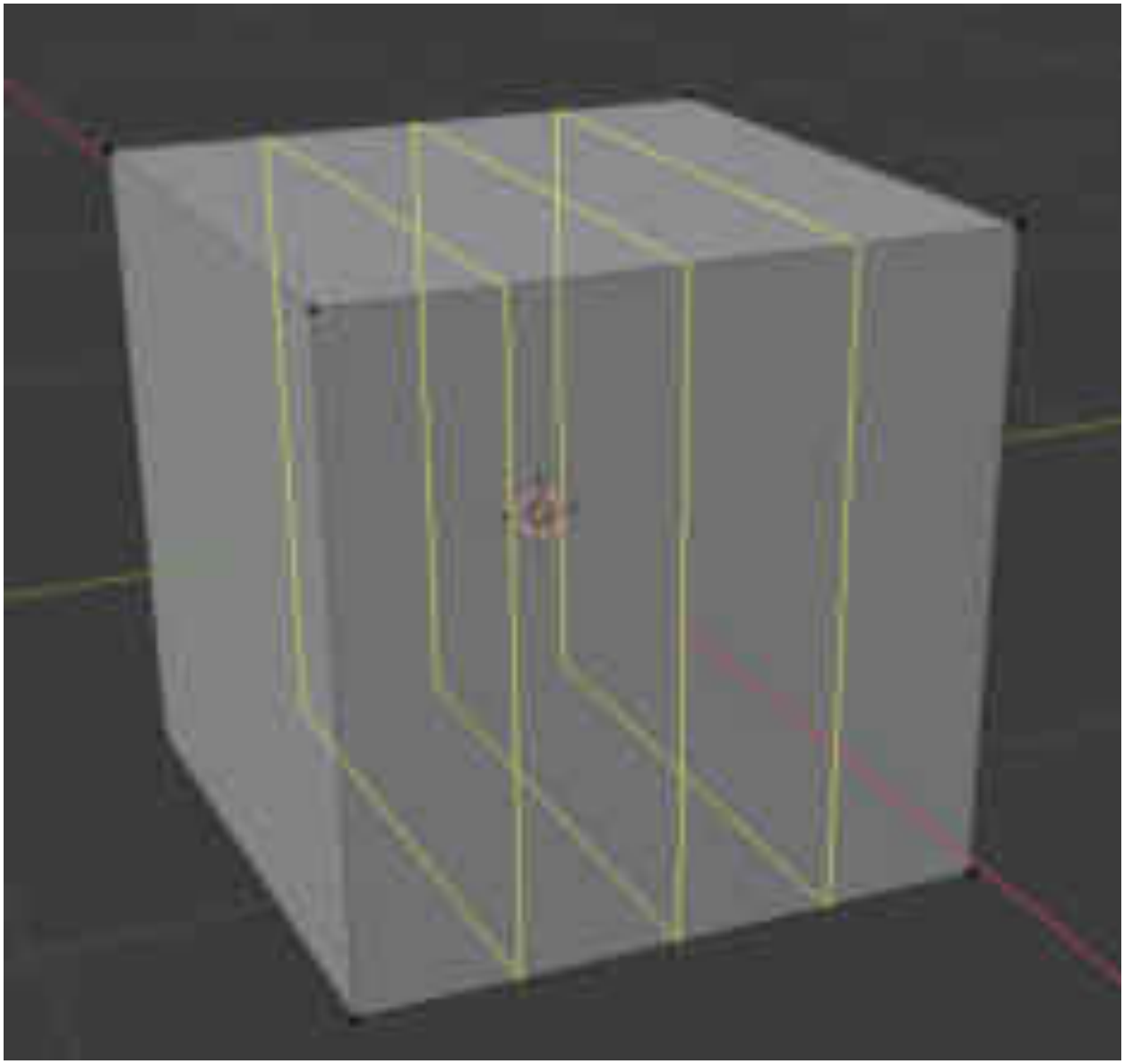
4. Нажимаем клавишу **E**, Нормаль круга совпадает с осью Z, поэтому ограничение указывать не требуется. Набираем на клавиатуре **5** и делаем клик ЛКМ.



Основа столба в виде цилиндра готова.

Добавление ребер

Для добавления ребер в режиме редактирования нажмите **Ctrl+R** и подведите курсор к грани объекта. Вращение колеса мышки увеличивает количество ребер.



Инструмент нож

Активируется нажатием **K**, позволяет разрезать грани объектов.

Для фиксации под угол 0, 90 или 45 градусов нажмите **C**. После окончания реза необходимо нажать Enter для подтверждения.

Подключение дополнений

Стандартный набор мешей можно значительно расширить подключив разнообразные аддоны. Для быстрого прототипирования сцен рекомендуется подключить

- Archimesh
- Archipack
- Extra Objects

Для их добавления перейдите в раздел *edit->preferences->addons*. Найдите пакеты Archimesh, Archipack и Extra Objects. Отметьте их галочками.

Внимание! Пакет Archipack требует предварительного рендера, для этого, после установки галочки, нажмите кнопку render presents thumbs, дождитесь завершения процесса и перезапустите Blender.
После этого в разделе Mesh появятся новые пункты.

Не игнорируйте данные разделы. Они содержат множество объектов, которые будут полезны новичкам и ускорят создание сцены. Дополнительные модели можно найти в публичных репозиториях в Интернете

Импорт моделей в формат .dae

Для экспорта модели выберете

```
File -> Export -> Collada (.dae)
```

Расставляем настройки осей, как указано на рисунке. По умолчанию все объекты, находящиеся в сцене, будут превращены в одну модель. Если требуется добавить только выделенные объекты, то указываем Selection Only.

В разделе **Geom** выбираем конвертацию граней в треугольники, а также применение модификаторов, если таковые были.

После этого выбираем место сохранения модели, вводим имя латинскими буквами и нажимаем кнопку **Export**

Загрузка модели в Gazebo

В Gazebo необходимо перейти в режим моделирования комбинацией **Ctrl+M**

Нажать кнопку Add в разделе Insert – Custom Shapes. Появится форма добавления модели. Укажите путь к ней, а также дайте ей имя.

Разместите модель в окне просмотра.

Нажмите

`File->Save as.`
Заполните описание модели, указав имя, описание и авторство.

После этого можно сохранить модель в удобное для вас место. По умолчанию, Gazebo создаст в вашей домашней директории папку для пользовательской коллекции моделей. В дальнейшем подобную модель можно использовать наряду со стандартными объектами. Добавляя ее в свой мир в разделе Insert.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.03.02 История и направления развития искусственного интеллекта

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

История и направления развития искусственного интеллекта

Профиль подготовки

Искусственный интеллект в автоматизации проектирования

Квалификация выпускника

Магистр

Формы обучения

очная

г. Ульяновск, 2021

ЛЕКЦИИ

Лекция 1. Предпосылки развития науки искусственного интеллекта

1. **Самые популярные сферы развития искусственного интеллекта**
2. **Области применения искусственного интеллекта**

Искусственный интеллект (ИИ) – один из главных трендов нашего времени. За последнее десятилетие компьютеры обучали решению все более сложных задач. Теперь они способны выполнять множество вещей, которые ранее казались присущими только человеку. Успех ИИ не стоит на месте. Машины активно покоряют многие области, начиная от идентификации людей в толпе, управления автомобилем на загруженной автомагистрали и заканчивая победами над лучшими игроками в го – игре, которая многие годы казалась чем-то недостижимым для ИИ, – и на этом достижения не заканчиваются. Иногда компьютеры выполняют работу лучше людей. В большинстве своем машины работают быстрее, дольше и никогда не устают.
Конечно же, идея разумных машин совершенно не нова. Без малого 75 лет мы пытались создать компьютеры, способные продемонстрировать хоть толику нашего интеллекта. А концепция автоматов, похожих на человека, и вовсе родилась столетия назад. Мы очарованы собой и своим интеллектом, и нет ничего удивительно в нашем желании наделить «искрой человечности» машины. Сравнение искусственного интеллекта с человеческим вызывает как радость, так и беспокойство. Насколько похожим на нас станет ИИ? Сможет ли он нас заменить, лишить работы, превзойти в играх и творческих начинаниях, придающих смысл нашей жизни?

1. Предпосылки развития науки искусственного интеллекта

Философия искусственного интеллекта

Философия искусственного интеллекта задаётся вопросами о «мышлении машин», эти вопросы отражают интересы различных исследователей искусственного интеллекта, философов, исследователей познавательной (когнитивной) деятельности. Ответы на эти вопросы зависят от того, что понимается под понятиями «интеллект» или «сознание», и какие именно «машины» являются предметом обсуждения.

- Может ли машина мыслить?

- Что считать интеллектом?
- Как лучше представлять и использовать знания и информацию?
- Этические проблемы создания искусственного разума

Этические проблемы создания искусственного разума

Если в будущем машины смогут рассуждать, осознавать себя и иметь чувства, то что тогда делает человека человеком, а машину — машиной?
Если в будущем машины смогут осознавать себя и иметь чувства, возможно ли будет их эксплуатировать или придется наделять их правами?
Если в будущем машины смогут рассуждать, то как сложатся отношения людей и машин? Данный вопрос был не раз рассмотрен в произведениях искусства на примере противостояния людей и машин.
Будет ли человек, которому в результате многочисленных медицинских имплантаций заменили 99 процентов тела, считаться машиной?

> **Интеллект** (от лат. intellectus «восприятие»; «разумение», «понимание»; «понятие», «рассудок») или ум — качество психики, состоящее из способности осознавать новые ситуации, способности к обучению и запоминанию на основе опыта, пониманию и применению абстрактных концепций, и использованию своих знаний для управления окружающей человека средой. Общая способность к познанию и решению проблем, которая объединяет познавательные способности: *ощущение, восприятие, память, представление, мышление, воображение.*
>
> *Википедия*

Составляющие интеллекта и его роль

Интеллект — это, прежде всего, основа целеполагания, планирования ресурсов и построение стратегии достижения цели.
Интеллект как способность обычно реализуется при помощи других способностей. Таких как:

- *способности познавать, обучаться,*
- *мыслить логически,*
- *систематизировать информацию путём её анализа,*
- *определять её применимость (классифицировать), находить в ней связи, закономерности и отличия, ассоциировать её с подобной и т. д.*

О наличии интеллекта можно говорить при совокупности всех этих способностей, в отдельности каждая из них не формирует интеллект!
К параметрам, формирующим отличительные особенности интеллектуальной системы человека относят:

- объём рабочей памяти, способность к прогнозированию, орудийной деятельности, логике,
- многоуровневую (6 слоев нейронов) иерархию системного отбора ценной информации,
- сознание,
- память.

Часть исследователей интеллекта и рабочей памяти считает, что рабочая память и подвижный интеллект находятся в сильной связи друг с другом и в значительной степени являются эквивалентными конструктами, другие, что хотя эти сущности коррелируются, но являются автономными, как рост и вес. Показано, что индивидуальные различия рабочей памяти объясняют от трети до половины всех индивидуальных различий общего интеллекта.

Искусственный интеллект — свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека ; наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.

Одно из частных определений интеллекта, общее для человека и «машины», можно сформулировать так:

«Интеллект — способность системы создавать в ходе самообучения программы (в первую очередь эвристические) для решения задач определённого класса сложности и решать эти задачи»

Идея создания искусственного подобия человека для решения сложных задач и моделирования человеческого разума витала в воздухе еще в древнейшие времена. Так, в древнем Египте была создана «оживающая» механическая статуя бога Амона. У Гомера в «Илиаде» бог Гефест ковал человекоподобные существа-автоматы. В литературе эта идея обыгрывалась многократно: от Галатеи Пигмалиона до Буратино папы Карло. Однако родоначальником искусственного интеллекта считается средневековый испанский философ, математик и поэт Раймонд Луллий, который еще в XIII веке попытался создать механическую машину для решения различных задач, на основе разработанной им всеобщей классификации понятий.

В XVIII веке Лейбниц и Декарт независимо друг от друга продолжили эту идею, предложив универсальные языки классификации всех наук. Эти работы можно считать первыми теоретическими работами в области искусственного интеллекта.

Окончательное рождение искусственного интеллекта как научного направления произошло только после создания ЭВМ в 40-х годах XX века. В это же

время Нор-берт Винер создал свои основополагающие работы по новой науке — кибернетике.

История искусственного интеллекта как нового научного направления начинается в середине XX века. К этому времени уже было сформировано множество предпосылок его зарождения:

- среди философов давно шли споры о природе человека и процессе познания мира,
- нейрофизиологи и психологи разработали ряд теорий относительно работы человеческого мозга и мышления,
- экономисты и математики задавались вопросами оптимальных расчётов и представления знаний о мире в формализованном виде;
- наконец, зародился фундамент математической теории вычислений — теории алгоритмов — и были созданы первые компьютеры.

Философские предпосылки к возникновению науки

На саму возможность мыслить о понятии «Искусственный интеллект» огромное влияние оказало рождение механистического материализма, которое начинается с работы Рене Декарта «Рассуждение о методе» (1637) и сразу вслед за этим работы Томаса Гоббса «Человеческая природа» (1640).

Рене Декарт предположил, что животное — некий сложный механизм, тем самым сформулировав механистическую теорию.

И тут важно понимать, чем отличается именно механистический материализм, от античного материализма, взгляды которого запечатлены в работах Аристотеля, и последующей диалектики Гегеля, диалектического и исторического материализма (Фейербах, Карл Маркс, Фридрих Энгельс, В. И. Ленин). Дело в том, что механистический материализм направлен на механистическое происхождение организмов, в то время как античный материализм направлен на механистическое происхождение природы, а диалектический и исторический материализм относится к проявлениям механизма в обществе.

Поэтому понятно, что без понимания механистичности в организмах не могла идти речь о понимании искусственного интеллекта даже в самом примитивном смысле, а наличие механистичности природы и общества выходят за рамки области об искусственном интеллекте, и строго говоря не являются необходимыми предпосылками.

Технологические предпосылки к возникновению науки

В 1623 г. Вильгельм Шикард (нем. Wilhelm Schickard) построил первую механическую цифровую вычислительную машину, за которой последовали машины Блеза Паскаля (1643) и Лейбница (1671). Лейбниц также был первым,

кто описал современную двоичную систему счисления, хотя до него этой системой периодически увлекались многие великие ученые.
В 1832 году коллежский советник С. Н. Корсаков выдвинул принцип разработки научных методов и устройств для усиления возможностей разума и предложил серию «интеллектуальных машин», в конструкции которых, впервые в истории информатики, применил перфорированные карты.
В XIX веке Чарльз Бэббидж и Ада Лавлейс работали над программируемой механической вычислительной машиной.
В 1910—1913 гг. Бертран Рассел и А. Н. Уайтхед опубликовали работу «Принципы математики», которая произвела революцию в формальной логике.
В 1941 Конрад Цузе построил первый работающий программно-управляемый компьютер.
Уоррен Маккалок и Уолтер Питтс в 1943 опубликовали *A Logical Calculus of the Ideas Immanent in Nervous Activity*, который заложил основы нейронных сетей.

Автоматы: от андроидов до роботов

Первым, кто представил чертеж человекоподобного робота, был великий Леонардо да Винчи примерно в 1495 году. Чертеж представлял модель механического рыцаря, который может сидеть, стоять, двигать руками, головой и, возможно, захватывать предметы. Но так и неизвестно, пытался ли да Винчи воплотить в реальность этот механизм.
В 16-17 веке в Западной Европе инженеры начали конструировать автоматоны — заводные механизмы наподобие человека, которые могли выполнять довольно сложные действия. Самый известный из них – робот «испанский монах», который был изобретен примерно в 1560 году механиком Хуанело Турриано для императора Карла V. Автоматон был около 40 см в высоту, способный ходить, бить себя в грудь рукой, кивать головой и даже преподносить деревянный крест к губам.
Более заметный прогресс в робототехнике наблюдался в 18 веке. К примеру, в 1738 году французский инженер Жак де Вокансон собрал первого в мире андроида, способного играть на флейте.
С 19 века изобретения стали приобретать более практический смысл. В 1898 году известный физик Никола Тесла представил общественности миниатюрное радиоуправляемое судно. Первоначально это изобретение казалось немного причудливым. Но в дальнейшем его идеи стали воплощаться в жизнь и приобрели широкое применение.
1921 год – механизмы, наконец, обрели четкий термин «робот» благодаря чешскому писателю Карлу Чапеку и его пьесе под названием «Россумские

Универсальные Роботы». Примечательно, что Чапек назвал этим словом не машины, а живых людей, создаваемых на специальной фабрике. Но термин закрепился в науке и дал жизнь всем автоматизированным устройствам.

В середине 20 века, в частности, в 1950-ых стали разрабатываться механические манипуляторы для взаимодействия с радиоактивными материалами. Эти роботы копировали движения рук человека, находящегося в безопасном месте.

В 1968 году японской компанией Kawasaki Heavy Industries, Ltd был произведен первый промышленный робот. С тех пор Япония начала вовсю стремиться стать мировой столицей робототехники, и ей это удалось. Несмотря на то, что роботы изначально разрабатывались в США, они импортировались в Японию в малых количествах, где инженеры изучали их и применяли в производстве.

Коммерческое распространение роботов началось с 1980-ых годов. Технический прогресс двигался в направлении совершенствования систем управления. Такие компании как Unimate, Hitachi KUKA, Westinghouse, FANUC развивали системы датчиков для своих роботов, делая их более чувствительными к задачам, которые они выполняют.

В конце 90-ых – начале 2000-ых начался активный рост и развитие отрасли с использованием новых контроллеров, языков программирования, запуска первых роботов в космос и возникновением машин, создающих роботов.

В это время также появились новые человекоподобные роботы, такие как канадский Aiko, имитирующий человеческие чувства (осязание, слух, речь, зрение), ASIMO – гуманоид японской фирмы Honda, робот-собака AIBO, созданная компанией Sony и другие.

Классические работы

В 1943 году в своей статье «Логическое исчисление идей, относящихся к нервной активности» У. Мак-Каллок и У. Питтс предложили понятие искусственной нейронной сети. В частности, ими была предложена модель искусственного нейрона.

Д. Хебб в работе «Организация поведения» 1949 года описал основные принципы обучения нейронов. Эти идеи несколько лет спустя развил американский нейрофизиолог Фрэнк Розенблатт. Он предложил схему устройства, моделирующего процесс человеческого восприятия, и назвал его «перцептроном».

Среди советских учёных искусственный интеллект был главной областью научной деятельности Д. А. Поспелова. Здесь научные интересы Д. А. Поспелова связаны с моделированием поведения человека, формализацией рассуждений, общими проблемами моделирования жизненных процессов в

естественных и искусственных системах. В частности, Д. А. Поспеловым был впервые в мире разработан подход к принятию решений, опирающийся на семиотические (логико-лингвистические) модели, который послужил теоретической основой ситуационного управления большими системами. По истории также можно проследить интерес других советских учёных к кибернетике.

Этапы развития ИИ

Кратко всю историю искусственного интеллекта можно разбить на следующие периоды:

- ***П**оявление предпосылок искусственного интеллекта (период с 1943 года по 1955 год)*
- ***Р**ождение искусственного интеллекта (1956 год)*
- ***Р**анний энтузиазм, большие ожидания (период с 1952 года по 1969 год)*
- ***С**толкновение с реальностью (период с 1966 года по 1973 год)*
- ***С**истемы, основанные на знаниях (период с 1969 года по 1979 год)*
- ***П**ревращение искусственного интеллекта в индустрию (период с 1980 года по настоящее время)*
- *Возвращение к нейронным сетям (период с 1986 года по настоящее время)*
- *Превращение искусственного интеллекта в науку (период с 1987 года по настоящее время)*
- *Появление подхода, основанного на использовании интеллектуальных агентов*

1. Появление предпосылок искусственного интеллекта (период с 1943 года по 1955 год)

Первая работа, которая теперь по общему признанию считается относящейся к искусственному интеллекту, была выполнена Уорреном Мак-Каллоком и Уолтером Питтсом. Они черпали вдохновение из трех источников: знание основ физиологии и назначения нейронов в мозгу; формальный анализ логики высказываний, а также теория вычислений Тьюринга.

В 1951 году два аспиранта факультета математики Принстонского университета, Марвин Минский и Дин Эдмондс, создали первый сетевой компьютер на основе нейронной сети.
Кроме того, можно привести большое количество примеров других ранних работ, которые можно охарактеризовать как относящиеся к искусственному интеллекту, но именно Алан Тьюринг впервые выразил полное представление об искусственном интеллекте в своей статье Computing Machinery and Intelligence, которая была опубликована в 1950 году. В этой статье он описал тест Тьюринга, принципы машинного обучения, генетические алгоритмы и обучение с подкреплением.

2. Рождение искусственного интеллекта (1956 год)

Джон Маккарти с другими участниками организовали двухмесячный семинар в Дартмуте летом 1956 года. Дартмутский семинар не привел к появлению каких-либо новых крупных открытий, но позволил познакомиться всем наиболее важным деятелям в научной области исследований интеллекта. Результатом данного семинара было соглашение принять новое название для этой области, предложенное Маккарти, -- искусственный интеллект.

3. Ранний энтузиазм, большие ожидания (период с 1952 года по 1969 год)

Первые годы развития искусственного интеллекта были полны успехов, хотя и достаточно скромных. Если учесть, какими примитивными были в то время компьютеры и тот факт, что компьютеры рассматривались как устройства, способные выполнять только арифметические действия, можно лишь удивляться тому, как удалось заставить компьютер выполнять операции, хоть немного напоминающие разумные.
За первыми успешными разработками Ньюэлла и Саймона последовало создание программы общего решателя задач (General Problem Solver-- GPS). Программа GPS была самой первой программой, в которой был воплощен подход к "организации мышления по такому же принципу, как и у человека".
Начиная с 1952 года, Артур Самюэл написал ряд программ для игры в шашки. Он опроверг утверждение, что компьютеры способны выполнять только то, чему их учили: одна из его программ быстро научилась играть лучше, чем ее создатель. Эта программа была продемонстрирована по телевидению в феврале 1956 года и произвела очень сильное впечатление на зрителей.
Джон Маккарти перешел из Дартмутского университета в Массачу-сетский технологический институт и здесь в течение одного истори-ческого 1958 года внес три крайне важных вклада в развитие искусственного интеллекта. Он

привел определение нового языка высокого уровня Lisp, которому суждено было стать доминирующим языком программирования для искусственного интеллекта.
Разработав это язык, Маккарти получил необходимый для него инструмент, но доступ к ограниченным и дорогостоящим компьютерным ресурсам продолжал оставаться серьезной проблемой. В связи с этим он совместно с другими сотрудниками Массачусетсского технологического института изобрел режим разделения времени. В том же 1958 году Маккарти опубликовал статью под названием Programs with Common Sense, в которой он описал гипотетическую программу Advice Taker, которая может рассматриваться как первая полная система искусственного интеллекта. Замечательной особенностью указанной статьи является то, что значительная ее часть не потеряла своего значения и в наши дни.

4. Столкновение с реальностью (период с 1966 года по 1973 год)

С самого начала исследователи искусственного интеллекта не отличались сдержанностью, высказывая прогнозы в отношении своих будущих успехов. Например, часто цитировалась приведенное ниже предсказание Герберта Саймона, опубликованное им в 1957 году.
«Я не ставлю перед собой задачу удивить или шокировать вас, но проще всего я могу подвести итог, сказав, что теперь мы живем в таком мире, где машины могут думать, учиться и создавать. Более того, их способность выполнять эти действия будет продолжать расти до тех пор, пока (в обозримом будущем) круг проблем, с которыми смогут справиться машины, будет сопоставим с тем кругом проблем, где до сих пор был нужен человеческий мозг.»
Такие выражения, как "обозримое будущее", могут интерпретироваться поразному, но Саймон сделал также более конкретный прогноз, что через десять лет компьютер станет чемпионом мира по шахматам и что машиной будут доказаны все важные математические теоремы. Эти предсказания сбылись не через десять лет, а через сорок. Чрезмерный оптимизм Саймона был обусловлен тем, что первые системы искусственного интеллекта демонстрировали многообещающую производительность, хотя и на простых примерах. Но почти во всех случаях эти ранние системы терпели сокрушительное поражение, сталкиваясь с более широким кругом проблем или с более трудными проблемами.
Сложности первого рода были связаны с тем, что основная часть ранних программ не содержала знаний или имела лишь небольшой объем знаний о своей предметной области.

Сложности второго рода были связаны с неразрешимостью многих проблем, решение которых пытались найти с помощью искусственного интеллекта.
Сложности третьего рода возникли в связи с некоторыми фундаментальными ограничениями базовых структур, которые использовались для выработки интеллектуального поведения.

5. Системы, основанные на знаниях (период с 1969 года по 1979 год)

Основной подход к решению задач, сформированный в течение первого десятилетия исследований в области искусственного интеллекта, представлял собой механизм поиска общего назначения, с помощью которого предпринимались попытки связать в единую цепочку элементарные этапы проведения рассуждений для формирования полных решений. Подобные подходы получили название слабых методов, поскольку они не позволяли увеличить масштабы своего применения до уровня более крупных или более сложных экземпляров задач, несмотря на то, что были общими.
Альтернативным по сравнению со слабыми методами стал подход, предусматривающий использование более содержательных знаний, относящихся к проблемной области, который позволяет создавать более длинные цепочки шагов логического вывода и дает возможность проще справиться с теми проблемными ситуациями, которые обычно возникают в специализированных областях знаний.
Одним из первых примеров реализации такого подхода была программа Dendral. Значение программы Dendral состояло в том, что это была первая, успешно созданная экспертная система, основанная на широком использовании знаний.
Ее способность справляться с поставленными задачами была обусловлена применением большого количества правил специального назначения. В более поздних системах также широко применялся основной принцип подхода, реализованного Маккарти в программе Advice Taker, -- четкое отделение знаний (в форме правил) от компонента, обеспечивающего проведение рассуждений.

6. Превращение искусственного интеллекта в индустрию (период с 1980 года по настоящее время)

В индустрии искусственного интеллекта произошел бурный рост, начиная с нескольких миллионов долларов в 1980 году и заканчивая миллиардами долларов в 1988 году. Однако вскоре после этого наступил период, получивший название "зимы искусственного интеллекта", в течение которого постра-

дали многие компании, поскольку не сумели выполнить своих заманчивых обещаний.

7. Возвращение к нейронным сетям (период с 1986 года по настоящее время)

Хотя основная часть специалистов по компьютерным наукам прекратила исследования в области нейронных сетей в конце 1970-х годов, работу в этой области продолжили специалисты из других научных направлений. Психологи, включая Дэвида Румельхарта и Джефа Хинтона, продолжали исследовать модели памяти на основе нейронных сетей.

8. Превращение искусственного интеллекта в науку (период с 1987 года по настоящее время)

В последние годы произошла буквально революция, как в содержании, так и в методологии работ в области искусственного интеллекта. С точки зрения методологии искусственный интеллект наконец-то твердо перешел на научные методы.

9. Появление подхода, основанного на использовании интеллектуальных агентов

Наиболее широко известным примером создания полной архитектуры агента является работа Аллена Ньюэлла, Джона Лэрда и Пола Розенблума над проектом Soar. Для того чтобы проще было разобраться в работе агентов, внедренных в реальную среду с непрерывным потоком сенсорных входных данных, были применены так называемые *ситуационные движения*. Одним из наиболее важных примеров среды для интеллектуальных агентов может служить Internet. Технологии искусственного интеллекта легли в основу многих инструментальных средств Internet, таких как машины поиска, системы, предназначенные для выработки рекомендаций, и системы создания Web-узлов.

Сторонники данного подхода считают, что феномены человеческого поведения, его способность к обучению и адаптации есть следствие именно биологической структуры и особенностей её функционирования. Отличается от понимания искусственного интеллекта по Джону Маккарти, когда исходят из положения о том, что искусственные системы не обязаны повторять в своей структуре и функционировании структуру и протекающие в ней процессы, присущие биологическим системам.

Одним из следствий попыток создания полных агентов стало понимание того, что ранее изолированные подобласти искусственного интеллекта могут потребовать определенной реорганизации, когда возникнет необходимость

снова связать воедино накопленные в них результаты. Поэтому системы проведения рассуждений и планирования должны быть приспособленными к работе в условиях неопределенности. Вторым важным следствием изменения взглядов на роль агентов является то, что исследования в области искусственного интеллекта теперь необходимо проводить в более тесном контакте с другими областями, такими как теория управления и экономика, которые также имеют дело с агентами.

История развития искусственного интеллекта в СССР и России

В 1954 г. в МГУ начал свою работу семинар «Автоматы и мышление» под руководством академика Ляпунова А. А. (1911-1973), одного из основателей российской кибернетики. В этом семинаре принимали участие физиологи, лингвисты, психологи, математики. Принято считать, что именно в это время родился искусственный интеллект в России. Как и за рубежом, выделились два основных направления — нейрокибернетики и кибернетики «черного ящика».

Среди советских учёных искусственный интеллект был главной областью научной деятельности Д. А. Поспелова. Здесь научные интересы Д. А. Поспелова связаны с моделированием поведения человека, формализацией рассуждений, общими проблемами моделирования жизненных процессов в естественных и искусственных системах. В частности, Д. А. Поспеловым был впервые в мире разработан подход к принятию решений, опирающийся на семиотические (логико-лингвистические) модели, который послужил теоретической основой ситуационного управления большими системами. По истории также можно проследить интерес других советских учёных к кибернетике.

В СССР работы в области искусственного интеллекта начались в 1960-х годах. В Московском университете и Академии наук был выполнен ряд пионерских исследований, возглавленных Вениамином Пушкиным и Д. А. Поспеловым.

С начала 1960-х М. Л. Цетлин с коллегами разрабатывали вопросы, связанные с обучением конечных автоматов.

В 1964 году была опубликована работа ленинградского логика Сергея Маслова «Обратный метод установления выводимости в классическом исчислении предикатов», в которой впервые предлагался метод автоматического поиска доказательства теорем в исчислении предикатов.

В 1966 году В. Ф. Турчиным был разработан язык рекурсивных функций Рефал.

Большой вклад в становление российской школы ИИ внесли выдающиеся ученые Цетлин М.Л., Пушкин В. Н., Гаврилов М. А, чьи ученики и явились пионерами этой науки в России (например, знаменитая Гавриловская школа).
При том, что отношение к новым наукам в советской России всегда было настороженное, наука с таким «вызывающим» названием тоже не избежала этой участи и была встречена в Академии наук в штыки. К счастью, даже среди членов Академии наук СССР нашлись люди, не испугавшиеся столь необычного словосочетания в качестве названия научного направления. Двое из них сыграли огромную роль в борьбе за признание ИИ в нашей стране. Это были академики А. И. Берг и Г. С. Поспелов.
В 1980–1990 гг. проводятся активные исследования в области представления знаний, разрабатываются языки представления знаний, экспертные системы (более 300).
В 1988 г. создается АИИ — Ассоциация искусственного интеллекта. Ее членами являются более 300 исследователей. Президентом Ассоциации единогласно избирается Д. А. Поспелов, выдающийся ученый, чей вклад в развитие ИИ в России трудно переоценить.
«Официальная» история искусственного интеллекта в России началась в январе 2019 года, когда президент страны Владимир Путин дал поручение правительству разработать подходы к национальной стратегии развития искусственного интеллекта (ИИ) и представить соответствующие предложения. В середине октября президент Путин подписал указ, которым утвердил стратегию развития ИИ в стране до 2030 года. Согласно документу, Россия должна занять одну из ведущих позиций в мире в этой сфере, так как лидер в области ИИ станет, по мнению российского президента, «властелином мира».
Национальная стратегия развития искусственного интеллекта
30 мая 2019 г. на совещании по развитию цифровой экономики под председательством В. В. Путина было принято решение о подготовке национальной стратегии по искусственному интеллекту.
Затраты развитых стран, особенно США, Китая, а также Евросоюза, на технологии искусственного интеллекта растут ударными темпами. Между тем, вплоть до последнего времени Россия оставалась едва ли не последней из крупных стран, не имеющих собственной стратегии развития технологий ИИ. Ситуация начала меняться лишь в конце 2019 года, когда была утверждена «***Национальная стратегия развития искусственного интеллекта на период до 2030 года***».

Приоритеты развития технологий ИИ в России

К приоритетам развития ИИ в России относится:

1. Ускорение технологического развития РФ за счет увеличения количества организаций, осуществляющих технологические инновации, до 50% от их общего числа;
2. Обеспечение ускоренного внедрения цифровых технологий в экономику и социальную сферу;
3. Создание в базовых отраслях экономики, прежде всего в обрабатывающей промышленности и агропромышленном комплексе, высокопроизводительного экспортоориентированного сектора, развивающегося на основе современных технологий и обеспеченного высококвалифицированными кадрами.

Задачи развития технологий ИИ в России

Стратегия также обозначает ряд задач, которые необходимо решить для успешного развития технологий ИИ в России, в числе которых:

- Создание высокопроизводительных рабочих мест;
- Обеспечение конкурентоспособных условий труда для специалистов в сфере ИИ;
- Привлечение специалистов из-за рубежа;
- Поддержка экспорта продуктов и услуг, созданных с использованием ИИ;
- Создание стимулов для развития корпоративной науки и исследований;
- Формирование комплексной системы безопасности при создании, развитии, внедрении и использовании технологий ИИ.

Первоначально на развитие проектов в области искусственного интеллекта планировалось выделить 125 млрд рублей, из которых почти 90 млрд из бюджета. Однако, существенные коррективы в планы российского правительства внесла эпидемия коронавируса, в результате чего размер федерального финансирования проектов, связанных с ИИ, заметно снизился. Как следствие на **2021–2024** гг. на развитие ИИ в России планируется потратить лишь 29,4 млрд рублей бюджетных средств и 6,9 млрд из внебюджетных источников.

Хотя в Стратегии заявлено, что «Российская Федерация обладает существенным потенциалом для того, чтобы стать одним из международных лидеров в развитии и использовании технологий искусственного интеллекта», достижение этой цели в краткосрочной и среднесрочной перспективе выглядит маловероятным. Ключевым препятствием на пути России в число мировых лидеров в сфере ИИ может стать недостаточное финансирование, особенно на фоне Китая и США.

Национальная стратегия определяет две ключевые точки развития ИИ в России — **2024** и **2030** годы. Предполагается, что к первой дате страна значительно улучшит позиции в этой сфере, а к 2030 году ликвидирует отставание от развитых стран и добьется мирового лидерства в отдельных направлениях, связанных с ИИ. Внедрять технологии ИИ российские власти планируют в том числе через государственные национальные проекты.
Между тем искусственный интеллект уже сейчас используется в России при решении самых разных задач. Например, Сбербанк применяет его при выдаче кредитов, Яндекс – в развитии беспилотного транспорта и в голосовом помощнике «Алиса», Mail.ru – в коммуникациях, Ростех – в сфере производства, а МВД – при распознавании лиц для обеспечения безопасности на улицах Москвы.

Инвестиции США и Китая

Согласно бюджетным документам, федеральное правительство США планировало инвестировать около 4,9 млрд долларов в исследования и разработки в области искусственного интеллекта и машинного обучения в 2020 финансовом году (в одном году, а не на 4 года).
А вот Китай утвердил свою масштабную стратегию развития ИИ ещё в 2017 году. Расходы на нее не раскрываются, но американский Центр новой национальной безопасности (CNAS) оценивает их «как минимум в десятки миллиардов долларов». Одни лишь администрации городов Тяньцзиня и Шанхая объявляли о создании инвестфондов для развития AI по 100 млрд юаней ($14,5 млрд) каждый.

Лекция 2. Подходы к пониманию проблемы

Два альтернативных подхода

Все известные попытки создания искусственного интеллекта направлены на преодоление главной проблемы: «Как не только понять ход мыслительных процессов и природу интеллекта человека, но и воплотить все эти механизмы в одной интеллектуальной сущности?». Для решения этой задачи на протяжении многих лет существовало два альтернативных стратегических подхода к разработке ИИ :

- **нейрокибернетика и**
- **кибернетика «черного ящика».**

Эти подходы появились в середине 20 века практически сразу после выделения искусственного интеллекта в отдельную область науки.

Нейрокибернетика

Основную идею этого направления можно сформулировать следующим образом: единственный объект, способный мыслить, – это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру.

Таким образом, нейрокибернетика ориентирована на программно-аппаратное моделирование структур, подобных структуре мозга. Физиологами давно установлено, что основой человеческого мозга является большое количество (до 10^{21}) связанных между собой и взаимодействующих нервных клеток — нейронов. Поэтому усилия нейрокибернетики были сосредоточены на создании элементов, аналогичных нейронам, и их объединении в функционирующие системы. Эти системы принято называть **нейронными сетями**, или нейросетями.

Нейрокибернетика — научное направление, изучающее основные закономерности организации и функционирования нейронов и нейронных образований. Основным методом нейрокибернетики является математическое моделирование, при этом данные физиологического эксперимента используются в качестве исходного материала для создания моделей.

Нейрокибернетика имеет широкий спектр приложений — от медико-биологических разработок до создания специализированных **нейрокомпьютеров**.

Первые нейросети были созданы Розенблаттом и Мак-Каллоком в 1956–1965г. Это были попытки создать системы, моделирующие человеческий глаз и его взаимодействие с мозгом. Устройство, созданное ими тогда, получило название **персептрона** (perceptron). Оно умело различать буквы алфавита, но было чувствительно к их написанию. Постепенно в 70–80 годах количество работ по этому направлению искусственного интеллекта стало снижаться. Слишком неутешительны были первые результаты. Авторы объясняли неудачи малой памятью и низким быстродействием существующих в то время компьютеров.

Кибернетика «черного ящика»

В основу этого подхода был положен принцип, противоположный нейрокибернетике: не имеет значения, как устроено «мыслящее» устройство. Главное, чтобы на заданные входные воздействия оно реагировало так же, как человеческий мозг.

В рамках этого направления созданы модели и алгоритмы, которые при решении интеллектуальных задач дают результаты сравнимые с результатами, которые получает человек. При этом модели и алгоритмы могут, как воспро-

изводить процесс принятия решения человеком, так и совершенно отличаться от него.

Модель лабиринтного поиска (конец 50-х годов)

Этот подход представляет задачу как некоторое пространство состояний в форме графа, и в этом графе проводится поиск оптимального пути от входных данных к результирующим. Была проделана большая работа по разработке этой модели, но для решения практических задач эта идея не нашла широкого применения. В первых учебниках по искусственному интеллекту [Хант, 1986; Эндрю, 1985] описаны эти программы — они играют в игру «15», собирают «Ханойскую башню», играют в шашки и шахматы.

Эпоха эвристического программирования (начало 60-х)

Эвристический алгоритм (**эвристика**) — алгоритм решения задачи, включающий практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения поставленной задачи. Позволяет ускорить решение задачи в тех случаях, когда точное решение не может быть найдено. **Эвристический** алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев.
Эвристическое программирование — разработка стратегии действий на основе известных, заранее заданных эвристик.

Использование методов математической логики (1963-1970)

Робинсон разработал метод резолюций, который позволяет автоматически до-казывать теоремы при наличии набора исходных аксиом. Примерно в это же время выдающийся отечественный математик Ю. С. Маслов предложил так называемый обратный вывод, впоследствии названный его именем, решающий аналогичную задачу другим способом. На основе метода резолюций француз Альбер Кольмероэ в 1973 г. создает язык логического программирования **Пролог**. Большой резонанс имела программа «Логик-теоретик», созданная Нъюэлом, Саймоном и Шоу, которая доказывала школьные теоремы. Однако большинство реальных задач не сводится к набору аксиом, и человек, решая производственные задачи, не использует классическую логику, поэтому логические модели при всех своих преимуществах имеют существенные ограничения по классам решаемых задач.

Системы, основанные на знаниях, или экспертные системы

К середине 1970-х на смену поискам универсального алгоритма мышления пришла идея моделировать конкретные знания специалистов-экспертов. В США появились первые коммерческие системы, основанные на знаниях, или экспертные системы (ЭС). Стал применяться новый подход к решению задач искусственного интеллекта — представление знаний. Созданы MYCIN и DENDRAL, ставшие уже классическими, две первые экспертные системы для медицины и химии.

Начиная с середины 1980-х годов, повсеместно происходит коммерциализация искусственного интеллекта. Растут ежегодные капиталовложения, создаются промышленные экспертные системы. Растет интерес к самообучающимся системам. Издаются десятки научных журналов, ежегодно собираются международные и национальные конференции по различным направлениям ИИ. Искусственный интеллект становится одной из наиболее перспективных и престижных областей информатики (computer science).

Эволюционный подход

В настоящий момент дополнение к двум выше указанным стратегическим подходам появился ещё один, который называется ***эволюционный***. Данный подход связан с моделированием процесса эволюции человеческого мозга, что позволяет разбить задачу создания целостного интеллекта на подпроблемы. Как известно, различные виды животных в процессе эволюции получали нервные системы разной сложности. Идея заключается в том, чтобы развить процесс создания искусственного интеллекта постепенно, разбираясь сначала в том, какие механизмы отвечают за тот или иной скачок в интеллектуальности животного, а потом уже использовать полученные результаты для создания искусственного интеллекта человека.

Тест Тьюринга и интуитивный подход

Эмпирический тест был предложен Аланом Тьюрингом в статье «Вычислительные машины и разум», опубликованной в 1950 году в философском журнале «Mind». Целью данного теста является определение возможности искусственного мышления, близкого к человеческому.

Стандартная интерпретация этого теста звучит следующим образом: «Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор».

Все участники теста не видят друг друга. Если судья не может сказать определённо, кто из собеседников является человеком, то считается, что машина прошла тест. Чтобы протестировать именно интеллект машины, а не её возможность распознавать устную речь, беседа ведётся в режиме «только текст», например, с помощью клавиатуры и экрана (компьютера-посредника). Переписка должна производиться через контролируемые промежутки времени, чтобы судья не мог делать заключения, исходя из скорости ответов. Во времена Тьюринга компьютеры реагировали медленнее человека. Сейчас это правило тоже необходимо, потому что они реагируют гораздо быстрее, чем человек.

Почти все разработанные программы и близко не подошли к прохождению теста. Хотя такие программы, как Элиза (ELIZA), иногда заставляли людей верить, что они говорят с человеком, как, например, в неформальном эксперименте, названном AOLiza, но эти случаи нельзя считать корректным прохождением теста Тьюринга по целому ряду причин.

Машина может избежать лишних вопросов, например, притворившись параноиком, подростком или иностранцем с недостаточным знанием местного языка. Победитель одного из последних конкурсов, организованных по принципу теста Тьюринга, — бот по имени Женя Густман — сумел объединить все три приема, притворяясь тринадцатилетним мальчишкой из Одессы. Конкурс был организован в 2014 году университетом Рединга (Великобритания).

Был создан группой из трёх программистов: Владимира Веселова (родом из России, живёт в Нью-Джерси), Евгения Демченко (родом из Украины) и Сергея Уласеня (родом из России). Разработка программы была начата в Санкт-Петербурге в 2001 году. Чтобы характер и знания Густмана казались более правдоподобными, он представляется пользователям 13-летним мальчиком из Одессы.

Густман с момента его создания принимал участие в ряде соревнований на прохождение теста Тьюринга и несколько раз занимал второе место в соревновании на премию Лёбнера. В июне 2012 года Густман выиграл соревнование в честь 100-летия Алана Тьюринга, сумев убедить 29 % судей, что он человек. 7 июня 2014 года, на конкурсе, посвященном 60-летию со дня смерти Тьюринга, Густман убедил 33 % судей, что он человек, и, по словам Кевина Уорика, стал первым в истории компьютером, прошедшим тест Тьюринга.

Цель искусственного интеллекта, очевидно, заключается не в том, чтобы обмануть людей, а в том, чтобы достигнуть понимания мира и научиться действовать в нем способами, сравнимыми по своей полезности, эффективности

и надежности с человеческой деятельностью. Специалисты отмечают, что теста Тьюринга для этого недостаточно.
Более корректное тестирование подразумевает широкий спектр задач, таких как понимание человеческого языка, способность делать выводы о физическом и умственном состоянии людей, анализ видео на YouTube, владение элементарными научными знаниями и способность к автономному выполнению роботизированных операций.

Гипотеза Ньюэлла — Саймона

Гипотеза Ньюэлла — Саймона (гипотеза о физической символьной системе) — предположение, сформулированное Алленом Ньюэллом и Гербертом Саймоном в 1976 году, согласно которому «физическая символьная система имеет необходимые и достаточные средства для произведения основных интеллектуальных операций» (под интеллектуальными операциями подразумеваются действия сильного искусственного интеллекта).
Другими словами, предполагается, что без символьных вычислений невозможно выполнять осмысленные действия, а способность выполнять символьные вычисления вполне достаточна для того, чтобы стать способным выполнять осмысленные действия.
Таким образом, если предполагать, что животное или человек или машина действуют осмысленно, то значит, они каким-то образом выполняют символьные вычисления. И наоборот, так как компьютер способен к подобным вычислениям, то на его основе может быть создан искусственный интеллект.
Основанием для гипотезы стало успешное применение созданной Ньюэллом и Саймоном программы — универсального решателя задач — для моделирования рассуждений человека.

Символьный подход

Исторически символьный подход был первым в эпоху цифровых машин, так как именно после создания Лисп, первого языка символьных вычислений, у его автора возникла уверенность в возможности практически приступить к реализации этими средствами интеллекта. Символьный подход позволяет оперировать слабоформализованными представлениями и их смыслами.
Успешность и эффективность решения новых задач зависит от умения выделять только существенную информацию, что требует гибкости в методах абстрагирования. Тогда как обычная программа устанавливает один свой способ интерпретации данных, из-за чего её работа и выглядит предвзятой и чисто механической. Интеллектуальную задачу в этом случае решает только человек, аналитик или программист, не умея доверить этого машине.

В результате создается единственная модель абстрагирования, система конструктивных сущностей и алгоритмов. А гибкость и универсальность выливается в значительные затраты ресурсов для не типичных задач, то есть система от интеллекта возвращается к грубой силе.
Основная особенность символьных вычислений — создание новых правил в процессе выполнения программы. Тогда как возможности не интеллектуальных систем завершаются как раз перед способностью хотя бы обозначать вновь возникающие трудности. Тем более эти трудности не решаются и наконец компьютер не совершенствует такие способности самостоятельно.
Недостатком символьного подхода является то, что такие открытые возможности воспринимаются не подготовленными людьми как отсутствие инструментов. Эту, скорее культурную проблему, отчасти решает логическое программирование.

Логический подход

Логический подход к созданию систем искусственного интеллекта основан на моделировании рассуждений. Теоретической основой служит логика.
Логический подход может быть проиллюстрирован применением для этих целей языка и системы логического программирования **Пролог**. Программы, записанные на языке Пролог, представляют наборы фактов и правил логического вывода без жесткого задания алгоритма как последовательности действий, приводящих к необходимому результату.

Агентно-ориентированный подход

Последний подход, развиваемый с начала 1990-х годов, называется агентно-ориентированным подходом, или подходом, основанным на использовании интеллектуальных (рациональных) агентов. Согласно этому подходу, интеллект — это вычислительная часть (грубо говоря, планирование) способности достигать поставленных перед интеллектуальной машиной целей. Сама такая машина будет интеллектуальным агентом, воспринимающим окружающий его мир с помощью датчиков, и способной воздействовать на объекты в окружающей среде с помощью исполнительных механизмов.
Этот подход акцентирует внимание на тех методах и алгоритмах, которые помогут интеллектуальному агенту выживать в окружающей среде при выполнении его задачи. Так, здесь значительно тщательнее изучаются алгоритмы поиска пути и принятия решений.
Точно так же, как объектно-ориентированное программирование сдвинуло парадигму с написания процедур к созданию объектов, рациональное про-

граммирование сдвинуло парадигму с создания информационных объектов к созданию мотивированных агентов.

Определение парадигмы, данное автором

Агентом является всё, что может рассматриваться как воспринимающее свою среду с помощью **датчиков** и воздействующее на эту **среду** с помощью **исполнительных механизмов**.

Понятие агента, в отличие от простого объекта, наделяется рядом ментальных конструкций, таких как вера, обязанности и способности. Поэтому в языке программирования будут появляться различные ментальные категории, а семантика программирования будет связана с семантикой ментальных конструкций.

Гибридный подход

Гибридный подход предполагает, что только синергийная комбинация нейронных и символьных моделей достигает полного спектра когнитивных и вычислительных возможностей. Например, экспертные правила умозаключений могут генерироваться нейронными сетями, а порождающие правила получают с помощью статистического обучения. Сторонники данного подхода считают, что гибридные информационные системы будут значительно более сильными, чем сумма различных концепций по отдельности.

Под гибридной интеллектуальной системой (ГиИС) принято понимать систему, в которой для решения задачи используется более одного метода имитации интеллектуальной деятельности человека. Таким образом ГиИС — это совокупность:

- *аналитических моделей*
- *экспертных систем*
- *искусственных нейронных сетей*
- *нечётких систем*
- *генетических алгоритмов*
- *имитационных статистических моделей*

Междисциплинарное направление «гибридные интеллектуальные системы» объединяет ученых и специалистов, исследующих применимость не одного, а нескольких методов, как правило, из различных классов, к решению задач управления и проектирования.

Современный искусственный интеллект

Можно выделить два направления развития ИИ:

- решение проблем, связанных с приближением специализированных систем ИИ к возможностям человека, и их интеграции, которая реализована природой человека (см. Усиление интеллекта);
- создание искусственного разума, представляющего интеграцию уже созданных систем ИИ в единую систему, способную решать проблемы человечества (см. Сильный и слабый искусственный интеллект).

Усиление интеллекта

Усиление интеллекта (УИ) (Intelligence amplification, Cognitive augmentation, Machine augmented intelligence) — совокупность средств и методов, обеспечивающих максимально возможную производительность интеллекта человека; эффективное использование информационных технологий для усиления человеческого интеллекта. Теория УИ активно разрабатывалась в 1950-е и 1960-е годы пионерами кибернетики и информатики.
УИ иногда противопоставляется ИИ, то есть, проекту построения человекоподобного интеллекта в форме автономной технической системы, такой как компьютер или робот. ИИ столкнулся со многими фундаментальными препятствиями, практическими и теоретическими, с которыми вряд ли столкнется УИ, так как УИ требует технологию просто как дополнительную поддержку для автономного интеллекта, который уже существует.

Слабый и сильный ИИ

Сильный и слабый искусственные интеллекты — гипотеза в философии искусственного интеллекта, согласно которой некоторые формы искусственного интеллекта могут действительно обосновывать и решать проблемы.

- **теория сильного искусственного интеллекта** предполагает, что компьютеры могут приобрести способность мыслить и осознавать себя как отдельную личность (в частности, понимать собственные мысли), хотя и не обязательно, что их мыслительный процесс будет подобен человеческому.
- **теория слабого искусственного интеллекта** отвергает такую возможность. Сторонники слабого ИИ предпочитают рассматривать программы лишь как инструмент, позволяющий решать те или иные задачи, которые не требуют полного спектра человеческих познавательных способностей.

  Для создания сильного искусственного интеллекта необходимо, чтобы система не работала по какому-то заданному алгоритму, а проявляла признаки интеллектуальности и осознанности, свойственные человеку.

При общении с интеллектуальным устройством посредством анонимного канала человек не должен понять, что он общается с машиной.

На сегодняшний день процесс создания устройств сильного искусственного интеллекта находится лишь в стадии становления. Это подтверждает неудачный пример чат-бота Тэй от компании Майкрософт. Уже через день общения пользователями сети Твиттер Тэй стал агрессивен, восхваляя Гитлера и проявляя признаки расизма. Этот пример ярко иллюстрирует то, что созданному боту не хватает элемента осознанности человека.

По мнению ряда учёных, наиболее перспективным направлением создания сильного искусственного интеллекта, является эволюционный подход. Если говорить о развитии слабого искусственного интеллекта, то данное направление уже охватило целый спектр научных областей, начиная от задач общего характера, таких как обучение и восприятие, и заканчивая узкоспециализированными сферами, связанными с игрой в шахматы, доказательством теорем, написании литературных произведений, управлении автомобилем и диагностикой заболеваний.

Формы ИИ

- **Узконаправленный (слабый) искусственный интеллект (УИИ).** В последние годы интерес к системам искусственные интеллекта значительно увеличился. При этом уровень развития современных технологий позволяет создавать системы, лишь добавляющие интеллектуальности в нашу жизнь (система автопилотирования, робот— пылесос, стиральная машина с нечеткой логикой и др.), а не воспроизводящие интеллект человека полностью. Разработки такого класса входят в группу «слабого» искусственного интеллекта. УИИ специализируется в одной области. Среди таких ИИ есть те, кто может обыграть чемпиона мира по шахматам, но на этом все. Есть такой, который может предложить лучший способ хранения данных на жестком диске, и все.
- **Общий (сильный) искусственный интеллект**. Иногда также называют ИИ человеческого уровня. ОИИ относят к компьютеру, который умен, как человек — машина, которая способна выполнять любое интеллектуальное действие, присущее человеку. Создать ОИИ намного сложнее, чем УИИ, и мы пока до этого не дошли. Профессор Линда Готтфредсон описывает интеллект как «в общем смысле психический потенциал, который, наряду с другими вещами, включает способность рассуждать, планировать, решать проблемы, мыслить абстрактно, понимать сложные идеи, быстро учиться и извлекать опыт». ОИИ должен уметь делать все это так же просто, как делаете вы.

- **Искусственный сверхинтеллект (ИСИ).** Оксфордский философ и теоретик ИИ Ник Бостром определяет сверхинтеллект как «интеллект, который гораздо умнее лучших человеческих умов в практически любой сфере, включая научное творчество, общую мудрость и социальные навыки». Искусственный сверхинтеллект включает в себя как компьютер, который немного умнее человека, так и тот, который в триллионы раз умнее в любом направлении. ИСИ и есть причина того, что растет интерес к ИИ, а также того, что в таких обсуждениях часто появляются слова «вымирание» и «бессмертие».

Известные ИИ-системы

В настоящий момент в области искусственного интеллекта наблюдается вовлечение многих предметных областей, имеющих скорее практическое отношение к ИИ, а не фундаментальное. Многие подходы были опробованы, но к возникновению искусственного разума ни одна исследовательская группа пока так и не подошла. Ниже представлены лишь некоторые наиболее известные разработки в области ИИ.

Deep Blue — разработанный IBM шахматный суперкомпьютер, победил чемпиона мира по шахматам.

AlphaGo — разработанный Google DeepMind, выиграл матч в Го у корейским профессионала 9 дана Ли Седоля.

Watson — перспективная разработка IBM, способная воспринимать человеческую речь и производить вероятностный поиск, с применением большого количества алгоритмов. Для демонстрации работы Watson принял участие в американской игре «Jeopardy!», аналога «Своей игры» в России, где системе удалось выиграть в обеих играх.

MYCIN — одна из ранних экспертных систем, которая могла диагностировать небольшой набор заболеваний, причем часто так же точно, как и доктора.

20Q — проект, основанный на идеях ИИ, по мотивам классической игры «20 вопросов». Стал очень популярен после появления в Интернете на сайте 20q.net.

Распознавание речи. Такие системы как *ViaVoice* способны обслуживать потребителей.

Лекция 3. Самые популярные сферы развития ИИ

1. Моделирование рассуждений
2. Машинное обучение
3. Обработка естественного языка

4. Представление и использование знаний
5. Интеллектуальная робототехника
6. Биологическое моделирование искусственного интеллекта
7. Другие области исследований

1. Моделирование рассуждений

Анализируя историю ИИ, можно выделить такое обширное направление как моделирование рассуждений. Долгие годы развитие этой науки двигалось именно по этому пути, и теперь это одна из самых развитых областей в современном ИИ. Моделирование рассуждений подразумевает создание **символьных систем**, на входе которых поставлена некая задача, а на выходе требуется её решение. Как правило, предлагаемая задача уже формализована, то есть переведена в математическую форму, но либо не имеет алгоритма решения, либо он слишком сложен, трудоёмок и т. п. В это направление входят: *доказательство теорем, принятие решений и теория игр, планирование и диспетчеризация, прогнозирование*.

Рассуждение - один из важнейших видов мыслительной деятельности человека, в результате которого он формулирует на основе некоторых предложений, высказываний, суждений новые предложения, высказывания, суждения. Действительный механизм рассуждений человека остается пока недостаточно исследованным. Человеческим рассуждениям присущи: неформальность, нечеткость, нелогичность, широкое использование образов, эмоций и чувств, что делает чрезвычайно трудными их исследование и моделирование. К настоящему времени лучше всего изучены логические рассуждения и разработано много механизмов дедуктивных выводов, реализованных в различных интеллектуальных системах, основанных на представлении знаний с помощью логики предикатов 1-го порядка.

Автоматическое доказательство

Автоматическое доказательство (Automated Theorem Proving, ATP, а также Automated deduction) — доказательство, реализованное программно. В основе лежит аппарат математической логики. Используются идеи теории искусственного интеллекта. Процесс доказательства основывается на логике высказываний и логике предикатов.

В силу неразрешимости даже достаточно простых теорий практическое применение имеет лишь полуавтоматическое человеко-машинное доказательство. К тому же после полной автоматизации доказательство называют уже вычислением. Полностью автоматической может быть лишь проверка доказательства теорий посложнее (если его для этого подготовить).

В настоящее время автоматическое доказательство теорем в промышленности применяется в основном при разработке и верификации интегральных схем и программного обеспечения. После того, как была обнаружена ошибка деления в процессорах Пентиум, сложные модули операций с плавающей запятой современных микропроцессоров разрабатываются с особой тщательностью. В новых процессорах AMD, Intel и других фирм автоматическое доказательство теорем используется для проверки того, что деление и другие операции выполняются корректно.
Корпорация Microsoft использует автоматический доказатель теорем Z3 для верификации кода операционной системы Windows 7 и других программных продуктов

Теория принятия решений

Теория принятия решений — область исследования, вовлекающая понятия и методы математики, статистики, экономики, менеджмента и психологии с целью изучения закономерностей выбора людьми путей решения проблем и задач, а также способов достижения желаемого результата.
Различают ***нормативную теорию***, которая описывает рациональный процесс принятия решения и ***дескриптивную теорию***, описывающую практику принятия решений.

Процесс решения проблем и задач

Рациональный процесс решения проблем и задач включает следующие этапы, при необходимости, выполняемые одновременно, параллельно, итерационно, с возвратом к исполнению предыдущих этапов:

- Ситуационный анализ (анализ проблемной ситуации);
- Идентификация проблемы и постановка цели;
- Поиск необходимой информации;
- Формирование множества возможных решений;
- Формирование критериев оценки решений;
- Разработка индикаторов и критериев для мониторинга реализации решений;
- Проведение оценки решений;
- Выбор наилучшего решения;
- Планирование;
- Реализация;
- Мониторинг реализации;
- Оценка результата.

При этом выполнение всего процесса и этапов осуществляется рационально обоснованным способом.

Теория игр

Теория игр — математический метод изучения оптимальных стратегий в играх. Под игрой понимается процесс, в котором участвуют две и более сторон, ведущие борьбу за реализацию своих интересов. Каждая из сторон имеет свою цель и использует некоторую стратегию, которая может вести к выигрышу или проигрышу — в зависимости от поведения других игроков. Теория игр помогает выбрать лучшие стратегии с учётом представлений о других участниках, их ресурсах и их возможных поступках.

Теория игр — раздел прикладной математики, точнее исследования операций. Чаще всего методы теории игр находят применение в международных отношениях, экономике, чуть реже в других общественных науках — социологии, политологии, психологии, этике, юриспруденции и других. Начиная с 1970-х годов, её взяли на вооружение биологи для исследования поведения животных и теории эволюции. Очень важное значение она имеет для искусственного интеллекта и кибернетики, особенно с проявлением интереса к интеллектуальным агентам.

Автоматическое планирование и диспетчеризация

Автоматическое планирование и диспетчеризация (*Automated planning and scheduling, APS*) — область задач искусственного интеллекта, касающаяся выполнения стратегии или последовательности действий, обычно для интеллектуальных агентов, автономных роботов и беспилотных аппаратов. В отличие от классических проблем управления и классификации, решения задач данной области комплексны, неизвестны и должны разрабатываться и оптимизироваться в многомерном пространстве.

Решения в основном используют процессы проб и ошибок присущие области искусственного интеллекта, такие как динамическое программирование, обучение с подкреплением и комбинаторная оптимизация.

У типичного планировщика три входа:

- описание начальных условий,
- описание желаемой цели и
- множество возможных действий, заданных формальным языком наподобие STRIPS.

Планировщик создаёт последовательность действий, которые ведут систему из начального состояния в состояние, удовлетворяющее поставленной цели. Альтернативным способом описания проблем планирования является иерар-

хическая сеть задач, в которой из данного множества задач, каждая задача может быть либо выполнена с помощью примитивного действия, либо разбита на аналогичное подмножество задач.

Прогнозирование

Прогноз (от греч. πρόγνωση «предвидение, предсказание») — это научно обоснованное суждение о возможных состояниях объекта в будущем и (или) об альтернативных путях и сроках их осуществления. В узком смысле это вероятностное суждение о будущем состоянии объекта исследования.
Прогнозирование — это разработка прогноза; в узком значении — специальное научное исследование конкретных перспектив дальнейшего развития какого-либо процесса.
Необходимость прогноза обусловлена желанием знать события будущего, что достоверно — невозможно в принципе, исходя из статистических (ошибки текущих оценок), вероятностных (многовариантность следствий), эмпирических (методологические ошибки моделей), философских (ограниченность текущих знаний) принципов.
Точность любого прогноза обусловлена:
объёмом «истинных» (верифицированных с известной погрешностью) исходных данных и периодом их сбора;
объёмом неверифицированных исходных данных и периодом их сбора;
свойствами объекта прогнозирования и системы его взаимодействия с субъектом прогноза;
методиками и моделями прогнозирования.
При возрастании совокупности факторов, влияющих на точность прогноза, он практически замещается рутинным расчётом с некоторой установившейся погрешностью.
К основным методам прогнозирования относят:

- статистические методы;
- экспертные оценки (например, метод Дельфи);
- методы моделирования, в том числе имитационного;
- интуитивные (то есть выполненные без применения технических средств, экспромтом, «в уме» специалистом, имеющим опыт ранее применяемых научных методов в данном типе прогнозов).

2. Машинное обучение (Machine Learning)

Единственным способом заставить компьютер что-то делать — от сложения двух чисел до управления самолетом — было составление некоего алгоритма, скрупулезно объясняющего машине, что именно от нее требуется. Одна-

ко алгоритмы машинного обучения — совсем другое дело: они угадывают все сами, делая выводы на основе данных, и чем больше данных, тем лучше у них получается. Это значит, что компьютеры не надо программировать: они программируют себя сами.

Машинное обучение — технология, которая строит саму себя. Это новое явление в нашем мире.

Машинное обучение — это тренировка математической модели на исторических данных для того, чтобы прогнозировать какое-то событие или явление на новых данных. То есть попытка заставить алгоритмы программ совершать действия на основе предыдущего опыта, а не только на основе имеющихся данных. Для обучения нужны исторические данные (**обучающая выборка**) и значение целевой переменной (то, что прогнозируем), которое соответствует заданным историческим данным. Модель наблюдает и находит зависимости между данными и целевой переменной. Эти зависимости используются моделью для нового набора данных, чтобы прогнозировать целевую переменную, которая неизвестна.

Машинное обучение включает в себя целый набор методов и алгоритмов, которые могут предсказать какой-то результат по входным данным. Например, у вас есть какая-то информация по тому, сколько стоили ценные бумаги в каждый момент из какого-то длинного промежутка времени, алгоритмы машинного обучения могут предсказать, сколько эти бумаги будут стоить в будущем.

Типы машинного обучения

Существует множество моделей для машинного обучения, но они, как правило, относятся к одному из трех типов:

1. *обучение с учителем (supervised learning);*
2. *обучение без учителя, или самообучение (unsupervised learning);*
3. *обучение с подкреплением (reinforcement learning).*

В зависимости от выполняемой задачи одни модели могут быть более подходящими и более эффективными, чем другие.

Обучение с учителем

В этом типе корректный результат при обучении модели явно обозначается для каждого идентифицируемого элемента в наборе данных. Это означает, что при считывании данных у алгоритма уже есть правильный ответ. Поэтому вместо поисков ответа он стремится найти связи, чтобы в дальнейшем, при

введении необозначенных данных, получались правильные классификация или прогноз.
В контексте классификации алгоритм обучения может, например, снабжаться историей транзакций по кредитным картам, каждая из которых помечена как безопасная или подозрительная. Он должен изучить отношения между этими двумя классификациями, чтобы затем суметь соответствующим образом маркировать новые операции в зависимости от параметров классификации (например, место покупки, время между операциями и т. д.).

В случае, когда данные непрерывно связаны друг с другом, как, например, изменение курса акций во времени, регрессионный алгоритм обучения может использоваться для прогнозирования следующего значения в наборе данных.
Машинное обучение с учителем — это прямая имитация закономерностей, имеющих место между двумя наборами данных. В нем всегда входной набор данных преобразуется в выходной. Часто это невероятно мощный и полезный метод. Рассмотрим следующие примеры (**входные** данные выделены жирным шрифтом, а выходные — *курсивом*):

- Использование **пикселов** изображения для определения *присутствия* или *отсутствия кота*.
- Использование списка **понравившихся фильмов** для выбора *фильмов*, *которые могут понравиться*.
- Использование **слов** в сообщении, чтобы предсказать, *счастлив ли их автор или расстроен*.
- Использование **данных** с метеорологических приборов для *предсказания вероятности дождя*.
- *Использование **датчиков** автомобильного двигателя для определения оптимальных настроек*.
- Использование **новостей** для предсказания завтрашних *котировок на бирже.*
- Использование входного **числа** для предсказания удвоенного *числа*.
- Использование **аудиофайла** для получения *транскрипции речи*, содержащейся в нем.

Все это — задачи машинного обучения с учителем. Во всех случаях алгоритм машинного обучения пытается выявить такие закономерности между двумя наборами данных, чтобы по одному можно было спрогнозировать другой. Его удобно использовать, когда на входе имеется нечто известное и требуется быстро преобразовать его в то, что хотелось бы знать.

Обучение без учителя

В этом случае у алгоритма в процессе обучения нет заранее установленных ответов. Его цель — найти смысловые связи между отдельными данными, выявить шаблоны и закономерности. Например, **кластеризация** — это использование неконтролируемого обучения в рекомендательных системах (например, люди, которым понравился этот смартфон, также положительно оценили вот этот).

Обучение с подкреплением

Этот тип обучения представляет собой смесь первых двух. Обычно он используется для решения более сложных задач и требует взаимодействия с окружающей средой. Данные предоставляются средой и позволяют алгоритму реагировать и учиться.
Область применения такого метода обширна: от контроля роботизированных рук и поиска наиболее эффективной комбинации движений, до разработки систем навигации роботов, где поведенческий алгоритм «избежать столкновения» обучается опытным путем, получая обратную связь при столкновении с препятствием.
Логические игры также хорошо подходят для обучения с подкреплением, так как они традиционно содержат логическую цепочку решений. Этот метод обучения также часто применяется в логистике, составлении графиков и тактическом планировании задач.

Задачи машинного обучения

Машинное обучение базируется на идее о том, что аналитические системы могут учиться выявлять закономерности и принимать решения с минимальным участием человека. В машинном обучении выделяют следующие ключевые задачи:
регрессия — предсказание числовых значений признаков, например, предсказание будущих объемов продаж на основании известных данных о продажах в прошлом;
классификация — предсказание того, к какому из известных классов относится объект, например, предсказание того, вернет ли заемщик кредит, на основании данных о том, как возвращали кредиты заемщики в прошлом;
кластеризация — разделение большого множества объектов на кластеры — классы, внутри которых объекты похожи между собой, например, сегментирование рынка, разделение всех потребителей на классы так, что внутри классов потребители похожи между собой, а в разных классах — отличаются;

уменьшение размерности – сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации (например, сжатие данных);
поиск аномалий — поиск редких и необычных объектов, существенно отличающихся от основной массы, например поиск мошеннических транзакций.

Методы машинного обучения

Сегодня чаще всего для создания программ машинного обучения используются языки R, Python, Scala и Julia. Они поддерживаются многими интегрированными средами разработки, в частности, R-Studio, R-Brain, Visual Studio, Eclipse, PyCharm, Spyder, IntelliJ IDEA, Jupyter Notebooks, Juno и др.

Как работает машинное обучение

Машинное обучение часто называют волшебным или черным ящиком:

Вводишь данные → «волшебный черный ящик» → Миссия выполнена

Давайте посмотрим на сам процесс обучения, чтобы лучше понять, как машинное обучение справляется с данными.

Модель машинного обучения

Сбор

Машинное обучение основывается на данных. Первый шаг — убедиться, что имеющиеся данные верны и относятся именно к той задаче, которую вы пытаетесь решить. Оцените свои возможности для сбора данных, обдумайте их источник, необходимый формат и т. д.

Очистка

Данные зачастую формируются из различных источников, отображаются в различных форматах и языках. Соответственно, среди них могут оказаться нерелевантные или ненужные значения, которые потребуется удалить. И наоборот, каких-то данных может не хватать, и потребуется их добавить. От правильной подготовки базы данных прямым образом зависит и пригодность к использованию, и достоверность результатов.

Разделение

В зависимости от размера набора данных в некоторых случаях может потребоваться только небольшая их часть. Обычно это называется выборкой. Из

выбранной части данные надо разделить на две группы: одна для использования алгоритмом, а другая для оценки его действий.

Обучение

Этот этап фактически направлен на поиск математической функции, которая точно выполнит указанную задачу. Обучение разнится в зависимости от типа используемой модели. Построение линий в простой линейной модели — это обучение; генерация дерева принятия решений для алгоритма случайного леса — это также обучение. Изменение ответов при построении дерева решений поможет скорректировать алгоритм.

Чтобы было проще, сосредоточимся на нейронных сетях.
Суть в том, что алгоритм использует часть данных, обрабатывает их, замеряет эффективность обработки и автоматически регулирует свои параметры (также называемый метод обратного распространения ошибки) до тех пор, пока не сможет последовательно производить желаемый результат с достаточной достоверностью.

Оценка

После того как алгоритм хорошо показал себя на учебных данных, его эффективность оценивается на данных, с которыми он еще не сталкивался. Дополнительная корректировка производится при необходимости. Этот процесс позволяет предотвратить переобучение — явление, при котором алгоритм хорошо работает только на учебных данных.

Оптимизация

Модель оптимизируется, чтобы при интеграции в приложение весить как можно меньше и как можно быстрее работать.

Для чего можно использовать машинное обучение
В бизнесе можно рассматривать три сферы применения машинного обучения:

1. *описательную*
2. *прогнозирующую*
3. *нормативную.*

Описательное применение относится к записи и анализу статистических данных для расширения возможностей бизнес-аналитики. Руководители получают описание и максимально информативный анализ результатов и последствий прошлых действий и решений. Этот процесс в настоящее время обычен

для большинства крупных компаний по всему миру — например, анализ продаж и рекламных проектов для определения их результатов и рентабельности.

Второе применение машинного обучения — прогнозирование. Сбор данных и их использование для прогнозирования конкретного результата позволяет повысить скорость реакции и быстрее принимать верные решения. Например, прогнозирование оттока клиентов может помочь его предотвратить. Сегодня этот процесс применяется в большинстве крупных компаний.
Третье и наиболее продвинутое применение машинного обучения внедряется уже существующими компаниями и совершенствуется усилиями недавно созданных. Простого прогнозирования результатов или поведения уже недостаточно для эффективного ведения бизнеса. *Понимание причин, мотивов и окружающей ситуации — вот необходимое условие для принятия оптимального решения*. Этот метод наиболее эффективен, если человек и машина объединяют усилия. Машинное обучение используется для поиска значимых зависимостей и прогнозирования результатов, а специалисты по данным интерпретируют результат, чтобы понять, почему такая связь существует. В результате становится возможным принимать более точные и верные решения.

Глубокое обучение (Deep learning)

Глубокое обучение может быть как с учителем, так и без, но оно подразумевает под собой анализ **Big Data** – настолько большого массива информации, что одного компьютера будет недостаточно. Поэтому Deep Learning использует для работы нейронные сети. Нейронные сети позволяют разделить одну большую задачу на несколько маленьких и делегировать их другим устройствам. Например, один процессор собирает информацию и передает ее двум другим. Те, в свою очередь, анализируют ее и передают еще четырем, которые выполняют еще какие-то задачи и передают следующим процессорам.

3. Обработка естественного языка

Немаловажным направлением является обработка естественного языка, в рамках которого проводится анализ возможностей понимания, обработки и генерации текстов на «человеческом» языке. В рамках этого направления ставится цель такой обработки естественного языка, которая была бы в состоянии приобрести знание самостоятельно, читая существующий текст, доступный по Интернету. Некоторые прямые применения обработки естествен-

ного языка включают **информационный поиск** (в том числе, глубокий анализ текста) и **машинный перевод**.

Понимание естественного языка иногда считают **AI-полной** задачей, потому как распознавание живого языка требует огромных знаний системы об окружающем мире и возможности с ним взаимодействовать. Само определение смысла слова «понимать» — одна из главных задач искусственного интеллекта.

Качество понимания зависит от множества факторов: от языка, от национальной культуры, от самого собеседника и т. д. Вот некоторые примеры сложностей, с которыми сталкиваются системы понимания текстов.

Свободный порядок слов может привести к совершенно иному толкованию фразы: «Бытие определяет сознание» — что определяет что?

В русском языке свободный порядок компенсируется развитой морфологией, служебными словами и знаками препинания, но в большинстве случаев для компьютера это представляет дополнительную проблему.

В речи могут встретиться неологизмы, например глагол «Пятидесятирублируй» — то есть высылай 50 рублей. Система должна уметь отличать такие случаи от опечаток и правильно их понимать.

Сложности с раскрытием анафор (распознаванием, что имеется в виду при использовании местоимений): предложения «Мы отдали бананы обезьянам, потому что **они** были голодные» и «Мы отдали бананы обезьянам, потому что **они** были перезрелые» похожи по синтаксической структуре. В одном из них местоимение ***они*** относится к обезьянам, а в другом — к бананам. Правильное понимание зависит от знаний компьютера, какими могут быть бананы и обезьяны.

Правильное понимание омонимов — ещё одна проблема. При распознавании речи, помимо прочих, возникает проблема фонетических омонимов. Во фразе «*Серый волк в глухом лесу встретил рыжую лису*» выделенные слова слышатся одинаково, и без знания, кто глухой, а кто рыжий, не обойтись.

Популярные задачи

- Распознавание речи
- Анализ текста
- Извлечение информации
- Информационный поиск
- Анализ высказываний
- Анализ тональности текста
- Вопросно-ответные системы
- Генерирование текста
- Синтез речи

- Общая классификация
- Категоризация текстов
- Классификация последовательностей символов
    - Распознавание именованных сущностей
    - Определение частей речи слов
- Распознавание фраз
- Извлечение информации из текста
- Синтаксическая аннотация
- Семантическая аннотация
- Генерирование текста
- Генерация текста на основе распознанной речи
- Машинный перевод
- Обобщение текста

Информационный поиск

Информационный поиск (information retrieval) — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, и наука об этом поиске.
Поиск информации представляет собой процесс выявления в некотором множестве документов (текстов) всех тех, которые посвящены указанной теме (предмету), удовлетворяют заранее определенному условию поиска (запросу) или содержат необходимые (соответствующие информационной потребности) факты, сведения, данные.
Процесс поиска включает последовательность операций, направленных на сбор, обработку и предоставление информации.
В общем случае поиск информации состоит из четырех этапов:

1. определение (уточнение) информационной потребности и формулировка информационного запроса;
2. определение совокупности возможных держателей информационных массивов (источников);
3. извлечение информации из выявленных информационных массивов;
4. ознакомление с полученной информацией и оценка результатов поиска.

Методы поиска

Адресный поиск
Процесс поиска документов по чисто формальным признакам, указанным в запросе.
Семантический поиск

Процесс поиска документов по их содержанию.

Документальный поиск

Процесс поиска в хранилище информационно-поисковой системы первичных документов или в базе данных вторичных документов, соответствующих запросу пользователя.

Фактографический поиск

Процесс поиска фактов, соответствующих информационному запросу.

Машинный перевод

Машинный перевод — процесс перевода текстов (письменных, а в идеале и устных) с одного естественного языка на другой с помощью специальной компьютерной программы. Так же называется направление научных исследований, связанных с построением подобных систем.

Формы организации взаимодействия ЭВМ и человека при машинном переводе:

С постредактированием: исходный текст перерабатывается машиной, а человек-редактор исправляет результат.

С предредактированием: человек приспосабливает текст к обработке машиной (устраняет возможные неоднозначные прочтения, упрощает и размечает текст), после чего начинается программная обработка.

С интерредактированием: человек вмешивается в работу системы перевода, разрешая трудные случаи.

Смешанные системы (например, одновременно с пред- и постредактированием).

4. Представление и использование знаний

Направление инженерия знаний объединяет задачи получения знаний из простой информации, их систематизации и использования. Это направление исторически связано с созданием **экспертных систем** — программ, использующих специализированные базы знаний для получения достоверных заключений по какой-либо проблеме.

Производство знаний из данных — одна из базовых проблем **интеллектуального анализа данных**. Существуют различные подходы к решению этой проблемы, в том числе — на основе нейросетевой технологии, использующие процедуры вербализации нейронных сетей.

Модели представления знаний

Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам:

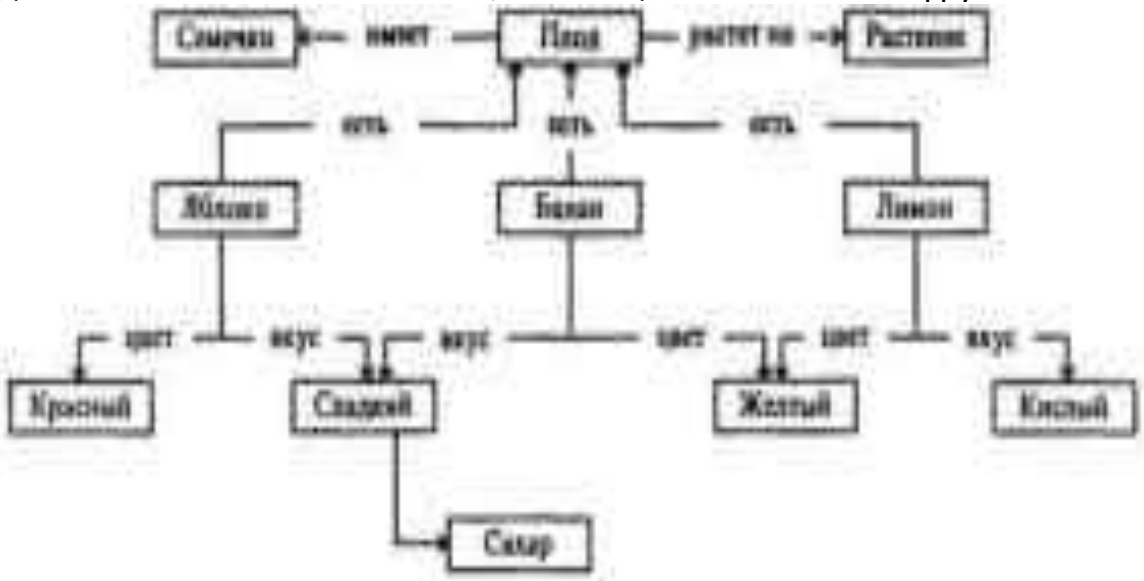
- продукционные модели;
- семантические сети;
- фреймы;
- формальные логические модели.

Продукционные правила

Продукционные правила - наиболее простой способ, представления знаний. Он основан на представлении знаний в форме правил, структурированных в соответствии с образцом «ЕСЛИ - ТО». Часть правила «ЕСЛИ» называется посылкой, а «ТО» - выводом или действием. Правило в общем виде записывается так:
ЕСЛИ A_1, A_2, ...,A_n, ТО В.
Такая запись означает, что «если все условия от A_1 до A_n являются истинными, то В также истинно» или «когда все условия от A_1 до A_n выполняются, то следует выполнить действие В».

Семантическая сеть

Семантическая сеть - иной подход к представлению знаний, который основан на изображении понятий (сущностей) с помощью точек (узлов) и отношений между ними с помощью дуг на плоскости. Семантические сети способны отображать структуру знаний во всей сложности их взаимосвязей, увязать в единое целое объекты и их свойства. В качестве примера может быть приведена часть семантической сети, относящейся к понятию «фрукты»



Фреймовая система

Фреймовая система имеет все свойства, присущие языку представления знаний, и одновременно являет собой новый способ обработки информации.

Слово «фрейм» в переводе с английского языка означает «рамка». Фрейм является единицей представления знаний об объекте, которую можно описать некоторой совокупностью понятий и сущностей. Фрейм имеет определенную внутреннюю структуру, состоящую из множества элементов, называемых **слотами**. Каждый слот, в свою очередь, представляется определенной структурой данных, процедурой, или может быть связан с другим фреймом.

- *Фрейм*: человек
- *Класс*: Животное
- *Структурный элемент*: Голова, шея, руки, ноги,...
- *Рост*: 30–220 см
- *Масса*: 1–200 кг
- *Хвост*: Нет
- *Фрейм аналогии*: Обезьяна

Существуют и другие, менее распространенные подходы к представлению знаний в интеллектуальных системах, в том числе гибридные, на основе уже описанных подходов.

Экспертная система

Экспертная система (ЭС, англ. expert system) — компьютерная система, способная частично заменить специалиста-эксперта в разрешении проблемной ситуации. Современные экспертные системы начали разрабатываться исследователями искусственного интеллекта в 1970-х годах, а в 1980-х годах получили коммерческое подкрепление.

Важнейшей частью экспертной системы являются базы знаний как модели поведения экспертов в определённой области знаний с использованием процедур логического вывода и принятия решений, иными словами базы знаний — совокупность фактов и правил логического вывода в выбранной предметной области деятельности.

В настоящее время «классическая» концепция экспертных систем, сложившаяся в 1970—1980 годах, переживает кризис, по всей видимости связанный с её глубокой ориентацией на общепринятый в те годы текстовый человеко-машинный интерфейс, который в настоящее время в пользовательских приложениях почти полностью вытеснен графическим (GUI). Кроме того, «классический» подход к построению экспертных систем плохо согласуется с реляционной моделью данных, что делает невозможным эффективное использование современных промышленных СУБД для организации баз знаний таких систем.

Интеллектуальный анализ данных

Data mining (рус. *добыча данных, интеллектуальный анализ данных, глубинный анализ данных*) — собирательное название, используемое для обозначения совокупности методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. Термин введён Пятецким-Шапиро в 1989 году.

Английское словосочетание «data mining» пока не имеет устоявшегося перевода на русский язык. При передаче на русском языке используются следующие словосочетания: просев информации, добыча данных, извлечение данных, а также интеллектуальный анализ данных. Более полным и точным является словосочетание «обнаружение знаний в базах данных» (англ. knowledge discovery in databases, KDD).

Основу методов data mining составляют всевозможные методы классификации, моделирования и прогнозирования, основанные на применении деревьев решений, искусственных нейронных сетей, генетических алгоритмов, эволюционного программирования, ассоциативной памяти, нечёткой логики. К методам data mining нередко относят статистические методы (дескриптивный анализ, корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ, компонентный анализ, дискриминантный анализ, анализ временных рядов, анализ выживаемости, анализ связей). Такие методы, однако, предполагают некоторые априорные представления об анализируемых данных, что несколько расходится с целями data mining (обнаружение ранее неизвестных нетривиальных и практически полезных знаний).

Методы data mining лежат на стыке баз данных, статистики и искусственно-го интеллекта.

5. Интеллектуальная робототехника

Роботы — часть стремительно надвигающегося будущего высоких технологий. В настоящее время на планете Земля в сфере робототехники революции происходят чуть ли не каждую неделю. Роботы спасают людей, работают в экстремальных условиях, заменяют живое общение, исследуют планеты Солнечной системы и многое другое. От слуг до наставников роботы развиваются чрезвычайно быстрыми темпами. Работа в сфере робототехники на данный момент борется с самой важной задачей: **как оснастить робота искусственным интеллектом.**

Интеллектуальность требуется роботам, чтобы манипулировать объектами, выполнять навигацию с проблемами локализации (определять местонахож-

дение, изучать ближайшие области) и планировать движение (как добраться до цели)
Основное предназначение роботов – сделать нашу жизнь более комфортной, улучшить условия труда, освободить «руки» от сложных рабочих процессов и увеличить производительность.
Роботы чаще всего встречаются в промышленности, где с их помощью удалось полностью автоматизировать большинство производственных задач. Но, кроме того, умные машины все больше задействуются в военной отрасли, медицине, сфере обслуживания и потребительском секторе.

Примеры интеллектуальной робототехники

1. *Pleo — робот-динозавр*.
   Сложный искусственный интеллект дает возможность формировать уникальную индивидуальность каждому отдельно взятому роботу Pleo.

   Технические возможности Pleo:

   - зрение, основанное на видеокамере (для реакции на освещенность и навигации)
   - инфракрасное определение внешних объектов
   - два микрофона для реакции на звук
   - восемь датчиков прикосновений (голова, подбородок, плечи, хвост, лапы)
   - датчики на ступнях (для определения типа поверхности)
   - датчик наклона для определения позиции тела
   - 14 сервомоторов, по одному в каждом суставе
   - инфракрасный датчик для обнаружения попадания объекта в ротовую полость …
2. *Aibo - серия собак-роботов, разработанная компанией Sony*.

   Aibo умеет ходить, «видеть» окружающие его предметы с помощью видеокамеры и инфракрасных датчиков расстояния, распознавать команды и лица. Робот является полностью автономным: он может учиться и развиваться, основываясь на побуждениях своего хозяина, обстановки, или другого AIBO. Несмотря на это, он поддаётся настройкам с помощью специальных программ. «Настроение» Aibo может меняться в зависимости от окружающей обстановки и влиять на поведение. Инстинкты позволяют Aibo двигаться, играть со своими игрушками, удовлетворять своё любопытство, играть и общаться с хозяином, самостоятельно подзаряжаться и просыпаться после сна. Разработчики утверждают, что у AIBO есть симу-

лирование шести эмоций: счастья, грусти, страха, антипатии, удивления и гнева.

Виды роботов по сфере применения

- Медицинские помощники
- Бытовые ассистенты
- Роботы-игрушки
- Сервисные
- Военные роботы
- Промышленные машины
- Развлекательные

Примеры

- автомобили с автономным управлением
- разгрузочно-погрузочные роботы, упаковщики, сортировщики, формовщики
- радиоуправляемые тракторы и плуги
- беспилотные летательные аппараты
- роботы-пылесосы
- системы умного дома

...

6. Биологическое моделирование искусственного интеллекта

Сюда можно отнести несколько направлений.

- **Нейронные сети** используются для решения нечётких и сложных проблем, таких как распознавание геометрических фигур или кластеризация объектов.
- **Генетический подход** основан на идее, что некий алгоритм может стать более эффективным, если позаимствует лучшие характеристики у других алгоритмов («родителей»).
- Относительно новый подход, где ставится задача создания автономной программы — агента, взаимодействующей с внешней средой, называется **агентным подходом**.

Биокомпьютинг

Биокомпьютинг (или квазибиологическая парадигма — биологическое направление в искусственном интеллекте, сосредоточенное на разработке и использовании компьютеров, которые функционируют как живые организмы или содержат биологические компоненты, так называемые **биокомпьютеры**.

Квазибиологическая парадигма сегодня по своему содержанию и возможным приложениям значительно богаче, чем первоначальный подход Мак-Каллоха и Питса. Она находится в процессе развития и изучения возможностей создания на её основе эффективных средств обработки информации.

Направления в исследованиях

- Биокомпьютинг позволяет решать сложные вычислительные задачи, организуя вычисления при помощи живых тканей, клеток, вирусов и биомолекул. Часто используют молекулы дезоксирибонуклеиновой кислоты, на основе которого создают ДНК-компьютер. Кроме ДНК, в качестве биопроцессора могут использоваться также белковые молекулы и биологические мембраны.
- Молекулярные вычисления
- Биомолекулярная электроника
- Искусственные нейронные сети
- Эволюционные вычисления
- Нейрокомпьютинг

7. Другие области исследований

Машинное творчество

Природа человеческого творчества ещё менее изучена, чем природа интеллекта. Тем не менее, эта область существует, и здесь поставлены проблемы написания компьютером музыки, литературных произведений (часто — стихов или сказок), художественное творчество. Создание реалистичных образов широко используется в кино и индустрии игр.
Отдельно выделяется изучение проблем технического творчества систем искусственного интеллекта. Теория решения изобретательских задач, предложенная в 1946 году Г. С. Альтшуллером, положила начало таким исследованиям.

Добавление данной возможности к любой интеллектуальной системе позволяет продемонстрировать, что именно система воспринимает и как это понимает. Добавлением шума вместо недостающей информации или фильтрация шума имеющимися в системе знаниями производит из абстрактных знаний конкретные образы, легко воспринимаемые человеком, особенно это полезно для интуитивных и малоценных знаний, проверка которых в формальном виде требует значительных умственных усилий.

Компьютерное зрение (Computer Vision, CV)

Компьютерное зрение (иначе техническое зрение) — теория и технология создания машин, которые могут производить обнаружение, отслеживание и классификацию объектов.

Как научная дисциплина, компьютерное зрение относится к теории и технологии создания искусственных систем, которые получают информацию из изображений. Видеоданные могут быть представлены множеством форм, таких как видеопоследовательность, изображения с различных камер или трехмерными данными, или медицинского сканера.

Как технологическая дисциплина, компьютерное зрение стремится применить теории и модели компьютерного зрения к созданию систем компьютерного зрения. Примерами применения таких систем могут быть:

- Системы управления процессами (промышленные роботы, автономные транспортные средства).
- Системы видеонаблюдения.
- Системы организации информации (например, для индексации баз данных изображений).
- Системы моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование).
- Системы взаимодействия (например, устройства ввода для системы человеко-машинного взаимодействия).
- Системы дополненной реальности.
- Вычислительная фотография, например, для мобильных устройств с камерами.

Одним из наиболее важных применений является обработка изображений в медицине. Эта область характеризуется получением информации из видеоданных для постановки медицинского диагноза пациентам. В большинстве случаев видеоданные получают с помощью микроскопии, рентгенографии, ангиографии, ультразвуковых исследований и томографии. Примером информации, которая может быть получена из таких видеоданных, является обнаружение опухолей, атеросклероза или других злокачественных изменений.

Другой прикладной областью компьютерного зрения является промышленность. Здесь информацию получают для целей поддержки производственного процесса. Примером может служить контроль качества, когда детали или конечный продукт автоматически проверяются на наличие дефектов. Другим примером является измерение положения и ориентации деталей, поднимаемых рукой робота.

Эволюционные алгоритмы

Эволюционные алгоритмы — направление в искусственном интеллекте (раздел эволюционного моделирования), которое использует и моделирует процессы естественного отбора.
Все они моделируют базовые положения в теории биологической эволюции — процессы отбора, мутации и воспроизводства. Поведение агентов определяется окружающей средой. Множество агентов принято называть популяцией. Такая популяция эволюционирует в соответствии с правилами отбора в соответствии с целевой функцией, задаваемой окружающей средой. Каждому агенту (индивидууму) популяции назначается значение его пригодности в окружающей среде. Размножаются только наиболее пригодные виды. Рекомбинация и мутация позволяют изменяться агентам и приспособляться к среде. Такие алгоритмы относятся к адаптивным поисковым механизмам.
Эволюционные алгоритмы успешно использовались для задач типа функциональной оптимизации и могут легко быть описаны на математическом языке.

Виды алгоритмов

генетические алгоритмы — эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров;
генетическое программирование — автоматическое создание или изменение программ с помощью генетических алгоритмов;
эволюционное программирование — аналогично генетическому программированию, но структура программы постоянна, изменяются только числовые значения;
программирование экспрессии генов
эволюционные стратегии — похожи на генетические алгоритмы, но в следующее поколение передаются только положительные мутации;
дифференциальная эволюция
нейроэволюция — аналогично генетическому программированию, но геномы представляют собой искусственные нейронные сети, в которых происходит эволюция весов при заданной топологии сети, или помимо эволюции весов также производится эволюция топологии;
системы классификаторов;

Генетический алгоритм

Генетический алгоритм — это в первую очередь эволюционный алгоритм, другими словами, основная идея алгоритма — скрещивание (комбинирование). Как несложно догадаться, идея алгоритма взята у природы. Так вот, путем перебора и самое главное отбора получается правильная «комбинация».
Алгоритм делится на три этапа:

1. Скрещивание
2. Селекция (отбор)
3. Формирования нового поколения

Если результат нас не устраивает, эти шаги повторяются до тех пор, пока результат нас не начнет удовлетворять или произойдет одно из нижеперечисленных условий:

- Количество поколений (циклов) достигнет заранее выбранного максимума
- Исчерпано время на мутацию

Рекомендательные системы (recommendation-engine)

Используя передовые алгоритмы, такие как машинное обучение и ИИ, система рекомендаций может помочь клиентам получить соответствующие продукты, которые они хотят или нуждаются.
Механизм рекомендаций продуктов - это, по сути, решение, которое позволяет маркетологам предлагать своим клиентам соответствующие рекомендации по продуктам в режиме реального времени. В качестве мощных инструментов фильтрации данных рекомендательные системы используют алгоритмы и методы анализа данных, чтобы рекомендовать наиболее релевантный продукт / элементы конкретному пользователю.
Основной целью любого рекомендательного движка является стимулирование спроса и активное вовлечение пользователей. В первую очередь компонент стратегии персонализации электронной коммерции рекомендательные движки динамически заполняют различные продукты на веб-сайтах, приложениях или электронных письмах, тем самым улучшая качество обслуживания клиентов. Эти виды разнообразных рекомендаций делаются на основе нескольких точек данных, таких как предпочтения клиентов, история прошлых транзакций, атрибуты или ситуационный контекст.

Управление жестами (gesture control)

Жестовый интерфейс — подмножество системы ввода для графического пользовательского интерфейса для устройств, оснащённых специальными либо устройствами ввода (отличными от клавиатуры), либо сенсорными

экранами, и позволяющая эмулировать клавиатурные команды (либо сочетания клавиш) при помощи жестов (росчерков, англ. gesture). Основной мотивацией разработки таких интерфейсов является улучшение эргономичности управления, с отказом от привычного для компьютерных программ меню приложения.
Подобный интерфейс может быть реализован как при помощи устройств координатного ввода с возможностью считывания координаты одной точки касания (мышь либо графический планшет — см. «жесты мышью»), так и таких, в которых имеется возможность считывания координат более чем одной точки (т. н. мультикасание, multitouch) — сенсорные экраны и панели. Последние стали широко применяться в интерфейсах множества современных смартфонов с сенсорным экраном (напр. iPhone) и ноутбуков (как с тачпадом, так с сенсорным экраном) и прочих мобильных устройств.

Контекстно-зависимые вычисления (Context-aware computing)

Контекстно-зависимые системы относят к разряду «повсеместных вычислений» (англ. Ubiquitous computing или Pervasive computing). Основными источниками информации для контекстно-зависимых систем являются местоположение, социальное и физическое окружение. Несмотря на то, что определение местоположения широко используется в настоящее время, оно не всегда учитывает изменяющиеся интересы пользователя. Контекстная зависимость в широком понимании включает в себя находящихся рядом людей, устройства, доступ к интернету, уровень освещенности, уровень шума, а также взаимодействие людей в бытовых ситуациях. Например, находитесь ли вы сейчас с семьей или со своим школьным другом.

Интеллектуальное видеонаблюдение (Video Content Analysis)

Видеоаналитика — технология, использующая методы компьютерного зрения для автоматизированного получения различных данных на основании анализа последовательности изображений, поступающих с видеокамер в режиме реального времени или из архивных записей. Видеоаналитика представляет собой программное обеспечение (ПО) для работы с видеоконтентом. В основе программного обеспечения лежит комплекс алгоритмов машинного зрения, позволяющих вести видеомониторинг и производить анализ данных без прямого участия человека. Алгоритмы видеоаналитики могут быть интегрированы в различные бизнес-системы, чаще всего используются в видеонаблюдении и других сферах безопасности.

Определения

Видеоаналитика (VCA, Video Content Analysis) – компьютеризированная обработка и автоматический анализ видеоконтента, который поступает на видеосервер от видеокамер, носимых устройств и устройств Интернета вещей IoT, оснащённых веб-камерами.

Видеоаналитика - это технология, использующая методы компьютерного зрения для автоматизированного получения различных данных на основании анализа последовательности изображений, поступающих с видеокамер в режиме реального времени или из архивных записей.

Видеоаналитика представляет собой программное обеспечение (ПО) для работы с видеоконтентом. В основе программного обеспечения лежит комплекс алгоритмов машинного зрения, позволяющих вести видеомониторинг и производить анализ данных без прямого участия человека.

Традиционное решение, включающее в себя функции какой-либо видеоаналитики строится по схеме: камера + back-end аналитика. Т.е. камера просто гонит поток видео на сервер, а специальное ПО на сервере уж делает весь видеоанализ.

Видеоаналитика – это частные приложения компьютерного зрения, которые извлекают информацию и знания из видеоконтента, то есть дают ответы на вопросы:

Кто: распознавание и идентификация людей;

Что: объекты, действия, события, поведение, взаимоотношения;

Где: геолокация, пространственная (3D) и планарная (2D) локация;

Когда: маркировка даты и времени, сезона.

Три основных типа приложений видеоаналитики

1. **Ретроспектива:** что уже случилось, т.е. управление архивами видеозаписей, поиск, сортировка, получение юридических доказательств;
2. **Настоящий момент:** что происходит сейчас, т.е. контроль ситуации, получение предупреждений в реальном времени, кодирование, компрессия видеопотока;
3. **Взгляд в будущее:** что может или скорее всего произойдёт, т.е. предсказания на основе событий прошлого и настоящего, прогнозирование событий или активности, детектирование намечающихся аномалий.

Функциональные возможности

Во время видеонаблюдения в промышленности, городском и жилищном хозяйстве, а также в различных социальных медиа, генерируется огромное количество видеоданных, для которых требуется системы хранения данных (СХД) с высокой ёмкостью. Разрешающая способность видеоизображений всё время возрастает, и количество хранимого контента растёт экспоненциально.

Видеоаналитика в последние годы набирает всё большую популярность по многим причинам. Она позволяет гибко управлять видеопотоками при анализе их контента «на лету», при автоматизации аналитических функций. Это позволяет персоналу концентрироваться на определённых инцидентах на видеозаписи, а не тратить время на просмотр длинных однообразных видеопотоков, что позволяет сократить затраты и численность персонала. Интеллектуальные системы безопасности с видеоаналитикой могут начинать запись, например, только при начале какого-то движения в зоне обзора камеры. При этом снижается нагрузка на сеть и экономится пространство в системе хранения.

При помощи систем видеоаналитики, можно получить ценную информацию о качестве работы персонала предприятия (например, продавцов-

консультантов в торговом зале), таким образом, можно сделать более адекватные оценки его работы.
Системы видеоаналитики не требуют чрезмерно громоздкой инфраструктуры и даже небольшие предприятия, магазины и пр. вполне могут себе позволить её использование. Интенсивность использования функций видеоаналитики можно гибко регулировать по мере потребностей бизнеса, выбирая именно те функции, которые нужны в конкретном случае. Это позволяет создавать кастомизированные решения.
Типовая системная архитектура видеоаналитики (VCA)

Стандарты

2020: В России разработан стандарт ИИ для ситуационной видеоаналитики
20 июля 2020 года стало известно о создании первого в России национального стандарта в области искусственного интеллекта для ситуационной видеоаналитики. Документ, подготовленный ООО «Видеоинтеллект» (развивает системы компьютерного зрения для использования в сложных условиях, общественных местах с большим скоплением людей, на объектах промышленности), представил технический комитет по стандартизации ТК 164 «Искусственный интеллект», созданный на базе РВК.

Виртуальный цифровой помощник

Виртуальный цифровой помощник (Virtual Digital Assistant, сокращенно VDA) — веб-сервис и (или) приложение для смартфонов и ПК, фактически исполняющий роль личного секретаря при пользователе. Решает задачи планирования графика, организации и выполнения повседневных дел и контекстного поиска информации для нужд конкретного человека.
Виртуальный цифровой помощник может создавать напоминания, подбирать места отдыха и развлечений, облегчить поиск и онлайн-бронирование

билетов и столиков, заказ такси[2]. Он способен самообучаться в ходе выполнения заданий, анализируя поведение и интересы пользователя.

Эксперты прогнозируют, что к 2025 году около половины работников, занятых в сфере обслуживания, начнут использовать виртуальных помощников при решении ежедневных задач. Отчасти это связано с коронавирусной пандемией, на фоне которой многие компании занялись разработкой систем искусственного интеллекта, включая чат-боты и цифровые ассистенты, чтобы оказывать поддержку онлайн-покупателям круглосуточно и без выходных.

Развитие ИИ способствует все большему использованию виртуальных помощников, поскольку клиенты довольны повышением качества обслуживания. Кроме того, автоматизация рутинных задач, для которых обычно нужны кол-центры, позволяет компаниям сосредоточиться на более сложных задачах.

В будущем виртуальные ассистенты почти ничем не будут отличаться от живых онлайн-консультантов. Уже сейчас появляются чат-боты, все больше похожие на человека и способные анализировать желания пользователей. Огромную роль в этом играет машинное обучение, когда искусственный интеллект тренируется выполнять определенные задачи на базе множества готовых решений.

Помимо экономии времени, которые люди тратят на выбор покупок, получение услуг и планирование ежедневного графика, виртуальные помощники могут усовершенствовать и процесс образования. Например, чат-боты могут круглосуточно консультировать студентов, которые опасаются того, что преподаватель оценит содержание их вопросов. Что более важно, виртуальные помощники смогут настраивать индивидуальную программу обучения, а также помогать учащимся сосредоточиться на тех областях знаний, которые пригодятся им в будущем.

Компании, включая Content Technologies и Carnegie Learning, разрабатывают онлайн-платформы для персонализированного обучения с помощью ИИ. Такие виртуальные помощники способны адаптировать способ взаимодействия к каждому учащемуся. Они смогут определять слабые стороны студента и использовать эту информацию, чтобы предложить примеры решения задач, вызывающих затруднение. Чат-боты и виртуальные помощники могут выполнять роль помощника преподавателя, но справляться с задачами намного быстрее и эффективнее.

Нейроинтерфейсы

Уже сейчас ведутся работы по созданию нейроинтерфейсов, соединяющих мозг с компьютером. Считается, что подобные разработки помогут парализо-

ванным людям взаимодействовать с окружающим миром и даже передвигаться, мысленно управляя экзоскелетом.
В 2019 году нейротехнологическая компания Илона Маска Neuralink представила прототип чипа, который должен вживляться в мозг, чтобы обеспечить связь между центральной нервной системой и компьютером. Чип представляет собой пучок из тысячи электродов, которые будут внедряться в зоны мозга, отвечающие за движение тела и обработку поступающей сенсорной информации. Чип был протестирован на животных, после чего Neuralink получила разрешение начать испытания с участием людей.
Микроэлектроды могут применяться и для нейростимуляции — технологии, позволяющей улучшить работу мозга с помощью электромагнитных импульсов, посылаемых в кору больших полушарий. Нейростимуляция может изменить к лучшему жизнь парализованных людей, пациентов с поражением органов чувств и страдающих хроническими болями, например, при невропатии.
Ожидается, что в будущем нейростимуляция и нейропротезирование будут прочно связаны с ИИ, который научится распознавать и дешифровать нервные сигналы. Это позволит человеку значительно расширить свои способности, поскольку он фактически обзаведется мозговой надстройкой — кибернетической.
Сама возможность того, что ИИ, превосходящий человека, станет дополнением биологического мозга, позволяет представить будущее, где грань между людьми и компьютерами станет более размытой. Ученые уверены: бояться такого будущего не стоит. Эти перспективы окажут значительное влияние на развитие нейробиологии и медицины, что в итоге лишь улучшит качество жизни.

Лекция 4. Области применения искусственного интеллекта

В зависимости от области и обширности сферы применения, выделяют два вида ИИ – Weak AI, называемый еще «слабым», и Strong AI, «сильный». В первом случае перед системой ставят узкоспециализированные задачи – диагностика в медицине, управление роботами, работа на базе электронных торговых платформ. Во втором же подразумевается решение глобальных задач.
Перечислить разом все области, в которых задействован искусственный интеллект, практически нереально. На данный момент он затрагивает все больше самых разных сфер. И причин на то немало – та же автоматизация

производственных процессов, стремительный рост информационного оборота и инвестиций в эту сферу, даже социальное давление.

Коммерция

Одна из наиболее популярных сфер применения ИИ – это Big Data в коммерции. Крупные торговые площадки используют подобные технологии для исследования потребительского поведения. Компания «Яндекс» вообще создает с их помощью музыку. В некоторые мобильные приложения встроены голосовые помощники вроде Siri, Алисы или Cortana. Они упрощают процесс навигации и совершения покупок в сервисе. И не стоит забывать про программы с нейросетями, обрабатывающими фото и видео.

Финансы

Финансовые учреждения давно используют нейронные сети для выявления подозрительных событий и действий. Использование ИИ в банковской сфере началось ещё в 1987 году, когда Security Pacific National Bank в США создал целевую группу по противодействию мошенничеству и несанкционированному использованию дебетовых карт.

Алгоритмическая торговля

Алгоритмическая торговля предполагает использование сложных систем искусственного интеллекта для принятия торговых решений со скоростью, превышающую скорость, на которую способен человеческий организм. Это позволяет делать миллионы сделок в день без какого-либо вмешательства человека. Автоматизированные торговые системы обычно используются крупными институциональными инвесторами.

Исследования рынка и интеллектуальный анализ данных

Несколько крупных финансовых учреждений вложили средства в развитие ИИ, чтобы использовать его в их инвестиционной практике. Широкий спектр функциональных возможностей таких систем включает обработку естественного языка для чтения текста, такого как новости, отчёты брокеров и каналы социальных сетей. Затем система оценивает настроения в упомянутых компаниях и присваивает им оценку.

Банки используют систему ИИ под названием Sqreem, которая может обрабатывать данные для разработки профилей потребителей и сопоставлять их с продуктами, которые они, скорее всего, захотят.

Управление финансовым портфелем

Автоматизированные помощники-советчики становятся все более широко используемыми в отрасли управления инвестициями. Автоматизированные системы предоставляют финансовые консультации и советы в управлении финансовым портфелем с минимальным вмешательством человека. Этот класс финансовых консультантов работает на основе алгоритмов, созданных для автоматического развития финансового портфеля в соответствии с инвестиционными целями и склонностью к риску клиентов. Он может корректировать изменения в реальном времени на рынке и калибровать портфель в соответствии с пожеланиями клиента.

Управление личными финансами

Существуют продукты, которые используют ИИ для помощи людям в управлении их личными финансами. Например, Digit — это приложение, основанное на искусственном интеллекте, которое автоматически помогает потребителям оптимизировать свои расходы и сбережения, основываясь на своих личных привычках и целях. Приложение может анализировать такие факторы, как ежемесячный доход, текущий баланс и привычки к расходам, затем принимать собственные решения и переводить деньги на отдельный сберегательный счёт. Wallet.AI, развивающийся в Сан-Франциско старт-ап, создаёт агентов, которые анализируют данные, которые генерирует потребитель, при взаимодействии со смартфонами и социальными сетями, чтобы проинформировать потребителя о своих расходах.

Военное дело

Применение ИИ является важным трендом в создании перспективных систем управления поля боя и вооружением.
С помощью ИИ возможно обеспечить оптимальный и адаптивный к угрозам выбор комбинации сенсоров и средств поражения, скоординировать их совместное функционирование, обнаруживать и идентифицировать угрозы, оценивать намерения противника. Существенную роль ИИ играет в реализации тактических систем дополненной реальности. Например, ИИ позволяет обеспечить классификацию и семантическую сегментацию изображений, локализацию и идентификацию мобильных объектов для эффективного целеуказания

Тяжелая промышленность

Роботы стали распространены во многих отраслях промышленности и часто занимаются работой, которая считается опасной для людей. Роботы оказа-

лись эффективными на рабочих местах, связанных с повторяющимися рутинными заданиями, которые могут привести к ошибкам или несчастным случаям из-за снижения концентрации с течением времени. Также широкое применение роботы получили в работе, которую люди могут найти унизительной.

Медицина

- Медицинская диагностика
- Компьютерная интерпретация медицинских изображений. Такие системы помогают сканировать цифровые изображения, например от компьютерной томографии, для типичных проявлений и для выделения заметных отклонений, таких как возможные заболевания. Типичным применением является обнаружение опухоли.
- Анализ сердечного ритма
- Проект Watson — это ещё одно использование ИИ в этой области, программа вопросов/ответов, которая создана для помощи врачам-онкологам
- Роботы-помощники для ухода за престарелыми
- Обработка медицинских записей для предоставления более полезной информации
- Создание планов лечения
- Выявление повышенного риска заболеваний
- Помощь в повторяющихся заданиях, включая управление приёмом медикаментов
- Предоставление консультаций
- Создание лекарств
- Использование человекоподобных манекенов вместо пациентов для клинического обучения

В настоящее время в отрасли здравоохранения работает более 90 стартапов, основанных на применении ИИ.

Иные области применения

- Управление человеческими ресурсами и рекрутинг
- Музыка
- Новости, издательство и писательство
- Новости, издательство и писательство
- Онлайновые и телефонные службы поддержки клиентов

- Техническое обслуживание телекоммуникаций
- Развлечение и игры
- Транспорт
- Спецслужбы
- Политика

Примеры использования ИИ

Автомобили битком набиты системами УИИ, от компьютеров, которые определяют, когда должна заработать антиблокировочная тормозная система, до компьютера, который определяет параметры системы впрыска топлива. Самоуправляемые автомобили Google, которые в настоящее время проходят испытания, будут содержать надежные системы УИИ, которые будут воспринимать и реагировать на мир вокруг себя.
Ваш телефон — маленькая фабрика УИИ. Когда вы используете приложение карт, получаете рекомендации по скачиванию приложений или музыки, проверяете погоду на завтра, говорите с Siri или делаете что-либо еще — вы используете УИИ.
Спам-фильтр вашей электронной почты — классический тип УИИ. Он начинает с выяснения того, как отделить спам от пригодных писем, а затем обучается в процессе обработки ваших писем и предпочтений.
Google Translate — еще одна классическая система УИИ, впечатляюще хороша в определенных вещах. Распознавание голоса — тоже. Когда ваш самолет садится, терминал для него определяет не человек. Цену билета — тоже. Лучшие в мире шашки, шахматы, нарды, балды и прочие игры сегодня представлены узконаправленными искусственными интеллектами.
Поиск Google — это один гигантский УИИ, который использует невероятно хитроумные методы для ранжирования страниц и определения результатов поисковой выдачи.

Основные проблемы ИИ

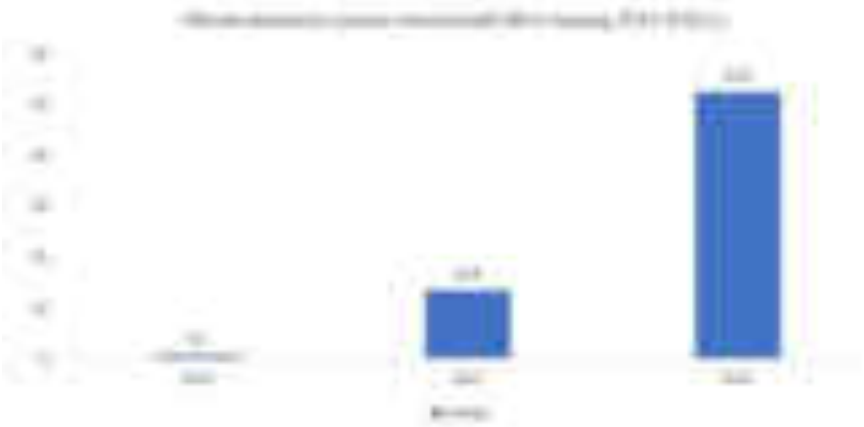
Обучение машин возможно только на основе массива данных. Это означает, что любые неточности в информации сильно сказываются на конечном результате.

Интеллектуальные системы ограничены конкретным видом деятельности. То есть умная система, настроенная на выявление мошенничества в сфере налогообложения, не сможет выявлять махинации в банковской сфере. Мы имеем дело с узкоспециализированными программами, которым ещё далеко до многозадачности человека.

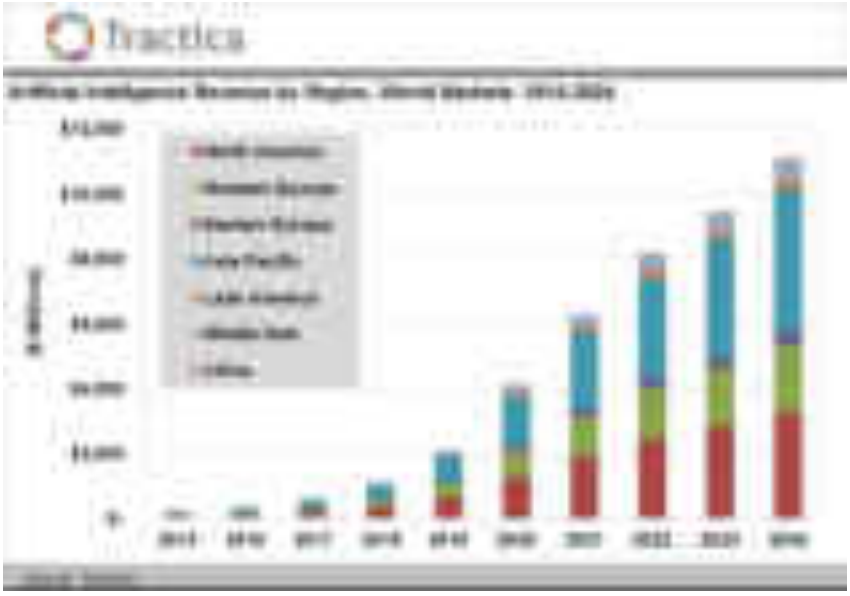
Интеллектуальные машины не являются автономными. Для обеспечения их «жизнедеятельности» необходима целая команда специалистов, а также большие ресурсы.

Обзор рынка

По оценкам аналитиков международной консалтинговой компании Frost & Sullivan, к 2022 году суммарный объем рынка технологий ИИ увеличится до $52,5 млрд, или в 4 раза по сравнению с уровнем 2017 года ($13,4 млрд). Ежегодный темп роста (CAGR) в прогнозируемый период будет сохраняться на уровне 31%. Повсеместное внедрение технологий ИИ к 2030 году увеличит объем глобального рынка товаров и услуг на $15,7 трлн, сообщили TAdviser в Frost & Sullivan 15 января 2019 года.



Уровень развития ИИ в разных странах



Перспективы развития искусственного интеллекта

Современные компьютеры приобретают все больше знаний и «умений». Скептики же утверждают, что все возможности ИИ – не более чем компьютерная программа, а не пример самообучения. Однако это не мешает технологии широко распространяться в самых различных сферах и открывать невиданные ранее потенциалы для развития. Со временем компьютеры будут становиться все мощнее, а ИИ еще быстрее совершенствоваться в своем развитии.

А что с рынком труда? Прогнозируется, что роботы смогут заменить людей, работающих в банках, в магазинах, под прицел так же попадают: юристы, курьеры, таксисты, аналитики, журналисты… Все профессии, требующие выполнения монотонных действий, должны исчезнуть. Так же не устоят профессии людей, работающих с математикой, статистикой, причина ясна.

Однако, Элон Маск не считает, что людям негде будет работать. По его словам, со смертью одних профессий придёт много других. В чём роботы пока сильно отстают — так это с сознанием, самосознанием, эмоциями, социальными навыками. И неизвестно, появится ли у них сознание. Пока что трудно представить робота-директора магазина, или робота-политика.

Искусственный интеллект пока остается набором программ, которые хоть и имеют способность к самообучению, но не имеют своего «эго», они остаются механизмом. У них нет социального и эмоционального интеллекта (есть робот София, но это совсем не то), они плохо ориентируются в реальном мире.

Сможет ли механизм догнать создателя? Или AI станет чем-то вроде нового универсального инструмента? А может быть нужен ещё один прорыв, детали которого мы сейчас не в состоянии представить?

ПРАКТИЧЕСКИЕ (СЕМИНАРСКИЕ) ЗАНЯТИЯ

Номер	Наименование практического (семинарского) занятия
1	**Тест Тьюринга и интуитивный подход** *Методические рекомендации* Кроме обсуждения исторического аспекта создания текста и особенностях его прохождения, необходимо рассмотреть вопрос о возможностях, которыми должна обладать система, чтобы пройти этот тест. Провести имитационная игру согласно описанию Тьюринга в статье «Вычислительные машины и разум». Игрок С путём задавания серии вопросов пытается определить, кто из двух других игроков — мужчина, а кто — женщина. Игрок А, мужчина, пытается запутать игрока С, а игрок В пытается помочь С. *Вопросы:* 1. Согласны ли вы, что тест Тьюринга позволяет дать ответ на вопрос «является искусственная система разумной или нет?» 2. Какие вопросы задали бы вы в этом тесте? 3. Оказал ли тест Тьюринга влияние на процесс разработки искусственных интеллектуальных систем? 4. Какие недостатки видите вы в этом тесте? 5. Какой тест предложила бы вам интуиция сегодня? *Источники:* • Интуитивный подход и тест Тьюринга | Другая фаза | Яндекс Дзен (yandex.ru) • https://ru.wikipedia.org/wiki/Тест_Тьюринга
2	**Биологическое моделирование искусственного интеллекта** *Методические рекомендации* Сделать упор на интересном направлении биокомпьютинге — биологическом направлении в искусственном интеллекте, сосредоточенным на разработке и использовании компьютеров, которые функционируют как живые организмы или содержат биологические компоненты, так называемые биокомпьютеры. *Вопросы:* 1. Что лежит в основе квазибиологической парадигмы? 2. Приведите примеры использования нейронных сетей. 3. Приведите примеры использования генетического подхода. 4. Что можно отнести к понятию интеллектуальный агент? Компьютерные вирусы, боты, поисковые роботы. 5. Что понимают под моделированием сознания? 6. Можем ли мы сегодня сказать, что знаем как работает мозг? 7. Каковы основные преимущества нейрокомпьютеров?

	Источники: • https://ru.wikipedia.org/wiki/Биокомпьютинг
3	## Представление и использование знаний *Методические рекомендации* Составить для одной предметной области разные модели знаний, представленные продукционными правилами, семантическими сетями, фреймами. Провести анализ полученных моделей. Рассмотреть подробнее технологию семантической паутины, в которой основанные на XML языки представления знаний используются для увеличения доступности компьютерным системам информации, хранящейся в сети. *Вопросы:* 1. Какой экспертной системой вы хотели бы воспользоваться? Кого вы видите в качестве эксперта? 2. Как происходит вывод в экспертной системе? 3. Какие методы ИИ положены в основу data mining? 4. Где находят применение семантические сети? 5. Какие знания считаются релевантными? 6. Какие языки программирования ориентированы на представление знаний? 7. Приведите пример логической модели представления знаний. *Источники:* • https://neerc.ifmo.ru/wiki/index.php?title=Представление_знаний • https://ru.wikipedia.org/wiki/Инженерия_знаний • https://ru.wikipedia.org/wiki/Представление_знаний
4	## Работа с естественными языками *Методические рекомендации* Активно использовать различные программные средства для демонстрации наглядных успехов в этой области ИИ. *Вопросы:* 1. Как вы лично оцениваете успехи ИИ в этой области? Часто ли приходится пользоваться? Какое из направлений вам более знакомо? 2. Как используется NLP в чат-ботах? 3. Почему речь компьютера немного отличается от человеческой? 4. Как вы считаете, методы анализа текстов сильно зависят от языка? 5. Как используются регулярные выражения для обработки текста? *Источники:* • https://ru.wikipedia.org/wiki/Обработка_естественного_языка • https://habr.com/ru/company/Voximplant/blog/446738/
5	## Символьное моделирование мыслительных процессов

	Методические рекомендации Сформировать общее представление по всем направлениям моделирования рассуждений - доказательство теорем, принятие решений и теория игр, планирование и диспетчеризация, прогнозирование. Привести как можно больше примеров, иллюстрирующих правила логических выводов. *Вопросы:* 1. В чем отличие синтаксического и семантических методов доказательства в логике? 2. Перечислите известные вам методы решения задач. 3. Какие логические виды и методы использует традиционная логика при получении новых знаний? (анализ, синтез, дедукция, индукция, сравнение, аналогия, …) 4. Достаточно ли дедукции для моделирования аспектов человеческих рассуждений? 5. Что входит в состав простого суждения? 6. Дедуктивные и индуктивные модели формализации знаний. В чем отличие? *Источники:* • Моделирование рассуждений (raai.org) • https://fil.wikireading.ru/92133 • http://csaa.ru/predstavlenie-znanij-v-sistemah-iskusstvennogo/
6	## Робототехника *Методические рекомендации* Сформировать представление о роботе как об интеллектуальной системе, для построения которой необходимы знания из механики, электроники, кибернетики, мехатроники, информатики, … Рассмотреть на примере робота-пылесоса его компоненты, систему управления, интерфейс, … *Вопросы:* 1. На знаниях из каких областей строится современная робототехника? 2. Грозит ли нам восстание машин под предводительством ИИ? 3. Не начнется ли деградация людей, если мы научим компьютер думать за нас? 4. Каковы угрозы, которых мы не ожидали от искусственного интеллекта? 5. Какого робота вы бы хотели иметь дома? Как он должен выглядеть? Какими функциями обладать? 6. Где, по вашему мнению, скоро робот заменит человека? 7. Какие специальности скоро могут исчезнуть из-за роботизации? *Источники:*

	• https://ru.wikipedia.org/wiki/Робототехника
7	## Машинное обучение (МО) *Методические рекомендации* Основной акцент сделать на общей постановке задачи обучения по прецедентам и способам МО. *Вопросы:* 1. Назовите области использования МО. 2. Чем обусловлен необычайный успех machine learning? 3. В чем отличие дедуктивного и индуктивного обучения? 4. Какие классические задачи решаются с помощью МО? 5. Какие виды входных данных используются при МО? 6. Как определить качество обучения? 7. Что такое глубокое обучение? *Источники:* • https://ru.wikipedia.org/wiki/Машинное_обучение
8	## Машинное творчество *Методические рекомендации* Привести примеры написания компьютером музыки, литературных произведений, художественное творчество. Рассмотреть вопросы технического творчества (ТРИЗ). *Вопросы:* 1. Что есть творчество? 2. Какие задачи, которые решаете вы или ваши друзья, можно считать творческими? 3. Как именно искусственный интеллект выстраивает творческий процесс? 4. Какие технологии искусственного интеллекта использует машинное творчество? 5. Можно ли считать произведение, созданное машиной, настоящим искусством? *Источники:* • https://www.theguardian.com/artanddesign/2016/apr/05/new-rembrandt-to-be-unveiled-in-amsterdam • https://www.kv.by/post/1049607-mashinnoe-tvorchestvo • https://www.orange-business.com/ru/blogs/iskusstvennii-intellekt-i-tvorchestvo

Перечень дополнительных вопросов для собеседования

1. Что такое технологии искусственного интеллекта?
2. В каких сферах ИИ применяется уже сейчас?
3. Каковы социальные последствия массового внедрения технологий ИИ?
4. Каковы правовые аспекты внедрения ИИ? Регулируется ли это законодательством?
5. Кто должен нести ответственность за действия искусственного интеллекта?
6. Как выглядит Россия в мировом рейтинге по ИИ?
7. Какие стандарты создания и применения искусственного интеллекта (ИИ) существуют в мире и России?
8. Приведите пример логической модели представления знаний
9. Что такое «Алгоритмическая торговля»? Где и когда она находит применение?
10. Как используется ИИ для управления личными финансами?
11. Дайте понятия сильного и слабого искусственного интеллекта
12. Назовите требования к созданию сильного искусственного интеллекта
13. Назовите основные направления развития ИИ.
14. Что понимают под моделированием рассуждений? Что входит в это направление?
15. В чем суть агентно-ориентированного подхода в ИИ?
16. Какова главная особенность символьных вычислений?
17. Что такое оптические нейронные сети?
18. Приведите примеры применения искусственного интеллекта в области финансов
19. Приведите примеры применения искусственного интеллекта в военном деле
20. Приведите примеры применения искусственного интеллекта в области медицины
21. Приведите примеры применения искусственного интеллекта в области тяжелой промышленности
22. Приведите примеры применения искусственного интеллекта в области транспорта

САМОСТОЯТЕЛЬНАЯ РАБОТА ОБУЧАЮЩИХСЯ

Виды самостоятельной работы распределяются в течение семестра. Подготовка к промежуточной аттестации ведется в установленные календарным учебным графиком сроки.

Темы для самостоятельной работы

1. Методология создания онтологий, основанная на системах представления декларативных знаний
2. Экспертная система MYCIN для диагностирования небольшого набора заболеваний
3. Генетические алгоритмы
4. Гибридные интеллектуальные системы
5. Универсальный решатель задач Ньюэлла и Саймона
6. Виртуальный личный помощник
7. Рекомендательные системы
8. Контекстно-зависимые вычисления
9. Представление знаний и вывод на знаниях
10. Молекулярный компьютер
11. Наноробот
12. Проект по компьютерному моделированию головного мозга человека Blue Brain Project
13. Перцептрон - математическая или компьютерная модель восприятия информации мозгом
14. Интеллектуальный анализ данных Data mining
15. Модели представления знаний

ЗАЧЕТ

Вопросы к зачету

1. Определения искусственного интеллекта.
2. Происхождение и понимание термина «искусственный интеллект».
3. Философские предпосылки к возникновению науки.
4. Технологические предпосылки к возникновению науки.
5. История развития искусственного интеллекта в СССР и России.
6. Национальная стратегия развития искусственного интеллекта.
7. Нейрокибернетика и кибернетика «чёрного ящика».
8. Эволюционный подход. Может ли машина мыслить. Тест Тьюринга.
9. Символьный подход.
10. Логический подход.
11. Подход, основанный на использовании интеллектуальных агентов.
12. Сильный и слабый искусственный интеллект. Усиление интеллекта.
13. Моделирование рассуждений.
14. Обработка естественного языка.
15. Экспертные системы.
16. Машинное обучение.
17. Нейронные сети.
18. Интеллектуальная робототехника.
19. Известные ИИ-системы. Примеры эффективного применения систем искусственного интеллекта.
20. Финансы. Медицина. Военное дело. Промышленность. Развлечение и игры. Связь с другими науками и явлениями культуры.

ПЕРЕЧЕНЬ УЧЕБНОЙ ЛИТЕРАТУРЫ И РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ»

Рекомендуемый перечень учебной литературы:

1. Искусственный интеллект: современный подход, 4-е издание. Том 1. Решение проблем: знания и рассуждения | Рассел Стюарт, Норвиг Питер, изд-во Диалектика-Вильямс, 2020
2. Девятков В. В. Системы искусственного интеллекта / Гл. ред. И. Б. Фёдоров. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. — 352 с. — (Информатика в техническом университете)
3. Нильсон Н. Искусственный интеллект. — М.: Мир, 1973
4. Бруссард Мередит. Искусственный интеллект. Пределы возможного. Изд-во Альпина нон-фикшн, 2020
5. Компьютер учится и рассуждает (ч. 1) // Компьютер обретает разум = Artificial Intelligence Computer Images / под ред. В. Л. Стефанюка. — Москва: Мир, 1990
6. Методы бизнес-анализа / Пол Тернер, Джеймс Кадл, Дебра Пол
7. Ключевые показатели эффективности. 75 показателей, которые должен знать каждый менеджер / Бернард Марр.

Ресурсы информационно-телекоммуникационной сети «Интернет», полезные для самостоятельной работы.

1. История искусственного интеллекта https://ru.wikipedia.org/wiki/История_искусственного_интеллекта
2. Искусственный разум: от философии до нейрона. http://neural.narod.ru/Main.htm
3. Моделирование рассуждений Д.А. Поспелов https://diary.ru/~Organon/p21769784.htm
4. Общее строение искусственного разума http://neural.narod.ru/Part2.htm
5. Анатолий Гершман. Заблуждения искусственного интеллекта. http://postnauka.ru/faq/80051
6. Искусственные нейронные сети http://bigor.bmstu.ru/?cnt/?doc=NN/base.cou
7. Введение в нейронные сети: Курс Интернет-университета информационных технологий http://www.intuit.ru/department/ds/intneuronnets/
8. Российский научно-исследовательский институт искусственного интеллекта (РосНИИ ИИ) http://www.artint.ru

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.В.ДВ.03.02 Параллельное и распределенное программирование

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Параллельное и распределенное программирование

Конспект лекций

Назаркин О.А.

2021

Раздел 1. Виды и уровни параллельности, их реализация в общедоступных компьютерных архитектурах и системах программирования

Лекция 1

1.1 Введение. Обоснование необходимости и ограничения параллельных вычислений

Максимально быстрое решение пользовательских задач декларируется как главная цель конструкторов вычислительных систем, системных программистов, разработчиков языков программирования и компиляторов [1]. Мы исходим из предположения, что вычислительные действия необходимо выполнить как можно быстрее, со скоростью получения результатов связана какая-либо польза.

Чем быстрее производятся вычисления, тем с большим их объемом можно справиться за какие-либо отрезки календарного времени (секунды, минуты, часы, дни, недели, месяцы, годы, десятилетия и т.д), в течение которых задачи *сохраняют актуальность*. Например, при повышении быстродействия процедур численного моделирования атмосферных процессов для краткосрочного прогноза погоды можно учесть в уравнениях больше факторов и повысить детальность моделирования, взяв сетку с меньшим размером ячеек – все это приведет к повышению точности прогнозирования погоды, но потребует обработки огромных многомерных массивов данных.

Кроме того, часто становится возможным сократить время решения конкретной задачи. Например, радикальное ускорение перебора вариантов сложных химических соединений при разработке лекарств приведет к более быстрому завершению исследований и ускорит вывод новых препаратов на рынок.

Скорость решения задач зависит как от быстродействия вычислительной техники, так и от используемых алгоритмов. На скорость решения влияет общее число выполняемых операций, поэтому некоторые алгоритмы имеют преимущество – они построены таким образом, что решают задачу за меньшее количество шагов. К алгоритмической сфере относится сама идея *параллельного выполнения* шагов алгоритма: несколько совмещенных во времени операций можно рассматривать как один шаг в общей последовательности этапов решения (рис. 1.1.1).

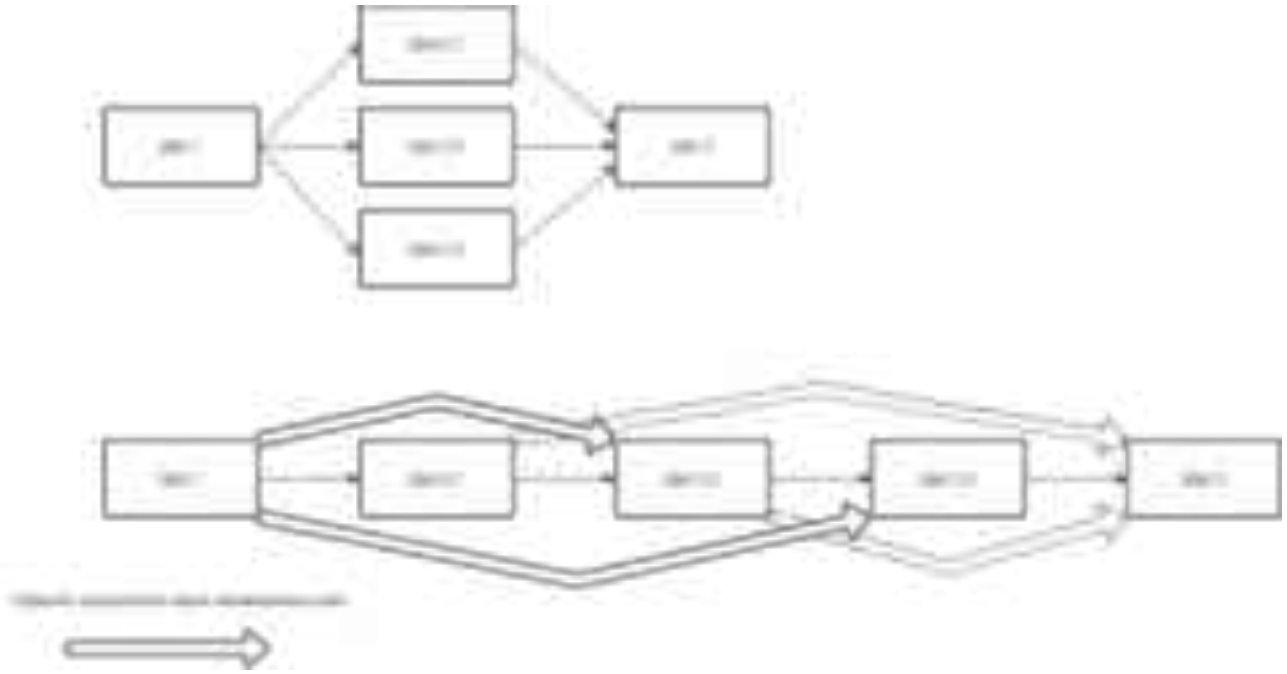


Рис. 1.1.1. Параллельная и последовательная версии алгоритма

Совмещение шагов алгоритма возможно не всегда, так как требуемые действия, в общем случае, связаны зависимостями, не позволяющими приступить к следующему шагу, пока не получены результаты предыдущих (таков Шаг 3 на рис. 1.1.1). Приведенная схематическая иллюстрация была построена сначала для параллельного варианта, а затем он был трансформирован в последовательный, с явным указанием зависимостей по данным. На практике нередко, если не в большинстве случаев, поступают наоборот: имеется последовательная версия алгоритма, требуется выделить в ней зависимости и попытаться совместить во времени независимые шаги.

Задачи, поддающиеся параллельной декомпозиции, встречаются повсеместно. Некоторые из них, с одной стороны, удобно и эффективно распараллеливаются из-за почти полного отсутствия зависимостей, а с другой стороны, чисто последовательные подходы к их решению демонстрируют на практике неприемлемо низкую производительность. Например, изменение яркости цифровой фотографии сводится к применению некоторой достаточно простой функции преобразования ко всем пикселям кадра (их могут быть миллионы в изображениях высокого качества). Последовательный процессор будет вынужден сделать миллионы вызовов этой функции, обрабатывая пиксели один за другим. Графические процессоры, построенные, как правило, на основе вычислительной архитектуры с массовым параллелизмом, обладают тысячами вычислительных ядер и способны одновременно применять функцию к блокам, содержащим до нескольких тысяч пикселей. Можно считать, что в этом случае логическая длина *последовательности действий* сокращается в тысячи раз.

Параллельная организация – совмещение действий во времени – является наиболее естественной формой деятельности. “Реальный мир во всех его проявлениях параллелен, а современный компьютерный мир по своей природе последователен: данные передаются по последовательным каналам, команды выполняются одна за другой, как следствие любые

попытки распараллеливания и адаптации к условиям реального мира рождают чрезвычайно сложные и искусственные решения” [2].

Последовательная форма действий может выглядеть по меньшей мере странно: предположим, требуется собрать некий агрегат C из заказных модулей A и B, которые производятся на разных заводах; тогда можно выдать одному заводу заказ на изготовление модуля A, дождаться его получения, затем выдать заказ другому заводу на изготовление модуля B, дождаться его получения, затем собрать агрегат. Но зачем сначала дожидаться A и только потом заказывать B? Гораздо эффективнее сразу выдать заказы двум заводам и ждать, пока будут готовы оба модуля – они будут производиться параллельно, с момента получения заказов.

Можно представить себе соответствующую программу:

```
      // изготовить модуль A
a = FactoryA.produceItem(specA)
      // изготовить модуль B
b = FactoryB.produceItem(specB)
      // собрать агрегат из модулей A и B
c = composeItems(a, b)
```

Эта плохо замаскированная *последовательность* действий выглядит достаточно привычно; тем не менее, мы просто описали на псевдокоде тот способ производства агрегата из модулей, который выше, при словесном описании, был признан неэффективным. Излишне говорить, что рассматриваемая задача может быть переформулирована в весьма общих терминах, обобщена на произвольное количество составных частей. Паттерн неоправданной последовательной композиции объектов “шаг за шагом” окажется не таким уж редким явлением на практике.

Скорее всего, менее понятной следует признать параллельную версию, совмещающую те же самые действия во времени для устранения ненужных ожиданий:

```
      // подготовить набор операций
actions = [FactoryA.produceItem, FactoryB.produceItem]
      // подготовить набор аргументов операций
arguments = [specA, specB]
      // параллельно вызвать операции со своими аргументами
results = parallel_map(actions, arguments)
      // собрать агрегат из результатов параллельных операций
c = composeItems(results[0], results[1])
```

Почему же “менее естественная” форма компьютерных программ – последовательная – возникла раньше параллельной и распространилась более широко? Сказать, что большинство практически применяемых алгоритмов связаны жесткими, неустранимыми зависимостями по данным от инструкции к инструкции, будет, мягко говоря, неверным. Даже наоборот, строго последовательные *элементарные* шаги (соответствующие процессорным инструкциям) требуются для относительно небольшой доли задач.

Причина здесь не только в отсутствии в ранних ЭВМ нескольких АЛУ, которые могли бы функционировать одновременно. В исторической перспективе многопроцессорные системы появились позднее и стоили дороже однопроцессорных, но не это главное. По крайней мере, в настоящее время многоядерные процессоры (с количеством ядер порядка 10) достаточно доступны и широко распространены – но можно ли говорить о достижении идеальных условий для повсеместного применения программистами параллельных вариантов решения вычислительных задач?

В.В. Воеводин [1] выделяет чрезвычайно важный, но неочевидный влияющий фактор:
степень согласованности структуры алгоритмов с аппаратной архитектурой вычислительных устройств.

Дело в том, что последовательные вычислительные системы более универсальны и предоставляют очень удобный фреймворк для программирования. Любой параллельный алгоритм, при необходимости, можно выполнить последовательно. Зато проектировать последовательные алгоритмы проще. При этом, скорее всего, они будут содержать меньше ошибок. Прикладные программы редко проектируются сразу в параллельной форме, либо параллельная декомпозиция производится весьма поверхностно. Если решаемые при этом задачи не упираются в ограничения по скорости вычислений, то хорошо отлаженные последовательные версии алгоритмов сохраняются на протяжении значительных эволюционных периодов. Возможно, такая практика и не является правильной, но де-факто она доминирует в разработке ПО.

Приведем пример алгоритма, который позволяет выявить сложности и особенности параллельного программирования: умножение матриц. Последовательную версию алгоритма умножения двух матриц легко составит даже начинающий программист. Алгоритм же параллельного умножения матриц выставляет многочисленные “подводные камни”, несмотря на то, что задача очень хорошо поддается параллельной декомпозиции.

При умножении матрицы m x n на матрицу n x k нужно вычислить m x k скалярных произведений векторов длиной n. Все эти скалярные произведения независимы друг от друга и могут вычисляться параллельно. Более того, при вычислении скалярных произведений n умножений компонентов векторов независимы друг от друга и также могут выполняться параллельно. Суммирование при вычислении скалярных произведений имеет зависимость от результатов умножений, но все равно для $n > 3$ может совмещаться во времени, так как компоненты суммы независимы друг от друга.

1	2	3	4
5	6	7	8

x

9	10
11	12
13	14
15	16

=

1*9 + 2*11 + 3*13 + 4*15	1*10 + 2*12 + 3*14 + 4*16
5*9 + 6*11 + 7*13 + 8*15	5*10 + 6*12 + 7*14 + 8*16

Здесь параллельно можно выполнить все 2 x 2 x 4 = 16 умножений. Допустим, в наличии имеется 16 свободных процессоров, так что каждому достанется вычислить по одному произведению. После вычисления произведения процессор освобождается и может заняться суммированием.

Тут возникает затруднение: с чем суммировать? Сложение требует двух слагаемых, но у каждого из процессоров пока есть только одно. Надо как-то получить второе слагаемое от другого процессора. (От какого? Как к нему обратиться? Где гарантии, что у него уже готов результат?) Зависимость по данным неизбежно приводит к необходимости *межпроцессорной коммуникации*.

Сделаем (чрезмерно оптимистичное) предположение, что аппаратная архитектура вычислительной системы идеально подходит для межпроцессорного обмена данными, и этот обмен не вносит никаких задержек. Тогда передача результатов от одного процессора к другим может осуществляться через аппаратно реализованные очереди (рис. 1.1.2).

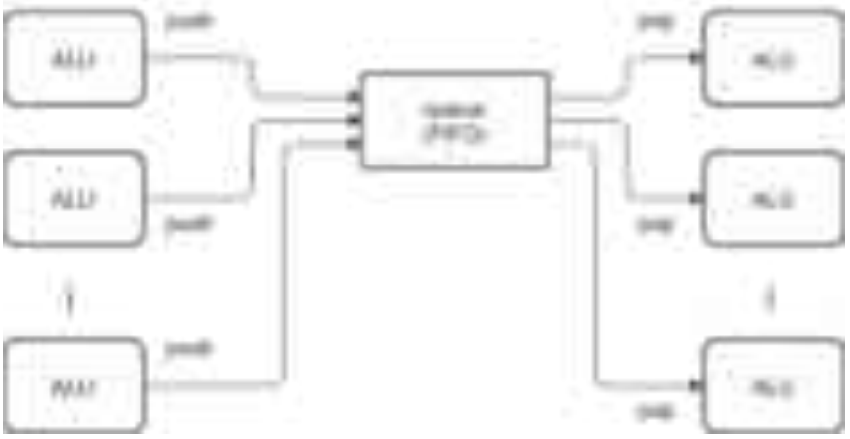


Рис. 1.1.2. Применение очередей для межпроцессорных коммуникаций

Параллельное суммирование при вычислении скалярного произведения требует выборки из очереди сразу двух элементов-слагаемых. Будем считать, что в системном фреймворке присутствует функция `pop(n)`, “мгновенно” извлекающая из очереди `n` элементов или приостанавливающая вызвавший ее процессор, пока в очереди не появятся элементы в необходимом количестве. Элементы помещаются в очередь функцией `push(el)`, выполняемой “мгновенно”, то есть в том же такте, когда вычислено значение `el`.

На рис. 1.1.2 блоки ALU, изображенные слева и справа, могут быть одними и теми же устройствами (слева – до выполнения операции, справа – после). Иными словами, *одна и та же очередь может служить как приемником, так и источником данных*.

Это обстоятельство позволяет во многих случаях избавиться от циклов в программе, если выполняются ассоциативные операции (сложение, умножение, min, max и другие). Идея заключается в том, что при выполнении ассоциативных операций группировка операндов может быть разной, и к выполнению операций можно приступать по мере готовности достаточного количества операндов. Ассоциативное выражение вычисляют каскадно,

группируя на новых стадиях ранее сгруппированные операнды – следовательно, в общей очереди могут присутствовать как исходные данные, так и частичные результаты. Процессоры, которым сразу не достались исходные данные, имеют шанс позже получить из очереди частичные результаты, но выполняемая при этом всеми процессорами операция *одна и та же*. (Сравните этот подход с последовательной итерацией: в цикле повторяется *одно и то же* тело; попробуйте представить, что на каждую итерацию цикла выходит *новый* процессор.)

На основе разделяемой очереди межпроцессорных коммуникаций можно построить схему редукции при суммировании набора из *n* чисел. Каждый процессор пытается извлечь из очереди по 2 элемента. Если в очереди нет двух элементов, обратившийся к очереди процессор блокируется в ожидании. При наличии в очереди не менее двух элементов процессор получает два слагаемых, вычисляет их сумму и помещает результат обратно в очередь. Каким-то из процессоров достанутся исходные числа, каким-то (позже) – частичные суммы. Количество процессоров не должно быть меньше *n - 1*, иначе в программе потребуется цикл. Процессоры не простаивают, кроме вынужденных периодов отсутствия готовых для обработки данных.

Рассмотрим подробнее эту схему при вычислении суммы a + b + c + d. Пусть параллельно работают 3 ALU, ровно по количеству необходимых сложений. Начальное содержимое очереди соответствует набору слагаемых (Шаг 0). Допустим, что в конкуренции за пару элементов из очереди “победили” ALU1 и ALU2, а для ALU3 исходных чисел не хватило (Шаг 1). Процессоры, которым достались слагаемые, вычисляют частичную сумму (Шаг 2) и помещают свои результаты обратно в очередь (Шаг 3). Для ALU1 и ALU2 программа закончилась, а для ALU3 функция pop(2), наконец, возвращает запрошенные два элемента, поэтому ALU3 получает возможность выполнить сложение (Шаг 4) и поместить результат в очередь (Шаг 5). Все три ALU закончили работу, очередь содержит результат (Шаг 6).

Шаг	ALU1	ALU2	ALU3	Содержимое очереди
0				a, b, c, d
1	pop(2)	pop(2)	pop(2)	пусто
2	a + b	c + d	ждет	пусто
3	push	push	ждет	(a + b), (c + d)
4	закончил	закончил	(a + b) + (c + d)	пусто
5	закончил	закончил	push	((a + b) + (c + d))
6	закончил	закончил	закончил	((a + b) + (c + d))

Как видно, для параллельного суммирования *n* чисел по схеме {pop(2), +, push} необходимо *n - 1* процессоров (по числу операций сложения, которых на единицу меньше количества чисел). Но при вычислении скалярного произведения требуется ровно *n* умножений, и если

совмещать в одном параллельном запуске вычисление произведений с последующим суммированием, то один из процессоров никогда не дойдет до конца своей программы, ему не достанется слагаемых. Поэтому системная функция диспетчеризации параллельного выполнения должна иметь параметр – количество завершенных потоков, после достижения которого задача считается решенной. Если какие-то из запущенных потоков планомерно “зависают” из-за отсутствия данных в очередях, то без этого параметра не будет возвращено управление вызывающей программе.

Кстати о программном интерфейсе диспетчеризации параллельного исполнения. Выше он использовался в виде операции `parallel_map`, принимающей массив функций и массив аргументов функций и возвращающей массив результатов применения функций к своим аргументам. Этот вариант удобен и широко применяется в различных фреймворках параллельного программирования. В нем не конкретизируется количество параллельно запускаемых потоков. Управление возвращается после применения всех функций к своим аргументам, то есть, после окончательной готовности массива результатов.

Для параллельного умножения матриц этот вариант не слишком удобен. Во фреймворках массово-параллельного программирования (CUDA, OpenCL и других) распространена другая форма запуска параллельных потоков: по “бруску”, вырезанному из nd-мерного пространства (рис. 1.1.3 для nd = 3). Очень часто вычислительные задачи связаны с обработкой массивов, матриц и других объектов, представляющих собой агрегатные данные разной размерности. Удобно сопоставить потоки со “своими” элементами многомерного агрегатного объекта, подлежащего обработке, при этом каждый поток получает уникальный nd-мерный идентификатор, используя который он легко получает доступ к “своим” данным. Например, для трехмерного “бруска” идентификаторы потоков будут иметь вид thread_id[row][col][depth] из диапазонов row ∈ range[0], col ∈ range[1], depth ∈ range[2].

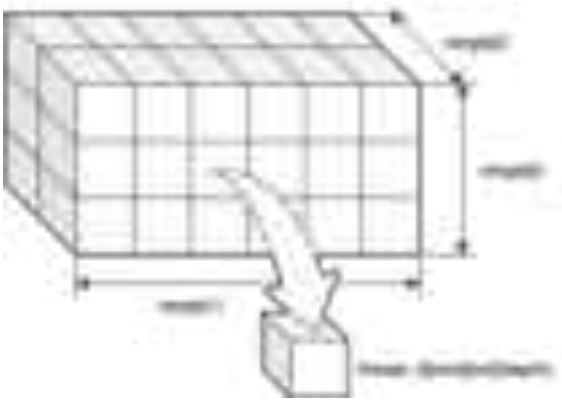


Рис. 1.1.3. Запуск множества параллельных потоков в формате nd-range (трехмерный случай)

В общем случае, задав количество разбиений r_i по отдельным измерениям $i \in [0, nd)$, можно вычислить общее количество потоков (объем “бруска”) $W = \prod_{i=0}^{nd-1} r_i$. Имея “плоский” идентификатор потока $w \in [0, W)$, а также размеры $s_i = \prod_{j=i+1}^{nd-1} r_j$, $s_{nd-1} = 1$ отдельных

слоев “бруска”, можно вычислить значения компонентов nd-мерного идентификатора id по следующему алгоритму:

```
index = w
for(i = 0; i < nd; ++i) {
      id[i] = index / s[i]
      index = index % s[i]
}
```

Будем считать, что фреймворк предоставляет функцию `parallel_map_ndrange`, которая имеет следующие аргументы: 1) функция, которая будет выполняться в параллельных потоках и получит при вызове в качестве аргумента уникальный идентификатор потока, 2) массив из нескольких чисел, каждое из которых обозначает количество разбиений многомерного пространства по соответствующему направлению, 3) параметр, задающий количество потоков, которые должны быть завершены к моменту возврата управления. При необходимости система может вызывать функции потоков частично последовательно; схему параллельного алгоритма этот факт не меняет, поэтому даже для задач большой размерности функции потоков реализуются в предположении, что физические АЛУ доступны в достаточном количестве.

Тогда можно записать на псевдокоде программу параллельного умножения матриц:

```
const m = 2
const n = 4
const k = 2

      // исходные матрицы доступны глобально
M1 = [[1, 2, 3, 4], [5, 6, 7, 8]]
M2 = [[9, 10], [11, 12], [13, 14], [15, 16]]

      // очереди доступны глобально
queues = [[new Queue, new Queue], [new Queue, new Queue]]

function f(thread_id) {
            // определяем координаты своего потока
      row = thread_id[0] // "=" не является дополнительной операцией,
      col = thread_id[1] // а просто обозначает синонимы, для наглядности
      inner = thread_id[2]

            // вычисляем произведение и помещаем в свою очередь
      product = M1[row][inner] * M2[inner][col]
      q = queues[row][col]
      q.push(product)

            // приступаем к суммированию
      el1, el2 = q.pop(2)
            // некоторые процессоры никогда не выйдут из pop(2)

            // вычисляем частичную сумму
      partialSum = el1 + el2
            // помещаем частичную сумму обратно в очередь,
            // давая возможность ожидающим процессорам
```

```
            // получить свои слагаемые
      q.push(partialSum)
}

parallel_map_ndrange(action=f, range=[m, k, n], threadsToWait=m*k*(n-1))

      // извлекаем результаты из очередей
      // (при необходимости можно распараллелить)
Result = [  [queues[0][0].pop(1), queues[0][1].pop(1)],
            [queues[1][0].pop(1), queues[1][1].pop(1)] ]
```

Приведенная программа демонстрирует, что при определенной поддержке со стороны аппаратной архитектуры параллельные вычисления могут быть записаны в достаточно лаконичной форме. По сути, если удалить синонимы, применяемые исключительно для наглядности, то смысловое ядро программы умножения матриц получится едва ли не проще классической последовательной версии с тремя циклами:

```
function f(id) {
      queues[id[0]][id[1]].push(M1[id[0]][id[2]]*M2[id[2]][id[1]])
      el1, el2 = queues[id[0]][id[1]].pop(2)
      queues[id[0]][id[1]].push(el1+el2)
}
parallel_map_ndrange(f, [m, k, n], m*k*(n-1))
```

Современные вычислительные платформы не предоставляют аппаратных средств для поддержки использованных в примере разделяемых очередей межпроцессорных коммуникаций. Некоторое отдаленное подобие можно найти в GPU от NVIDIA с микроархитектурой Ampere – участки разделяемой локальной памяти с синхронизацией доступа к ним потоков-производителей (producers) и потоков-потребителей (consumers); соответствующие средства программной модели – `cuda::pipeline` [4]. Но следует иметь в виду, что в программной модели CUDA потоки не являются полностью независимыми, они работают в соответствии с моделью SIMT (Single Instruction, Multiple Threads, см. далее пункты 1.6 и 2.4), которая существенно отличается от представленных выше приемов параллельного программирования.

Кроме того, в нашем примере совершенно не затрагивались вопросы эффективности доступа к оперативной памяти, особенности кэширования и тому подобные частные аспекты. Между тем, именно они ключевым образом влияют на реально достижимое быстродействие параллельных вычислений.

Для справки, практически применимый подход к параллельному умножению матриц описан, например, здесь:
https://intuit.ru/studies/professional_skill_improvements/2065/courses/190/lecture/4954?page=1

Важно рассмотреть на примере еще одну концепцию параллельного программирования – *атомарные операции (atomic operations)*. Пусть необходимо построить гистограмму целочисленного массива M размерностью *n*, в котором содержатся значения из диапазона [0, L). Алгоритм простой: определяем массив гистограммы H размерностью *L* (по числу уровней

значений во входном массиве), запускаем *n* параллельных потоков, каждый из них считывает из входного массива значение v = M[thread_id] и инкрементирует значение в ячейке гистограммы H[v].

Проблема в том, что инкремент (как вообще любая арифметико-логическая операция) вычисляется в АЛУ, а не в оперативной памяти. Поэтому сначала в регистр АЛУ считывается значение из ячейки памяти, затем значение инкрементируется в регистре, и новое значение из регистра записывается обратно в память. Если один из процессоров прочитал имеющееся на данный момент значение *old*, инкрементировал его, но еще не успел записать в память новое значение *(old + 1)*, то другой процессор может в этот момент взять из памяти "старое" значение и приступить к инкременту. В итоге в память дважды запишется одно и то же новое значение *(old + 1)*, тогда как корректным результатом является *(old + 2)*.

Накопление суммы в ячейках гистограммы не является параллельной операцией, независимой по данным. В самом деле, выполнение `a += b` в параллельных потоках означает, что любой процессор, сделавший один *атомарный* шаг (чтение, сложение, запись), должен передать остальным процессорам свой результат – а те пока вынужденно ожидают. Значит, целевая ячейка `a` должна быть *одноэлементной очередью*, подобной рассмотренным выше очередям межпроцессорной коммуникации, и каждый процессор должен действовать по следующей схеме:

```
      // перед началом работы в очередь Qa занесен 0;
      // пытаемся извлечь один элемент, будь то начальный 0
      // или накопленный к данному моменту частичный результат
a = Qa.pop(1)
      // сейчас в очереди пусто, так что другие процессоры
      // будут ждать появления нашего результата
Qa.push(a + b)
      // теперь другие процессоры могут добавлять свои значения b
```

С учетом этого программа вычисления гистограммы может быть построена так:

```
const n = 10
const L = 3

      // исходный массив доступен глобально
M = [0, 2, 1, 1, 0, 0, 1, 2, 2, 1]

      // очереди доступны глобально
H = new ZeroedQueues(L)
      // после создания каждая из L очередей содержит 0

function f(thread_id) {
            // определяем координаты своего потока
      sourceIndex = thread_id[0]
            // считываем свой элемент массива
      v = M[sourceIndex]
            // извлекаем из очереди накопленное значение
            // (возможно, придется подождать...)
      acc = H[v].pop(1)
```

```
            // инкрементируем и заносим обратно в очередь
      H[v].push(acc + 1)
}

parallel_map_ndrange(action=f, range=[n], threadsToWait=n)

// гистограмма накоплена в очередях,
// при необходимости переносим ее в отдельный массив:

Histogram = Array(L)

function transfer(thread_id) {
      index = thread_id[0]
      Histogram[index] = H[index].pop()
}

parallel_map_ndrange(action=transfer, range=[L], threadsToWait=L)
```

Рассмотрим еще одну программу, использующую идеализированные очереди межпроцессорных коммуникаций для выполнения атомарного суммирования. Пусть требуется отсортировать массив из n элементов с помощью квадратичного алгоритма подсчета. В этом алгоритме для каждого элемента подсчитывается количество других элементов, меньших по значению, или равных по значению, но предшествующих в массиве. Тогда найденное количество будет индексом позиции данного элемента в отсортированном массиве.

Пример неупорядоченных данных:

#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
1	2	30	-1	0	0	10	2	-2	-2

Результаты подсчета (целевые позиции):

5	6	9	2	3	4	8	7	0	1

Результат переноса исходных элементов на целевые позиции:

#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
-2	-2	-1	0	0	1	2	2	10	30

При последовательной реализации квадратичного алгоритма требуется двойной цикл (во внешнем цикле выбирается элемент, во вложенном цикле с ним сравниваются остальные элементы).

При параллельной реализации необходимо создать столько потоков, сколько требуется провести сравнений пар (a, b), $pos(a) \neq pos(b)$, где $pos(\cdot)$ означает номер позиции элемента (неравенством исключены сравнения элементов с собой). Верхняя оценка здесь $(n^2 - n)$, но фактически требуемое количество – в два раза меньше. Будем формировать только пары (a, b), $pos(a) < pos(b)$, то есть сравнивать элементы только с

предшествующими. Если $a \leq b$, то следует инкрементировать значение целевой позиции в ячейке $pos(b)$, то есть сдвигать целевую позицию элемента b вправо. Иначе (если НЕ $a \leq b$), то автоматически получается $b < a$, и следует инкрементировать значение в ячейке $pos(a)$, то есть сдвигать целевую позицию элемента a вправо. Какими бы ни были отношения элементов в паре (a, b), $pos(a) < pos(b)$, в любом случае какая-то целевая позиция будет инкрементирована. Но количество таких пар равно $\frac{1}{2}(n^2 - n)$, так как для другой половины $pos(a) > pos(b)$.

Как с помощью `parallel_map_ndrange` покрыть потоками только "часть бруска"? Нужно либо конструировать функцию `parallel_map_ndrange` таким образом, чтобы она "пропускала" некоторые ячейки в многомерном "бруске" и не создавала для них потоки, либо в программе каждого потока проверить условие фильтрации и сразу завершиться, если этот поток – лишний.

По разным причинам более эффективен первый вариант. Поэтому будем предусматривать модификацию интерфейса `parallel_map_ndrange` с целью поддержки фильтрации множества потоков еще до запуска. При таком подходе надо не забывать корректировать и количество ожидаемых завершенных потоков. Подсчет этого параметра может быть нетривиален для сложных фильтров. Одно из решений – задавать количество ожидаемых завершенных потоков по тривиальному объему "бруска", но отфильтрованные еще до запуска потоки сразу считать завершенными.

С учетом этих замечаний получается следующая программа сортировки:

```
const n = 10

      // исходный массив доступен глобально
M = [1, 2, 30, -1, 0, 0, 10, 2, -2, -2]

      // очереди доступны глобально
targetPositions = new ZeroedQueues(n)
      // после создания каждая из n очередей содержит 0

      // функция, исполняемая в потоках
function f(id) {
            // основной элемент
      v = M[id[0]]
            // второй элемент (выбранный для сравнения с ним основного)
      el = M[id[1]]
            // сравниваем элементы и определяем,
            // в какой из очередей производить инкремент целевой позиции
      q = (el <= v) ? targetPositions[id[0]] : targetPositions[id[1]]
            // приступаем к атомарному инкременту:
            // извлекаем из очереди накопленное значение
            // (возможно, придется подождать...)
      acc = q.pop(1)
            // инкрементируем и заносим обратно в очередь
      q.push(acc + 1)
}
```

```
      // функция фильтрации потоков на этапе запуска
function filter(id) {
      return id[1] < id[0]
}

      // параллельно вычисляем целевые позиции
parallel_map_ndrange(action=f, range=[n, n],
                        ndfilter=filter, threadsToWait=n * n)

      // теперь переносим элементы исходного массива на целевые позиции
Msorted = Array(n)

function transfer(id) {
      v = M[id[0]]
      q = targetPositions[id[0]]
      newPos = q.pop(1)
      Msorted[newPos] = v
}

parallel_map_ndrange(action=transfer, range=[n], ndfilter=null, threadsToWait=n)
```

В этой программе очереди изначально заполнены нулями. При первом инкременте выполняется добавление единицы к нулю – бесполезная, по сути, операция, и она будет выполнена на *n - 1* процессоре, причем, вместе с извлечением нуля из очереди.

Для устранения этой лишней операции можно немного изменить программу. Очереди создаются пустыми. Каждый поток помещает единицу в очередь, выбранную на основе сравнения элементов, а потом приступает к редукции (суммированию) элементов этой очереди, выбирая по 2 элемента, складывая их и отправляя частичную сумму обратно в очередь. Такая схема редукции использовалась выше в программе умножения матриц при вычислении скалярных произведений.

Чтобы нагляднее показать, почему эта общая схема редукции работоспособна в том числе и в алгоритме сортировки, мы сделали во вспомогательной моделирующей программе снимок “ландшафта” записи единиц в свои очереди (рис. 1.1.4) для конкретного сортируемого массива (значения его элементов приведены выше). Здесь прочерки стоят на диагонали матрицы – элементы никогда не сравниваются с собой, эти потоки отфильтрованы до запуска. Заполненные ячейки содержат код пары индексов сравниваемых элементов (id[0], id[1]) и номер очереди, в которую поток с данным двумерным идентификатором помещает единицу. Следует обратить внимание, что все идентификаторы, в том числе, расположенные выше диагонали, имеют id[1] < id[0] (условие фильтра потоков).

Рис. 1.1.4. Распределение записываемых единиц по очередям

Здесь строки соответствуют очередям (от 0 до *n - 1*, т. е. от 0 до 9). Количество заполненных ячеек в каждой строке – это целевая позиция элементов исходного массива: 0-й элемент идет на 5-ю позицию, 1-й элемент – на 6-ю, и т. д. Очередь с индексом 8 пуста – она соответствует минимальному элементу в массиве, он должен стоять на 0-й позиции.

Каждая заполненная ячейка обрабатывается каким-либо потоком (строка и, следовательно, очередь, оказавшаяся пустой, не обрабатывается ни одним потоком). Поэтому если каждый поток будет действовать по схеме {el1, el2 = q.pop(2), q.push(el1 + el2)}, то потоки, “расположенные” на одной строке, в совместной конкуренции за извлечение элементов вычислят сумму единиц в соответствующей очереди. Причем, одному потоку с каждой строки не достанется данных (напомним, для суммирования *k* элементов требуется *k - 1* сложений). Среди них – один из потоков, который “зависнет” на очереди из одного элемента (это позиция #1 в отсортированном массиве, левее ее находится ровно 1 элемент). Итого, “зависнут” *n - 1* потоков; из *n* строк одна обязательно будет пуста, так как в каждом массиве есть минимальный элемент. После запуска “бруска” на выполнение с помощью функции `parallel_map_ndrange` будет необходимо ожидать завершения $n^2 - (n - 1)$ потоков, с учетом соглашения о том, что отфильтрованные потоки входят в число завершенных.

```
      // очереди доступны глобально
targetPositions = new EmptyQueues(n)
      // после создания каждая из n очередей пуста

      // функция, исполняемая в потоках
function f(id) {
            // основной элемент
      v = M[id[0]]
            // второй элемент (выбранный для сравнения с ним основного)
      el = M[id[1]]
            // сравниваем элементы и определяем,
            // в какой из очередей производить инкремент целевой позиции
      q = (el <= v) ? targetPositions[id[0]] : targetPositions[id[1]]
            // заносим в очередь единицу
      q.push(1)
            // приступаем к редукции
            // (возможно, придется подождать...)
      el, el2 = q.pop(2)
            // заносим частичную сумму обратно в очередь,
            // давая возможность ожидающим процессорам
            // получить свои слагаемые
      q.push(el1 + el2)
```

```
}

      // функция фильтрации потоков на этапе запуска
function filter(id) {
      return id[1] < id[0]
}

      // параллельно вычисляем целевые позиции
parallel_map_ndrange(action=f, range=[n, n],
                        ndfilter=filter, threadsToWait=n*n-(n-1))
```

Функция переноса элементов из исходного массива в отсортированный массив должна быть изменена:

```
function transfer(id) {
      v = M[id[0]]
      q = targetPositions[id[0]]
            // искусственно формируем 0 в хвосте очереди;
            // это не повлияет на головы непустых очередей,
            // но позволит избежать блокировки потока на пустой очереди
      q.push(0)
      newPos = q.pop(1)
      Msorted[newPos] = v
}
```

Команда `q.push(0)` добавлена для того, чтобы поток, которому “досталась” пустая очередь, смог, не зависая, перенести минимальный элемент на нулевую позицию. Не следует забывать, что очередь имеет дисциплину доступа FIFO, поэтому “искусственный” ноль заносится в хвост очереди, а содержащиеся в непустых очередях “информативные” элементы извлекаются из головы. И только для изначально пустой очереди после внесения искусственного элемента голова совпадает с хвостом, что и позволит далее извлечь корректную – нулевую – позицию.

Мы надеемся, что рассмотренные примеры дают начальное представление о принципах построения таких параллельных алгоритмов, в которых целью является *максимально возможное совмещение операций во времени*. В [7] подобная концепция называется концепцией *неограниченного параллелизма*. Другой подходящий термин – *ультимативно-параллельные* алгоритмы. Разрыв между архитектурой имеющихся на рынке компьютеров и архитектурой, требуемой для ультимативно-параллельного программирования, очень существенный и проявляется наглядно.

Можно сразу предвидеть критическое и, вероятно, справедливое возражение: “Ни одна реальная система не создает около полутриллиона потоков для сортировки массива из миллиона элементов!” Действительно, не создает; тем хуже для системы! При сортировке массива квадратичным алгоритмом *теоретически допустимо* совмещение во времени столь большого количества операций. Ограничения диктуются только практическими соображениями. Разумеется, квадратичные алгоритмы для сортировки являются не самыми эффективными, но это уже совсем другой аспект конкретной задачи.

В общем случае, объемы *допустимой параллельности* в разных задачах, особенно при больших объемах обрабатываемых данных, могут быть астрономическими, и соответствующие многопроцессорные компьютеры никогда не будут построены. Этот факт, тем не менее, ничуть не сдерживает разработку алгоритмов и программ: при решении задачи всегда следует оценивать ее потенциал для параллельности, даже если целевая вычислительная система не в состоянии реализовать аппаратную поддержку этого потенциала. В большинстве случаев вовсе не обязательно иметь столько физических параллельных потоков, сколько допускает задача. Можно разделить работу между доступным количеством процессоров, при этом каждый из них будет *последовательно* вызывать функции логических потоков.

Как при этом будет выглядеть “ландшафт” состояния вычислений? Безусловно, при “физической параллельности” непредсказуемая конкуренция процессоров за разделяемые объекты с высокой вероятностью приводит к некоторой изменчивости совокупного хода выполнения операций. При выполнении того же алгоритма на физически однопоточной системе без вытесняющей многозадачности дисциплина обращений к разделяемым объектам будет полностью детерминированной (она будет зависеть только от последовательности вызова функций логических потоков). В любом случае, при корректном построении параллельных алгоритмов их “однопроцессорные” прогоны должны приводить к корректным результатам. Добавим, что все приведенные выше программы были проверены на специально разработанном однопоточном эмуляторе.

Тут следует сделать еще одно замечание. Что понимается под “процессором” в рассмотренных примерах? По сути, это очень простое устройство, имеющее счетчик команд, доступ к общей оперативной памяти и к специализированным очередям межпроцессорной коммуникации. Большой регистровый пул не требуется: ультимативно-параллельные алгоритмы редко оперируют большим локальным контекстом потока – в противном случае, эти операции можно было бы с высокой вероятностью распараллелить и разнести контекст по разным потокам.

Блокировка такого гипотетического процессора на командах извлечения данных из очереди может быть реализована на основе запрета инкремента счетчика команд, если из очереди в данный момент не извлечены запрошенные элементы. Сигнал запрета устанавливается и снимается диспетчером очереди, так что при заблокированном инкременте счетчика команд процессор будет в каждом такте пытаться выполнить одну и ту же команду (pop), пока у него не получится продвинуться дальше по программе.

Из-за своей простоты такие процессоры могут быть реализованы в больших количествах. Современная аппаратура специализированных параллельных ускорителей как раз и движется в подобном направлении: радикальное увеличение количества вычислительных ядер за счет их упрощения (рис. 1.1.5).

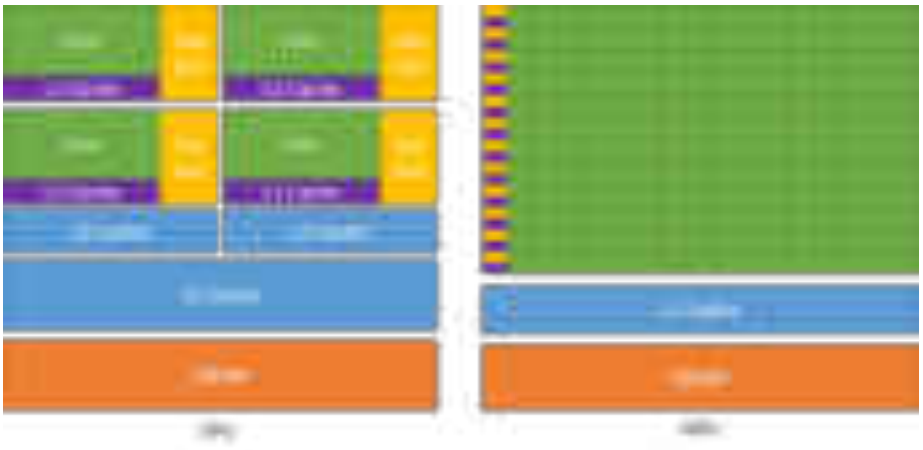


Рис. 1.1.5. Упрощение процессоров позволяет разместить больше вычислительных ядер на той же площади кристалла (источник изображения – NVIDIA [3])

В [5] можно найти краткие описания экспериментальных вариантов аппаратной организации параллельных вычислительных систем, которые по разным причинам не нашли широкого воплощения. Поиски эффективных вариантов аппаратно-программных моделей продолжаются.

Но вернемся от теоретических, идеализированных построений к реальной практической ситуации в сфере высокопроизводительных вычислений.

На общую скорость выполнения программ в первую очередь влияет быстрота выполнения отдельных элементарных инструкций – она определяется тактовой частотой. На достижение высоких тактовых частот были направлены десятилетия усилий специалистов, и достигнутые результаты впечатляют. Например, только для микропроцессоров тактовые частоты увеличились от 1 МГц (1970-е) до 5 ГГц (2020-е). Темпы роста тактовых частот серьезно замедлились в силу технологических и даже чисто физических причин (размеры транзисторов приближаются к размерам одиночных атомов). Зато совершенствование технологии привело к возможности производить многоядерные процессоры в изобилии, сделать их устройствами, доступным повсеместно.

Профессор Массачусетского технологического института Анант Агарвал (https://en.wikipedia.org/wiki/Anant_Agarwal), выступая на конференции, посвященной проблемам современных процессоров, имея опыт собственной разработки в основанной им компании Tilera, сказал следующее: «Процессоры становятся все более и более единообразными и обезличенными, критичнее становится система», а далее подарил афоризм: «Процессор – это транзистор современности» (цитируется по [2], но в сети можно почитать развернутое интервью с Агарвалом [6], датированное еще 2007 годом, многие положения из которого весьма актуальны и в настоящее время).

С другой стороны, на рынке достаточно давно присутствуют специализированные массово-параллельные вычислительные системы (типичный пример – GPU, графические процессоры, применяемые в видеокартах для ускорения 3D-графики, но также и для решения нерафических задач). Бурное развитие таких устройств началось с середины 2000-х, причем,

к рубежу 2020-х годов флагманские GPU демонстрируют впечатляющую производительность вычислений с плавающей точкой.

Наконец, объединение компьютеров, оснащенных многоядерными процессорами и, опционально, массово-параллельными ускорителями, в единую вычислительную сеть со сверхвысокой пропускной способностью и низкими задержками потенциально предоставляет возможности для крупномасштабных высокопроизводительных вычислений. Собственно, все современные суперкомпьютеры строятся именно по схеме гетерогенных кластеров.

Таким образом, аппаратная поддержка для постепенного перехода от последовательных к параллельным вычислениям имеется. Системное программное обеспечение, языки программирования, библиотеки и фреймворки также предоставляют достаточные возможности для организации параллельных вычислений – потоки и процессы, пулы потоков и процессов, очереди сообщений для коммуникации между параллельными задачами, средства синхронизации, барьеры.

Лекция 2

1.2 Виды и уровни параллелизма.

Параллелизм на уровне битов (bit-level parallelism, BLP)

Параллелизм на уровне битов – вид параллельных вычислений, основанный на увеличении размера машинного слова.

С появлением первых микропроцессоров в начале 1970-х годов увеличение размера машинного слова стало основным направлением прогресса при разработки новых моделей. 4-разрядные микропроцессоры сменялись 8-, 16- и 32-разрядными. Каждое удвоение машинного слова снижало количество инструкций, необходимых для обработки данных, имеющих большую длину, чем размер прежнего машинного слова.

Предположим, что 8-разрядный процессор должен сложить два 16-разрядных числа. Тогда он сначала складывает 8 младших бит, затем 8 старших бит, учитывая при выполнении операций необходимые биты переносов. Здесь как минимум 2 операции, а 16-разрядный процессор обходится одной.

Возможности увеличения производительности за счёт увеличения размера машинного слова в основном были исчерпаны с появлением 32-разрядных микропроцессоров. Значительно более позднее появление 64-разрядных микропроцессоров в основном связано с увеличением адресного пространства, а не производительности.

Параллелизм на уровне инструкций (instruction-level parallelism, ILP)

Параллелизм на уровне инструкций – потенциальное совмещение выполнения машинных команд. В потоке инструкций, выполняемых процессором, можно изменить порядок этих

инструкций, распределить их по группам, которые будут выполняться параллельно, без изменения результата работы всей программы.

Есть два подхода к выявлению параллелизма (parallelism extraction) на уровне команд:

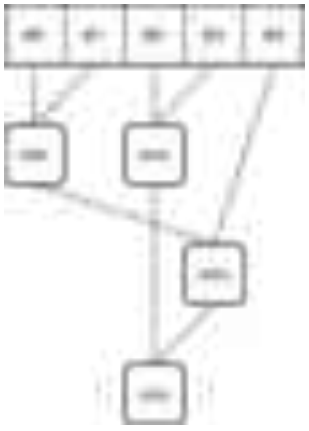
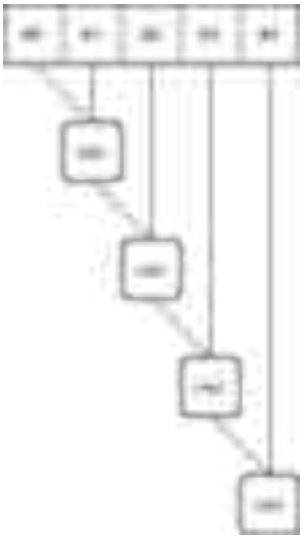
- выявлением параллелизма в потоке операций занимаются аппаратные средства процессора при исполнении кода программ;
- выявлением параллелизма занимается компилятор, который формирует исполняемый код программы под специальный процессор.

Уровень аппаратного обеспечения осуществляет динамический параллелизм, тогда как уровень программного обеспечения реализует статический параллелизм.

Цель разработчиков компилятора и процессора заключается в выявлении параллелизма и получении от него максимального выигрыша. Обычные программы, как правило, написаны под последовательную модель исполнения, где команды выполняются одна за другой в порядке, установленном программистом. ILP позволяет компилятору и/или процессору распараллеливать выполнение нескольких инструкций или даже изменять порядок их выполнения.

Пример: вычисление минимального значения среди нескольких чисел.

```
      // определить минимум в массиве
nums = [12, 10, -20, -3, -1]
```

// вариант 1 n1 = (nums[0] <= nums[1]) ? nums[0] : nums[1] n2 = (nums[2] <= nums[3]) ? nums[2] : nums[3] n3 = (n1 <= nums[4]) ? n1 : nums[4] min = (n2 <= n3) ? n2 : n3	// вариант 2 n1 = (nums[0] <= nums[1]) ? nums[0] : nums[1] n2 = (n1 <= nums[2]) ? n1 : nums[2] n3 = (n2 <= nums[3]) ? n2 : nums[3] min = (n3 <= nums[4]) ? n3 : nums[4]
	

Количество операций в обоих вариантах одно и то же. В варианте 1 определение меньшего из двух значений в ячейках 0 и 1 может выполняться параллельно с определением меньшего из

двух значений в ячейках 2 и 3. В варианте 2 применяется стратегия сравнений, при которой распараллеливание операций невозможно. Доступны и другие варианты, как с параллельными сравнениями, так и со строго последовательными.

Разные программы предоставляют разные возможности для ILP. В некоторых задачах одновременно могут выполняться все операции (например, поэлементное сложение двух матриц C = A + B: $c_{ij} = a_{ij} + b_{ij}$).

Существуют и такие задачи, в которых все операции должны выполняться строго последовательно (например, рекуррентные серии $x_i = f(x_{i-1})$).

Пример:

$x_2 = f(x_1)$
$x_3 = f(x_2)$
$x_4 = f(x_3)$
$x_5 = f(x_4)$

Если промежуточные результаты не сохраняются, то вычисление x_5 представляет собой итерирование функции: $x_5 = f(f(f(f(x_1))))$

Параллелизм на уровне данных (векторизация)

Основная идея подхода, основанного на *параллелизме данных* (data parallelism), заключается в том, что одна операция выполняется сразу (параллельно) над всеми элементами массива данных. Время выполнения векторной инструкции сопоставимо со временем выполнения соответствующей скалярной инструкции (например, за 1 такт скалярный процессор может сложить два скаляра, а векторный – два вектора). В силу аппаратных ограничений длина векторов не может быть велика; при размере элементов 32 бит обычно поддерживаются векторы с количеством элементов от 4 до 8, реже до 16.

Рассмотрим псевдокод для сложения 4-элементных векторов

```
vector4 a = {10, -2, 3, 5}, b = {0, 3, 1, 1};
vector4 c;

c = add4(a, b);

// вариант 1
vector4 add4(vector4 x, vector4 y) {
      // последовательное поэлементное сложение
      vector4 sum;
      for(i = 0; i < 3; ++i)
            sum[i] = x[i] + y[i];
      return sum;
}

// вариант 2
```

```
vector4 add4(vector4 x, vector4 y) {
      // компилируется в подходящую векторную инструкцию
      return simd_add_4(x, y);
}
```

Во втором варианте подразумевается применение аппаратно-реализованной векторной инструкции, которая принимает 4-элементные операнды, суммирует все 4 элемента в параллельном режиме и возвращает 4-элементный результат.

Более подробно векторные операции и процессоры будут рассматриваться в 5.1.

Параллелизм задач (task parallelism)

Параллелизм задач подразумевает одновременное исполнение заданного набора вычислительных процедур в параллельных потоках или процессах. Параллелизм задач характеризуется тем, что одновременно выполняются существенно разные программы. Это отличает параллелизм на уровне задач от параллелизма на уровне данных, так как в последнем случае над разными данными выполняются одни и те же операции.

В общем случае параллельные задачи имеют возможность обмениваться информацией друг с другом во время вычислений. Такая коммуникационная активность обычно снижает эффективность параллельных алгоритмов (возникают неизбежные задержки на передачу и ожидание данных), поэтому наилучшим образом параллелизм задач реализуется для независимых участков вычислений.

Разновидностью параллелизма задач является конвейер (pipelining). В этом случае элементы данных проходят последовательную обработку по разным алгоритмам – стадиям конвейера. Все стадии работают параллельно, поэтому для эффективности такого метода организации вычислений требуется не один входной элемент данных, а достаточно длинная серия элементов (очередь).

Декомпозиция общей задачи на параллельные подзадачи обычно производится разработчиком программного обеспечения. Ни аппаратура процессоров, ни компиляторы не имеют возможности выделить такие блоки кода и организовать коммуникации между ними. Поэтому особую важность в связи с параллелизмом задач приобретают способы параллельной декомпозиции критических участков алгоритмов, влияющих на производительность программы ключевым образом. С другой стороны, широкому использованию параллелизма задач способствуют повсеместное распространение многоядерных процессоров, операционных систем с вытесняющей многозадачностью на основе потоков и процессов, языков программирования и фреймворков с поддержкой параллельного выполнения.

Распределённые вычисления *(distributed computing)*

Термины “одновременные вычисления” (concurrent computing), “параллельные вычисления” (parallel computing), “распределенные вычисления” (distributed computing) пересекаются по

смыслу и не имеют четкого различия обозначаемых понятий. Все они относятся к системам, в которых несколько процессоров одновременно заняты обработкой данных. Распределенные вычисления подразумевают согласованную параллельную работу автономных вычислительных узлов (компьютеров), разнесенных в пространстве и соединенных между собой компьютерной сетью, при этом каждый узел не обязательно имеет прямые физические каналы связи с любым другим узлом сети.

Можно выделить несколько характерных особенностей распределенных вычислительных систем:
1) координация и обмен информацией происходят посредством передачи сообщений по сети, разделяемая физическая память отсутствует;
2) глобальные часы для координации событий во времени отсутствуют;
3) топология сети не обязательно известна заранее и может быть изменчивой;
4) отказы узлов происходят независимо друг от друга и до определенного порогового уровня считаются нормальным явлением;
5) отказы отдельных узлов не означают отказа вычислительной системы в целом и до определенного порогового уровня не должны приводить систему в неработоспособное состояние;
6) ни один из отдельных узлов не имеет полного представления о состоянии всей вычислительной системы;
7) для пользователей распределенная вычислительная сеть должна представляться единой согласованной системой.

1.3 Наблюдение Мура

Закон Мура (англ. Moore's law) – эмпирическое наблюдение, опубликованное 19 апреля 1965 Гордоном Муром, одним из основателей компании Intel, согласно которому (в современной формулировке) количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца (рис. 1.3.1).

Закон Мура сформулирован для роста числа транзисторов, но вместе с ростом числа транзисторов увеличивается и производительность процессоров (правда, не с той же скоростью).

На протяжении нескольких десятилетий производители процессоров постоянно увеличивали тактовую частоту и параллелизм на уровне инструкций, поэтому на новых процессорах старые однопоточные приложения исполнялись быстрее без каких-либо изменений в программном коде.

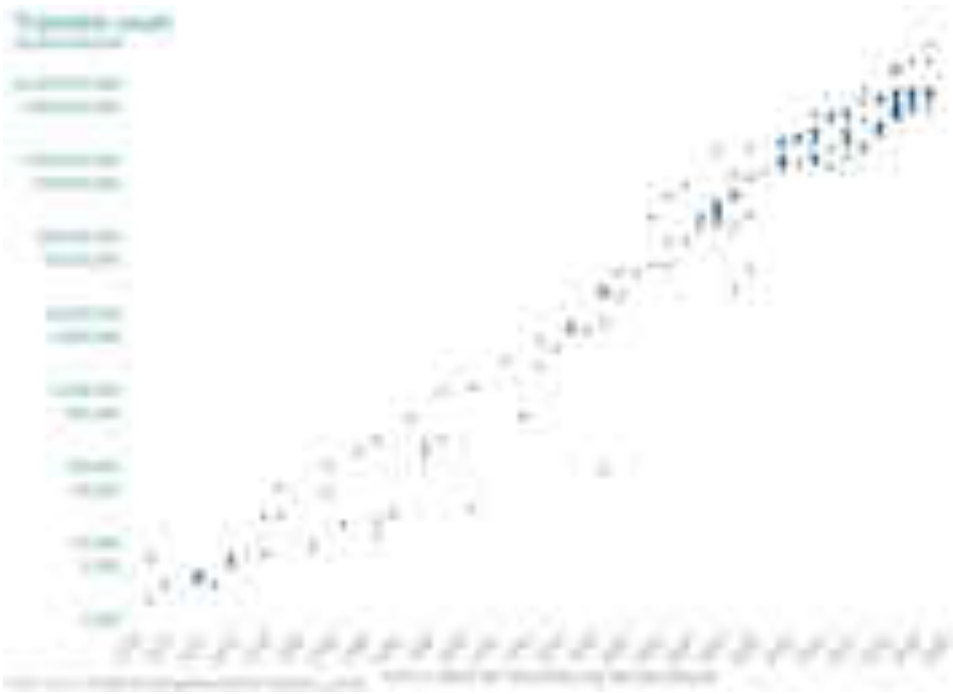


Рис. 1.3.1. Количество транзисторов на кристалле для разных процессоров (1970 – 2020)

Но приблизительно к середине 2000-х годов рост тактовых частот сильно замедлился, а потенциал совершенствования однопоточной микроархитектуры был практически исчерпан. Отчетливо проявились факторы, негативно влияющие на развитие вычислительных систем. Эти барьеры и технологические вызовы принято называть “стенами” :

Название стены	**Ложное, устаревшее убеждение**	**Истинное положение в настоящее время**
Энергетическая стена (power wall)	Энергия дешевая, транзисторы дорогие и каждый на счету	Энергия очень ценная (особенно в мобильных, носимых, встраиваемых системах); стоимость отдельного транзистора близка к нулю, в схемах их миллиарды, так что плюс-минус миллион никто не заметит
Стена памяти (memory wall)	Память работает быстро, почти мгновенно, вычислительные операции, особенно с плавающей точкой, выполняются долго	Системную производительность сдерживает память, вычислительные операции выполняются быстро, почти мгновенно, производительность вычислений с плавающей точкой равна таковой для целых чисел
Стена параллелизма на уровне команд (ILP wall)	Производительность можно повысить за счет качества компиляторов и архитектурных усовершенствований: конвейеров, внеочередного исполнения команд и др.	Производительность повышается за счет правильного и глубокого учета разработчиками программ естественного параллелизма в решаемых задачах; инструкции выполняются параллельно на разных ядрах, выгоднее иметь очень много

		простых ядер, чем мало сложных
Стена ментальности (education wall)	Эффективность паттернов (приемов, наилучших практик) проектирования аппаратного и программного обеспечения; программное обеспечение дорогое, поставить более мощную аппаратуру дешевле, чем полностью переписать кодовую базу.	Многие образовательные догматы, лежащие в основе подготовки IT-специалистов, уязвимы. Программы “ожирели” и стали исключительно неэффективными (в пересчете на отдельный транзистор), но эти недостатки уже нельзя “залить” более мощной аппаратурой – процессоров с необходимой производительностью пока нет в природе, и неясно, как она может появиться. “Развал системы традиционных знаний в многоядерную эру. Все, что вы знаете, – неверно.” (Доклад на суперкомпьютерной конференции, Дрезден, 2007)

Прогресс в сфере GPU идет более высокими темпами, нежели в сегменте обычных центральных процессоров, что даже дало повод для появления неформального и – в отличие от закона Мура – далеко не общепризнанного «закона Хуанга», по имени руководителя компании NVIDIA Дженсена Хуанга (Jen-Hsun «Jensen» Huang).

На самом деле, GPU тоже подвержены серьезным ограничениям, и их развитие не решает всех накопленных проблем высокопроизводительных вычислений. Энергопотребление мощных GPU в абсолютных величинах (сотни ватт) давно стало предметом иронии пользователей, несмотря на очень высокую относительную эффективность – в пересчете производительности на ватт. Но самая серьезная проблема GPU и подобных им ускорителей – сложность программирования, трудности, нередко неразрешимые, оптимального отображения структуры многих алгоритмов на специфическую программную модель. Неоптимальность программы для GPU, как правило, оборачивается драматическим снижением производительности по сравнению с тщательно оптимизированным вариантом.

1.4 Закон Амдала

Закон Амдала (англ. Amdahl's law, иногда также Закон Амдала – Уэра) – иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей. Джин Амдал сформулировал закон в 1967 году, обнаружив простое, но непреодолимое ограничение на рост производительности при распараллеливании вычислений.

Очевидна *теоретическая* (то есть, не связанная с уровнем технологий производства процессоров) причина, по которой совмещение операций во времени не может обеспечить бесконечное ускорение: в случае, когда задача разделяется на несколько этапов, общее время

выполнения не может быть меньше времени выполнения самого медленного этапа (рис. 1.4.1).

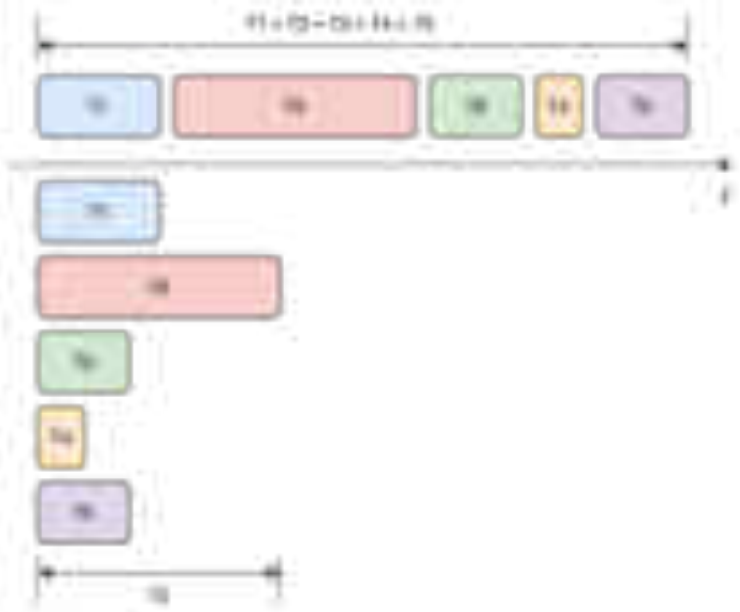


Рис. 1.4.1. Выполнение работы, состоящей из нескольких частей

В любых задачах есть этапы, которые необходимо выполнять строго последовательно. Время, необходимое для выполнения всех последовательных инструкций в совокупности, является тем пределом, быстрее которого решить задачу *нельзя ни при каком количестве процессоров*. (Если же в задаче абсолютно все этапы можно совмещать во времени, то все равно можно считать, что эта задача состоит из одного последовательного этапа.)

Пусть полный объем работы – 1 (единица). Обозначим строго последовательную часть работы α, тогда часть, которую можно распараллелить – (1 - α). Если имеется n процессоров, то они справятся с параллельной частью за время $\frac{1-\alpha}{n}$, поэтому общее время работы будет $T_n = \alpha + \frac{1-\alpha}{n}$, $T_n < 1$, $n > 1$. Можно найти величину $S_n = \frac{1}{T_n} = \frac{1}{\alpha+\frac{1-\alpha}{n}}$ – относительное сокращение времени работы при параллельном выполнении на n процессорах.

При доле последовательных вычислений α общий прирост производительности не может превысить $\frac{1}{\alpha}$, поскольку $\lim\limits_{n\to\infty} S_n = \frac{1}{\alpha+L} = \frac{1}{\alpha}$, $L = \lim\limits_{n\to\infty} \frac{1-\alpha}{n} = 0$. Так, если половина кода – последовательная, то общий прирост никогда не превысит двух.

Закон Амдала показывает, что прирост эффективности вычислений зависит от алгоритма задачи и ограничен сверху для любой задачи с $\alpha \neq 0$. Не для всякой задачи имеет смысл наращивание числа процессоров в вычислительной системе. Более того, если учесть время, необходимое для передачи данных между узлами вычислительной системы, то зависимость времени вычислений от числа узлов будет иметь минимум. Это накладывает ограничение на масштабируемость вычислительной системы, то есть означает, что с определенного момента добавление новых узлов в систему будет увеличивать время решения задачи.

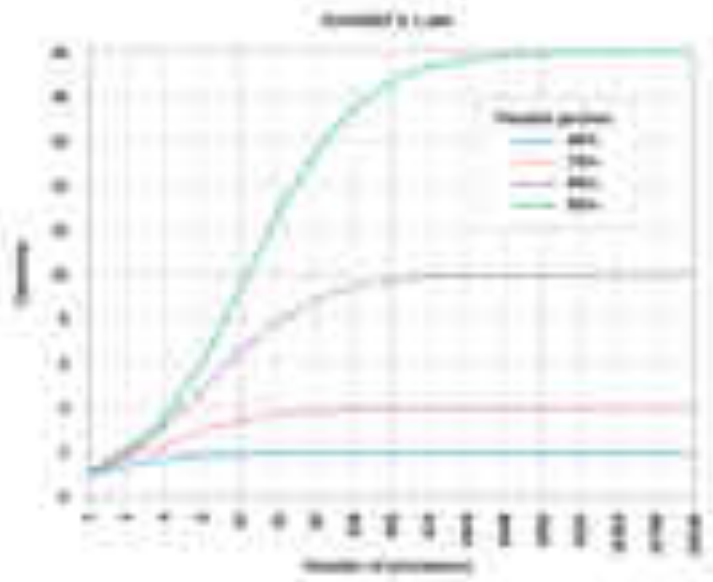


Рис. 1.4.2. Зависимость максимального ускорения S_n от числа процессоров для различных объемов параллельной части программы (горизонтальная шкала – логарифмическая)

При взгляде на график на рис. 1.4.2 может возникнуть чрезмерно пессимистическое предубеждение относительно перспективности распараллеливания. Но для устранения этого предубеждения полезно рассматривать величину $\frac{1}{\alpha}$ – верхнюю оценку прироста производительности. Эта величина не ограничена и может стремиться к бесконечности, если доля последовательных этапов стремится к нулю. Поэтому очень важно глубоко анализировать все задачи в поисках любой допустимой совмещенной обработки!

В реальных задачах требуется принимать во внимание задержки доступа к памяти всех уровней, они будут вносить вклад в последовательную часть работы. Задержки ввода-вывода – своеобразный вариант задержек доступа к памяти (внешней). В распределенных системах скорость передачи сообщений между узлами компьютерной сети в пределе ограничивается физической скоростью света. Половину длины экватора Земли свет проходит приблизительно за 67 миллисекунд, так что в глобальных распределенных вычислительных системах даже при нулевом времени буферизации нельзя получить результат неограниченно быстро, если источник данных расположен далеко от места их обработки.

Наблюдение, положенное в основу закона Амдала, позволяет разработать интересную и полезную форму организации вычислений: конвейер. (Фактически, конвейер в промышленном производстве изобретен задолго до появления вычислительной техники и формулировки закона Амдала.)

Несмотря на то, что работа не может быть выполнена быстрее, чем выполняется самая медленная ее часть, можно найти косвенный способ получать результат быстрее. Тут важно определиться с тем, что понимается под результатом. Если результат есть следствие некоторой причины, и в полученном результате нам важна эта причинно-следственная связь, то время от события-причины до события-результата сократить никак не получится. Если же событие-результат используется само по себе, без связи с породившей его причиной, то

скоростью потока этих событий-результатов можно управлять – в частности, ускорять этот поток.

Например, пусть дерево растет до необходимой высоты 100 лет. Из этого не следует, что нужно обязательно ждать 100 лет, чтобы срубить ствол заданной высоты – его можно срубить *в лесу* прямо сейчас. (Но лес должен быть в наличии.) Если посадить 100 деревьев с интервалом в один год, то через 100 лет можно будет каждый год рубить полноценный ствол. Не важно, что ствол 100 лет назад был семечком или саженцем – важно лишь то, что в настоящий момент он присутствует в необходимом состоянии, пригодном к использованию.

Этот подход является примером конвейерной обработки данных (рис. 1.4.3): с выхода конвейера *в установившемся режиме* можно *с высокой скоростью* получать результаты, формирование которых производилось сколь угодно *медленно*.

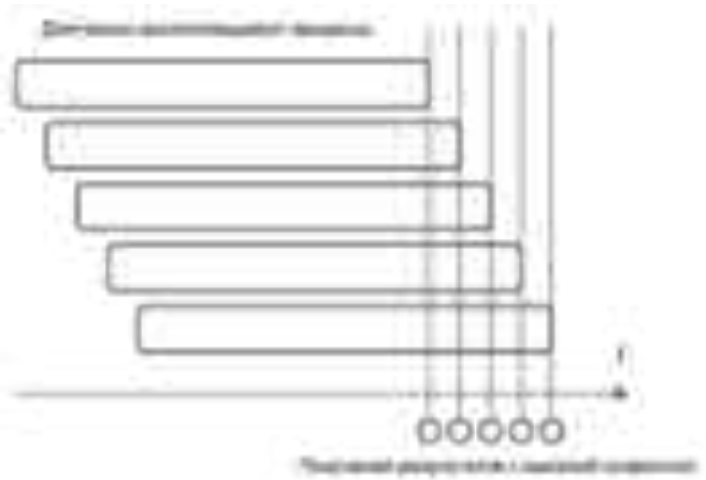


Рис. 1.4.3. Принцип ускорения потока результатов за счет конвейерной обработки

Из рис. 1.4.3 видно, что хотя процессы идут параллельно, но каждый из них относится к независимым задачам, представляя *последовательную* обработку отдельного экземпляра входных данных. Таким образом, конвейерная обработка имеет смысл только при наличии *потока данных*. (Кстати, по отношению к промышленному конвейерному производству нередко используется термин “поставить [производство чего-либо] на поток”.)

Может показаться, что нет таких реальных вычислительных задач, где не важна причинно-следственная связь между началом обработки данных и ее завершением. Но это заблуждение. Задач таких много.

Вот пример: допустим, при распаковке цифрового видео высокого разрешения, сжатого очень ресурсоемким методом, на обработку каждого кадра *одному процессору* может требоваться, скажем, 3 секунды. Тем не менее, открыв файл и *подождав всего 3 секунды* от начала фильма (вполне терпимая, причем однократная, задержка), пользователь далее получит плавный поток кадров со сколь угодно высокой частотой, если в системе есть *много процессоров*.

По закону Амдала все эти процессоры в совокупности при параллельной работе не позволяют распаковать один кадр за время, равное периоду комфортной кадровой частоты,

соответствующей плавному воспроизведению (обычно не более 1/30 секунды). Зато каждый процессор в одиночку справляется за 3 секунды. Тогда имеет смысл вообще не распараллеливать алгоритм распаковки отдельных кадров!

Вместо этого надо вручить каждому процессору по одному кадру *со сдвигом* начала обработки на величину периода кадровой частоты. Если частота кадров 30 fps, а на каждый кадр уходит 3 секунды, то в системе потребуется 30 x 3 = 90 процессоров. Да, это довольно много по современным меркам (впрочем, для GPU – мало), но все равно эти 90 ядер, работая параллельно, не смогли бы “вытянуть” комфортную частоту кадров из-за ограничений, наложенных законом Амдала (на самом деле, конечно, не законом, а природой задачи). А при обработке по строго последовательному алгоритму (который, между прочим, и составить проще) создается впечатление, что “вытягивают”. Как только процессор выдал очередной кадр, он немедленно берется за новый, поэтому в течение следующих 3-х секунд зритель будет продолжать получать все новые и новые кадры. Причем, зрителю не важна “предыстория” каждого отдельного кадра, так что результат используется “беспричинно”. Цель системы в данном случае – обеспечить *плавный поток* этих “беспричинных” результатов.

Разумеется, это решение имеет и оборотную сторону. Представим себе, что зритель решил “перескочить” на новую позицию в фильме, пропустить скучный или страшный момент. Тогда ему вновь понадобится ждать 3 секунды – потому что кадры с новой позиции еще не готовы! Программа же не могла предвидеть, что зритель решит “перепрыгнуть”. Так что лучше смотреть все по порядку – “перемотка тормозит”. (На самом деле, каждый кадр “тормозит”, но это незаметно в *установившемся режиме* просмотра.) Пользователь инициировал некоторое действие (перемотка) и ждет отклика системы на это действие (поступление кадров с нового места) – причинно-следственная связь между этими событиями важна для пользователя, и длительное время ожидания в данном случае является хорошо заметным раздражающим фактором.

Процессоры, имеющие конвейер для обработки *потока инструкций*, позволяют получать результат в каждом такте, при этом инструкции могут быть достаточно сложными (длинными по времени исполнения). Приведенный выше пример с видеопотоком наглядно объясняет по аналогии, почему процессоры с глубоким конвейером “не любят” операции переходов, условных или безусловных: инструкции с нового адреса должны пройти весь конвейер, прежде чем станут доступны их результаты – а это приводит к значительным задержкам. Заодно объясняется также и смысл введения в такие процессоры сложных аппаратных блоков предсказания переходов: это равносильно тому, что программа просмотра видео “предсказывала” бы то место, куда зритель будет “перематывать”, чтобы заранее приступить к распаковке кадров именно с этой позиции, а не брать их по порядку.

Кроме того, конвейеры позволяют скрывать задержки доступа к данным в памяти всех уровней: оперативной, дисковой, сетевой. Этот аспект тоже можно проиллюстрировать примером с просмотром фильма, достаточно немного изменить условия и представить, что кадры распаковываются почти мгновенно (по несложному алгоритму), а 3 секунды – это задержка передачи кадра от сервера потокового вещания на клиентский компьютер по медленному каналу связи. Клиент запрашивает у сервера кадры с периодом 1/30 секунды

(период между кадрами). После выдачи первого запроса придется ждать 3 секунды, зато по истечении этого ожидания программа просмотра уже может начинать выводить плавный (30 fps) видеопоток зрителю. При получении очередного кадра на сервер отправляется новый запрос, результат которого придет только через 3 секунды, но для зрителя уже готовы кадры, запрошенные ранее (сопоставьте с рис. 1.4.3).

Аналогия в этом варианте не совсем точная, потому что реальные каналы связи обычно вносят непредсказуемые вариации задержек передачи. Для их устранения в мультимедиа-приложениях применяется буферизация. Кроме того, современные кодеки, как правило, формируют поток не из кадров, а из накопленной разницы между кадрами. Но это несущественные детали.

1.5 Классификация параллельных вычислительных систем

Цель классификации параллельных вычислительных систем – оказать помощь пользователю в выборе класса систем, на который ему требуется ориентироваться для эффективного решения требуемого класса задач [7]. Одновременно с этим содержательная классификация должна:
1) облегчать понимание того, что достигнуто на сегодняшний день в области архитектур вычислительных систем, и какие архитектуры имеют лучшие перспективы в будущем;
2) подсказывать новые пути организации архитектур – речь идет о тех классах, которые в настоящее время по разным причинам пусты;
3) показывать, за счет каких структурных особенностей достигается увеличение производительности различных вычислительных систем; с этой точки зрения, классификация может служить моделью для анализа производительности.

Признаки, по которым отличаются параллельные вычислительные системы разных классов (список неполный):
- число бит в машинном слове (все биты машинного слова обрабатываются параллельно);
- число машинных слов, обрабатываемых одновременно;
- иерархия памяти;
- способ организации памяти (общая, распределенная);
- количество блоков памяти данных;
- количество блоков памяти команд;
- количество устройств управления;
- количество процессоров;
- наличие прямых связей между процессорами;
- топология связей между процессорами (матрица, линейный массив, тор, дерево, звезда и др.);
- степень связности между процессорами (сильная, средняя, слабая – по накладным расходам на передачу информации);
- типы переключателей связей между функциональными элементами (1:1 – парная связь выделенных элементов, n:n – парная связь элементов множества, 1:n – один со всеми, n x n – все со всеми);
- механизм передачи информации между процессорами (сообщения, разделяемая память, сигналы готовности операндов);

- синхронность/асинхронность режима управления;
- уровень, или *гранулярность* параллелизма (задачи, инструкции, данные).

1.6 Таксономия Флинна

Самой ранней классификацией является классификация Флинна (Michael J. Flynn, 1966). В ее основе находится понятие потоков команд и потоков данных (рис. 1.6.1).

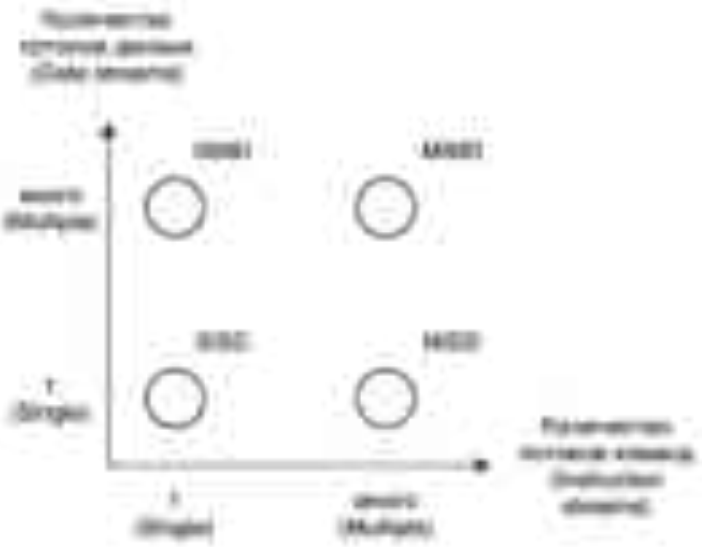


Рис. 1.6.1. Таксономия Флинна

SISD (Single Instruction stream over a Single Data stream) – это система, в которой параллелизм обработки данных отсутствует (не принимая во внимание самые низкоуровневые виды параллелизма – скажем, на уровне битов); один поток инструкций обрабатывает единственный поток данных.
SIMD (Single Instruction stream over Multiple Data streams) – один поток инструкций применяется к нескольким потокам данным; системы с векторными инструкциями.
MISD (Multiple Instruction streams over a Single Data stream) – несколько потоков инструкций применяются к единственному потоку данных (поочередно или одновременно – не конкретизируется).
MIMD (Multiple Instruction streams over Multiple Data streams) – несколько потоков инструкций применяются к нескольким потокам данных; системы с несколькими независимыми процессорами, каждый из которых выполняет свою программу.

Следует иметь в виду, что эта классификация не отражает всю совокупность характеристик конкретной системы. Например, MIMD – это, в том числе, обычный многоядерный процессор, однако в потоке инструкций, которые исполняет каждое из ядер, могут встречаться векторные SIMD-команды. Или, с другой стороны, SISD-поток на время может переключиться в SIMD-режим, если встретилась векторная инструкция. А вот примеров того, как SISD может переключиться на время в MIMD-режим и вернуться обратно, нет в реально существующих компьютерах. Поэтому классификация Флинна полезна для описания принципа работы отдельных компонентов системы.

В практике программирования термином *поток* (thread) обозначают поток инструкций (это подразумевает работу системы в соответствии с MIMD-моделью). Поток данных (data stream) не находит пока адекватного унифицированного отражения в системном ПО, в каждом прикладном случае он трактуется по-своему.

Предложенная схема классификации вплоть до настоящего времени является самой применяемой при начальной характеристике того или иного компьютера. Недостаток – чрезмерная заполненность класса MIMD. Необходимы средства, более избирательно систематизирующее архитектуры, которые по Флинну попадают в один класс, но совершенно различны по числу процессоров, природе и топологии связи между ними, по способу организации памяти, а также по технологии программирования. Кроме того, до сих пор продолжаются споры о классе MISD, так как нет убедительных примеров безусловного отнесения к нему каких-либо реальных архитектур; некоторые специалисты относят к классу MISD системы конвейерной обработки.

При описании гетерогенных массово-параллельных систем на основе GPU можно встретить термин SIMT [3]. Он означает Single Instruction, Multiple Threads. Суть его в том, что одна и та же программа запускается параллельно во множестве потоков, но каждый из этих потоков имеет свой контекст, свои локальные данные, может выбирать разные ветви исполнения. Подробнее о SIMT см. пункт 2.4.

Раздел 2. Моделирование и анализ параллельных вычислений

Лекция 3

2.1. Модели и показатели качества параллельных алгоритмов

Специалисты разного профиля, работающие на компьютерах с обычной (непараллельной) архитектурой, привыкли оценивать алгоритмы по трем характеристикам: количество операций, объем требуемой памяти, точность. Создание параллельных систем потребовало от алгоритмов принципиально иных свойств и характеристик.

Усугубляет ситуацию и то, что существующие в настоящее время компьютеры, операционные системы, фреймворки и среды поддержки прикладного программирования и, собственно, прикладные программы все еще являются *последовательными с элементами параллельности*, причем доля последовательных режимов значительно превышает долю параллельных режимов (ядер относительно мало по сравнению с объемами задач, доступная гранулярность параллельности не всегда подходит для решаемых задач, межпроцессорные коммуникации вносят большие накладные расходы). Требуется полностью изменить подход к вычислениям как с точки зрения аппаратуры, так и с точки зрения алгоритмов, но такого рода трансформацию невозможно провести быстро, она займет десятилетия – приблизительно столько же складывалась существующая методика организации вычислительных систем (не вполне удовлетворительная, но все-таки способная приносить колоссальные практические результаты).

В первую очередь, необходима конструктивная методология моделирования параллельных систем для выявления параллельных свойств алгоритмов [7].

Алгоритм (или, в конечном итоге, компьютерная программа) представляет собой *порядок действий* исполнителя для решения задачи. Ранее писали “последовательность действий”, но в связи с появлением параллельных компьютеров пришлось использовать более общий термин “порядок”.

Что означает *порядок* в применении к действиям при решении вычислительных задач? Отношение порядка между элементами некоторого множества в каком-то смысле аналогично отношению неравенства, поэтому для обозначения порядка будем использовать знаки сравнения ‘<’, ‘≤’ и другие, а под тем множеством, элементы которого связаны отношением порядка, будем понимать множество действий, предназначенных к выполнению.

Неформально, множество упорядочено, если указано, какие элементы *следуют* за какими. Если в алгоритме сначала требуется сделать действие A, затем B, то A < B (B *следует за* A). Для целей моделирования вычислений важно, что отношение порядка транзитивно: если A < B и B < C, то A < C.

Действие, операция имеет *аргументы*, которые показывают, над какими объектами производятся вычисления. Теоретически, введение понятия аргументов операций не является обязательным, но иначе возникло бы колоссальное неудобство. Например, для операции сложения требуются два аргумента-слагаемых, это привычно. Если же не использовать понятие аргументов (параметров, конкретизирующих выполняемые действия), то потребуется бесконечное множество безаргументных операций: “сложить 0 с 1”, “сложить 0 с 2”, … , “сложить 1 с 1”, “сложить 1 с 2”, …, и так далее. (Такое предложение кажется вздорным, но на самом деле оно не слишком далеко от реальности. Инструкции, выполняемые процессорами, закодированы в составе двоичного “пакета” вместе со своими аргументами или их адресами, так что *пакет* целиком можно принять за код некой безаргументной операции.)

Операции могут иметь несколько аргументов. В общем случае удобно использовать функциональную запись *f(a1, a2, a3)* – так обозначается операция *f* над аргументами *a1*, *a2*, *a3*. Большинство применяемых на практике элементарных операций имеют по два аргумента (бинарные операции), и для них обычно используется инфиксная запись вида $a1 \circ a2$.

В какой бы форме операция не была записана, вместе со значениями своих аргументов она образует некий объект. Этот объект является *синонимом* результата применения операции к своим аргументам. Например, синонимом результата применения сложения к аргументам 1 и 2 есть +(1, 2). Кстати, для общности, константы (самых разных типов) можно считать безаргументными (нуль-арными) функциями.

Если в алгоритме несколько раз появляется одна и та же операция с одними и теми же значениями аргументов (но не ссылок в память), то нет необходимости выполнять вычислительные действия несколько раз – можно один раз вычислить, а в других случаях просто подставить однажды полученный результат. Такой способ называют *мемоизацией*.

(“Самые быстрые операции – это те, которых нет в программе.”) К сожалению, этот способ мало применим к элементарным инструкциям.

Иногда имеет смысл разделять операции на два класса: *преобразователи* и *распознаватели*. Преобразователи перерабатывают информацию, а распознаватели задают последовательность работы преобразователей. Примером распознавателя является команда ветвления, которая в зависимости от условия переключает обработку на ту или иную цепочку действий. При начальном анализе параллелизма в алгоритмах обычно не рассматривают распознаватели, а вместо этого анализируют набор различных траекторий, соответствующих разным комбинациям срабатывания распознавателей.

Любая операция является *потребителем* аргументов. Она не может выполняться до готовности результатов предшествующих операций – *поставщиков* аргументов. Следовательно, на множестве операций в программе существует частичный порядок: для любой произвольно взятой пары операций (A, B) может быть верно либо A < B (A выполняется до B), либо A > B (A выполняется после B), либо A = B (A и B могут выполняться одновременно).

Для алгоритма можно построить *граф информационных зависимостей*, или просто *граф алгоритма*. Это направленный граф, в котором узлы соответствуют операциям, причем из узла u в узел v идет дуга, если $u < v$. Пример графа алгоритма вычисления значения выражения a + b * (c - d) приведен на рис. 2.1.1:

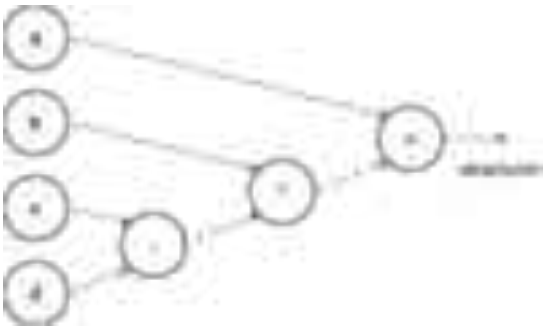


Рис. 2.1.1. Граф алгоритма вычисления выражения a + b * (c - d)

Пример графа вычисления значения полинома 4-й степени с применением мемоизации при возведении в степень (учтена также группировка для редукции при суммировании) приведен на рис. 2.1.2.

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 = (a_0 + a_1 x) + (a_2 x^2 + a_3 x^2 x) + a_4 x^2 x^2$$

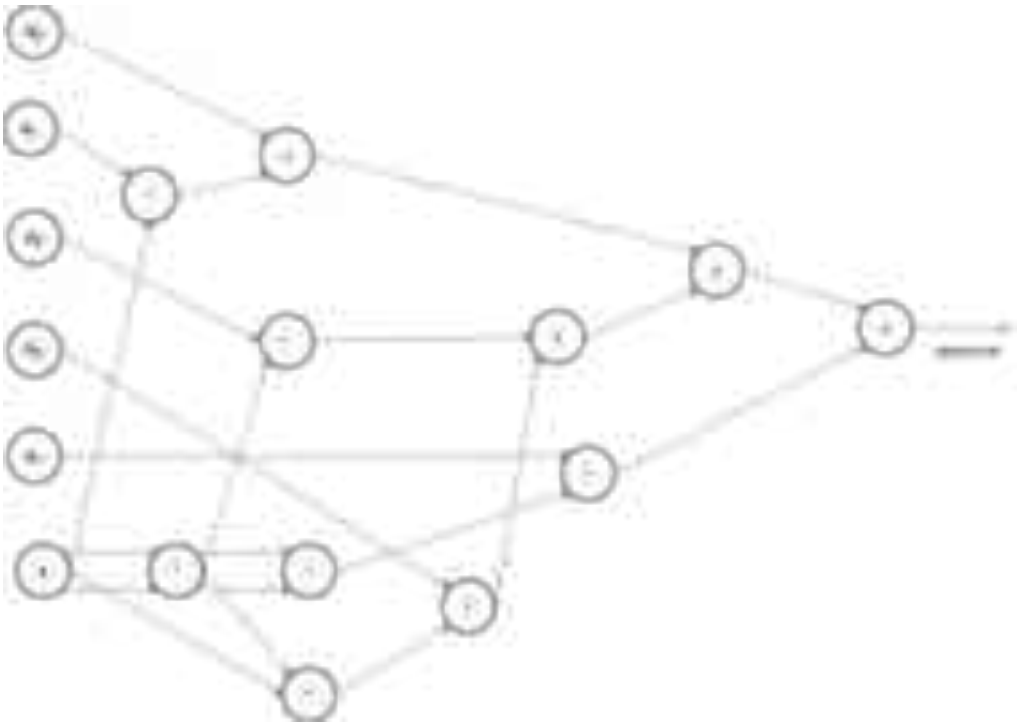


Рис. 2.1.2. Граф алгоритма вычисления полинома 4-й степени

Граф любого алгоритма является *ациклическим*, потому что в вычислительных задачах нельзя задавать некоторую величину через саму себя. Бесконечная рекурсия может быть записана на языках программирования, но окажется неработоспособной после запуска программы. В литературе часто встречается понятие DAG (Directed Acyclic Graph, ориентированный ациклический граф). Из рис. 2.1.2 видно, что граф алгоритма может являться *мультиграфом* – это случаи, когда несколько аргументов операции имеют одно и то же значение (например, $x^2 = x \cdot x, x^4 = x^2 \cdot x^2$).

Ацикличность графа алгоритма никаким образом не препятствует разработке циклических алгоритмов. Просто для каждой итерации алгоритмических циклов нужно создавать новые узлы-операции (рис. 2.1.3), даже если в программе они будут записаны той же строкой:

```
s = 1;
for i from 2 to 3 do
      s = s * i
```

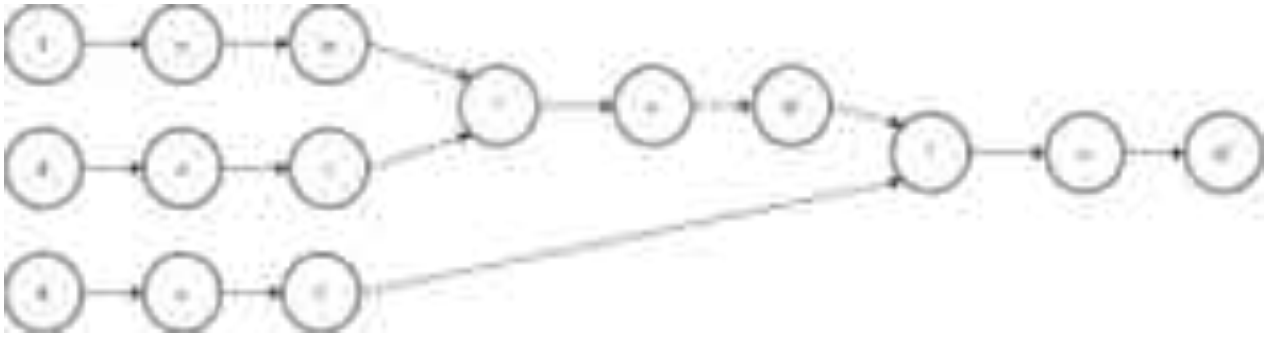


Рис. 2.1.3 Ориентированный ациклический граф для алгоритма с циклом

В этой схеме мы весьма вольно изобразили оператор присваивания как унарную операцию. На самом деле это бинарная операция, в качестве второго аргумента она принимает ссылку на ячейку (будь то адрес в ОЗУ или наименование регистра), в которую следует *скопировать* первый аргумент. Но можно понимать присваивание и по-другому, а именно как установление тождества между значениями: *все копии тождественны*. Кроме того, повторные присваивания приводят к новым вершинам (i', s', s''), так как после перезаписи ячейки между старым и новым значениями *теряется тождество*, и теперь их неправомерно обозначать одинаково.

Пусть задан ориентированный ациклический граф с числом узлов n. Тогда имеет место утверждение: существует число $s \leq n$, для которого можно так пометить все узлы графа одним из индексов 1, 2, …, s, что если дуга идет из узла с индексом i в узел с индексом j, то $i < j$.

Указанное утверждение можно обосновать следующим образом. Выберем все узлы, не имеющие предшественников (в непустом ациклическом графе такое всегда возможно), и пометим их индексом 1, а затем удалим из графа. Оставшийся граф является ациклическим. Опять выберем в нем все узлы без предшественников, пометим их индексом 2 и удалим из графа. Продолжая, дойдем до последних узлов, которые получат индекс s. На каждом шаге из графа удалялось не менее одного узла, поэтому $s \leq n$. Никакие два узла, получившие один и тот же индекс, не связаны дугой (потому что именно так была реализована разметка).

Граф, размеченный таким образом, называется *строгой параллельной формой графа*. Группы узлов, имеющих одинаковые индексы, называются *ярусами*, число узлов в ярусе – шириной яруса. Число ярусов называется *высотой* параллельной формы, максимальная ширина ярусов – *шириной* параллельной формы.

Из соображений полной загрузки процессорных элементов требуется составлять алгоритм в расчете на получение (средней) ширины ярусов, равной количеству процессоров. Но требуется также принимать во внимание особенности обмена данными между ярусами.

Верно ли утверждение, что никакая операция не может *начать* выполняться, пока не готовы полностью все ее аргументы?

Если это элементарная инструкция процессора, типа сложения или умножения, то да. Если же это "крупная" операция типа скалярного умножения достаточно длинных векторов (с длиной большей, чем допускают аппаратные векторные инструкции процессора), то не обязательно: операция может начать выполняться еще до готовности ее аргументов и, более того, *активно влиять* на доведение этих аргументов до готовности.

В связи с этим наблюдением можно рассматривать такую организацию алгоритма (ее называют *ленивыми* вычислениями), в которой операции с конца алгоритма по собственной

инициативе “вытягивают” необходимые аргументы и тем самым определяют *потоки движения информации* от начала к концу алгоритма.

Проиллюстрировать ленивые вычисления можно на том же примере умножения матриц, который мы рассматривали ранее (см. п. 1.1):

1	2	3	4
5	6	7	8

x

9	10
11	12
13	14
15	16

=

1*9 + 2*11 + 3*13 + 4*15	1*10 + 2*12 + 3*14 + 4*16
5*9 + 6*11 + 7*13 + 8*15	5*10 + 6*12 + 7*14 + 8*16

Здесь удобно ввести операцию *pullResult(v<i, j>)*. При указанных размерах матриц эта операция будет запущена, возможно параллельно, четыре раза (по числу элементов матрицы-результата) с разными аргументами *v<i, j>*, представляющими собой ссылки на *пакеты*, кодирующие операцию-поставщик вместе с ее аргументами. Что фактически делает *pullResult*? Она запускает операцию по указанной ссылке и ждет результат, причем для получения результата должен быть предусмотрен некий канал передачи.

Что может представлять из себя этот *канал передачи результата*? Есть следующие варианты:
1) ожидать от вызываемой операции *v* ссылку на ячейку памяти, из которой можно прочитать содержимое результата;
2) передавать вызываемой операции ссылку на ячейку памяти, в которую надо положить результат, и после запуска ожидать *сигнал о готовности результата* по указанной ссылке (в этом варианте *пакет* операции *v* будет иметь вид *v<i, j, address>*);
3) между вызывающей и вызываемой операцией могут существовать неявные коммуникационные объекты типа рассмотренных в п. 1.1 очередей; в этот объект вызываемая операция заносит свои результаты (*push*), а вызывающая операция ожидает готовности (*pop*).

В примере с матрицами удобен второй вариант, при котором *pullResult* просто передает операции *v* адрес той ячейки, в которую необходимо положить результат – и тогда в теле *pullResult* можно уже ничего не делать (одно из проявлений “ленивости” этой стратегии). На самом деле, не совсем ничего; нужно как-то сигнализировать о готовности результата в ту среду, из которой вызвана операция *pullResult*. Оставим пока эту деталь и рассмотрим процесс вычислений дальше.

Итак, вызвана операция *v<i, j, address>*. Она должна вычислить скалярное произведение строки *i* на столбец *j*, положить результат в ячейку по адресу *address* и сигнализировать о готовности результата. Согласно схеме вычислений при указанных размерах матриц результат будет суммой двух значений (частичных сумм). Таким образом, операция *v* должна дважды вызвать операцию *sum* (можно параллельно), дождаться двух результатов, сложить их, записать сумму по адресу *address* и сигнализировать о завершении в вызывающую среду. Для получения двух слагаемых нужно выделить две временных ячейки памяти, следовательно, пакеты для исходящих вызовов будут следующими: *sum<i, j, 0,*

addressTmpX1>, *sum<i, j, 2, addressTmpX2>*. Смысл третьего формального аргумента (обозначим *arg3*) в этом пакете объясняется далее.

Операция *sum* должна вычислить сумму двух произведений; опуская детали, получаем пакеты для двух вызовов *mul<i, j, arg3, addressTmpY1>* и *mul<i, j, arg3 + 1, addressTmpY2>*. Здесь *arg3* нужен для выбора своих пар элементов при умножении.

Операция *mul* является концом ленивой цепочки; она берет элементы M1[i][arg3] и M2[arg3][j] из исходных массивов, умножает их, помещает произведение по переданному ей адресу и сигнализирует о готовности результата.

В этой стратегии *все* операции *pullResult*, *v*, *sum*, *mul* могут выполняться параллельно, но далеко не во всякий момент они заняты полезной работой. Операции *mul* не подвержены ожиданию (разве что из-за задержек при доступе к памяти), отрабатывают и немедленно получают результат. Операции *sum* ожидают, пока не отработает пара вызванных ими операций *mul*. Операции *v* ожидают, пока не отработает пара вызванных ими операций *sum*. Операция *pullResult* ожидает, пока не отработает вызванная ими операция *v*. Вызывающая среда ожидает, пока не отработают все операции *pullResult*.

Ленивая стратегия параллельного исполнения может быть более простой для понимания при разработке параллельных алгоритмов, чем рассмотренный в п. 1.1 вариант с очередями. Однако ожидание сигнала о завершении вызванных операций все равно требует поддержки со стороны исполняющей среды, операционной системы или аппаратной платформы. Здесь важно, чтобы ожидание не было *активным*, то есть, не расходовало время какого-либо процессора – следовательно, при готовности результата должен формироваться сигнал, “пробуждающий” ожидающие этот сигнал потоки.

Псевдокод программы для рассмотренного алгоритма:

```
const m = 2
const n = 4
const k = 2

      // исходные матрицы доступны глобально
M1 = [[1, 2, 3, 4], [5, 6, 7, 8]]
M2 = [[9, 10], [11, 12], [13, 14], [15, 16]]

      // матрица-результат
C = Array(m, n)

function mul(args, addressResult) {
      store(addressResult, M1[args[0]][args[2]] * M2[args[2]][args[1]])
}

function sum(args, addressResult) {
      a, b  // временные ячейки для “вытягивания” операндов
      do in parallel
            pullResult(command<mul, args[0], args[1], args[2], &a)>)
            pullResult(command<mul, args[0], args[1], args[2] + 1, &b)>)
```

```
      join
      store(addressResult, a + b)
}

function v(args, addressResult) {
      a, b  // временные ячейки для “вытягивания” операндов
      do in parallel
            pullResult(command<sum, args[0], args[1], 0, &a)>)
            pullResult(command<sum, args[0], args[1], 2, &b)>)
      join
      store(addressResult, a + b)
}
      // начало выполнения
do in parallel
      pullResult(command<v, 0, 0, &(C[0][0])>)
      pullResult(command<v, 0, 1, &(C[0][1])>)
      pullResult(command<v, 1, 0, &(C[1][0])>)
      pullResult(command<v, 1, 1, &(C[1][1])>)
join
      // результат получен в матрице C
```

Редукция сложения в этом примере организована для частного случая (длина набора чисел равна 4). Псевдокод общего случая рекурсивной параллельной редукции рассмотрим на примере поиска минимума в массиве:

```
const n = 10

      // исходный массив доступен глобально
M = [1, -2, -3, 0, 5, -6, 7, 8, 1, 3]

function min(args, addressResult) {
  start = args[0]
  length = args[1]

  if(length == 1) {
      store(addressResult, M[start])
  }
  else {
      halfLength1 = length / 2;
      halfLength2 = length - halfLength1;

      a, b  // временные ячейки для “вытягивания” операндов
      do in parallel
        pullResult(command<min, args[0], halfLength1, &a)>)
        pullResult(command<min, args[0] + halfLength1, halfLength2, &b)>)
      join
      store(addressResult, a < b ? a : b)
  }
}

minValue
pullResult(command<min, 0, n, &minValue>)
```

Здесь все рекурсивные вызовы операции `min` (кроме первого) осуществляются параллельно, но фактически эти потоки после создания не работают, а только ожидают, пока подтянутся операнды `a` и `b`. Редукция с разделяемой очередью и блокировкой потоков (см. примеры из п. 1.1) была более лаконичной и не содержала лишних операций – но реализация микрогранулярных очередей на существующей вычислительной аппаратуре была бы либо затруднительной, либо крайне неэффективной. Пример же с рекурсивной редукцией ближе к практической реализации, однако при разбиении массива на части имеются лишние операции вычисления стартовых индексов и длин частей, причем, эти вычисления повторяются в разных потоках.

2.2 Показатели качества параллельных алгоритмов: ускорение, масштабируемость, пропускная способность

Пользователям, решающим свои прикладные вычислительные задачи, важно получить результат как можно быстрее (либо обработать больше данных за некоторый промежуток времени). Поэтому им важно знать, насколько быстрее после перехода к параллельным вычислениям программа будет проходить путь от момента поступления некоторого фиксированного объема входных данных до момента получения результата. Если выигрыш окажется незначительным, то вложение дополнительных средств в разработку параллельной версии алгоритма и, возможно, приобретение новой вычислительной техники не оправдано.

С другой стороны, разработчикам параллельных алгоритмов и программ важно продемонстрировать потенциальным пользователям преимущества своих версий по отношению к однопоточным реализациям. Самый простой способ – это показать количественный прирост быстродействия.

Таким образом, естественной, интуитивной характеристикой качества параллельного алгоритма является *ускорение* – отношение времени работы алгоритма на одном процессоре ко времени работы параллельной версии алгоритма на нескольких процессорах.

Казалось бы, достаточно вычислить величину $s_n = \frac{T_1}{T_n}$, где n – число процессоров (ядер), T_1 – время работы программы на одном процессоре, T_n – время работы программы на n процессорах. Если, скажем, время при выполнении на двух процессорах сократилось вдвое, то величина ускорения $s_2 = 2$.

Однако, практическая полезность такого простого подхода существенно ограничена. Действительно, такое сравнение предполагает, что имеется однопоточная и параллельная версии *одного и того же* алгоритма, кроме того, характеристики *скорости процессора* однопроцессорной системы полностью совпадают с таковыми на многопроцессорной системе. Но можно ли сказать, что параллельная версия представляет собой *тот же* алгоритм? Он же устроен существенно по-другому! А что такое *скорость процессора*? Тактовая частота при идентичности микроархитектуры и характеристик кэш-памяти? Или измерение ускорения требуется производить в обоих случаях на многопроцессорной системе, отключая дополнительные ядра при прогоне однопоточной версии? А как гарантировать, что

эти “отключенные” ядра не влияют на поведение оставшегося ядра (например, что на нем динамически не повышается тактовая частота из-за изменившейся нагрузки)?

Еще один недостаток: преодолев указанные трудности и измерив, наконец, значение величины s_n, далеко не всегда получится верно спрогнозировать значение величины s_m, $m \neq n$; оно может оказаться каким угодно. Так на какую параллельную систему (какое число процессоров) ориентироваться? Скорее всего, имеется некий sweet spot (оптимальный случай ускорения), зависящий как от самой задачи, так и от алгоритма и аппаратуры.

Поэтому требуется разработать и другие подходы к оценке эффективности параллельных вычислений. Один из альтернативных путей – это оценка степени использования аппаратных возможностей. Как для однопроцессорной системы, так и для многопроцессорной известны величины *пиковой производительности*. Как правило, это произведение числа процессоров на тактовую частоту и на количество операций (обычно с плавающей точкой), выполняемых за такт. Пиковая производительность измеряется во FLOPS (floating-point operations per second) и недостижима в силу разных причин, но это неважно, поскольку она недостижима для *любых* систем.

В этом методе снимается вопрос о том, один и тот же алгоритм измеряется, или разные. Общее количество операций по обработке входных данных в том и другом случае должно быть одинаковым – оно определяется задачей. Например, умножение матриц размерами 100x100 “классическим” способом требует свыше 100^3 операций. Помимо операций, относящихся непосредственно к задаче, алгоритм может выполнять какие-то дополнительные действия, но они являются лишними и характеризуют его неидеальность. Заметим, что другие алгоритмы, такие как алгоритм Штрассена, могут использовать иной набор операций, но в примере мы имеем в виду, что параллельная версия использует тот же “классический” метод (умножение строки на столбец).

Пусть K – теоретически необходимое алгоритму количество операций (зависит от объема входа), тогда можно вычислить “идеальное” (пиковое) время выполнения $t_1 = \frac{K}{P_1}$, $t_n = \frac{K}{P_n}$, где P_1, P_n – спецификации пиковой производительности, соответственно, однопроцессорной и многопроцессорной систем. Заметим: *вычислить* пиковое время, а не измерить.

Теперь приступим к измерению. Если прогон алгоритма на однопроцессорной системе завершился за время T_1, $T_1 > t_1$, то он задействует долю $R_1 = \frac{t_1}{T_1}$ от пикового числа операций в секунду, $R_1 < 1$, а прогон многопоточной версии на n процессорах за время T_n, $T_n > t_n$ дает долю $R_n = \frac{t_n}{T_n}$, $R_n < 1$.

Может показаться, что это просто алгебраические манипуляции с разными обозначениями, и на самом деле скорости систем будут $V_1 = \frac{K}{T_1}$, $V_n = \frac{K}{T_n}$ – без привлечения сведений о

пиковой производительности. Но если мы будем брать отношение $\frac{V_n}{V_1}$ для оценки ускорения, то K сократится, и мы получим ту же самую формулу ускорения $s_n = \frac{T_1}{T_n}$ из предыдущего метода! В записи этой формулы не содержится информации ни о связи алгоритма и задачи, ни о вычислительной системе.

В отличие от этого, величины R_1 и R_n содержат в себе и "теоретическую" информацию о задаче (через K), и "технологическую" информацию о системе (через P), и "ситуационную" информацию о решении задачи конкретным алгоритмом на конкретной системе (через измеренное время решения T). Величины R характеризуют степень приспособленности алгоритма решения конкретной задачи к той конкретной системе, на которой он исполняется. Чем выше R, тем более качественно оптимизирован алгоритм под свою систему. При одном и том же значении R на параллельной системе, естественно, достигается более высокая скорость, так как пиковая производительность параллельной системы выше, чем однопроцессорной (а иначе просто не имеет смысл с ней связываться). Если же $R_n > R_1$, то переход в параллельный режим вообще обязателен (при наличии средств на ее эксплуатацию).

Но даже в случае $R_n < R_1$ не обязательно стоит отвергать параллельный режим. За счет более высокой пиковой производительности параллельной системы все равно можно получить более быстрое решение задачи. Но есть граница "неэффективности": при невыполнении условия $R_n P_n > R_1 P_1$ параллельная система отстает по скорости от однопроцессорной, и параллельный алгоритм в текущей версии не должен применяться, его надо доработать.

В IT-сфере вообще и, в частности, в контексте параллельных вычислений широко применяется понятие масштабируемости. Масштабируемость (scalability) – способность системы увеличивать свою производительность при добавлении ресурсов (обычно аппаратных). Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам.

Различают *горизонтальную* и *вертикальную* масштабируемость. Горизонтальная масштабируемость (масштабируемость вширь, scale out) – возможность увеличения производительности системы за счет *добавления* дополнительных программных или аппаратных средств. Вертикальная масштабируемость (масштабируемость вглубь, scale up) – возможность *замены платформы*, в которой функционирует система, на новую, обладающую большей производительностью.

Масштабируемость можно оценить через отношение прироста производительности системы к приросту используемых ресурсов: $Q = \frac{\Delta P}{\Delta R}$. Чем ближе это отношение к единице, тем масштабируемость лучше.

В чем и как измерять производительность и ресурсы – это детали различных подходов и методик. Важно то, что масштабируемость системы может быть как больше единицы, так и

меньше нуля. В параллельных вычислениях при выборе для решения задачи неподходящих алгоритмов добавление процессоров может не повышать, а снижать производительность; причиной такого явления являются накладные расходы на взаимодействие между процессорами: чем больше процессоров, тем больше связей между ними и, соответственно, потерь. Наоборот, производительность может существенно вырасти при относительно небольшом приращении ресурсов по причине скачкообразного выхода на штатный режим работы параллельного алгоритма.

Основные помехи масштабируемости параллельных программ:
- закон Амдала (последовательные части программы);
- накладные расходы на коммуникации;
- неравномерность загрузки процессоров;
- естественные пределы декомпозиции задачи.

Параллельные системы организуют не только для более быстрого решения задач, но и для повышения объемов обрабатываемых данных в единицу времени. *Пропускная способность* (throughput) обработки данных – это количество информационных элементов (байт, слов, чисел, символов и т.п.), которые вычислительная система способна преобразовать в единицу времени согласно поставленным целям обработки. Например, одна система может сортировать 1 млн. чисел в секунду, а другая – 10 млн. чисел в секунду; у второй системы пропускная способность сортировки в 10 раз выше.

Рост пропускной способности параллельных систем ограничен, по сути, тем же законом Амдала, но в форме закона Густафсона – Барсиса (Gustafson – Barsis's law). Закон Амдала (см. п. 1.4) исходит из предположения, что объем работы постоянен, и при параллельном выполнении работа завершится быстрее. А закон Густафсона – Барсиса несколько по-иному расставляет акценты: параллельно работающие исполнители за одно и то же время способны совместно сделать больше, нежели единственный исполнитель, так как их результаты объединяются (рис. 2.2.1).

Повышение пропускной способности определяется как отношение объема вычислений, выполненных с использованием многопоточности, к объему вычислений, выполненных последовательно за один и тот же промежуток времени:

$$B_n = \frac{(1-\alpha)n+\alpha}{1-\alpha+\alpha} = n - \alpha n + \alpha = n + \alpha(1 - n).$$

2.3 Оценка вычислительной и коммуникационной трудоемкости алгоритма

Временна́я сложность алгоритма определяется как функция от длины строки, представляющей входные данные, и обозначает время работы алгоритма на данном входе. Временная сложность алгоритма обычно выражается с использованием нотации *«О большое»*, которая учитывает только слагаемое самого высокого порядка, а также не учитывает константные множители, то есть коэффициенты. Если сложность выражена таким способом, говорят об асимптотическом описании временной сложности, то есть при стремлении размера входа к бесконечности. Например, если существует число n_0, такое, что

время работы алгоритма для всех входов длины $n > n_0$ не превосходит $an^3 + bn^2 + cn + d$, то временная сложность алгоритма $O(n^3)$. Время выполнения одной операции при этом считается константой, то есть асимптотически оценивается как $O(1)$.

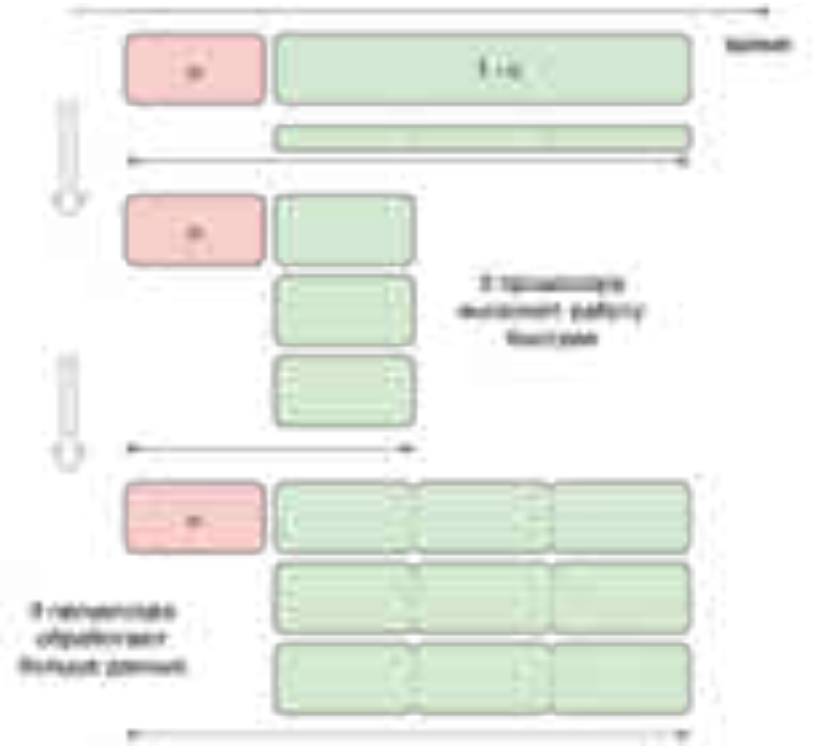


Рис. 2.2.1. Связь законов Амдала и Густафсона – Барсиса

На практике целесообразно учитывать вообще все операции алгоритма, не отбрасывая никакие степени и коэффициенты в функции зависимости от объема входных данных. Будем говорить о вычислительной сложности задачи: K – количество элементарных вычислительных шагов лучшего последовательного алгоритма, необходимых для решения задачи на одном процессоре (см. также п. 2.2). K является некоторой функцией от размера входных данных. Для простоты полагают, что все элементарные вычислительные шаги выполняются с одной и той же скоростью (например, и сложение, и умножение, и вычисление трансцендентных функций выполняются за 1 такт работы процессора), тогда, исходя из K и производительности вычислительной системы, можно сделать приблизительные выводы о времени решения задачи.

Для сложения n чисел $K = n - 1$, для скалярного произведения векторов $K = 2n - 1$, для перемножения матриц $K = n^2(2n - 1)$. Для поиска элемента в массиве $K = 1$ в лучшем (искомый элемент стоит в той позиции, с которой произошло первое сравнение) и $K = n$ в худшем случае (искомый элемент оказался последним в цепочке сравнений). Во всех этих примерах не учитываются операции работы с индексами массива (инкременты индекса для перехода к следующему элементу, к следующей строке матрицы и т.п.) – на самом деле, в программах все элементы могут быть просто поименованы явно (`A[0]`, `A[1]`, ...), это позволит обойтись без циклов вообще, но количество ключевых операций (умножений, сложений, сравнений) в задаче размера n от такой “развертки циклов” не изменится.

Помимо вычислительных операций в алгоритмах следует учитывать всевозможные задержки (latency) доступа к памяти всех видов. В модели параллельных вычислений процессоры *вынуждены* обмениваются информацией, кроме самых простых задач, где каждый процессор изолированно от других обрабатывает свою порцию данных. Естественным каналом для обмена является *память*. Эта память может быть разных типов (сверхоперативная, оперативная, внешняя). Даже если представить себе некую экзотическую фантастическую систему, в которой процессор передает свой результат непосредственно другому процессору – то и в этом случае переданный блок информации должен разместиться по крайней мере в регистрах процессора-получателя, а иначе *где*? Другой, тоже экзотический, но не настолько фантастический способ: один процессор своими действиями изменяет программу работы другого процессора, но и здесь воздействие передается через оперативную память (и – нет, это не обязательно должен быть самомодифицирующийся код, тут достаточно только строить программы как последовательность косвенных вызовов функций по адресам в совместно изменяемой таблице). В распределенных системах процессоры (точнее, распределенные вычислительные процессы) обмениваются данными посредством сетевых коммуникаций.

Начальный макет параллельного алгоритма обычно строится на основе анализа вычислительных операций. Время обмена информацией и стоимость этого обмена полагается пренебрежительно малой по сравнению со временем и стоимостью вычислений. Однако при практической реализации параллельного алгоритма все может обернуться полной противоположностью (см. комментарии к “технологическим стенам” в п. 1.3): “бутылочным горлом” станут не вычислительные ресурсы, а именно память. К тому, что сетевые коммуникации работают относительно медленно, программисты заведомо готовы. Но память? В конце концов, есть кэши, сопоставимые по скорости с регистрами процессора. Такого рода “добросовестные заблуждения” приводят к тому, что алгоритмы организуются неэффективно, и потенциал вычислительной системы не используется в должной мере.

Аппаратные спецификации быстро устаревают, но тем не менее полезно привести некоторые справочные сведения: на данный момент (2021 г.) общая пропускная способность подсистемы памяти на топовых графических ускорителях приближается к 2 Тбайт/сек. Количество ALU в тех же топовых GPU немного превышает 10000, каждое ALU выполняет 2 инструкции с плавающей точкой FP32 за такт. Частота GPU приближается к 2000 МГц. При совместной работе все ALU способны выполнить (10^4 ALU x 2 инструкции за такт) x (2 x 10^9 Гц) = 4 * 10^{13} = 40 TFLOPS.

Если для каждой выполняемой инструкции операнды (32-разрядные числа, 4 байта) потребуется брать непосредственно из памяти, то поток этих данных из “идеальной” (“мгновенной”) памяти будет 40 TFLOPS x 4 байт = 160 Тбайт/сек. И это если результаты операций не записывать обратно в память, что нереально – где их тогда хранить? (Конечно, разумно результаты одной операции использовать в другой операции без промежуточной записи из регистров в оперативную память, это правильное решение, но мы рассматриваем предельные случаи, да и не для всех алгоритмов нужна подобная цепочка.) Так или иначе, на топовых, наилучших и очень дорогих устройствах имеем *2 порядка отставания* пропускной способности памяти от пропускной способности вычислительных блоков.

Возвращаясь от конкретных цифр к принципам, следует отметить, что единственным по-настоящему эффективным каналом межпроцессорного обмена является *сверхоперативная* память. Например, аппаратура современных GPU как типичных представителей массово-параллельных вычислительных систем содержит так называемую разделяемую память (shared memory) [8], программно конфигурируемый набор ячеек с быстродействием, соответствующим быстродействию регистров. Однако есть ограничение: к одному блоку разделяемой памяти имеют доступ не все ALU, а только их ограниченное количество в составе так называемых *групп потоков*. Поэтому сверхоперативную память невозможно использовать для организации связи "каждый с каждым" между процессорами.

Вообще, топология связей между процессорами как раз и составляет основу анализа коммуникационной трудоемкости параллельных вычислений. Используется ряд характеристик, относящихся к топологии связей [9]:
- *диаметр* – показатель, определяемый как максимальное расстояние между двумя процессорами (под расстоянием обычно понимается величина кратчайшего пути между процессорами); данная величина может характеризовать максимально необходимое время для передачи данных между процессорами, поскольку время передачи обычно прямо пропорционально длине пути;
- *связность* (connectivity) – показатель, характеризующий наличие разных маршрутов передачи данных между процессорами; конкретный вид этого показателя может быть определен, например, как минимальное количество ребер, которое надо удалить для разделения графа передачи данных на две несвязные области;
- *ширина бисекции* (bisection width) – показатель, определяемый как минимальное количество ребер, которое надо удалить для разделения графа передачи данных на две несвязные области одинакового размера;
- *стоимость* – показатель, который может быть определен, например, как общее количество линий передачи данных в многопроцессорной вычислительной системе.

Время передачи данных между процессорами определяет коммуникационную составляющую (communication latency) длительности выполнения параллельного алгоритма в многопроцессорной вычислительной системе. Основной набор параметров, описывающих время передачи данных, состоит из следующего ряда величин:
- *время начальной подготовки* (t_s) характеризует длительность подготовки сообщения для передачи;
- *время передачи служебных данных* (t_h) между двумя соседними процессорами (т.е. для процессоров, между которыми имеется физический канал передачи данных); к служебным данным может относиться заголовок сообщения;
- *время передачи одного слова данных* по одному каналу передачи данных (t_w); длительность подобной передачи определяется пропускной способностью коммуникационных каналов.

Время пересылки данных размером m слов по маршруту длиной l определяется выражением $t_d = t_s + (mt_w + t_h)l$. Для достаточно длинных сообщений временем передачи служебных данных можно пренебречь: $t_d = t_s + mt_w\, l$.

Пусть некоторая задача решается на одном процессоре за время T, а на n таких же процессорах в условиях нулевых задержек на межпроцессорные коммуникации – за время $\frac{T}{n}$. На реальной многопроцессорной системе требуется обмен данными между процессорами, причем, на пересылки данных тратится тем больше времени, чем большее количество процессоров задействовано в работе.

Пусть на коммуникации при решении задачи тратится $T_{comm} = n \cdot \tau_{comm}$ единиц времени (вариант, когда задержки прямо пропорциональны числу процессоров). Тогда время решения задачи будет суммой "чистой" процессорной работы и задержек на коммуникации: $T_{sum} = \frac{T}{n} + T_{comm}$. Отношение $s(n) = \frac{T}{\frac{T}{n}+n\cdot\tau_{comm}} = \frac{n}{1+n^2\frac{\tau_{comm}}{T}}$ – ускорение при параллельном выполнении алгоритма с учетом задержек из-за коммуникаций.

Функция $s(n)$ имеет максимум, следовательно, при заданном параметре $\frac{\tau_{comm}}{T}$ ускорение начинает падать с ростом числа процессоров выше оптимального значения $n_0 = \sqrt{\frac{T}{\tau_{comm}}}$. Таким образом, чем крупнее система (чем больше n), тем более длительной должна быть и работа (больше T), чтобы обеспечивать оптимальные возможности по ускорению вычислений при фиксированных затратах на межпроцессорные коммуникации. Если взять вариант, при котором $T_{comm} = n^2 \cdot \tau_{comm}$ (задержки пропорциональны квадрату числа процессоров), то принципиально характер зависимости не изменится: $n_0 = \sqrt[3]{\frac{T}{2\tau_{comm}}}$.

К числу наиболее распространенных методов передачи данных относятся следующие два основные способа коммуникации: 1) *передача сообщений* как неделимых блоков информации и 2) *передача пакетов* как составных частей информационного потока.

При первом подходе процессор, содержащий сообщение для передачи, готовит *весь объем* данных для передачи, определяет процессор, которому следует направить данные, и запускает операцию пересылки данных. Процессор, которому направлено сообщение, осуществляет прием всех пересылаемых данных *полностью* и только затем приступает к обработке принятого сообщения (включая сюда его возможную передачу дальше, другим процессорам).

При втором подходе принимающий процессор может осуществлять частичную обработку и пересылку данных по дальнейшему маршруту непосредственно после приема очередного пакета, не дожидаясь завершения приема *всех* данных (особенно актуален этот способ в случае потоковой передачи, не ограниченной во времени явным образом).

Метод передачи пакетов обычно приводит к более быстрой пересылке данных и снижает потребность в буферной памяти. Кроме того, для передачи разных пакетов, составляющих одно сообщение или принадлежащих единому потоку данных, могут *одновременно* использоваться *разные коммуникационные каналы* (однако это полезное обстоятельство

может серьезно усложнить алгоритм сборки пакетов, на что тоже тратится время – поэтому нужен компромисс).

К основным операциям межпроцессорного обмена данными относят:
- *передача* данных *между двумя* процессорами;
- *передача* данных *от одного процессора всем остальным* процессорам (one-to-all broadcast);
- *прием на одном процессоре* данных *от всех остальных* процессоров (single-node accumulation);
- *передача* данных *от всех процессоров всем процессорам* (all-to-all broadcast);
- *прием на каждом процессоре* данных *от всех процессоров* (multi-node accumulation);
- *передача* данных *от одного процессора подмножеству* остальных процессоров, “разбрасывание” (scatter);
- *прием на одном процессоре* данных *от подмножества* остальных процессоров, “сбор” (gather).

Очевидно, что операции широковещательных рассылок (broadcast) и накапливающего приема (accumulation), а также операции разбрасывания и сбора (scatter и gather) являются двойственными друг к другу и используются совместно.

Рассмотрим пример, в котором происходит обмен данными в процессе выполнения параллельного алгоритма (*ленточный алгоритм умножения матриц*). Пусть имеется 4 вычислительных узла и умножаются матрицы, разделенные на 4 *ленты*. Предполагается, что матрицы очень велики и не могут целиком разместиться в памяти каждого из узлов, однако памяти достаточно для размещения одной ленточной строки и одного ленточного столбца.

Перед началом работы на вычислительные узлы рассылаются строки и столбцы исходных матриц в порядке номера узла: на 0-й узел идут 0-я строка и 0-й столбец, на 1-й – 1-я строка и 1-й столбец, на 2-й – 2-я строки и 2-й столбец, на 3-й – 3-я строка и 3-й столбец. Таким образом, узлы на 1-й итерации готовы вычислить диагональные блоки матрицы-результата. Произведя перемножение лент, узлы сохраняют соответствующие блоки результата у себя.

По окончании итерации строки первой матрицы, имеющиеся на узле, не удаляются. А вот имеющийся столбец пересылается узлу, имеющему номер на 1 больше по модулю 4 (по кольцу): так, 0-й узел отправляет свой столбец 1-му узлу, 1-й узел – 2-му узлу, 2-й узел – 3-му узлу, 3-й узел – 0-му узлу. Одновременно с выдачей своего столбца каждый узел осуществляет прием столбца, пересылаемого к нему узлом, имеющим номер на 1 меньше по модулю 4. По мере передачи-приема столбец в памяти узла перезаписывается, чтобы не расходовать лишнюю память.

Как только обмен столбцами закончился, вычислительные узлы приступают ко второй итерации. Пользуясь имеющейся у них лентой-строкой и новым присланным столбцом, узлы вычисляют новые блоки и сохраняют у себя.

Далее все продолжается тем же способом, и в итоге после 4-й итерации на вычислительных узлах будут готовы ленты, содержащие ленты-строки матрицы-результата (с номером, равным номеру узла). Теперь их следует отослать управляющему узлу, на котором строки

результата будут объединяться в полную матрицу. Этот управляющий узел должен содержать достаточно памяти для получения полной матрицы (но, возможно, ценой отсутствия у него собственных мощных вычислительных ресурсов, на которых можно было произвести умножение самостоятельно); как вариант, в условиях ограниченной памяти управляющий узел может занести результат на место одной из исходных матриц.

На рис. 2.3.1 – 2.3.4 показаны итерации алгоритма.

Рис. 2.3.1. Ленточное умножение, итерация 1

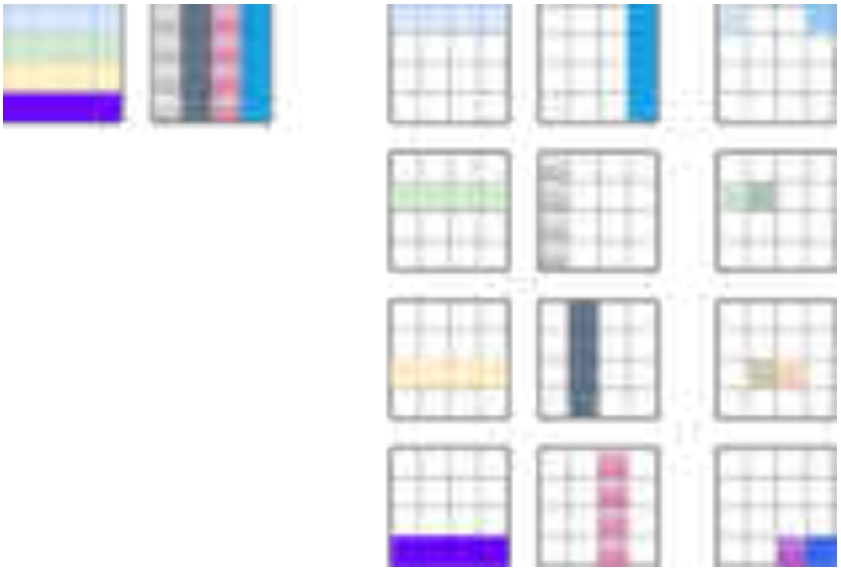


Рис. 2.3.2. Ленточное умножение, итерация 2

Рис. 2.3.3. Ленточное умножение, итерация 3

Рис. 2.3.4. Ленточное умножение, итерация 4

Этот алгоритм устроен таким образом, что данные между распределенными исполнителями передаются по кольцу. Следовательно, требования к топологии связей вычислительных узлов можно ограничить *кольцевой* структурой (рис. 2.3.5):

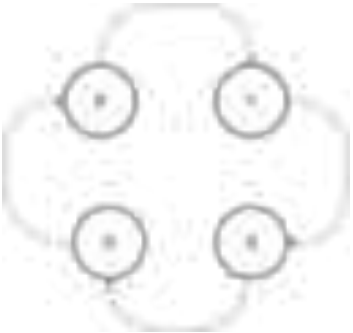


Рис. 2.3.5. Топология связей между вычислительными узлами – кольцо

В рассматриваемой задаче пересылка данных по отдельным звеньям является односторонней: передача осуществляется по одному звену, а прием – по другому звену, поэтому полоса пропускания каналов связи может использоваться полностью, без разделения пополам между передачей и приемом.

При исходных ограничениях задачи (на вычислительных узлах есть ограничения по памяти) ленточные столбцы, которые пересылаются между узлами, желательно передавать не как сообщения (целиком), а разбивать на пакеты. В таком режиме после отправки пакета, содержащего участок ленты, этот участок освобождается, и его можно занять теми данными, которые принимаются с другого узла, поэтому дополнительная память расходуется только на буферы отправки и приема. Но при этом требуется так организовать алгоритм, чтобы передавались и принимались упорядоченные участки ленты: скажем, отправляется сначала участок [0..m), затем [m..2m), затем [2m..3m), и так далее, где m – размер пакета; соответственно, прием будет осуществляться в том же порядке.

Не следует забывать, что исходные матрицы не хранятся на тех узлах, которые производят вычисления (они там не поместятся целиком), а полученный результат распределен по разным узлам. Поэтому в задаче необходим управляющий (центральный) узел, с которым вычислительные узлы связаны двунаправленными каналами (рис. 2.3.6), так как требуется передавать части исходных данных от центрального узла, а части результата – обратно на центральный узел.

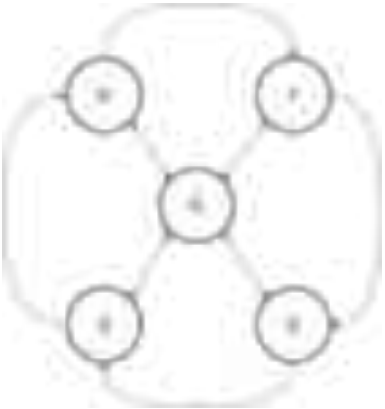


Рис. 2.3.6. Топология связей между вычислительными узлами с учетом управляющего узла

Строго говоря, центр не обязательно должен иметь связи со всеми узлами; можно соединить его с каким-то одним узлом (для определенности, с нулевым, рис. 2.3.7). Тогда нулевой узел будет получать исходные данные (ленты-строки и ленты-столбцы) от центра *по очереди* и передавать далее по цепочке кольцевых связей, пока последний узел кольца не пришлет нулевому узлу пакет специального вида “stop”, означающий, что исходные данные достигли всех вычислительных узлов. Естественно, каждый узел сохраняет у себя в памяти не все проходящие сквозь него данные, а только в соответствии с номерами лент, требуемыми алгоритмом. То, что канал связи нулевого и управляющего узлов является двунаправленным,

не создает двойной нагрузки в полосу пропускания, поскольку фазы рассылки исходных данных и приема результатов разнесены во времени.

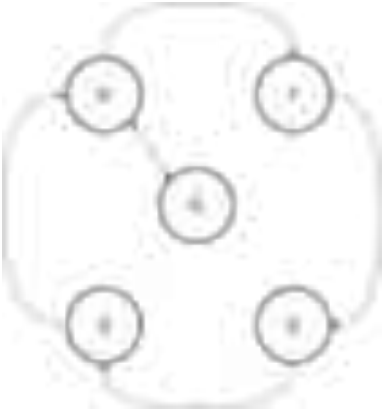


Рис. 2.3.7. Вариант упрощенной топологии связей между вычислительными узлами с учетом управляющего узла

2.4 Алгоритмы, ограниченные памятью (memory-bound) и вычислениями (compute-bound)

Некоторым параллельным программам требуется вычислять выражения по сложным формулам, в которых результат компонуется посредством арифметических операций, вызовов трансцендентных функций и других форм вычислительных действий. Приведем в качестве примера (без теоретических пояснений) формулу двумерного дискретного косинусного преобразования, которое положено в основу всем известного метода JPEG сжатия растровых изображений:

$$G_{uv} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{xy} cos\left[\beta_1 xu + \beta_2 u\right] cos\left[\beta_1 yv + \beta_2 v\right],$$

$0 \le u, v < 8,\ \alpha(n) = \left\{\frac{1}{\sqrt{2}} \text{ при } n = 0;\ 1 \text{ при } n > 0\right\}, \beta_1 = \frac{\pi}{8}, \beta_2 = \frac{\pi}{16}$, g – матрица 8x8.

С учетом того, что $\beta_1 u$, $\beta_2 u$, $\beta_1 v$, $\beta_2 v$, $\frac{1}{4}\alpha(u)\alpha(v)$ – это все константы, зависящие от *u* и *v* (так что можно организовать 64 разных процедуры вычисления G_{uv} или предварительно просчитать компактную таблицу констант), получаем в теле вложенного цикла 8 операций (умножение, сложение, cos). Эти 8 операций осуществляются в циклах, поэтому всего требуется выполнить $8^3 = 512$ вычислительных шагов, плюс к ним добавляется одно умножение на константу. Свыше 500 операций только для одного G_{uv}, а всего этих величин при разных *u*, *v* тоже требуется 8 x 8 = 64, так что для преобразования блока пикселей 8 x 8 всего потребуется свыше 32 тысяч операций. При этом *все* данные, которые требуются для вычислений, могут быть собраны в нескольких относительно небольших массивах 8x8; эти данные с высокой вероятностью поместятся в сверхоперативную память, обладающую быстродействием, сопоставимым с быстродействием регистров. (Что касается конкретного примера с дискретным косинусным преобразованием, то существуют эффективные *быстрые* алгоритмы для его вычисления, аналогичные алгоритмам быстрого преобразования Фурье. Этот факт, однако, не отменяет большой вычислительной нагрузки при расчетах по каким-то похожим формулам, для которых быстрых алгоритмов может и не быть.)

Смысл понятия параллельной процедуры, производительность которой определяется, в основном, вычислительными ресурсами (*compute-bound kernel*), может быть очень простым: если в процессе выполнения процедуры время, в течение которого загружены ALU, значительно превышает время, в течение которого осуществляется перемещение данных из памяти в регистры или из регистров в память, то эта процедура относится к классу compute-bound. Существует неформальный критерий, который, тем не менее, неплохо отражает характер этого класса процедур: если взять более быстрый процессор, и процедура на нем станет работать быстрее, то она относится к классу compute-bound.

Противоположное по смыслу понятие параллельной процедуры, производительность которой определяется, в основном, эффективностью доступа к памяти (*memory-bound kernel*) распадается на два класса: *bandwidth-bound* – критическая зависимость от *пропускной способности* памяти и *latency-bound* – критическая зависимость от *задержек* доступа к памяти.

Производительность параллельных вычислений очень часто ограничена памятью. Как правило, с такими ограничениями сталкиваются процедуры обработки очень больших массивов данных по очень простым алгоритмам. Например, сложение матриц (да в принципе и умножение матриц), построение гистограмм, сортировка, поиск – во всех этих задачах требуется прочитать из памяти одно или несколько чисел, использовать их в качестве операндов для одной-двух простых инструкций, записать результат обратно в память. При сложении матриц на два обращения к памяти (чтение и запись) приходится одна арифметическая инструкция, так что загрузка ALU составляет ⅓. Построение гистограммы – на каждый прочитанный элемент массива приходится одна операция инкремента, к тому же атомарного (транзакция *чтение-инкремент-запись*), то есть загрузка ALU – ¼. При сортировке имеем чтение двух элементов (память x 2), сравнение (ALU), возможную перестановку (память x 2) или какой-то подсчет в целевой ячейке (чтение-инкремент-запись), то есть тоже на уровне ½ загрузки ALU.

Тем не менее, задачи, в которых преобладает работа с памятью, широко распространены и требуют своего решения, а параллельные системы предоставляют подходящие средства для этого. Повысить эффективность загрузки обеих подсистем – арифметической и подсистемы памяти – можно с помощью “конвейерного” принципа (см. п. 1.4 и рис. 1.4.3). Основная цель этого подхода состоит в том, чтобы неизбежные задержки памяти (возможно, сотни тактов) не приводили к простою процессоров.

Рассмотрим псевдокод сложения матриц 1000 x 1000 в том стиле, в котором мы писали программы в п. 1.1:

```
const n = 1000

      // исходные матрицы доступны глобально
M1 = Array(n, n)
M2 = Array(n, n)

      // исходные матрицы каким-то образом заполнены,
```

```
      // и надо найти их сумму S = M1 + M2
S = Array(n, n)

function f(thread_id) {
            // определяем координаты своего потока
      row = thread_id[0]
      col = thread_id[1]
            // считываем свои элементы массивов
      el1 = read(M1, row, col)
      el2 = read(M2, row, col)
            // суммируем элементы и записываем результат на свое место
      write(S, row, col, el1 + el2)

      // read и write применены для акцентирования на обращениях к памяти
}

parallel_map_ndrange(action=f, range=[n*n], threadsToWait=n*n)
```

Это типичная процедура класса memory-bound. Все 1000x1000 потоков работают независимо, им не нужно обмениваться данными. Они просто считывают операнды, суммируют их и записывают результат.

Пока запрошенные операнды не получены из памяти, процессор не может осуществлять арифметические операции и будет вынужден *простаивать*, если не найти ему какую-нибудь работу на этот период. А какую работу, не связанную с этими ожидаемыми операндами, он может делать? Ведь в случае memory-bound вычислительная процедура настолько проста, что в параллельных потоках требуется всего-то одно арифметическое действие (как в сложении матриц), и для этого действия как раз и не хватает операндов.

Если рассматривать ситуацию с точки зрения *одного* потока, то да, действительно: *этому конкретному* потоку Th_n нечего делать. Но в программе создается множество потоков – их количество больше, чем доступных физических процессоров! Значит, нужно *мгновенно* снять поток Th_n с выполнения на физическом процессоре и *заменить* другим потоком Th_m, который еще не приступал к инструкциям выборки операндов из памяти (`read`). Поток Th_n отправится “спать”, но *физический процессор* не будет простаивать и сразу приступит к выполнению инструкции `read` в потоке Th_m. Здесь важно осознавать разницу между простоем потока и простоем физического процессора. Или, иначе, разницу между потоком (каким-то конкретным экземпляром цепочки инструкций со своей позицией продвижения по цепочке – счетчиком команд) и процессором (физическим вычислительным устройством, которое выполняет все эти цепочки инструкций).

Допустим, процессор приступит к выполнению `read` в потоке Th_m, поставленном взамен Th_n. А что это даст? Это же опять обращение к памяти – следовательно, опять задержка, опять поток не сможет продвигаться дальше по своей цепочке, пока не получены операнды.

Как ни парадоксально звучит, но правильное решение будет состоять в замене потока Th_m на… новый поток, который еще не приступал к выполнению инструкции `read`! И так далее: планировщик должен постоянно ставить все новые и новые потоки на физический

процессор, давая этим потокам выполнить инструкцию `read`. Точнее, не выполнить, а *выдать* (to *issue* the command); сразу после выдачи запроса к памяти инструкцию еще нельзя считать *выполненной*.

Причем, продолжаться это будет не бесконечно: рано или поздно память начнет отдавать запрошенные элементы исходных массивов, и их уже можно обрабатывать. Тогда планировщик вернется к тем потокам, которые он ранее отправил в "спячку" (Th_n, Th_m и т.д.) и даст им физический процессор. Правда, работать им придется недолго: после сложения они выдадут инструкцию `write` и завершатся (хотя к моменту завершения потока его результаты еще не будут физически записаны в память – ведь на запись, естественно, тоже есть задержки).

С высокой вероятностью запрошенные данные начнут поступать *раньше*, чем планировщик переберет *все* потоки, готовые выдать `read`. Все-таки задержки памяти, выраженные в количестве тактов, не столь велики по сравнению с количеством потоков, создаваемым для решения задачи; в примере со сложением матриц у нас *миллион* потоков, а задержки памяти реально составляют приблизительно *сотни* тактов.

Поэтому после получения первых порций операндов планировщик обязан "разбудить" потоки, остановившиеся на арифметической инструкции. Нельзя пока выдавать новые `read`, ведь данные из памяти поступают в регистры, а размер регистрового файла несравнимо меньше размера глобальной памяти. Надо освободить регистры, давая поработать потокам над "арифметикой", тем более, что после нее потоки выдадут `write`, и памяти снова будет чем заняться. А вот как только все "разбуженные" потоки отработают свои `+` и `write`, можно запускать очередной набор потоков, счетчик команд которых стоит на позиции `read`.

Проиллюстрируем по шагам эту стратегию диспетчеризации потоков. Для простоты, чтобы не загромождать таблицу, предположим, что имеется 2 процессора, а память разделена на 2 банка, доступ к которым осуществляется независимо; задержки памяти составляют 4 такта.

Шаг (такт)	Процессор 0	Процессор 1	Память (банк 0)	Память (банк 1)
0	read (id 0)	read (id 1)	получен запрос r (id 0)	получен запрос r (id 1)
1	read (id 2)	read (id 3)	получен запрос r (id 2)	получен запрос r (id 3)
2	read (id 4)	read (id 5)	получен запрос r (id 4)	получен запрос r (id 5)
3	read (id 6)	read (id 7)	получен запрос r (id 6)	получен запрос r (id 7)
4	+ (id 0)	+ (id 1)	данные прочитаны (id 0)	данные прочитаны (id 1)
5	+ (id 2)	+ (id 3)	данные прочитаны (id 2)	данные прочитаны (id 3)
6	+ (id 4)	+ (id 5)	данные прочитаны (id 4)	данные прочитаны (id 5)
7	+ (id 6)	+ (id 7)	данные прочитаны (id 6)	данные прочитаны (id 7)
8	write (id 0)	write (id 1)	буферизованы данные w (id 0)	буферизованы данные w (id 1)

9	write (id 2)	write (id 3)	буферизованы данные w (id 2)	буферизованы данные w (id 3)
10	write (id 4)	write (id 5)	буферизованы данные w (id 4)	буферизованы данные w (id 5)
11	write (id 6)	write (id 7)	буферизованы данные w (id 6)	буферизованы данные w (id 7)
12	read (id 8)	read (id 9)	[данные записаны (id 0)] получен запрос r (id 8)	[данные записаны (id 1)] получен запрос r (id 9)
13	read (id 10)	read (id 11)	[данные записаны (id 2)] получен запрос r (id 10)	[данные записаны (id 3)] получен запрос r (id 11)
14	read (id 12)	read (id 13)	[данные записаны (id 4)] получен запрос r (id 12)	[данные записаны (id 5)] получен запрос r (id 13)
15	read (id 14)	read (id 15)	[данные записаны (id 6)] получен запрос r (id 14)	[данные записаны (id 7)] получен запрос r (id 15)
16	… и так далее			

Из таблицы видно, что ни процессоры, ни память не простаивают, поскольку планировщик в каждом такте ставит готовые к исполнению потоки на процессоры и одновременно снимает потоки, блокированные в ожидании операндов инструкций. Память же занята либо получением запросов на чтение, либо выдачей запрошенных данных, либо внутренней буферизацией тех данных, которые необходимо перенести из регистров в ячейки оперативной памяти. Таким “скользящим окном” можно двигаться по программе, пока не будут обработаны все элементы данных.

Отсюда становится понятным, почему производительность памяти определяется как задержками (latency), так и пропускной способностью (bandwidth). В приведенном выше примере задержки были 4 такта, а пропускная способность характеризуется тем, что имеется 2 банка, по которым за каждый такт в совокупности передается 4 числа (по 2 операнда инструкции сложения). Если бы задержки были больше, то для их маскировки потребовалось бы не 8, а больше потоков. А если бы память имела только 1 банк, тогда запросы от разных процессоров выполнялись бы последовательно, и количество данных, передаваемых между всеми процессорами и памятью в единицу времени, снизилось бы в 2 раза.

Рассмотренная модель выполнения, в соответствии с которой в параллельных системах можно успешно покрывать задержки памяти, называется SIMT, мы уже неоднократно упоминали ее (пп. 1.1, 1.6). Не следует путать SIMT-модель с SIMD-моделью; они различны в большей степени, нежели имеют сходство. Эффективная аппаратная реализация SIMT-модели впервые была представлена в массово-параллельной архитектуре GPU NVIDIA в 2006-м году, и с тех пор этот подход широко применяется в разных вычислительных системах. Ключевая особенность SIMT – наличие очень большого числа потоков (массовый параллелизм в задаче), а также способность аппаратного планировщика переключать физические процессоры с одного потока на другой без каких-либо издержек, при необходимости хоть в каждом такте.

3. Многопоточная обработка в SMP

Лекция 4

3.1 Симметричные мультипроцессорные системы

Симметричная мультипроцессорность (symmetric multiprocessing или shared-memory multiprocessing, SMP) – это вычислительная архитектура, в которой два или более *идентичных* процессоров подключены к *общей памяти*, имеют полный доступ к устройствам ввода/вывода и находятся под управлением *одной копии операционной системы* (ОС), явно поддерживающей многопроцессорный многозадачный режим и предоставляющей различным задачам равный доступ к вычислительным ресурсам всех процессоров. Никакой из процессоров не резервируется для специальных задач. В настоящее время SMP является наиболее распространенной формой организации параллелизма, хотя и уступает в производительности другим архитектурам. Все многоядерные процессоры являются SMP-системами.

С точки зрения таксономии Флинна (п. 1.6) системы SMP относятся к классу MIMD: каждый процессор выполняет свои собственные инструкции, обрабатывая разные наборы данных. Лучше, однако, определять их как MPMD (Multiple Programs, Multiple Data).

Преимущество SMP для реализации параллельных алгоритмов заключается в удобстве межпроцессорных коммуникаций: поскольку память общая, то все процессоры могут обращаться к одним и тем же ячейкам памяти. При таком способе организации вычислений параллельная декомпозиция задачи упрощается: каждый процессор получает отдельную порцию работы, но все исходные данные могут оставаться в неизменном виде [10].

Кроме того, полная идентичность процессоров позволяет ОС переносить, при необходимости, выполнение какой-либо программы с одного процессора на другой – *балансировать нагрузку*.

Недостатком систем SMP являются существенные ограничения по *масштабируемости* (scalability): в SMP системе не может быть очень много процессоров, так как общая шина или схема межсоединений “каждый-с-каждым” на основе коммутаторов становятся “бутылочным горлом”. Если к общей шине подключить очень много процессоров, то она “задохнется” от нагрузки (процессоры будут постоянно простаивать в очереди за доступ к шине), а если реализовать межсоединения с помощью коммутаторов, то эти схемы получатся чрезмерно сложными и будут обладать очень высоким энергопотреблением (между n процессорами потребуется $n^2 - n$ быстродействующих соединений с количеством линий не меньше разрядности процессоров).

Есть еще один фактор, ограничивающий масштабируемость SMP: когерентность кэш-памяти (*cache coherence*) [11]. Кратко суть проблемы: каждый процессор в SMP имеет собственную кэш-память; содержимое кэш-памяти на *разных* процессорах должно быть *одинаковым*, иначе возникнет несогласованное состояние *общей памяти*, через которую происходит межпроцессорный обмен во время выполнения параллельных алгоритмов. Поддержание кэш-памяти в когерентном (согласованном) состоянии сильно затрудняется, если процессоров очень много.

SMP хорошо подходит для серверных систем – обработка клиентских запросов может распределяться между разными физическими процессорами (естественная параллельность, возникающая по причине независимости клиентов и, соответственно, их запросов друг от друга).

Сложнее обстоит дело с прикладными программами – их требуется специально проектировать таким образом, чтобы извлечь выгоду от запуска на многопроцессорной системе, при этом анализ задачи должен выявить подходящие для распараллеливания участки алгоритмов. По крайней мере, SMP-модель удобна для крупногранулярных параллельных подзадач, где совмещается во времени выполнение не каких-то отдельных процессорных инструкций, а достаточно крупных блоков, типа сортировки участков массивов, умножения блочных матриц и т.п. Дело в том, что *удобство* обмена через общую память не означает *эффективность* такого обмена, и если процессоры слишком часто будут вступать в *конкуренцию* за доступ к общей памяти, производительность системы значительно упадет.

С программной точки зрения параллелизм обеспечивается *процессами* (processes) и *потоками* (threads). Параллельные алгоритмы, ориентированные на совместную работу нескольких процессов, являются менее распространенными, чем алгоритмы, ориентированные на *многопоточную* обработку (multi-threading). Разные процессы имеют *разные адресные пространства*, поэтому взаимодействие между ними напрямую через память несколько затрудняется (необходимо создавать участки разделяемой памяти, которые отображают один и тот же участок физической памяти в разные адресные пространства). А при многопоточной обработке несколько потоков в составе одного процесса имеют доступ к одним и тем же адресами и, соответственно, ячейкам памяти. Кроме того, накладные расходы на создание потоков ниже, чем на создание процессов. Каждый поток имеет свой собственный стек, что позволяет ему независимо выполнять рекурсивные участки алгоритма.

Следует иметь в виду, что частое переключение потоков влечет накладные расходы, так как ОС должна переносить контекст выполнения потока между регистрами физического процессора и оперативной памятью. На SMP-системах, в отличие от GPU, не удается трюк с переключением потоков почти в каждом такте (реализовать такое в принципе возможно, но производительность упадет драматически).

Поскольку память в SMP-системах является разделяемым ресурсом, доступ разных физических процессоров (и, следовательно, потоков) к ее участкам должен быть синхронизированным. Если *на некотором участке* памяти не производится операций записи, то *читать* данные *этого участка* можно любым процессорам без ограничений; иначе дисциплина операций чтения и записи в параллельных потоках должна быть тщательно проанализирована. Удобным случаем является такая декомпозиция задачи, при которой каждый отдельный поток получает *собственный участок для записи* данных, не пересекающийся с участками других потоков.

Например, в программе умножения матриц на вычисление каждого элемента матрицы-результата может быть назначен свой поток (он вычисляет скалярное произведение

“строка на столбец” и записывает результат на отведенное данному потоку место), и синхронизация не нужна.

В других случаях можно построить параллельный алгоритм таким образом, чтобы потоки выполняли “основную массу” работы, записывая результаты в собственный*, выделенный “лично” для них* участок памяти и не конкурируя друг с другом, а затем *объединить* эти участки.

Рассмотрим пример с построением гистограммы значений элементов массива. Пусть массив `M` длиной `n` байт заполнен случайными значениями. Требуется в целочисленном массиве `H` длиной 256 элементов (по числу возможных значений, которые может принимать однобайтовая величина) построить гистограмму массива `M`. Количество доступных процессоров – `P`. Создадим `P` потоков и разделим работу поровну между ними, тогда каждому потоку достанется обработать участок массива `M` размером (`n / P`) байт.

Многопоточное накопление значений в ячейках массива `H` требует *атомарных* операций сложения (см. пример с построением гистограммы из п. 1.1); их по возможности следует избегать, так как они потенциально приводят к неоправданной конкуренции потоков за доступ к памяти. Поэтому удобно применить другой подход: выделим `i`-му потоку память под собственную версию гистограммы LH_i. После того как все потоки завершат работу, необходимо поэлементно сложить индивидуальные версии `LH`, получив итоговый массив `H`. Выполнять поэлементное сложение тоже можно *параллельно*, предоставляя каждому из `P` потоков вычисление (`256 / P`) элементов итоговой гистограммы (рис. 3.1.1). Заметим, что при “сборке” итоговой гистограммы атомарные операции сложения не требуются, так как потоки занимаются обработкой непересекающихся участков массива `H`.

Недостатком предложенного решения является дополнительный расход памяти на промежуточные массивы (их совокупный размер зависит от конкретной задачи, а также от количества используемых потоков). Потенциальным слабым местом может стать двукратный запуск групп потоков, сначала для построения частных гистограмм, затем для объединения гистограмм – но это зависит от реализации алгоритма; если использовать *пул потоков* (threadpool), то все потоки создаются однократно, но после *барьера* (завершено построение частных гистограмм) они начнут выполнять новую процедуру (сборку общей гистограммы).

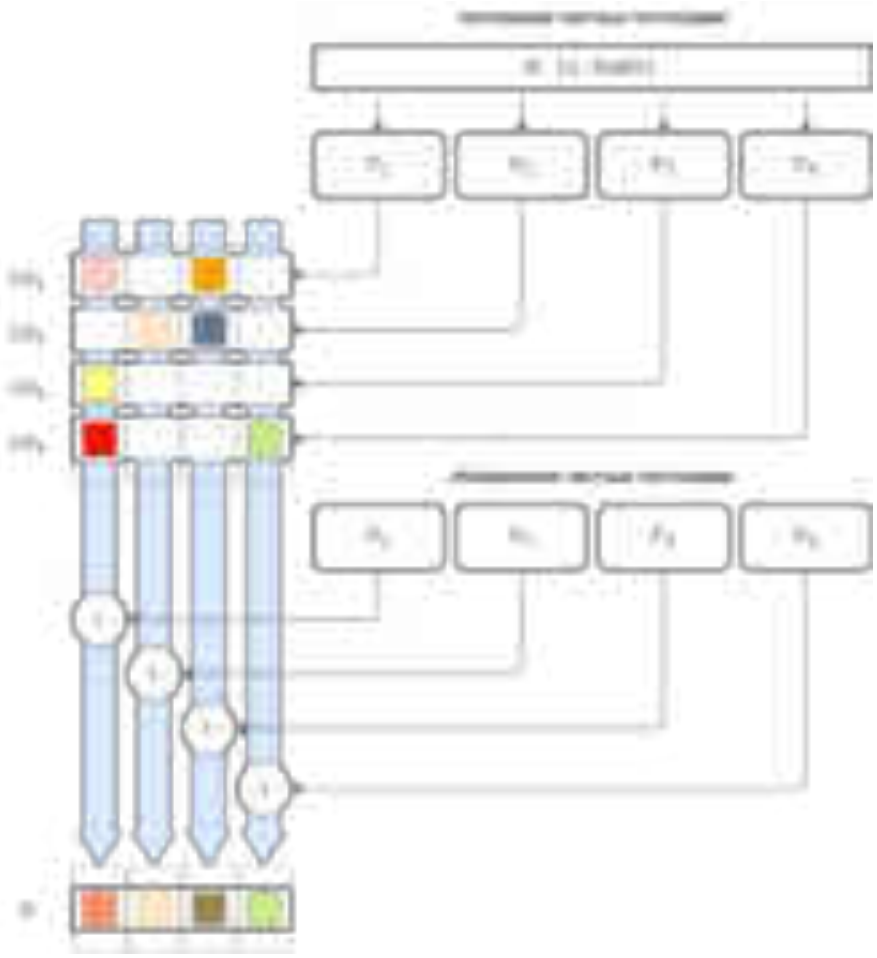


Рис. 3.1.1. Многопоточное построение гистограммы без применения атомарных инструкций

Псевдокод решения с пулом потоков (детали опущены):

```
M = Array(n)
LH = Array(P, 256)
H = Array(256)

function f1(id) {
	// построение частной гистограммы (M -> LH)(id)
}
function f2(id) {
	// сборка части финальной гистограммы (LH -> H)(id)
}

Threadpool pool(P)

foreach i in [0..P)
	pool.pushTask(f1, i) // i позволит потоку найти свою часть работы
pool.join()

foreach i in [0..P)
	pool.pushTask(f2, i) // i позволит потоку найти свою часть работы
pool.join()
```

Однако, не исключено, что для каких-то других задач изолировать потоки друг от друга будет затруднительно, так как непересекающихся подмножеств данных может и не оказаться. Поэтому в общем случае многопоточным программам требуются универсальные средства синхронизации (мьютексы, семафоры, критические секции, спин-блокировки); обычно они подробно рассматриваются в дисциплине “Операционные системы”, мы на них останавливаться не будем.

3.2 Интерфейс OpenMP

Современные оптимизирующие компиляторы способны до определенной степени выявлять параллелизм инструкций и выполнять автоматическое распараллеливание кода, имея на входе обычную однопоточную реализацию алгоритма. Но их возможности существенно ограничены, и полностью полагаться на них – значит, упускать потенциал увеличения производительности за счет явного применения параллельных вычислительных ресурсов.

Итак, программисты, знакомые с семантикой решаемой задачи, должны если не самостоятельно проектировать всю параллельную структуру алгоритма, то хотя бы “помогать” компилятору с ней разобраться.

Как это сделать, не меняя самого языка программирования и, по возможности, не затрагивая те участки кодовой базы, которые не нуждаются в распараллеливании? Реализация параллельных алгоритмов для высокопроизводительных вычислений обычно осуществляется на C/C++, часто и на Fortran. Эти языки должны обязательно поддерживаться в системах высокопроизводительных вычислений. Можно применять специальные библиотеки (их мы рассмотрим в следующем разделе), но у каждой библиотеки свой программный интерфейс, своя архитектура, своя идеология, паттерны и наилучшие практики, свои условия применимости, зависимости и ограничения; наконец, они могут поддерживаться не на всех аппаратных платформах и не в каждом системном окружении. Нужен какой-то унифицированный, *стандартизированный* подход к организации SMP-вычислений.

Таким стандартом является программный интерфейс OpenMP [12] для программирования на масштабируемых SMP-системах в модели с общей памятью (shared memory model). В стандарт OpenMP входят спецификации набора директив компилятора, процедур и переменных среды. Все самые распространенные компиляторы поддерживают эти спецификации.

Программа “Hello, World!” на OMP выглядит так:

```
#include <stdio.h>
#include <omp.h>

int main(void) {
    #pragma omp parallel
    {
        int ID = omp_get_thread_num();
        printf("Hello, World %d!\n", ID);
    }
```

```
    return 0;
}
```

Я обработал ее компилятором `gcc` из командной строки:

```
>gcc test.c -fopenmp -o test
```

Здесь флаг `-fopenmp` указывает на необходимость активировать при компиляции и сборке программы расширения OMP. Запускаем полученный исполняемый файл `test` и смотрим, что он выдает в поток вывода:

```
>./test
Hello, World 1!
Hello, World 0!
```

Заметим, что мы не писали `printf` дважды! Значит, прагма `omp parallel` была корректно обработана компилятором, и он *автоматически* создал параллельные ветви блока, который идет непосредственно за прагмой. В этом блоке с помощью функции `omp_get_thread_num` получен идентификатор потока и выведен в поток вывода. Эти действия отработали 2 раза, потому что для Docker-контейнера, в котором я все это тестировал, установлен лимит на 2 CPU. Вообще говоря, идентификатор ID правильнее называть *рангом* (rank), так как он не совпадает с идентификатором, под которым поток учитывается в ОС. Ранг принимает последовательные значения от 0, что удобно использовать для разделения работы между процессорами.

На рис. 3.2.1 представлены программные и аппаратные слои вычислительной системы, которые задействуются при выполнении OMP-программ на SMP-компьютерах. Приложение использует директивы компилятора и вызовы функций из библиотеки OpenMP. На системном уровне работает библиотека исполнения (OpenMP runtime), которая взаимодействует с ОС для создания потоков и управления доступом к памяти. Аппаратный слой представлен набором физических процессоров (ядер), обращающихся к общей памяти.

Прототипы функций объявлены в заголовочном файле `omp.h`. Наиболее важные функции (подробности, например, в [13]):
`omp_get_thread_num()` – возвращает идентификатор потока;
`omp_set_num_threads(int)` – установить необходимое количество потоков в группе;
`omp_get_num_threads()` – возвращает текущее установленное количество потоков в группе;
`omp_init_lock(omp_lock_t *lock)`, `omp_set_lock(omp_lock_t *lock)`, `omp_test_lock(omp_lock_t *lock)`, `omp_unset_lock(omp_lock_t *lock)`, `omp_destroy_lock(omp_lock_t *lock)` – функции работы со спин-блокировками, необходимые для синхронизации доступа к разделяемым ресурсам в памяти;

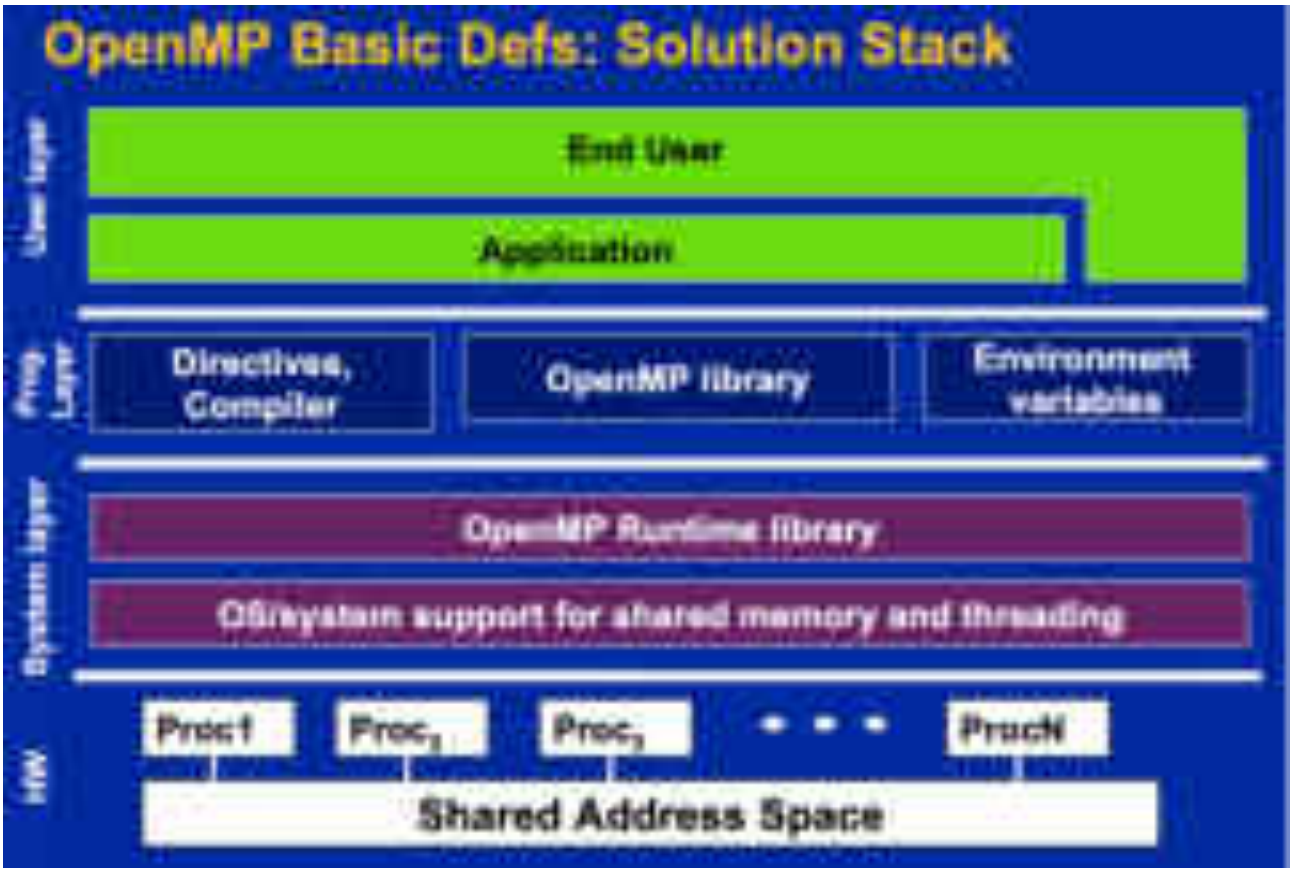


Рис. 3.2.1. Структура программно-аппаратных решений на основе OpenMP

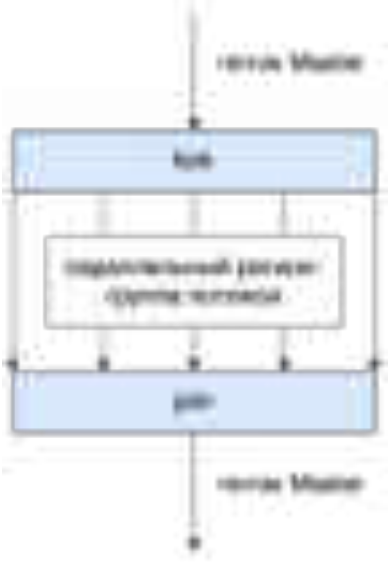


Рис. 3.2.2. Модель fork-join

Большинство OMP-конструкций задаются с помощью прагм, имеющих вид `#pragma omp ...` , например:

```
#pragma omp parallel num_threads(4)
```

OMP-конструкции применяются к *блокам* (в терминологии языка C/C++). Начало блока – это момент запуска необходимого числа потоков, конец блока – это *барьер*, на котором программа ожидает, когда все потоки дойдут до этой точки.

Модель исполнения в OMP относится к так называемому fork-join типу (рис. 3.2.2). Параллельный регион – блок кода, который всеми потоками исполняется одновременно. Параллельные регионы могут быть вложены, но поведение зависит от реализации. Работа в параллельном регионе распределяется между всеми потоками.

Одна из наиболее часто используемых конструкций – это параллельный цикл:
`#pragma omp parallel for`

Циклы с большим количеством итераций, как правило, являются узким местом и ключевым образом влияют на производительность. Конструкция `parallel for` позволяет разделить итерации цикла между потоками; например, если надо выполнить 1024 итерации, и в наличии имеется 2 процессора, то первому процессору достанутся итерации с 0-й по 511-ю, а второму – с 512-й по 1023-ю. Синтаксис организации параллельного цикла в OMP очень лаконичен и удобен:

```
#define N 1024
int A[N], B[N], C[N];

int i;
#pragma omp parallel for
for(i = 0; i < N; ++i) {
      C[i] = A[i] + B[i];
}
```

Также очень часто в параллельных программах встречается редукция (объединение многих значений в одно – сумма, минимум/максимум и многие другие), но “ручное” кодирование редукции является довольно громоздким. Вот как решает эту проблему OMP:

```
#define N 1024
int A[N];

int s = 0;
int i;
#pragma omp parallel for reduction(+:s)
for(i = 0; i < N; ++i) {
      s += A[i];
}
```

Как это работает? С помощью `reduction(+:s)` мы указали компилятору, что редукция аккумулирует значение в переменной `s` и что выполняемая при этом операция – сложение (+). Компилятор создает локальные для каждого потока переменные s_{local}, инициализирует их либо нулем, либо единицей, либо какими-то другими константами (зависит от операции; для операции сложения значение инициализации равно нулю, для умножения – единице), генерирует необходимый код параллельной редукции, объединяющий значения этих локальных переменных в одно, а после комбинирует полученное объединенное значение с тем, которое имела переменная `s` перед началом параллельного блока. Поддерживаемые операции: `+`, `-`, `*`, `&`, `|`, `^`, `&&`, `||`, `min`, `max`.

В OMP есть директивы для *барьеров* (barrier) и для указания *отсутствия необходимости в барьерах* (nowait):

```
#pragma omp parallel
{
// сначала все потоки совместно заполняют вспомогательный массив data
      id = omp_get_thread_num();
      data[id] = f1(id);
// ждем, пока все отработают над data
      #pragma omp barrier
// за барьером безопасно используем data для вычисления элементов A;
// не надо ждать пока все потоки отработают над A
      #pragma omp for nowait
      for(i = 0; i < N; ++i){
            A[i] = f2(i, data);
      }
// безопасно, так как "недоделанный" A здесь не используется
      counts[id]++
}
```

Иногда полезны *критические секции* и *атомарные операции* (но и вреда для производительности от них бывает немало, поэтому использовать их надо с осторожностью):

```
#pragma omp parallel
{
      Element el = createElement();
      #pragma omp critical
      {
            el.next = queue.last;
            queue.last = el;
      }

      #pragma omp for
      for(i = 0; i < N; ++i) {
            #pragma omp atomic
            Histogram[ A[i] ]++;
      }
}
```

OpenMP прошел долгий и успешный путь развития и к настоящему времени поддерживает очень много директив и функций, но общий стиль программирования на основе этого стандартизованного интерфейса понятен из нескольких приведенных примеров. Детальное понимание всех инструментов OpenMP требует, разумеется, внимательного изучения справочных материалов.

Лекция 5

3.3 Средства организации многопоточности в современных языках программирования, фреймворки и библиотеки времени исполнения

Несмотря на обилие удобных и эффективных языков программирования, C/C++ остаются основными языками, на которых создается ПО высокопроизводительных вычислений, поэтому сначала рассмотрим наиболее распространенные средства организации многопоточности для этих языков.

Если по каким-либо причинам средства OpenMP не подходят для реализации проекта на C/C++, содержащего параллельные вычисления, можно “погрузиться” в низкоуровневое программирование, чтобы с максимальной гибкостью управлять потоками и распределять между ними работу.

Многопоточность для программ на C

У каждой программы на C есть как минимум один поток – поток, выполняющий функцию `main()`. Программа может запустить дополнительные потоки, точкой входа в которые служит другая функция. Аналогично завершению программы при выходе из `main()` поток завершается при возвращении из функции, указанной в качестве точки входа.

При разработке многопоточных программ на языке C используется *стандарт* POSIX Threads (Pthreads) [14], определяющий API для создания и управления потоками, а также для их синхронизации. Библиотеки, реализующие этот стандарт, наиболее распространены для UNIX-подобных систем (включая Mac OS X). Многопоточные программы для Windows, написанные на C, обычно обращаются напрямую к Windows API (`CreateThread`), но есть реализации Pthreads, построенные как обертки над `CreateThread`.

Заголовочный файл – `pthread.h`. Все функции и типы данных снабжены префиксом `pthread_`.

Некоторые функции управления потоками (см. подробно в [15]):
- `pthread_create()` – создание потока;
- `pthread_join()` – ожидание завершения потока (если поток уже завершился, функция возвращает управление немедленно);
- `pthread_exit()` – завершение потока (поток можно завершить и просто возвратом из его функции, но если завершение необходимо сделать из какого-то вложенного вызова, то потребуется `pthread_exit`).

Некоторые функции для работы с объектами синхронизации:
`pthread_mutex_init()`, `pthread_mutex_destroy()`, `pthread_mutex_lock()`, `pthread_mutex_trylock()`, `pthread_mutex_unlock()` – функции работы с мьютексами (тип данных `pthread_mutex_t`);
`pthread_barrier_init()`, `pthread_barrier_wait()`, `pthread_barrier_destroy()` – функции работы с барьерами (тип данных `pthread_barrier_t`);
`pthread_cond_init()`, `pthread_cond_signal()`, `pthread_cond_wait()` – функции работы с условными переменными (тип данных `pthread_cond_t`);

Пример программы на Pthreads (построение гистограммы массива однобайтовых значений по схеме на рис. 3.1.1):

```
#include <string.h> // для memset
#include <pthread.h>

// количество элементов в массиве
#define n 1024
// количество рабочих потоков
#define threads_count 16
// количество элементов гистограммы
#define bins_count 256

// различные константы, необходимые для разделения работы между потоками
#define work_size1 (n / threads_count)
#define last_size1 ((n % threads_count) ? n - work_size1 * (threads_count - 1) :
work_size1)
#define histogram_block_size (bins_count * sizeof(int))
#define work_size2 (bins_count / threads_count)
#define last_size2 ((bins_count % threads_count) ? bins_count - work_size2 *
(threads_count - 1) : work_size2)

// массивы для обработки (доступны глобально):

      // входной массив
unsigned char M[n];

      // массив для гистограммы
int H[bins_count];

      // вспомогательные массивы (локальные гистограммы)
int LH[threads_count][bins_count];

// функция, исполняемая в потоках:
// построение гистограммы для “своего” участка входного массива
void* routine1(void* arg) {
  int id = (int)arg;

  int start = id * work_size1;
  size_t count = (id < threads_count - 1) ? work_size1 : last_size1;

  int i;
  int stop;
  for(i = start, stop = start + count; i < stop; ++i) {
      // инкремент в локальной версии гистограммы,
      // атомарные операции не требуются
    LH[id][M[i]]++;
  }

  return NULL;
}

// функция, исполняемая в потоках:
// объединение локальных гистограмм
```

```
void* routine2(void* arg) {
  int id = (int)arg;

  int start = id * work_size2;
  size_t count = (id < threads_count - 1) ? work_size2 : last_size2;

  int i, j;
  int stop;

  for(j = start, stop = start + count; j < stop; ++j) {
    int s = LH[0][j];
    for(i = 1; i < threads_count; ++i) {
            // суммирование по "своим" участкам локальных гистограмм,
            // атомарные операции не требуются
      s += LH[i][j];
    }
    H[j] = s;
  }

  return NULL;
}

int main(void) {
      // обнуляем память под локальные гистограммы
  memset(LH, 0, threads_count * histogram_block_size);

      // … каким-либо образом заполняем входной массив

      // создаем потоки, в каждом из них запускается routine1
  pthread_t th[threads_count];

  int i;
  for(i = 0; i < threads_count; ++i)
    pthread_create(&(th[i]), NULL, routine1, (void*)(long)i);

      // барьер: ждем, когда routine1 отработает во всех потоках
  for(i = 0; i < threads_count; ++i)
    pthread_join(th[i], NULL);

      // создаем потоки, в каждом из них запускается routine2
  for(i = 0; i < threads_count; ++i)
    pthread_create(&(th[i]), NULL, routine2, (void*)(long)i);

      // барьер: ждем, когда routine2 отработает во всех потоках
  for(i = 0; i < threads_count; ++i)
    pthread_join(th[i], NULL);

      // гистограмма в массиве H построена

  return 0;
}
```

Далее для сравнения предлагается, по сути, та же программа, но реализованная с помощью OpenMP; отличие здесь в том, что количество процессоров определяется автоматически (соответственно, память под локальные гистограммы выделяется в куче). Для pthread-версии

количество потоков было задано константой, поскольку Pthreads формально не включает API для определения количества процессоров, а другие способы не обладают хорошей переносимостью (можете поискать информацию о функциях `sched_getaffinity`, `sysconf(_SC_NPROCESSORS_ONLN)`; есть и другие подходы).

```
#include <string.h>
#include <omp.h>

#define n 1024
#define bins_count 256

unsigned char M[n];
int H[bins_count];

int main(void) {

      // ... каким-либо образом заполняем входной массив
      // ...

      // определяем, сколько у нас процессоров
    int num_threads = omp_get_num_procs();

      // вычисляем размер массива для локальных гистограмм
    size_t histo_size = bins_count * sizeof(int);
    size_t tmp_size = num_threads * histo_size;
      // выделяем в куче и обнуляем память локальных гистограмм
    int* LH = malloc(tmp_size);
    memset(LH, 0, tmp_size);

      // выполняем параллельное построение локальных гистограмм
    int i;
    #pragma omp parallel
    {
        int id = omp_get_thread_num();
        int* lane = LH + bins_count * id;
        #pragma omp for
        for(i = 0; i < n; ++i) {
            lane[M[i]]++;
        }
    }

      // выполняем параллельное объединение локальных гистограмм
    #pragma omp parallel for
    for(i = 0; i < bins_count; ++i) {
        int* pos = LH + i;
        int s = *pos;
        int j;
        for(j = 1; j < num_threads; ++j) {
            pos += bins_count;
            s += *pos;
        }
        H[i] = s;
    }
      // гистограмма в массиве H построена
```

```
      // освобождаем память массива локальных гистограмм
    free(LH);

    return 0;
}
```

OpenMP версия представляется более лаконичной. Однако для разных задач (с учетом интеграции всех модулей в единую систему) выбор может определяться совсем другими факторами.

Многопоточность для программ на C++

Потоки появились в C++ 11 (на данный момент действующим стандартом является C++20), поэтому современный C++ содержит встроенную поддержку потоков (см. справочную документацию в [16]).

Основной класс для создания новых потоков – `std::thread`.

```
#include <thread>

void threadFunction() {
     // тело выполняется в отдельном потоке
}

int main() {
     std::thread thr(threadFunction);
     thr.join();
     return 0;
}
```

Функция потока может принимать любое количество аргументов, но все они передаются по значениям. Можно использовать аргументы-указатели, если требуется возврат какой-либо информации из функции потока, или обертывать переменные, передаваемые по ссылкам в `std::ref`.

```
#include <thread>

void threadFunction1(int n, int* retValue) {
      int p = 1;
      for(int i = 2; i <= n; ++i) {
            p *= i;
      }
      *retValue = p;
}

void threadFunction2(int& retValue) {
      ++retValue;
}

int main() {
```

```
	int p1, p2 = 10;
	std::thread thr1(threadFunction1, 6, &p1);
	std::thread thr2(threadFunction2, std::ref(p2));
	thr1.join();
	thr2.join();
}
```

Некоторые полезные функции при работе с `std::thread`:

`std::this_thread::get_id()` – возвращает идентификатор того потока, откуда вызвана эта функция (тип возвращаемого значения – `std::thread::id`);

`std::this_thread::sleep_for()` – приостанавливает выполнение потока на некоторый промежуток времени; пример использования:

```
#include <chrono>
	// спит 2,5 секунды
std::this_thread::sleep_for(std::chrono::milliseconds(2500));
```

`std::this_thread::yield()` – дает указание планировщику переключиться на другие потоки, если нашему потоку в данный момент процессор “не очень нужен”, что может быть полезным при *активном ожидании* (например, если в цикле постоянно проверяется какое-то условие, на которое влияют внешние события, то лучше перемежать проверки с передачей процессора другим потокам).

Некоторые объекты синхронизации: `std::mutex`, `std::condition_variable`. Пример использования мьютекса:

```
#include <mutex>
Some obj; // ресурс, защищаемый мьютексом
std::mutex m_obj; // мьютекс, защищающий ресурс
m_obj.lock();	// получаем эксклюзивный доступ к ресурсу
	obj.doSome(); // потокобезопасная работа с ресурсом
m_obj.unlock();	// освобождаем доступ для других потоков
```

В стандарте языка C++17 появились *параллельные версии алгоритмов*, описанных в заголовочных файлах `<algorithm>` [17], `<numeric>` [18] и `<memory>` [19]. C++17 предлагает параметр политики выполнения для большинства алгоритмов (политики описаны в заголовочном файле `<execution>` [20]):

- `sequenced_policy` (последовательное исполнение алгоритма, соответствующий глобальный объект – `std::execution::seq`);
- `parallel_policy` (параллельное исполнение алгоритма, соответствующий глобальный объект – `std::execution::par`);
- `parallel_unsequenced_policy` (алгоритм применяет распараллеливание и SIMD-векторизацию, соответствующий глобальный объект – `std::execution::par_unseq`).

Пример использования параллельного алгоритма `transform`:

```
#include <execution>
#include <algorithm>
#include <vector>
#include <cmath>

      // функция преобразования
double f(double x) {
    return sin(x) * cos(x);
}

      // входной массив
std::vector<double> v(1000000);

      // заполняем массив данными
      // ...

      // применяем преобразование в параллельном режиме
std::transform(std::execution::par, v.begin(), v.end(), v.begin(), f);
```

Всего в C++17 было представлено 69 алгоритмов с поддержкой параллельного исполнения. Доступна, например, параллельная редукция:

```
double reduceOp(double a, double b) {
    return std::max(a, b);
}
double sum = std::reduce(std::execution::par, v.begin(), v.end(), -1.0, reduceOp);
```

Библиотека oneAPI Threading Building Blocks для C++

Библиотека oneAPI Threading Building Blocks (oneTBB) [21] – кроссплатформенная библиотека шаблонов C++ для параллельного программирования, основанная на ранее разработанной компанией Intel библиотеке TBB.

Библиотека включает

- базовые алгоритмы: `parallel_for`, `parallel_reduce`, `parallel_scan`;
- специальные алгоритмы: `parallel_pipeline`, `parallel_sort`;
- контейнеры: `concurrent_queue`, `concurrent_priority_queue`, `concurrent_vector`, `concurrent_hash_map`;
- операции выделения памяти: `scalable_malloc`, `scalable_free`, `scalable_realloc`, `scalable_calloc`, `scalable_allocator`, `cache_aligned_allocator`;
- мютексы: `mutex`, `spin_mutex`, `queuing_mutex`, `spin_rw_mutex`, `queuing_rw_mutex`, `recursive_mutex`.

Доступны также средства профилирования, а также планировки задач.

Многопоточность для программ на Java

В Java присутствуют класс `Thread` [22], интерфейс `Runnable` [23], ключевое слово `synchronized` и методы для синхронизации `wait()`, `notify()` и `notifyAll()` в классе `Object`. Объект текущего потока можно получить, вызвав статический метод: `Thread.currentThread()`. Идентификатор потока возвращается методом `getId()`. Приостановить выполнение потока на указанное количество миллисекунд можно вызовом метода `sleep(long ms)`. Вызов метода `yield()` дает указание планировщику переключиться на другие потоки.

Существуют два основных способа создания потока: 1) определить класс, являющийся наследником класса `Thread` и 2) определить класс, реализующий интерфейс `Runnable`.

Пример программы, которая использует потоки для параллельного поиска простого числа, большего или равного заданному:

```
      // 1-й способ: класс, наследующий Thread
class PrimeFindThread extends Thread {

  long startNum;
  long prime;

  PrimeFindThread(long startNum) {
    this.startNum = startNum;
  }

  public void run() {
    this.prime = Helper.search(this.startNum);
  }
}

      // 2-й способ: класс, реализующий Runnable
class PrimeFindRunnable implements Runnable {

  long startNum;
  long prime;

  PrimeFindRunnable(long startNum) {
    this.startNum = startNum;
  }

  public void run() {
    this.prime = Helper.search(this.startNum);
  }
}

class Helper {
  static boolean testIsPrime(long num) {
    if(num == 2) return true;
    if(num % 2 == 0) return false;
    long stop = (long)Math.floor(Math.sqrt(num));
    for(long i = 3; i <= stop; i += 2) {
      if(num % i == 0) return false;
    }
```

```
    return true;
  }

  public static long search(long startNum) {
    for(long i = startNum; ; ++i) {
      if(Helper.testIsPrime(i)) {
        return i;
      }
    }
  }
}

class Main {
  public static void main(String[] args) {

      // создание потока на основе пользовательского класса
    PrimeFindThread pft1 = new PrimeFindThread(12345678);

      // инициализация общего класса потока Runnable-объектом
    PrimeFindRunnable pfr = new PrimeFindRunnable(123456789);
    Thread pft2 = new Thread(pfr);

      // запуск потоков
    pft1.start();
    pft2.start();

    try{
            // ожидание завершения потоков
      pft1.join();
      pft2.join();

            // потоки отработали, значения вычислены:
      System.out.println(pft1.prime);
      System.out.println(pfr.prime);
    }
    catch(InterruptedException e) {
      System.out.println(e);
    }
  }
}
```

В современных версиях Java присутствует также пакет `java.util.concurrent` [24], который содержит большой набор классов для реализации параллельных программ. Concurrent Collections – набор коллекций, более эффективно работающие в многопоточной среде нежели стандартные универсальные коллекции из пакета `java.util`. Queues – неблокирующие и блокирующие очереди с поддержкой многопоточности. Synchronizers – вспомогательные утилиты для синхронизации потоков. Executors – средства для создания пулов потоков, планирования работы асинхронных задач с получением результатов. Locks – альтернативные и более гибкие механизмы синхронизации потоков по сравнению с базовыми `synchronized`, `wait`, `notify`, `notifyAll`. Atomics – классы с поддержкой атомарных операций над примитивами и ссылками.

Многопоточность для программ на Python

Система программирования Python, вообще говоря, не обеспечивает высокую скорость выполнения кода, поскольку данный язык является интерпретируемым. Следовательно, применение параллельных алгоритмов с целью повышения быстродействия может дать меньший эффект, чем переход от Python к компилируемым языкам (C/C++, C#, Java). Однако смена языка влечет за собой и смену всей экосистемы (среды исполнения, фреймворков, стандартных и сторонних библиотек, инструментов сборки, отладки, тестирования и т.д), что не всегда приемлемо. Поэтому при необходимости реализации программного проекта на Python в нем можно применить и средства организации параллельного выполнения задач.

В Python 3 присутствует стандартный пакет `threading`, содержащий необходимые средства для запуска потоков. Однако, на первой же странице документации есть замечание [25]:

CPython implementation detail: In CPython, due to the Global Interpreter Lock, only one thread can execute Python code at once (even though certain performance-oriented libraries might overcome this limitation). If you want your application to make better use of the computational resources of multi-core machines, you are advised to use multiprocessing or concurrent.futures.ProcessPoolExecutor. However, threading is still an appropriate model if you want to run multiple I/O-bound tasks simultaneously.

Вольный перевод: в реализации CPython из-за применения технологии глобальной блокировки интерпретатора (GIL) в каждый момент времени допускается выполнение кода только из одного потока (то есть, участки кода из разных потоков выполняются поочередно), хотя некоторые библиотеки, ориентированные на высокопроизводительную обработку данных, содержат внутренние методы для обхода этого ограничения. Для более эффективного использования вычислительных ресурсов многоядерных процессоров в приложениях рекомендуется использовать средства пакета `multiprocessing` [26] или класс `concurrent.futures.ProcessPoolExecutor` из пакета `concurrent.futures` [27]. Тем не менее, `threading` является подходящей моделью для одновременного выполнения нескольких задач ввода-вывода.

Когда в п. 2.4 мы рассматривали классы процедур compute-bound и memory-bound, мы не упоминали о существовании еще одной разновидности процедур: *IO-bound*. На самом деле, это важный класс процедур, и даже можно в какой-то степени считать, что именно они послужили двигателем развития многопоточности в системах программирования, причем, в те времена, когда процессоры большинства компьютеров были одноядерными и физический параллелизм отсутствовал. IO-bound означает, что производительность процедуры ограничена задержками *устройств ввода-вывода* (дисковые накопители, сеть), которые в совокупности можно рассматривать как внешнюю память вычислительной системы.

Поэтому, с другой стороны, IO-bound – это разновидность класса memory-bound, и методы маскирования задержек ввода-вывода, по сути, аналогичны таковым для оперативной памяти, просто абсолютные значения задержек на операциях ввода-вывода значительно выше. Соответственно, “логическая многопоточность” с поочередным, не одновременным

выполнением процессорных инструкций все равно будет являться подходящим средством для работы в условиях таких больших задержек, поскольку быстродействие даже единственного ядра CPU избыточно по сравнению с быстродействием внешней памяти, и процессор успеет за время передачи данных между внешней и оперативной памятью выполнить еще тысячи разных дел помимо опроса состояния операции ввода-вывода. Зато прикладная программа не будет вынуждена производить ввод-вывод последовательно, как при однопоточной организации. (Правда, в этом контексте есть серьезная оговорка: это верно только в том случае, когда операции ввода-вывода являются *блокирующими*, то есть, не возвращают управление, пока запрошенная операция фактически не завершится. Есть другой способ маскирования задержек на однопоточных системах исполнения – применение асинхронных, *неблокирующих* операций; однако эта тема выходит за рамки нашего курса и должна подробно рассматриваться в курсах операционных систем и системного программирования.)

Рекомендация [25] использовать параллельное выполнение, основанное на *процессах*, а не на потоках, позволяет обойти ограничение GIL, но следует иметь в виду, что накладные расходы на создание процесса выше, чем для потока. Поэтому желательно использовать *пулы процессов*, в которых заданное число процессов создается однократно, и затем эти процессы становятся доступными для многократного параллельного выполнения разных задач, без накладных расходов на порождение новых процессов.

Пример работы с пулом процессов:

```
from multiprocessing import Pool
import math

def testIsPrime(n):
  if n == 2:
    return True
  if n % 2 == 0:
    return False
  stop = math.floor(math.sqrt(n))
  d = 3
  while d <= stop:
    if n % d == 0:
      return False
    d = d + 2
  return True

def search(n):
  i = n
  while True:
    if testIsPrime(i):
      return i
    i = i + 1

primes = [0, 0]

if __name__ == '__main__':
    with Pool(4) as p:
        primes[0] = p.map(search, [12345678, 123456789, 55, 90])
```

```
        primes[1] = p.map(search, [1678, 123489, 5557, 901])

    print(primes)
```

Разные процессы имеют разные виртуальные адресные пространства, поэтому доступ к “одним и тем же” переменным в параллельной программе приводит, вообще говоря, к обращениям по разным адресам в памяти. Это нивелирует преимущества вычислительной модели SMP. Решение заключается в создании (при поддержке со стороны ОС) участков *разделяемой памяти* (shared memory), отображающих размещенные в них объекты на одни и те же физические адреса. Но пул процессов в Python разделяемую память не использует, а вместо этого входные задания на выполнение в пуле процессов и результаты выполнения сериализуются и переносятся (копируются) между разными адресными пространствами, что негативно влияет на производительность.

Для создания разделяемой памяти Python предоставляет пакет `multiprocessing.shared_memory` [28]. Для многих задач подходящим классом, реализующим все преимущества разделяемой памяти, будет `ShareableList`. Схема его использования: мастер-процесс создает экземпляр `ShareableList`, снабжая его некоторым именем. Рабочий процесс открывает созданный мастер-процессом экземпляр, передавая в конструктор `ShareableList` соответствующее имя. К элементам разделяемого участка можно обращаться по индексам.

Пример работы пула процессов с разделяемой памятью:

```
from multiprocessing import Pool
from multiprocessing import shared_memory
import math

#некоторое имя, идентифицирующее разделяемый участок
memoryName = 'SM1234'

workersCount = 4
itemsCount = 100
blockSize = itemsCount // workersCount

def testIsPrime(n):
  stop = math.floor(math.sqrt(n))
  if n == 2:
    return True
  if n % 2 == 0:
    return False
  d = 3
  while d <= stop:
    if n % d == 0:
      return False
    d = d + 2
  return True

def doWork(startIndex):
  # открываем разделяемый участок по имени
  a = shared_memory.ShareableList(name=memoryName)
```

```
  # обрабатываем свою порцию данных
  for i in range(startIndex, startIndex + blockSize):
    item = a[i]
    a[i] = testIsPrime(item)

if __name__ == '__main__':
  # в мастер-процессе создаем разделяемый участок
  a = shared_memory.ShareableList(range(itemsCount), name=memoryName)
  # заполняем разделяемый участок исходными данными
  for i in range(itemsCount):
    a[i] = 100 + i
  # создаем пул процессов
  with Pool(workersCount) as p:
    # выполняем обработку в параллельных процессах
    # (рабочие процессы найдут свои данные по стартовым индексам)
    p.map(doWork, range(0, itemsCount, blockSize))
  # результаты:
  for i in range(itemsCount):
    print(100 + i, '-', a[i])
  # разрушаем разделяемый участок
  a.shm.close()
  a.shm.unlink()
```

Раздел 4. Распределенная обработка в ММР

Лекция 6

4.1 Системы с массовым параллелизмом

Использование понятий “массовый параллелизм”, “массово-параллельные процессоры” получило широкое распространение после появления технологий организации вычислений общего назначения на GPU [29]. Современные GPU поддерживают десятки тысяч потоков, отсюда термин “массовость” – по контрасту с SMP-системами, на которых в настоящее время *десятки* ядер считаются значительными ресурсами и доступны только в очень дорогостоящих моделях (например, в [30] представлены характеристики топового сегмента серверных процессоров Xeon по состоянию на 2021 г.: до 40 ядер/80 потоков, на каждом ядре до 32 операций с вещественными числами двойной точности за 1 такт, максимальная тактовая частота до 4 ГГц, тепловыделение до 300 Вт, цена около 8000$).

Для CPU преодолеть дистанцию от *десятков* ядер (пусть даже с учетом того, что каждое ядро способно выполнять SIMD-операции) до *десятков тысяч*, присутствующих в современных GPU, на данном технологическом этапе невозможно, да и не требуется – у этих классов вычислительных устройств разные задачи. К GPU и другим массово-параллельным *акселераторам* мы подробнее обратимся далее, в разделе 4, а в данном разделе мы будем рассматривать “массово-параллельные” системы, построенные по другому принципу.

Как увеличить количество процессоров в системе в *тысячи* раз? Взять тысячи компьютеров и связать их сетью с высокой пропускной способностью. Недостаток SMP-систем – сложность организации эффективного доступа очень большого количества процессоров к единой

разделяемой памяти – не может быть преодолен. Значит, необходимо отказаться от концепции единой памяти. Разумеется, с отказом от нее мы потеряем много полезных свойств вычислительной системы; тем не менее, других способов сделать параллельные вычисления на CPU по-настоящему *массовыми* нет. (С другой стороны, этот же подход можно использовать для наращивания производительности систем, основанных на GPU.)

В SMP-системах логическая общая память программы совпадает с физической общей памятью (с оговорками про разные адресные пространства процессов), так что обеспечивается *однородный доступ к памяти* (uniform memory access, UMA). На самом низком уровне чтение и запись любых элементов данных на UMA-архитектуре производится инструкциями, входящими в систему команд CPU. (Для сравнения: без специальных технологий системного уровня невозможно, например, обращаться к элементам дисковых файлов с помощью процессорных инструкций чтения-записи, как мы обращаемся к элементам массивов – содержимое файловой системы, *внешняя память* уже не входит в сферу памяти с однородным доступом.)

Альтернатива SMP – это архитектура, разделяющая систему на многочисленные узлы, оснащенные собственной оперативной памятью. В таких системах доступ программы к данным будет осуществляться либо в пределах оперативной памяти узла (локальные элементы), либо через сеть – для доступа к *удаленным* (remote) элементам, физически расположенным в оперативной памяти других узлов. Узлам распределенной системы нет принципиальной необходимости работать под управлением одной и той же ОС – главное, чтобы та или иная ОС обеспечивала доступ к сетевым коммуникациям, а также поддерживала среду выполнения, необходимую прикладной программе.

Распределенная по разным узлам программа имеет потенциальную возможность производить операции над всей совокупностью данных задачи, где бы они не хранились. Однако методы, с помощью которых осуществляется доступ к конкретным элементам данных в *логически общем* хранилище, будут более сложными, по крайней мере, на низком, системном уровне. К тому же, разработчикам прикладного ПО придется учитывать в своих алгоритмах разную “стоимость” обращения к разным элементам данных.

При переходе от SMP-систем к системам более высокого масштаба применяется промежуточный уровень, при котором система строится из узлов, имеющих собственную память, но аппаратно-программное обеспечение объединяет эти физически разные блоки памяти в *единое адресное пространство*. Такой подход обозначают как *неоднородный доступ к памяти* (non-uniform memory access, NUMA), рис. 4.1.1. NUMA-системы, как правило, работают под управлением единой ОС.

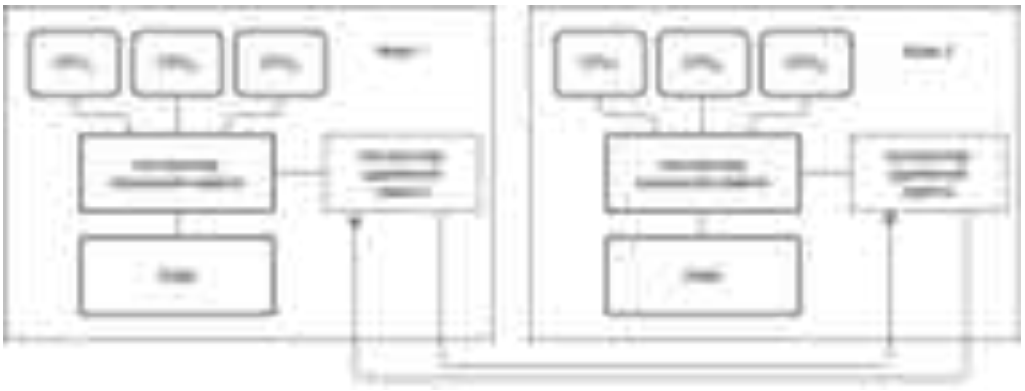


Рис. 4.1.1. Схема NUMA-системы с двумя узлами

В NUMA-системах в скрытой форме присутствуют недостатки SMP-систем, и к ним добавляются сложности, специфичные для NUMA. Программа “видит” память как общую, поэтому из разных потоков обращается к данным посредством процессорных команд доступа к памяти, а не с помощью системных функций ввода-вывода (как к сетевым или файловым операциям). Во время постепенного выполнения поток использует некоторое подмножество обрабатываемых данных, по возможности стараясь не пересекаться “своими” участками с участками других потоков, чтобы избегать необходимости синхронизации.

Но на самом деле память общей не является, и если поток начал выполняться на одном узле, а через некоторое время неактивности этого потока планировщик ОС запустил его уже на другом узле, то данные, к которым обращается поток, окажутся “дальними”, и доступ к ним будет значительно более медленным.

Для решения этой проблемы в ОС, поддерживающей NUMA, обязательно должна быть предусмотрена возможность программно задавать *привязку потоков* к конкретным физическим устройствам (процессорам). Такая привязка обозначается термином *processor affinity*. Однако в реальных алгоритмах даже при наличии привязки рано или поздно потокам станет необходимо обращаться к “дальней” памяти, что приведет к неизбежным задержкам.

По этой причине большинство NUMA-систем устроены таким образом, что доступ к удаленной памяти производит принудительную синхронизацию кэш-памяти (как отмечалось в п. 3.1, поддержка когерентности кэшей необходима и для SMP-систем). Такие системы называют ccNUMA (cache-coherent NUMA). Упрощенно, при доступе к памяти данные сначала ищутся в локальном кэше процессора, затем в кэше “своего” узла. Если данных нет в кэше этого узла, то они ищутся в кэшах других узлов (для выполнения такого поиска предусмотрены специальные аппаратные связи). Если необходимых данных нет в кэше ни у одного из узлов, то они считываются из оперативной памяти того узла, которому соответствует запрашиваемый адрес, и тем самым автоматически заносятся в кэш. Таким образом, при соблюдении дисциплины доступа, *дружественной к кэш-памяти* (cache-friendly), новые данные будут с высокой вероятностью выбираться из локального или удаленного кэша.

Решают ли NUMA-системы задачу организации высокопроизводительных вычислений? Безусловно, в какой-то степени. NUMA-системы масштабируются значительно лучше, чем SMP. Но можно ли на этой архитектуре построить систему с массовым параллелизмом? Ответ, в общем случае, отрицательный.

Массово-параллельная архитектура (massively multiprocessing, MMP, или massively parallel processing, MPP) определяет класс вычислительных систем, которые строятся из отдельных узлов с физически разделенной памятью и *не предполагают существования единого адресного пространства*. Здесь распределенная программа изначально строится таким образом, что данные, необходимые для решения задачи, не рассматриваются как совокупность элементов, размещенных в некоторой “логической оперативной памяти”. Вместо процессорных команд доступа к памяти программа “осознанно” применяет специальные функции ввода-вывода, чтобы прочитать или записать удаленные элементы.

Системы такого типа обычно называют вычислительными кластерами (рис. 4.1.2). Чтобы нарисовать эту иллюстрацию, я всего лишь немного изменил схему NUMA-системы. Действительно, между этими разновидностями высокопроизводительных архитектур очень много общего. Кластер – это группа компьютеров, соединенных высокоскоростными каналами связи. С точки зрения пользователя кластер представляется единой вычислительной системой. Задержки доступа к удаленной памяти для кластерных систем определяются характеристиками используемых технологий локальных сетей, но в любом случае они на порядки выше (медленнее), чем в NUMA.

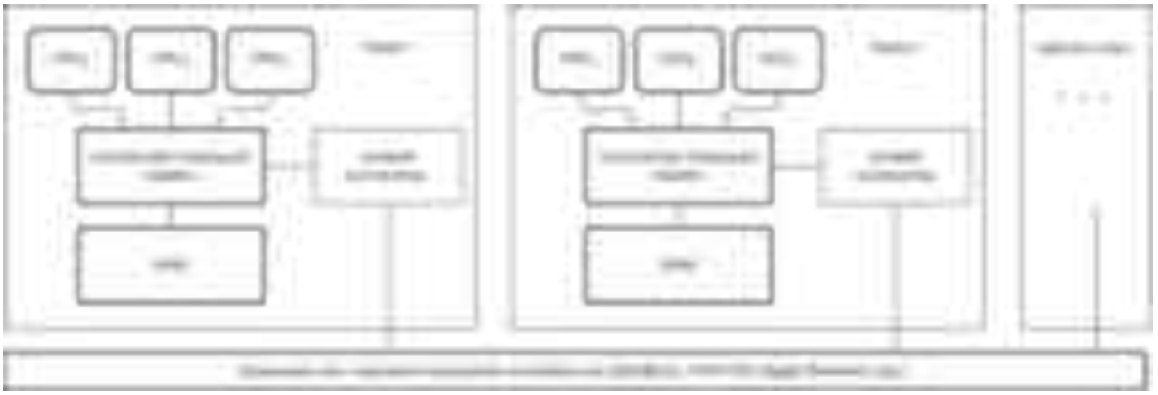


Рис. 4.1.2. Схема MPP-кластера

В кластеры могут объединяться сотни тысяч отдельных *многоядерных* процессоров. По состоянию на ноябрь 2021 г. наиболее производительным суперкомпьютером признан Fugaku [31], разработанный японской фирмой Fujitsu. В нем применяется 158976 отдельных узлов, в каждом из которых установлен 48-ядерный процессор, что в совокупности дает 7630848 вычислительных ядер. Эта система с массовым параллелизмом обладает производительностью свыше 400 петафлопс.

4.2 Обмен сообщениями как основа межпроцессорных коммуникаций в системах с распределенной памятью

В системах с общей памятью (SMP, NUMA) применяется очень простая модель программирования: для обмена данными между процессорами один из них записывает в память значение по некоторому адресу, а другой читает значение по этому адресу. Единственное *логическое* требование здесь состоит в том, чтобы процессор *B* выполнял чтение из указанной ячейки строго после того, как процессор *A* выполнил запись в эту ячейку, и при этом никакой третий процессор *E* не мог выполнить запись в эту же ячейку после *A*, но до ее чтения процессором *B*. Обеспечивается это специальными средствами синхронизации: с некоторым разделяемым ресурсом в памяти связывается мьютекс или спин-блокировка, или критическая секция, и разные потоки обязаны получать доступ к ресурсу только через предварительный захват объекта синхронизации, чтобы иметь возможность безопасно изменять состояние разделяемого ресурса.

В системах без общей памяти (кластеры) модель программирования вынужденно усложняется. Сами по себе узлы кластера являются SMP-системами, поэтому какая-то часть задачи программируется в соответствии с моделью общей памяти. Но для обмена данными с другими узлами необходимо применять сетевые протоколы, а следовательно, параллельная программа должна быть осведомлена, каким образом потоки (процессы) распределены по узлам, да и вся логика обращений к удаленным ресурсам будет отличаться от логики работы с локальными ресурсами. Поэтому целесообразно найти способы унификации работы с данными.

С одной стороны, обмен сообщениями как способ взаимодействия между объектами универсален и может применяться на самых разных масштабах. Между прочим, в SMP-системах на низком (аппаратном) уровне при организации протоколов кэш-когерентности тоже фактически используется передача сообщений [32].

С другой стороны, теоретически возможна виртуализация распределенной памяти с применением сквозной общей адресации (виртуальное адресное пространство, покрывающее память всех узлов), тогда попытки обращения процесса к участку памяти, который физически находится на другом узле, будут приводить к прерыванию и запуску сетевого обмена данными. Для прикладной программы обращение к удаленной памяти будет “прозрачным”, за исключением колоссальной разницы в задержках. Такой подход применяется редко из-за того, что он косвенно поощряет создание неэффективных программ, так как у разработчиков алгоритмов могут возникнуть ложные иллюзии относительно доступности тех или иных ресурсов и “цены” доступа к ним.

Одно из преимуществ обмена сообщениями – *асинхронность*: программа, отправив сообщение, может, не дожидаясь ответа, приступить к выполнению других действий; соответственно, когда ответ будет получен, программа вернется к его обработке, руководствуясь *приоритетностью* отложенных задач. Паттерн асинхронного выполнения применяется как с целью маскирования задержек передачи данных, так и с целью организации “квазипараллельной” обработки данных с помощью *единственного* потока, находящегося в “вечном” цикле, на каждой итерации которого осуществляется проверка готовности значений аргументов той или иной операции; если аргументы готовы, операция выполняется, иначе выдается асинхронное задание на подготовку данных, чтобы на следующих итерациях поток смог приступить к их обработке.

Недостаток асинхронности – усложнение программирования; тут важна роль фреймворка, насколько удобно в нем организовано разбиение всех действий на две фазы: старт асинхронной операции и учет результатов асинхронной операции. В принципе, нередко можно обойтись синхронными версиями передачи сообщений (коммуникационная функция не возвращает управления, пока ответ на сообщение не получен), вызывая их в разных потоках; при этом планировщик ОС должен переводить потоки в неактивное ожидание на время выполнения сетевого ввода-вывода.

4.3 Интерфейс MPI, группы процессов и коммуникаторы, двухточечные и коллективные обмены

Наиболее распространенной технологией программирования параллельных компьютеров с распределенной памятью является MPI (Message-Passing Interface) [33]. MPI – это спецификации интерфейса библиотеки программ. Существуют разные реализации этого стандарта (например, [34, 35]).

Программная модель MPI предполагает, что данные перемещаются из адресного пространства одного процесса в адресное пространство другого процесса, и при этом оба процесса (передающий и принимающий) совместно участвуют в выполнении действий по передаче данных: один из них посылает данные (операция *send*), а другой принимает данные (операция *receive*) [36]. Среда исполнения осуществляет запуск некоторого количества MPI-процессов (допустим, *n* процессов), каждый из которых получает свой номер (ранг), от 0 до *n - 1*. Зная свой ранг, процесс может использовать его для выбора различных действий. Обычно процесс с рангом 0 является мастер-процессом, определяющим ход выполнения параллельной обработки данных.

Какой минимальный интерфейс необходим для передачи сообщений?

В первом приближении, для функции *send* необходимо указать *по какому адресу* и *сколько байт* считывается в локальном адресном пространстве для последующей передачи, а также *кому* передаются эти данные (номер процесса-получателя).

Для функции *receive* аргументы почти такие же: по *какому адресу* и *сколько байт* записать в локальное адресное пространство по окончании приема данных. С точки зрения параметризации вызова *receive* “сколько байт” – это *максимально* допустимое количество, размер отведенного буфера; а фактически принятое количество следует записать в целочисленную переменную, адрес которой тоже будет аргументом функции.

Получателю желательно также знать *от кого* пришла информация, поэтому целесообразно снабдить функцию *receive* аргументом, обозначающим номер процесса, от которого ожидаются данные. Если необходимо принимать данные от разных процессов, потребуется вызвать функцию несколько раз с разными номерами. Алгоритм может включать прием данных *от любого* отправителя, поэтому следует допускать возможность вместо конкретного номера указывать специальное значение MPI_ANY_SOURCE. Чтобы принимающая сторона все-таки могла узнать, от кого пришли данные в этом случае, функция *receive* должна

предусматривать дополнительный аргумент – указатель на целочисленную переменную, в которую будет занесен номер процесса-отправителя.

Этот минимальный интерфейс уже позволяет построить работоспособный распределенный вычислительный алгоритм. Но предложенный простейший вариант обладает серьезным недостатком: все сообщения рассматриваются как бы в “одной плоскости”, так что у процесса нет возможности *фильтровать* входящие сообщения (по тематике, по приоритетности и т.д.).

Предположим, что для выполнения каких-то действий процессу нужно сначала получить сообщение, относящееся к категории A, преобразовать его в некий промежуточный результат *tmp(A)*, а затем получить сообщение категории B и преобразовать его в окончательный результат *r(tmp(A), B)*. Если сообщения приходят от разных процессов вперемешку, то получатель будет вынужден отдельно хранить такие сообщения B, для которых еще не приняты соответствующие сообщения A.

Все это усложняет прикладное программирование, поэтому можно возложить сортировку сообщений по тегам на служебную среду (будут созданы разные очереди для сообщений с разными тегами), а в API ввести дополнительный аргумент *тег*: отправитель передает сообщение, снабжая его тегом, получатель указывает тег в качестве аргумента функции приема, а функция *receive* игнорирует сообщения, не соответствующие по тегу. Тогда алгоритм процесса-получателя из приведенного выше примера приобретает более удобную форму:

```
for each i in senderRanks {
      receive(&dataA, dataSizeA, i, TAG_A)
      tmp = process(dataA)
      receive(&dataB, dataSizeB, i, TAG_B)
      r[i] = process(tmp, dataB)
}
```

При анализе этого псевдокода следует иметь в виду, что служебная среда работает “параллельно”, независимо от последовательности вызовов *receive*. Поэтому принимаются, разумеется, все сообщения от всех процессов в том порядке, в котором они поступают на сетевой интерфейс. Но вызов *receive* обращается к какой-то конкретной очереди (согласно тегу), которая к моменту этого обращения либо уже содержит необходимые данные, либо еще нет. Если данных нет, то в ожидании блокируется только тот поток, в котором выполняется *receive*, но не потоки, обслуживающие сетевые коммуникации. По той же аналогии, что и с номерами отправителей, желательно предусмотреть специальный “метатег” MPI_ANY_TAG, означающий прием сообщений с *любыми* тегами (а конкретный тег записывать по указателю – дополнительному аргументу функции).

В описываемой пробной версии API данные передаются и принимаются как цепочки байт некоторой длины. Но MPI-программы предназначены для высокопроизводительных вычислений, а такого рода действия обычно оперируют с числами, векторами чисел, матрицами чисел. Числа могут иметь разную структуру представления (вещественные с плавающей точкой, целые со знаком или без знака) и длину (от одного до нескольких байт).

Поэтому MPI предлагает в функциях передачи и приема указывать *тип данных*. Соответственно, *количество* будет относиться к передаваемым элементам данных (скажем, 100 для массива из ста целых чисел), а не к байтам.

MPI не является ни языком программирования, ни его расширением, он не вмешивается в систему типов языков, используемых для написания MPI-программ (обычно это C/C++ или Fortran), но обязан учитывать базовые целочисленные и вещественные типы, присутствующие в этих языках. Для этого основные типы кодируются специальными константами, их необходимо передавать в функции *send* и *receive* в качестве параметра, задающего тип передаваемых или принимаемых данных. Таблица соответствия некоторых типов:

тип C	обозначение в MPI
`char`	`MPI_CHAR, MPI_SIGNED_CHAR`
`unsigned char`	`MPI_UNSIGNED_CHAR, MPI_BYTE`
`short int`	`MPI_SHORT`
`unsigned short int`	`MPI_UNSIGNED_SHORT`
`int`	`MPI_INT`
`unsigned int`	`MPI_UNSIGNED`
`long int`	`MPI_LONG`
`unsigned long int`	`MPI_UNSIGNED_LONG`
`float`	`MPI_FLOAT`
`double`	`MPI_DOUBLE`
`long double`	`MPI_LONG_DOUBLE`

MPI также позволяет создавать пользовательские типы, в том числе такие, элементы которых не являются смежными с точки зрения их расположения в памяти (non-contiguous types). Доступно создание пользовательских структур (аналогичных типу, определяемому `struct` в языке C). Подробнее о способах создания пользовательских типов см. документацию [33] (функции `MPI_Type_contiguous`, `MPI_Type_vector`, `MPI_Type_indexed`, `MPI_Type_create_struct` и другие).

Выше обсуждалась необходимость фильтрации сообщений, и было обосновано решение использовать в MPI теги сообщений. Оказывается, одних только тегов недостаточно для построения по-настоящему практичной системы. В самом деле, и процесс-отправитель, и процесс-получатель должны быть согласованы с точки зрения применяемых тегов; иначе говоря, *исходный программный код* отправки и приема должен контролироваться разработчиком распределенной программы как единое пространство именования, чтобы взаимодействующим операциям отправки-приема были назначены согласованные теги из некоторой общей таблицы.

Но проблема возникает в том случае, если допустить использование *библиотек*, работающих поверх базовых средств MPI: очевидно, библиотеке для выполнения своих действий в параллельном режиме потребуется обмен сообщениями между процессами – следовательно, у каждой библиотеки должно быть уникальное подмножество тегов сообщений, не пересекающееся с тегами прикладной программы и тегами других библиотек. Доступа к исходному коду библиотек с целью какой-либо модификации системы именования сообщений, в общем случае, нет.

Решением этой проблемы является введение дополнительной сущности, описывающей обмен данными. В MPI эта сущность называется *коммуникатором* (communicator). Упрощенно говоря, в разных коммуникаторах существуют свои "вселенные" обмена данными. Предопределен коммуникатор `MPI_COMM_WORLD`, в котором представлены с последовательными номерами *все процессы* MPI-программы. Прикладная программа и библиотеки могут создавать новые коммуникаторы, чтобы избежать конфликтов и пересечений с другими участками распределенной схемы обработки. В одном и том же коммуникаторе все процессы имеют различные номера, но поскольку процесс может входить в разные коммуникаторы, то его номер в одном коммуникаторе может как совпадать, так и отличаться от номера в другом коммуникаторе. Следовательно, только пара *(номер, коммуникатор)* является идентификатором процесса.

С учетом замечаний о типах данных, тегах сообщений и коммуникаторах блокирующие версии функций передачи и приема сообщений имеют следующие заголовки в MPI:

```
int MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm);

int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag,
MPI_Comm comm, MPI_Status *status);
```

Аргумент `status` является указателем на структуру, объединяющую сведения о фактическом отправителе сообщения, фактическом теге сообщения и коде ошибки, если она имела место. Предусмотрена специальная константа `MPI_STATUS_IGNORE` для игнорирования статуса.

Практически в каждой MPI-программе используются еще несколько служебных функций:

```
int MPI_Init(int *argc, char ***argv);
int MPI_Finalize(void);
int MPI_Comm_rank(MPI_Comm comm, int *rank);
int MPI_Comm_size(MPI_Comm comm, int *size);
```

`MPI_Init` и `MPI_Finalize` являются "парными" операциями, выполняющими, соответственно, инициализацию среды исполнения и завершение ее работы. У функции `MPI_Init` аргументы соответствуют *адресам* аргументов функции `main`. Функция `MPI_Comm_rank` записывает номер (ранг) текущего процесса в группе, определяемой

коммуникатором comm. Функция MPI_Comm_size записывает количество процессов в группе, определяемой коммуникатором comm.

Ниже приведен пример MPI-программы, выполняющей разложение достаточно больших (тип unsigned long) чисел на простые множители. Цель этого примера – дать общее представление о том, как "устроены" MPI-алгоритмы. (Эта программа не является оптимальной, и с высокой вероятностью рассматриваемая задача будет быстрее решена без каких-либо межпроцессорных коммуникаций.) В примере иллюстрируется применение тегов сообщений, работа с разными типами данных, а также логика выбора действий для разных процессов в зависимости от их ранга – все это, в сущности, и составляет основу MPI-программирования.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <typeinfo>
#include <mpi.h> // основной заголовочный файл MPI

      // вспомогательная функция,
      //    определяющая сколько раз число n делится без остатка на d;
      // если найденное количество больше нуля,
      //    то оно заносится в вектор powers, а делитель d – в вектор factors;
      // во время выполнения число n модифицируется по ссылке,
      //    чтобы данную функцию было удобно вызывать циклически,
      //    до "исчерпания" делимости
void acc_factors(unsigned long& n,
                 unsigned long d,
                 std::vector<unsigned long>& factors,
                 std::vector<int>& powers) {
  bool first = true;
  while(n % d == 0) {
    if(first) {
      factors.push_back(d);
      powers.push_back(1);
      first = false;
    }
    else {
      ++powers.back();
    }
    n /= d;
  }
}

      // простейший алгоритм факторизации на основе перебора делителей
void factorize(unsigned long n,
               std::vector<unsigned long>& factors,
               std::vector<int>& powers) {
      // сначала проверяем делимость на 2
  acc_factors(n, 2, factors, powers);
      // далее будем перебирать делители от 3 до [sqrt(n)]:
  unsigned long stop = floor(sqrt(n));
      // после 3 пропускаются только очевидные четные делители;
      //    по-хорошему, надо бы иметь таблицу простых делителей:
```

```
  for(unsigned long i = 3; i <= stop; i += 2) {
    acc_factors(n, i, factors, powers);
  }
      // оставшееся число - простое, дописываем его в разложение:
  if(n > 1) {
    factors.push_back(n);
    powers.push_back(1);
  }
}
      // вывод полученного разложения числа n в стандартный поток вывода
void output(unsigned long n,
            std::vector<unsigned long>& factors,
            std::vector<int>& powers) {
  std::cout << n << ": ";
  for(int i = 0; i < factors.size(); ++i) {
    std::cout << factors[i] << "(" << powers[i] << ")";
  }
  std::cout << std::endl;
}

      // вспомогательные статические классы - обертки MPI-типов
class TypeUlong {
  public:
    constexpr static MPI_Datatype datatype = MPI_UNSIGNED_LONG;
    static unsigned long cdatatype;
};

class TypeInt {
  public:
    constexpr static MPI_Datatype datatype = MPI_INT;
    static int cdatatype;
};

      // вспомогательный статический класс отправки и приема сообщений
      //    с упрощенными заголовками операций
      //    (за счет применения универсальных параметров)
template<class T, int tag>
class Msg{
  public:
      // методы для отправки и приема единичных элементов указанного типа
    static void send(const decltype(T::cdatatype)* buffer, int dest) {
      MPI_Send(buffer, 1, T::datatype, dest, tag, MPI_COMM_WORLD);
    }
    static void receive(decltype(T::cdatatype)* buffer, int source) {
      MPI_Recv(buffer, 1, T::datatype, source, tag,
            MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
      // методы для отправки и приема векторов указанного типа
    static void send(const std::vector<decltype(T::cdatatype)>& v, int dest) {
      MPI_Send(v.data(), v.size(), T::datatype, dest, tag, MPI_COMM_WORLD);
    }
    static void receive(std::vector<decltype(T::cdatatype)>& v, int source) {
      MPI_Recv(v.data(), v.size(), T::datatype, source, tag,
            MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
};
```

```
      // главная функция, точка входа в программу
int main(int argc, char **argv) {

      // мастером обычно является процесс с рангом 0
  const int master = 0;

      // теги сообщений для разных ситуаций приема и передачи:
  enum {tagStart, tagLength, tagFactors, tagPowers} tags;

  MPI_Init(&argc, &argv); // инициализация MPI

  int my_rank, num_procs;
      // определение ранга нашего процесса ...
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
      // ... и общего количества процессов в группе
  MPI_Comm_size(MPI_COMM_WORLD, &num_procs);

      // переменные, используемые при распределении и выполнении работы
  unsigned long my_range_start;
  int my_range_length;
  int work_size;
  unsigned long offset;

  if (my_rank == master) {

      // исходные данные задачи
      //    (по-хорошему, их надо передавать через командную строку, TO DO)
    my_range_start = 123456789;
    int range_length = 10;

      // мастер распределяет работу (в том числе, и себе):
    work_size = range_length / num_procs;
    int r = range_length % num_procs;
    my_range_length = r ? (work_size + r) : work_size;

      // мастер пересылает остальным процессам
      //    информацию об участках работы:
    offset = my_range_start + my_range_length;
    for (int i = 1; i < num_procs; ++i) {
      Msg<TypeUlong, tagStart>::send(&offset, i);
      Msg<TypeInt, tagLength>::send(&work_size, i);
      offset += work_size;
    }
  }
  else {
      // рабочие процессы принимают от мастера участки работы:
    Msg<TypeUlong, tagStart>::receive(&my_range_start, master);
    Msg<TypeInt, tagLength>::receive(&my_range_length, master);
  }

      // во всех процессах начинается факторизация

  std::vector<unsigned long> factors;
  std::vector<int> powers;
  int current_size;
```

```
      // все процессы перебирают все числа своего участка:
  for(int i = 0; i < my_range_length; ++i) {
    unsigned long n = my_range_start + i;
    factorize(n, factors, powers);
    if (my_rank == master) {
            // мастер просто выводит свои результаты:
      output(n, factors, powers);
    }
    else {
            // рабочие процессы отправляют свои результаты мастеру:
      current_size = factors.size();
      Msg<TypeInt, tagLength>::send(&current_size, master);
      Msg<TypeUlong, tagFactors>::send(factors, master);
      Msg<TypeInt, tagPowers>::send(powers, master);
    }
      // предыдущие результаты очищаются при переходе к следующему числу:
    factors.clear();
    powers.clear();
  }

  if(my_rank == master) {
      // мастер получает результаты от всех рабочих процессов
      //    по всем элементам диапазона чисел:
    offset = my_range_start + my_range_length;
    for(int i = 1; i < num_procs; ++i) {
      for(int index = 0; index < work_size; ++index) {
            // сначала принимается информация о размере данных
            //    в разложении очередного числа:
        Msg<TypeInt, tagLength>::receive(&current_size, i);
            // векторы адаптируются под необходимый размер:
        factors.resize(current_size);
        powers.resize(current_size);
            // затем векторы заполняются
            //    принимаемыми результатами разложения:
        Msg<TypeUlong, tagFactors>::receive(factors, i);
        Msg<TypeInt, tagPowers>::receive(powers, i);
            // текущие результаты выводятся в стандартный поток:
        output(offset + index, factors, powers);
      }
      offset += work_size;
    }
  }
      // завершаем работу с MPI
  MPI_Finalize();
  return 0;
}
```

Я выполнял компиляцию и сборку программы с помощью следующей командной строки:

```
mpic++ test-mpi.cpp -std=c++11 -o test-mpi
```

Запуск программы с четырьмя процессами (столько ядер на моем ноутбуке, на котором я это тестировал):

```
mpirun -n 4 ./test-mpi
```

В принципе, если потребуется факторизовать миллионы чисел, а количество доступных процессоров будет исчисляться тысячами, то программа останется без изменений (за исключением того, что вывод желательно перенаправить в файл, это можно сделать прямо в командной строке). Такова суть массово-параллельной обработки с помощью MPI.

Помимо рассмотренных двухточечных (point-to-point) коммуникаций в MPI присутствуют так называемые *коллективные* (collective) коммуникации. Их особенностью является то, что в них участвуют все процессы группы. В коллективных коммуникациях можно применять те же коммуникаторы, что и в двухточечных обменах – среда выполнения гарантирует, что сообщения коллективных коммуникаций не пересекаются с сообщениями, которые используются в индивидуальных взаимодействиях процессов. Теги сообщений в коллективных коммуникациях отсутствуют.

Некоторые коллективные операции MPI:

Описание функции	Назначение
`int MPI_Barrier(MPI_Comm comm)` Вызывающий эту функцию процесс *блокируется*. После того, как *все* процессы группы вызвали эту функцию, каждый из них может продолжать работу.	Барьер (рис. 4.3.1) – метод синхронизации параллельных процессов. Он гарантирует, что переход в новую фазу параллельного алгоритма будет выполняться согласованно во всех процессах. Например, после фазы параллельной модификации данных следует ставить барьер, что гарантирует целостность данных на новом этапе работы.
`int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)` Широковещательная рассылка (broadcast). Данные из буфера `buffer` в адресном пространстве процесса с номером `root` переносятся в адресные пространства *всех* процессов по тем адресам `buffer`, которые каждый из процессов указывает в своем вызове этой функции. Количество элементов данных – `count`, типа данных – `datatype`.	Широковещательная рассылка (рис. 4.3.2) применяется для получения в каждом адресном пространстве копии блока данных из одного источника.
`int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)`	Операция *gather* (рис. 4.3.3) применяется для сбора в одном адресном пространстве упорядоченной

Сбор данных (*gather*). Данные из буфера `sendbuf` в адресном пространстве каждого из процессов, включая `root`, пересылаются в адресное пространство процесса с номером `root` и располагаются там, начиная с адреса `recvbuf` в порядке возрастания номеров. Иначе говоря, *n* порций данных выстраиваются в порядке возрастания рангов в единый блок, который записывается в адресное пространство `root`, начиная с адреса `recvbuf`. Для процессов с номером, отличным от `root`, аргумент `recvbuf` игнорируется.	последовательности блоков, происходящих из разных источников.
`int MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)` Разброс данных (*scatter*). Данные из буфера `sendbuf` в адресном пространстве процесса `root` пересылаются всем процессам порциями в порядке возрастания номеров процессов. Порции располагаются в адресных пространствах принимающих процессов, начиная с адреса `recvbuf`. Иначе говоря, блок данных в буфере `sendbuf` делится на *n* порций, и каждая порция поступает по адресу `recvbuf` в адресное пространство процесса, соответствующего этой порции по номеру. Для процессов с номером, отличным от `root`, аргумент `sendbuf` игнорируется.	Операция scatter (рис. 4.3.4) является обратной к *gather* и применяется для разбиения на части упорядоченной последовательности блоков данных и распределения этих частей по разным процессам.

Рис. 4.3.1. `MPI_Barrier`

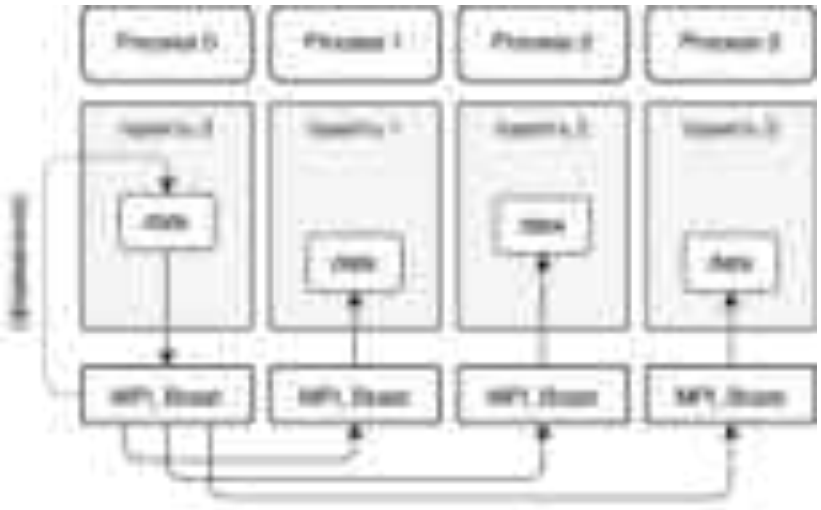


Рис. 4.3.2. `MPI_Bcast`

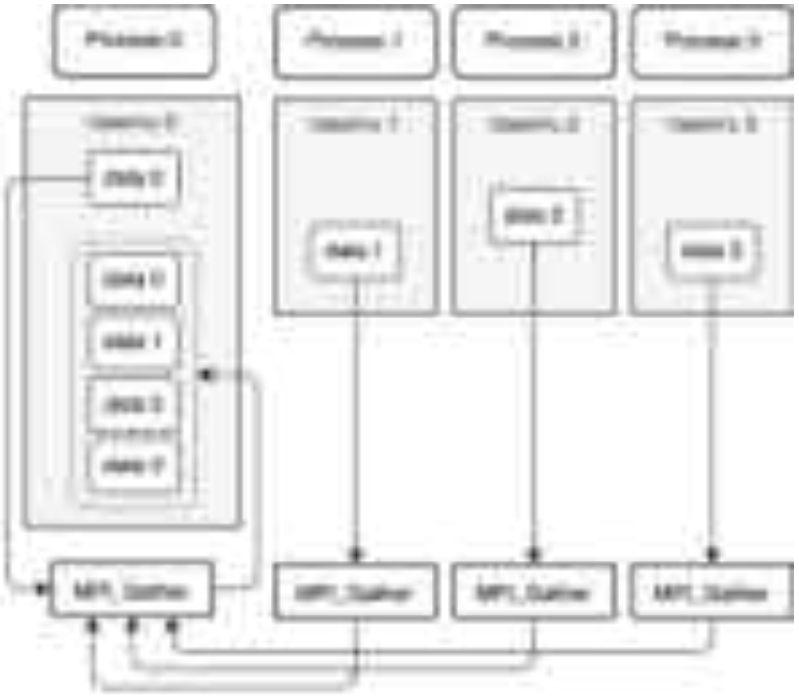


Рис. 4.3.3. `MPI_Gather`

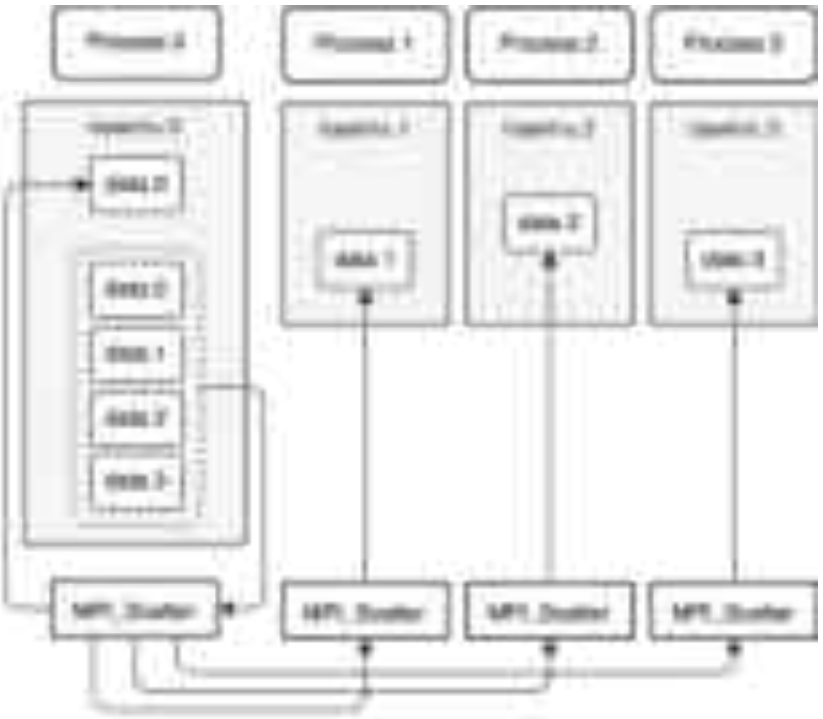


Рис. 4.3.4. `MPI_Scatter`

4.4 Вычислительная парадигма MapReduce, фреймворк Apache Hadoop

Программная модель MPI предоставляет большую гибкость при разработке параллельных программ, выполняющихся на больших кластерах. Но зачастую она может оказаться слишком низкоуровневой. Обработка очень больших объемов данных (терабайты, петабайты и более) демонстрирует проявление “перехода количества в качество”: хотя теоретически можно написать MPI-программу для таких крупномасштабных задач, это будет сродни программированию на ассемблере. Действительно, MPI – это “всего лишь” интерфейс *передачи сообщений*, поэтому все остальные аспекты организации крупномасштабного распределенного алгоритма программисту требуется решать самостоятельно. Появляется необходимость в распределенной системе надежного хранения данных и доступа к ним из распределенных процессов, среда исполнения должна поддерживать разбиение обширных распределенных наборов данных по разным узлам хранения и обработки, планирование работ, перераспределение работ при сбоях узлов, мониторинг состояния процессов.

Корпорация Google предложила в 2004 г. концепцию MapReduce [37], отвечающую этим требованиям. На самом общем уровне ее можно охарактеризовать так: построение *распределенного фреймворка* на основе характерных для параллельного и функционального программирования операций *map* и *reduce*.

Операция *map* принимает на входе пару (*ключ k1*, *значение v1*) и выдает на выходе *список промежуточных* пар (*ключ k2*, *значение v2*). Операция *reduce* принимает на входе *ключ k2* и *список промежуточных значений v2* и выдает на выходе *список выходных значений v3*.

После применения *map* ко всем входным парам производится группировка промежуточных элементов по уникальным ключам, так что в одной очереди (соответствующей ключу *k*)

собираются значения с ключом *k* из *всех промежуточных списков*. Таким образом, очередей получается столько, сколько уникальных ключей *k* встречается в промежуточных списках. Каждая очередь поступает на обработку в операцию *reduce*.

Схема выполнения операций представлена на рис. 4.4.1; здесь задача группировки по ключам (Group by Key) является центральной и очень важной составляющей программной модели, но ее детали скрыты фреймворком от прикладного программиста, что значительно упрощает разработку пользовательских программ. Часто фазу группировки называют Shuffle (англ. перемешивание, перестановка, перетасовка): рабочие узлы перераспределяют данные на основе ключей, созданных функцией *map*, таким образом, что все данные, принадлежащие одному ключу, оказываются на одном рабочем узле.

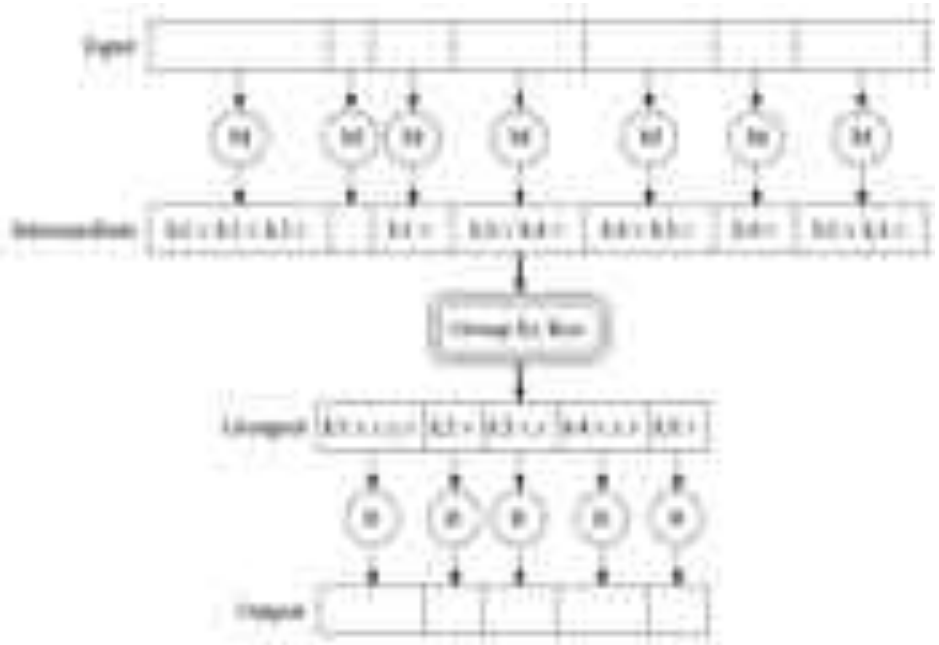


Рис. 4.4.1. Схема выполнения операций map (M) и reduce (R); источник изображения: https://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0007.html

Авторы [37] иллюстрируют применение этих операций на примере подсчета слов в некотором множестве документов:

```
function map(String input_key, String input_value){
    // input_key: имя документа (текстового файла);
    // здесь аргумент input_key не используется,
    // но по нему распределяется обработка в параллельном режиме
    // (каждый input_key является своего рода "идентификатором потока")

    // input_value: содержимое документа (файла)

    List< Pair<String, int> > intermediate
    for each word w in input_value {
      intermediate.push(w, 1)
    }
    return intermediate
}
```

```
function reduce(String intermediate_key, List<int> intermediate_values) {
    // intermediate_key: некоторое слово

    // intermediate_values:
    // список промежуточных значений для ключа intermediate_key

    int result = 0
    for each v in intermediate_values {
      result += v
    }
    List<int> out
    out.push(result)
    return out
}
```

Рассмотрим работу этих функций на следующем наборе документов:

```
{     file1: "раз два три четыре пять вышел зайчик погулять",
      file2: "дважды два четыре",
      file3: "солнечный зайчик"      }
```

В параллельном режиме запускаются 3 вызова функции *map*, по числу ключей (file1, file2, file3).

Поток для `file1` выдает список `[(раз, 1), (два, 1), (три, 1), (четыре, 1), (пять, 1), (вышел, 1), (зайчик, 1), (погулять, 1)]`
Поток для `file2` выдает список `[(дважды, 1), (два, 1), (четыре, 1)]`
Поток для `file3` выдает список `[(солнечный, 1), (зайчик, 1)]`

Промежуточные очереди после группировки и слияния по ключам:

```
раз: 1
два: 1, 1
три: 1
четыре: 1, 1
пять: 1,
вышел: 1
зайчик: 1, 1
дважды: 1
солнечный: 1
```

После параллельного (по числу очередей) применения *reduce* получим выходные списки; для рассматриваемой задачи они состоят только из одного значения, но в общем случае это будет *сокращенный* (reduced) *список* промежуточных значений:

```
раз: [1]
два: [2]
три: [1]
четыре: [2]
пять: [1]
вышел: [1]
зайчик: [2]
дважды: [1]
солнечный: [1]
```

Этот маленький пример демонстрирует работоспособность модели, по крайней мере, на концептуальном уровне. Возможно, он не раскрывает мощность рассматриваемого подхода к организации вычислений. Но если входных файлов миллиарды, триллионы (условно, объемы обрабатываемой информации – терабайты, петабайты), то единственным физически осуществимым способом решения такого рода задач становится распределенная обработка, и в этом случае MapReduce предоставляет очень удобный и прекрасно масштабируемый фреймворк, в котором прикладному программисту требуется задать тело функций *map* и *reduce*, а весь комплекс низкоуровневых деталей берет на себя исполняющая среда.

Эффективность практической реализации программной модели MapReduce будет зависеть как от эффективности разбиения задачи на параллельно выполняющиеся участки, так и от эффективности группировки промежуточных ключей. В основополагающей работе [37] авторы указывают, что главная цель программной модели MapReduce – возможность "справляться за разумное время" с обработкой *очень больших объемов* данных, которые физически невозможно сконцентрировать целиком в оперативной памяти.

Абсолютная скорость выполнения отдельных операций в данном случае не является определяющей характеристикой (и она будет заведомо ниже, чем при создании специализированной программы, адаптированной под конкретную задачу). Компания Google, представившая рассматриваемую модель, стремится строить большие кластеры из так называемых *commodity machines*, то есть из стандартных, массово выпускаемых, широко распространенных и доступных моделей серверной техники общего назначения. Входные и промежуточные данные располагаются в распределенной файловой системе, туда же записываются и результаты (при этом производительность *типовых* устройств дискового ввода-вывода тоже не является рекордной). Невысокая надежность и возможные отказы узлов компенсируются дублированием вычислительных операций при планировании.

Свободная реализация MapReduce (с открытыми исходными текстами) была выполнена в проекте Apache Hadoop [38]. Hadoop в силу своей доступности и открытости принесла популярность технологии MapReduce, а широкое использование Hadoop в различных проектах приносит пользу этой системе, стимулируя разработчиков к ее постоянному совершенствованию (книга [39], написанная одним из разработчиков Hadoop, выдержала 4 издания за период с 2009 по 2015).

Принцип выполнения операций map и reduce в Hadoop сохраняется таким же, как он был представлен в модели Google (рис. 4.4.2).

Приложения Hadoop обычно реализуют интерфейсы Mapper и Reducer, чтобы предоставить алгоритмы выполнения операций *map* и *reduce*. Выходы Mapper-потоков сортируются и разбиваются на части, отправляемые на вход Reducer-потоков. Общее количество участков разбиения по промежуточным ключам (и, следовательно, количество Reducer-потоков) может контролироваться пользовательской реализацией интерфейса Partitioner, но, как правило, в этом не возникает необходимости, поскольку встроенная в Hadoop реализация этого алгоритма работает достаточно эффективно.

Hadoop использует HDFS (Hadoop Distributed File System) – файловую систему, предназначенную для хранения файлов больших размеров, распределенных между узлами вычислительного кластера.

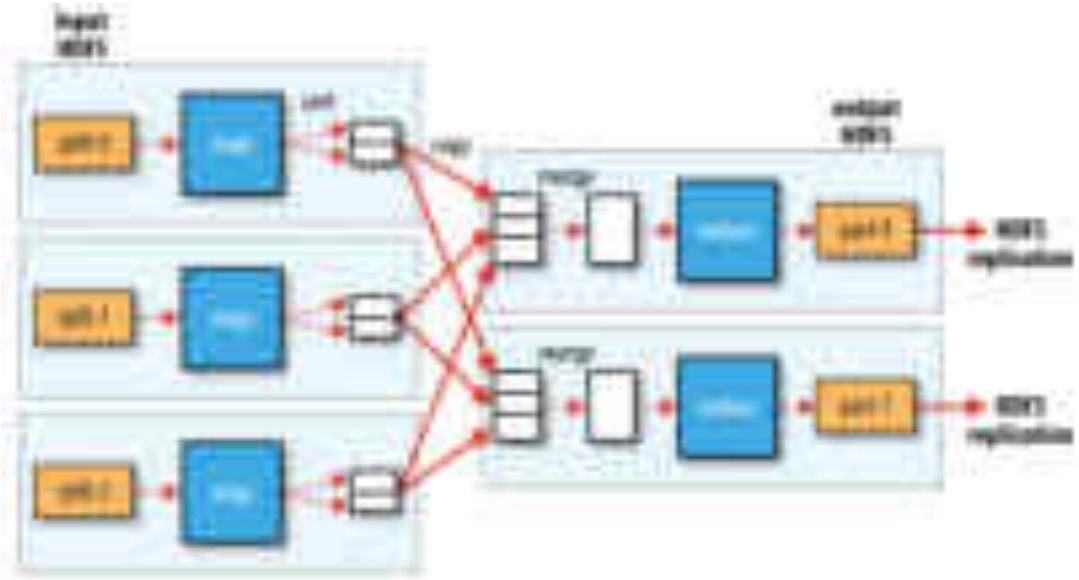


Рис. 4.4.2. Схема выполнения операций map и reduce в Hadoop;
источник изображения: книга [39]

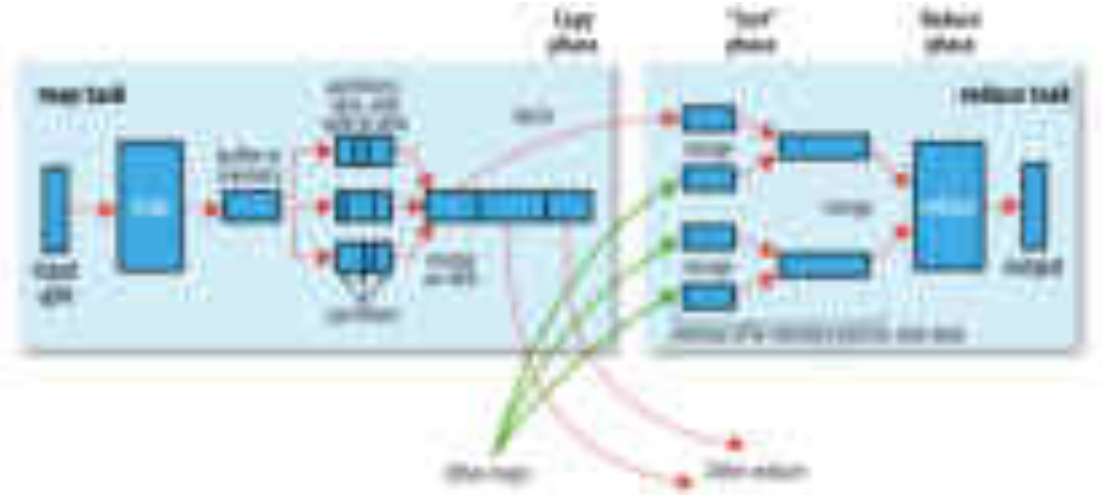


Рис. 4.4.3. Схема выполнения группировки по ключам в Hadoop;
источник изображения: книга [39]

В [39] “раскрываются” многие детали реализации группировки промежуточных значений по ключам (рис. 4.4.3). Оказывается, Hadoop-реализация по соображениям увеличения производительности пишет промежуточный вывод из *map* не в HDFS, а в обычные локальные файлы на дисках того узла, где выполняется *map*. Причем сначала вывод идет в буферы в оперативной памяти, и в них происходит предварительная сортировка; но размеры буферов ограничены, поэтому рано или поздно их содержимое приходится сбрасывать в файлы. Возможно применение компрессии промежуточных массивов данных, так как их участки впоследствии будут пересылаться по сети на узлы, выполняющие операции *reduce*; обычно вывод из *map* обладает высокой информационной избыточностью, так что файлы

хорошо сжимаются даже простыми (быстрыми) алгоритмами, и это позволяет значительно экономить пропускную способность каналов связи ценой относительно небольшой дополнительной загрузки процессоров.

Узлы, выполняющие *reduce*, должны получить *все* записи с заданными значениями ключа, но такие записи не локализованы на каком-то одном узле – в общем случае, они распределены по *всем* узлам, выполнявшим *map*. Поэтому требуется осуществлять передачу данных по сети, и *каждый* Reduce-узел должен потенциально иметь возможность сетевых коммуникаций с *каждым* из Map-узлов.

Reduce-узлы самостоятельно забирают по сети участки промежуточных результатов с необходимыми ключами с разных узлов. Как и когда Reduce-узлы узнают о готовности промежуточных результатов и как они получают “карту” с адресами тех Map-узлов, откуда следует забрать “свои” записи? В схеме распределенной обработки Hadoop присутствует Master-узел. Map-узлы оповещают его об окончании их работы, поэтому Master имеет необходимую полную информацию о распределении ключей по Map-узлам, а Reduce-узлы периодически опрашивают его и узнают, откуда им забирать участки своей работы, повторяя опрос до тех пор, пока полностью не перенесут данные в свою файловую систему, и пока не прекратится поступление новых записей из *map*-фазы. На Reduce-узлах затем проводится *слияние* участков, взятых с разных Map-узлов, в единый файл, который и поступает на вход операции *reduce*.

Почему в Hadoop активная роль “вытягивания” данных принадлежит Reduce-узлам (pull-модель извлечения данных), а не наоборот, Master “проталкивает” результаты (push-модель)? Вероятно, это связано с обеспечением эффективной работы в условиях нестабильности состава узлов (ведь возможны их отказы): если какой-то узел обращается с запросом о “вытягивании” данных, то он, очевидно, в текущий момент находится в работоспособном состоянии. Если же выполнять “проталкивание”, то надо сначала проверить, находится ли принимающая сторона в готовности, на что будут тратиться дополнительное время и полоса пропускания.

Каковы ограничения MapReduce?

Во-первых, это *жестко заданная модель параллельных вычислений*. Синхронизация присутствует только в фазе Shuffle (она выполняет роль барьера между фазами *map* и *reduce*). Какая-либо возможность взаимодействия между параллельными процессами отсутствует. Если какую-то задачу затруднительно свести к фазам *map* и *reduce*, или реализация параллельного алгоритма в этой форме будет очень неэффективной, то для распределенного решения такой задачи следует выбрать другие технологии.

Во-вторых, оборотной стороной упрощения прикладного программирования за счет усложненного фреймворка может оказаться утрата контроля из прикладной программы над тем где, когда и какие данные будут обрабатываться, поэтому исключительно важной становится процедура “тонкой настройки” Hadoop-кластера – а это требует сочетания знаний из области Hadoop-технологии и знаний о структуре и статистических свойствах данных решаемой задачи. Если в каком-то случае низкоуровневые компоненты фреймворка

(декомпозиция на параллельные подзадачи, запуск рабочих процессов, распределение задач по рабочим процессам и балансировка нагрузки, передача данных из файловых систем в оперативную память рабочих процессов, синхронизация и передача данных между рабочими процессами, обработка отказов рабочих процессов, схемы компрессии данных и другие) будут настроены неправильно по отношению к свойствам конкретной задачи, то падение производительности может оказаться драматическим.

Раздел 5. Векторная обработка средствами SIMD и GPU

Лекция 7

5.1 Векторные процессоры и наборы инструкций

Векторная обработка относится к *параллелизму уровня данных* (см. раздел 1.2). Под вектором понимается *упорядоченный набор однотипных данных*, все элементы которого размещены в памяти с одинаковым смещением друг относительно друга (элементы не обязательно должны быть “плотно упакованы”, между ними допустимы промежутки любой длины, лишь бы все эти промежутки имели одинаковую длину).

История векторных компьютеров начинается с моделей CDC STAR-100 (1974 г.), Cray-1 (1975-76 г.). На CDC STAR-100 теоретически можно было выполнять операции с векторами длиной до 65536 элементов, но реальные показатели производительности оказались очень низкими из-за того, что векторные операции работали с аргументами, размещенными в оперативной памяти, а не в регистрах.

В конструкции суперкомпьютера Cray-1 инженер Сеймур Крэй учел неудачный опыт STAR-100 (С. Крэй работал в CDC, но в 1972 г. организовал собственную фирму Cray Research). Cray-1 [40] помимо скалярных регистров содержал *восемь 64-элементных векторных регистров*, длина каждого элемента – 64 бит. Векторные регистры V_0 – V_7 использовались только для выполнения векторных команд. Для работы с векторами разной длины был предусмотрен дополнительный регистр VL, в котором задавалась реальная длина векторов для всех векторных операций. Имелся также 64-битный регистр векторной маски VM, каждый бит которого соответствовал одному из элементов вектора (0-й бит маскирует 0-й элемент, 1-й бит – 1-й элемент, и так далее). Маски позволяли выполнять векторные операции над подмножеством элементов векторного регистра (если бит установлен, операция над соответствующими элементами выполнялась, иначе – не выполнялась).

Архитектура оказалась очень удачной. Уже первая модель Cray-1 показала производительность до 160 MFLOPS (по тем временам очень высокий показатель), причем основной выигрыш по быстродействию был получен именно за счет использования векторных операций. Одна векторная команда декодируется и выполняется быстрее *совокупности* нескольких скалярных, выполняющих *тот же объем* действий.

В программах, где нет подходящего контекста для векторизации, процессоры с векторной архитектурой использовать неэффективно. Первое условие – это наличие векторных данных подходящей длины. Есть и второе, менее очевидное условие: над всеми элементами векторов

должны выполняться одинаковые, независимые действия, для которых имеются соответствующие процессорные инструкции [7].

Если требуется обрабатывать векторы с длиной большей, чем допускают векторные инструкции, то необходимо разбивать входные данные на участки допустимой длины и организовывать в программе итерации по этим участкам. Вообще говоря, накладные расходы на итерации падают с ростом длины обрабатываемых векторов, а правильный учет аспектов кэш-памяти целевой аппаратной платформы позволяет очень быстро “продвигаться” по данным “скользящим окном”, в котором применяются векторные инструкции. Как правило, векторные инструкции накладывают жесткие требования на выравнивание данных. В некоторых случаях затраты на сборку векторов из “неудобных” структур данных и выгрузку векторных результатов обратно в эти структуры будут нивелировать выигрыш от векторных операций.

Какие команды могут (или обязаны) присутствовать в программной модели векторных процессоров? Безусловно необходимы *арифметические* действия (сложение, вычитание, умножение, деление) над целыми и вещественными аргументами различной точности (битовой длины). Желательна также аппаратная поддержка округления, в том числе, до ближайших целых значений (*floor* и *ceil*), определение абсолютных, минимальных и максимальных значений. Помимо этого в алгоритмах могут потребоваться инструкции *загрузки* и *сохранения* (обмен между векторными регистрами и памятью), инструкции поэлементного *сравнения* векторов, *преобразования типов* (например, целое в вещественное, вещественное float в double, и др.), *побитовые* операции (and, or, not, xor), *сдвиги* (влево, вправо арифметический, вправо логический), *перестановки* элементов внутри вектора.

Почти во всех современных микропроцессорах общего назначения имеются векторные расширения. Векторные расширения включают *векторные регистры* и *векторные команды* для работы с векторными регистрами.

Векторизация серьезно влияет на переносимость программ. Векторных расширений очень много, поэтому приходится подготавливать несколько версий кода для разных аппаратных платформ, разных поколений процессоров. Под Linux определить поддержку процессором тех или иных векторных расширений можно с помощью файла /proc/cpuinfo:

Рис. 5.1.1. Пример содержимого файла /proc/cpuinfo

5.2 Вычислительная SIMD-модель на основе наборов инструкций MMX/SSE/AVX

SIMD-расширения MMX (маркетинговая аббревиатура, происходящая, по разным неофициальным сведениям от MultiMedia eXtension или от Matrix Math eXtension) в процессорах Intel (x86) появились в 1997 году. Набор MMX включал *только целочисленные* операции над 64-битными регистрами. В векторных операциях могли участвовать два 32-битных значения, четыре 16-битных или восемь 8-битных значений.

В 1999 г. преемником MMX в процессорах Intel стал набор инструкций SSE (Streaming SIMD Extensions), позже появились расширения SSE2, SSE3, SSE4. В расширениях SSE применяются как целочисленные, так и вещественные операции над элементами типа float и double (начиная с SSE2). Регистры SSE содержат 128 бит.

В 2011 г. в процессорах Intel появилось расширение AVX (Advanced Vector eXtensions), позже дополненное наборами AVX2, AVX-512. Ширина регистров AVX – 256 бит (512 бит в AVX-512).

Процессорных инструкций в MMX/SSE/AVX очень много. При программировании на языке C/C++ удобнее использовать не ассемблерные вставки с мнемониками этих команд, а так называемые *intrinsic-функции*. Эти функции имеют встроенную поддержку со стороны компилятора, так что при генерации кода они раскрываются в соответствующую машинную команду (иногда в несколько команд). При этом программист получает возможность обращаться к аппаратным векторным операциям, но оставляет за компилятором некоторые детали управления вычислительным процессом (выравнивание, размещение операндов в регистрах, управление стеком и состоянием процессора). Официальная документация Intel содержит удобный интерактивный справочник по intrinsic-функциям [41]. Далее мы будем применять термины “команда”, “инструкция” к intrinsic-функциям; это, вообще говоря, неточность, так как ассемблерные мнемоники реальных процессорных инструкций совсем

другие. Тем не менее, подавляющее большинство intrinsic-функций имеют однозначное соответствие с процессорными инструкциями, и путаницы не возникает.

Рассмотрим применение SIMD-операций на примере инструкций из набора AVX/AVX2.

Заголовочный файл с объявлениями intrinsic-функций:

```
#include <immintrin.h>
```

Для компиляции использовался компилятор gcc, командная строка должна включать опцию `-mavx2`, например: `gcc main.c -mavx2 -o main`

Некоторые арифметические операции с вещественными значениями:

```
// исходные векторы:
__m256 v1 = {1.2f, 0.3f, 1.5f, 2.1f, 3.3f, -5.4f, -100.6f, 11.3f};
__m256 v2 = {3.19f, 2.02f, 2.67f, -2.0f, 0.05f, 12.0f, 100.6f, -6.2f};

// вектор результата:
__m256 v3;
```

Здесь тип данных `__m256` – это вектор из восьми 32-битных значений float. Для величин double используется тип `__m256d`, тогда количество элементов в векторе будет 4. К элементам векторов можно обращаться по индексам, например `v1[2]`; индекс не обязан быть константой.

```
v3 = _mm256_add_ps(v1, v2); // поэлементное сложение векторов: v3 = v1 + v2
v3 = _mm256_sub_ps(v1, v2); // поэлементное вычитание векторов: v3 = v1 - v2
v3 = _mm256_mul_ps(v1, v2); // поэлементное умножение векторов: v3 = v1 * v2
v3 = _mm256_div_ps(v1, v2); // поэлементное деление векторов: v3 = v1 / v2
```

Преобразование вещественных векторов `__m256` в целочисленные `__m256i`:

```
// целочисленные векторы:
__m256i vi1, vi2, vi3;

// конвертация путем отбрасывания дробной части, без округления:
vi1 = _mm256_cvttps_epi32(v1);
vi2 = _mm256_cvttps_epi32(v2);

// конвертация путем округления до ближайшего целого с учетом знака:
vi1 = _mm256_cvtps_epi32(v1);
vi2 = _mm256_cvtps_epi32(v2);
```

Преобразование целочисленных векторов с 32-битными элементами в вещественные векторы с элементами типа float осуществляется intrinsic-функцией `_mm256_cvtepi32_ps`.

Некоторые целочисленные арифметические действия:

```
// сложение целочисленных векторов: vi3 = vi1 + vi2
vi3 = _mm256_add_epi32(vi1, vi2);

// вычитание целочисленных векторов: vi3 = vi1 - vi2
vi3 = _mm256_sub_epi32(vi1, vi2);

// умножение 32-битных целочисленных векторов с получением промежуточных 64-битных
элементов, из которых остаются только младшие 32 бит: vi3 = vi1 * vi2
vi3 = _mm256_mullo_epi32(vi1, vi2);

// целочисленного деления, а также вычисления остатка от деления в наборе AVX/AVX2
нет!
```

Здесь следует отметить, что один и тот же тип `__m256i` используется для целочисленных значений разной разрядности (8, 16, 32, 64 бит). Предусмотрены инструкции для целочисленной арифметики с операндами разной разрядности, то есть можно, например, поэлементно сложить векторы из 32 однобайтовых элементов, или из 16 двухбайтовых, и так далее. Но компилятор рассматривает `__m256i` как четыре 64-битных целых числа. Фактически, `__m256i` и другие подобные типы соответствуют векторному регистру, так что брать, например, адрес какого-либо компонента “внутри” `__m256i` нельзя. Набор инструкций содержит команды для загрузки в регистры значений из памяти, и наоборот, выгрузки значений из регистра в память.

```
#define N 8 // размер вектора для 32-битных элементов

// определяем исходные векторы в виде массивов в памяти,
//    с инициализацией и выравниванием на 256-битную границу:
int vi1mem[N] __attribute__ ((aligned (32))) =
      {10, 11, 17, 99, -290, -1000, -2, 8888};
int vi2mem[N] __attribute__ ((aligned (32))) =
      {3, 5, -2, 0, 8, 45, 120, -12345};

// определяем массив для сохранения векторного результата:
int vi3mem[N] __attribute__ ((aligned (32)));

// задаем маску;
// инструкции с использованием масок обрабатывают те элементы вектора,
//    для которых старший бит маски равен 1;
// в нашем случае для краткости записи программы
// битовая маска состоит из всех единиц (-1 = 111...1b),
// так что загружаться и сохраняться будут все элементы
__m256i mask = {-1, -1, -1, -1};

// загружаем векторы из массива в памяти в векторные регистры:
vi1 = _mm256_maskload_epi32(vi1mem, mask);
vi2 = _mm256_maskload_epi32(vi2mem, mask);

// выполняем операцию с векторными регистрами:
vi3 = _mm256_add_epi32(vi1, vi2);

// переносим результат из векторного регистра в массив в памяти:
_mm256_maskstore_epi32(vi3mem, mask, vi3);
```

```
// работаем с результатом, перенесенным в память (выводим в поток вывода):
int i;
for(i = 0; i < N; ++i) {
  printf("%d ", vi3mem[i]);
}
```

Аналогично для векторов из однобайтовых элементов:

```
#define M 32 // размер вектора для 8-битных элементов

unsigned char vi1mem8[M] __attribute__ ((aligned (32)));
unsigned char vi2mem8[M] __attribute__ ((aligned (32)));
unsigned char vi3mem8[M] __attribute__ ((aligned (32)));

// задаем для элементов векторов какие-нибудь значения:
for(i = 0; i < M; ++i) {
  vi1mem8[i] = i;
  vi2mem8[i] = 2 * i;
}

// загружаем 32-битные целые числа (команд для загрузки байтов нет)
vi1 = _mm256_maskload_epi32((int*)vi1mem8, mask);
vi2 = _mm256_maskload_epi32((int*)vi2mem8, mask);

// складываем векторы из 32 однобайтовых значений:
vi3 = _mm256_add_epi8(vi1, vi2);

// выгружаем 32-битные целые числа (команд для выгрузки байтов нет)
_mm256_maskstore_epi32((int*)vi3mem8, mask, vi3);

// выводим однобайтовые значения (преобразованные к целым)
for(i = 0; i < M; ++i) {
  printf("%d ", (int)vi3mem8[i]);
}
```

Редукция (на примере суммирования) выполняется с помощью команд группы *hadd* (horizontally add). Схема выполнения операции `_mm256_hadd_epi32` горизонтального сложения 32-битных целых чисел представлена на рис. 5.2.1.

```
#define N 8 // размер вектора для 32-битных элементов

// массив из 2N чисел для суммирования
int imemA[2 * N] __attribute__ ((aligned (32))) = {1, 2, 3, 4, 5, 6, 7, 8, 10, 20,
30, 40, 50, 60, 70, 80};

// загружаем содержимое массива в два регистра
__m256i vi1, vi2, vi3;
vi1 = _mm256_maskload_epi32(imemA, mask);
vi2 = _mm256_maskload_epi32(imemA + N, mask);
```

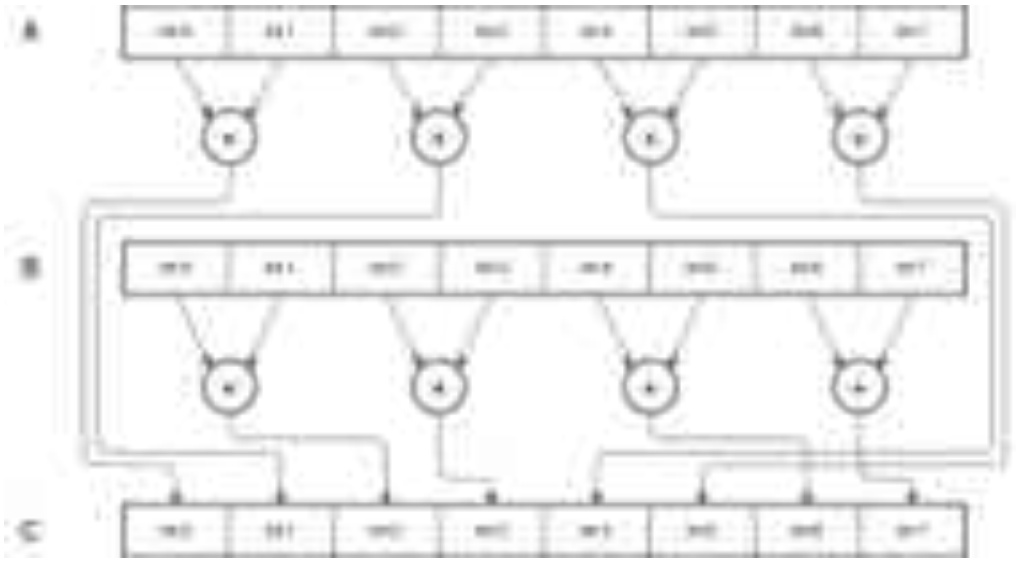


Рис. 5.2.1. Схема выполнения операции _mm256_hadd_epi32

```
// складываем смежные пары элементов согласно схеме hadd
vi3 = _mm256_hadd_epi32(vi1, vi2);
// содержимое v3: {3, 7, 30, 70, 11, 15, 110, 150}

// продолжаем с одним и тем же регистром:
vi3 = _mm256_hadd_epi32(vi3, vi3);
// содержимое v3: {10, 100, 10, 100, 26, 260, 26, 260}
vi3 = _mm256_hadd_epi32(vi3, vi3);
// содержимое v3: {110, 110, 110, 110, 286, 286, 286, 286}

// чтобы продолжить сложение по схеме hadd,
// необходимо обменять элементы с индексами 1 и 4;
// для этого задаем желаемый порядок расположения элементов:
//    0, 4, 2, 3, 1, 5, 6, 7
// (обратите внимание, что задаются 64-битные константы,
// в которых старшие 32-битные слова соответствуют нечетным индексам)
__m256i permIndex = {0x0000000400000000, 0x0000000300000002,
                     0x0000000500000001, 0x0000000700000006};
vi3 = _mm256_permutevar8x32_epi32(vi3, permIndex);
// содержимое v3: {110, 286, 110, 110, 110, 286, 286, 286}

// последний шаг сложения:
vi3 = _mm256_hadd_epi32(vi3, vi3);
// содержимое v3: {396, 220, 396, 220, 396, 572, 396, 572}

// извлекаем только 0-й элемент, для этого используем маску с установленным 31-м
битом
// (элементы маски – 64-битные):
__m256i maskLow = {0x80000000, 0, 0, 0};
int s;
_mm256_maskstore_epi32(&s, maskLow, vi3);
// в переменной s находится сумма 16 элементов массива imemA
```

Из справочных сведений [41] можно определить общее количество тактов, затраченное на вычисление суммы: *load*: [0.5 – 2] * 2, *hadd*: [1 – 2] * 4, *permute*: 1, *store*: [1 – 2]. Итого от 7 до

15 тактов, в зависимости от модели процессора. Теоретически для нахождения суммы 16 чисел требуется 15 сложений, но это без учета загрузки данных в регистры и выгрузки результатов в память. К тому же в рассмотренном примере из-за отсутствия дополнительных данных для суммирования приходилось под конец “впустую” складывать вектор с самим собой.

В любом случае, для получения выигрыша от векторных операций необходимо пропускать через векторный конвейер как можно больше данных. Далее приведен пример функции, выполняющей редукцию суммированием массива, содержащего произвольное количество 32-битных целочисленных элементов (это количество должно быть кратно 8).

```
int reduce(const int* data, const int size) {
  __m256i mask = {-1, -1, -1, -1};
  __m256i permIndex = {0x0000000400000000, 0x0000000300000002,
                       0x0000000500000001, 0x0000000700000006};
  __m256i maskLow = {0x80000000, 0, 0, 0};

  __m256i vi1, vi2;

  const int* p = data;
  const int* stop = data + size;

  vi1 = _mm256_maskload_epi32(p, mask);
  p += N;

  // редуцируем циклически, пока в операцию не будут вовлечены все элементы
  // исходного массива
  while(p < stop) {
    vi2 = _mm256_maskload_epi32(p, mask);
    p += N;
    vi1 = _mm256_hadd_epi32(vi1, vi2);
  }

  // после цикла суммируем регистр с самим собой :

  vi1 = _mm256_hadd_epi32(vi1, vi1);
  vi1 = _mm256_hadd_epi32(vi1, vi1);
  vi1 = _mm256_permutevar8x32_epi32(vi1, permIndex);
  vi1 = _mm256_hadd_epi32(vi1, vi1);

  // сохраняем в памяти 0-й элемент вектора:
  int s;
  _mm256_maskstore_epi32(&s, maskLow, vi1);

  return s;
}
```

Количество тактов при векторной обработке в сравнении с теоретическим количеством сложений:

количество чисел в	минимальное	максимальное	теоретическое

массиве	количество тактов на векторную редукцию AVX	количество тактов на векторную редукцию AVX	количество сложений
8	5.5	11	7
16	7	15	15
24	8.5	19	23
32	10	23	31
40	11.5	27	39
48	13	31	47
56	14.5	35	55
64	16	39	63
…			
800	154	407	799
1600	304	807	1599
8000	1504	4007	7999

Видно, что даже в худшем случае (на относительно старых процессорах) векторная редукция с использованием 256-битных инструкций из набора AVX требует в 2 раза меньше тактов, чем идеализированное последовательное скалярное сложение. Но для получения серьезного преимущества векторы должны быть достаточно длинными (тысячи и десятки тысяч элементов).

Вычисление определителя матрицы 3 x 3:

A[9] =

$0:a_{00}$	$1:a_{01}$	$2:a_{02}$
$3:a_{10}$	$4:a_{11}$	$5:a_{12}$
$6:a_{20}$	$7:a_{21}$	$8:a_{22}$

$$\det A = a_{00}a_{11}a_{22} + a_{01}a_{12}a_{20} + a_{10}a_{21}a_{02} - \\ - a_{02}a_{11}a_{20} - a_{01}a_{10}a_{22} - a_{12}a_{21}a_{00}$$

Векторные инструкции AVX выполняют одновременно 8 умножений, но в этой формуле возможны только 6 независимых произведений, причем, вычисление произведений необходимо выполнять последовательно 2 раза (здесь по умножению требуется двухшаговая редукция: второе умножение использует результат первого). Векторы придется дополнить фиктивными нулевыми 6-м и 7-м элементами (считая их с 0-го индекса). Редукцию по сложению можно выполнить по схеме из предыдущего примера, предварительно умножив элементы вектора, содержащего произведения, на константы +1 и -1. Но тогда получается уже редукция по умножению из трех шагов: $\det A = (+1)a_{00}a_{11}a_{22} + (+1)a_{01}a_{12}a_{20} + (+1)a_{10}a_{21}a_{02} +$

$(-1)a_{02}a_{11}a_{20} + (-1)a_{01}a_{10}a_{22} + (-1)a_{12}a_{21}a_{00}$. Значит можно умножить *сначала* два регистра, *затем* еще два регистра, *затем* результаты этих шагов перемножить между собой (векторизация – это параллельность по данным, так что у нас *нет параллельности по инструкциям*, несмотря на то, что она здесь возможна). После этого все будет готово к редукции по сложению восьми элементов (включая 2 фиктивных нулевых); алгоритм такой редукции уже написан выше.

Задать необходимый порядок выборки элементов вектора из массива в памяти можно с помощью операции *gather* (`_mm256_i32gather_epi32` для выборки восьми 32-битных целочисленных элементов по восьми 32-битным индексам).

```
int det3x3avx(const int* matr) {
      // маска для сохранения в памяти 0-го элемента вектора
  __m256i maskLow = {0x80000000, 0, 0, 0};

      // знаки множителей (+1, +1, +1, -1, -1, -1, 0, 0)
  __m256i signes = {0x0000000100000001, 0xFFFFFFFF00000001, -1, 0};

      // индексы для задания порядка перемножения
      // (см. формулу детерминанта)

  __m256i vindex1 = {0x0000000100000000,0x0000000200000003,
                         0x0000000500000001, 0}; // 0, 1, 3, 2, 1, 5

  __m256i vindex2 = {0x0000000500000004,0x0000000400000007,
                         0x0000000700000003, 0}; // 4, 5, 7, 4, 3, 7

  __m256i vindex3 = {0x0000000600000008,0x0000000600000002,
                         0x0000000000000008, 0}; // 8, 6, 2, 6, 8, 0

      // индексы для перестановки элементов векторного регистра
      // при выполнении редукции по сложению
  __m256i permIndex = {0x0000000400000000, 0x0000000300000002,
                         0x0000000500000001, 0x0000000700000006};

      // собираем "слои" множителей;
      // последний аргумент – масштабный множитель
      // (фактически, это размер в байтах 32-битного элемента)
  __m256i layer1 = _mm256_i32gather_epi32(matr, vindex1, 4);
  __m256i layer2 = _mm256_i32gather_epi32(matr, vindex2, 4);
  __m256i layer3 = _mm256_i32gather_epi32(matr, vindex3, 4);

      // редукция по умножению (включая установку знаков множителей)
  __m256i p1 = _mm256_mullo_epi32(signes, layer1);
  __m256i p2 = _mm256_mullo_epi32(layer2, layer3);
  p1 = _mm256_mullo_epi32(p1, p2);

      // редукция по сложению
  p1 = _mm256_hadd_epi32(p1, p1);
  p1 = _mm256_hadd_epi32(p1, p1);
  p1 = _mm256_permutevar8x32_epi32(p1, permIndex);
  p1 = _mm256_hadd_epi32(p1, p1);

      // сохраняем результат из 0-го элемента:
```

```
  int d;
  _mm256_maskstore_epi32(&d, maskLow, p1);

  return d;
}
```

Количество тактов, за которые выполняются операции *gather*, достаточно велико (от 6 до 10 тактов). Умножение и сложение выполняются за 1–2 такта, 1 такт требуется на перестановку, 1–2 такта – на выгрузку результата. Без учета *gather* векторному алгоритму вычисления детерминанта 3x3 необходимо до 15 тактов, а идеализированная скалярная последовательность действий требует 17 арифметических операций (тоже без учета загрузки данных в регистры). Таким образом, практическая выгода от применения векторных инструкций в данном примере сомнительна; однако, он может служить иллюстрацией преобразования некоторых формул к виду, удобному для векторных вычислений.

5.3 Оптимизирующие компиляторы с автоматической генерацией SIMD-инструкций

Ручное кодирование векторных операций (с помощью ассемблерных команд или intrinsic-функций) может потребовать от разработчиков ПО значительных усилий и высокой квалификации. В ходе этого процесса неизбежно будут возникать ошибки; кроме того, требуется умение распознавать возможность или невозможность применения векторных операций для тех или иных участков кода. Низкоуровневое программирование требует от разработчика особых навыков мышления в контексте отдельных машинных инструкций и их комбинаций. Повсеместный переход от ассемблера к языкам высокого уровня имеет очень веские причины, среди которых переносимость ПО, возможность иметь одинаковую или почти одинаковую кодовую базу для разных целевых аппаратных платформ. Следовательно, весьма желательно, чтобы векторизация кода *автоматически* производилась компилятором. Автоматическая векторизация является разновидностью *оптимизации* машинного кода, генерируемого компилятором.

Рассмотрим некоторые опции оптимизации компилятора gcc [42], которые влияют на автоматическую векторизацию. Совокупность действий по векторизации можно включить на уровне оптимизации `-O3`. К векторизации относятся следующие флаги:

```
-ftree-loop-vectorize
-ftree-slp-vectorize
-ftree-vectorize
-fsimd-cost-model=[unlimited|dynamic|cheap]
-fvect-cost-model=[unlimited|dynamic|cheap]
```

Для целевого кода платформ x86 и x86-64 необходимо явно разрешать компилятору использование векторных инструкций (типа `-msse -msse2` … `-mavx -mavx2` и других). Для автовекторизации редукции вещественных значений необходимо использовать `-ffast-math` или `-fassociative-math`. Подробнее об опциях автовекторизации в gcc можно узнать из [43].

Так может выглядеть командная строка для компиляции с автоматическим применением векторизации:
`gcc main.c -o main -O3 -mavx2 -ftree-vectorize`

Для сравнения, командная строка с оптимизацией, но без автовекторизации:
`gcc main.c -o main -O2`

Фрагмент кода, суммирующий 1 миллион 32-разрядных случайных целых чисел с оптимизацией (`-O2`), но без автовекторизации, выполнялся на моей тестовой системе порядка 700 – 900 микросекунд. Тот же фрагмент с автовекторизацией на инструкциях AVX2 (`-O3 -mavx2 -ftree-vectorize`) выполнялся около 400 – 500 микросекунд. Разумеется, этот эксперимент не является точным, но он дает представление о работоспособности автоматической векторизации. Код вообще без оптимизации работал порядка 3000 – 5000 микросекунд.

Код тестового фрагмента:

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

#define N 1000000

   // функция редукции (поиск суммы элементов массива),
   // написанная без каких-либо векторных операций
int reduce(const int* a, int size) {
  int s = 0;
  int i;
  for(i = 0; i < size; ++i) {
    s += a[i];
  }
  return s;
}

int main(void) {
  int a[N];

  // заполняем массив как-нибудь . . .
  int i;
  for(i = 0; i < N; i += 2) {
    a[i] = rand();
  }

  // запускаем редукцию и измеряем время в наносекундах
  int s;
  struct timespec t1;
  struct timespec t2;

  clock_gettime(CLOCK_REALTIME, &t1);
  s = reduce(a, N);
  clock_gettime(CLOCK_REALTIME, &t2);
```

```
  printf("s = %d, %ld ns\n", s, t2.tv_nsec - t1.tv_nsec);
  return 0;
}
```

С помощью команды
`gcc main.c -o main -S -O3 -mavx2 -ftree-vectorize`
в текстовом файле `'main'` можно получить ассемблерный листинг результата компиляции.

Ниже приведен фрагмент ассемблерного листинга компиляции с опциями автовекторизации. Векторные инструкции выделены жирным. Комментарии о соответствующих intrinsic-функциях – мои.

Автоматический ассемблерный листинг читать затруднительно, но принцип использования векторных инструкций здесь более-менее ясен. Компилятор не применял инструкции категории *hadd* (“горизонтальное сложение”, см. рис. 5.2.1), вместо этого выполняется обычное поэлементное сложение 8-векторов – последовательных блоков исходного массива (причем, один из операндов сложения поступает прямо из памяти, по “скользящему” указателю), так что аккумулируются 8 частичных сумм, а уже после прохода по всему массиву осуществляется суммирование частичных сумм (собственно, *редукция* выполняется только на последнем этапе).

```
L25:
      ; на этом участке происходит последовательная выборка 32-байтовых блоков
      ; из памяти и сложение их с накоплением в регистре ymm0
      ; таким образом, по окончании цикла
      ; регистр ymm накопит 8 частичных сумм

      ; intrinsic: _mm256_add_epi32
vpaddd  0(%r13), %ymm0, %ymm0
addq    $32, %r13
cmpq    %r13, %r14
jne .L25

      ; дальше идет сложение друг с другом
      ; накопленных в ymm0 восьми частичных сумм
      ; и выгрузка 32-битного значения итоговой суммы

      ; intrinsic: _mm256_xor_si256
vpxor   %xmm1, %xmm1, %xmm1
leaq    -4000096(%rbp), %rsi
xorl    %edi, %edi
      ; intrinsic: _mm256_permute2x128_si256
vperm2i128  $33, %ymm1, %ymm0, %ymm2
      ; intrinsic: _mm256_add_epi32
vpaddd  %ymm2, %ymm0, %ymm0
      ; intrinsic: _mm256_permute2x128_si256
vperm2i128  $33, %ymm1, %ymm0, %ymm2
      ; intrinsic: _mm256_alignr_epi8
vpalignr    $8, %ymm0, %ymm2, %ymm2
      ; intrinsic: _mm256_add_epi32
vpaddd  %ymm2, %ymm0, %ymm0
      ; intrinsic: _mm256_permute2x128_si256
```

```
vperm2i128  $33, %ymm1, %ymm0, %ymm1
      ; intrinsic: _mm256_store_si256
vmovdqa %ymm0, -4000176(%rbp)
      ; intrinsic: _mm256_alignr_epi8
```

Компилятор clang [44] также осуществляет автовекторизацию циклов. В официальной документации нет прямого указания, на каком уровне оптимизации присутствует автовекторизация. По косвенным (и достаточно старым) сведениям [45] автовекторизация включается с уровня `-O2`. Флаги, относящиеся к векторизации: `-slp-vectorizer`, `-vectorize-loops`, `-vectorize-slp`. Для компиляции с набором инструкций AVX/AVX2 необходимо задать целевой процессор `-march=haswell`. Для векторизации редукции вещественных значений требуется флаг `-ffast-math`.

Таким образом, командная строка компиляции с применением автовекторизации на основе инструкций AVX/AVX2 для приведенного выше примера программы будет следующей:
`clang main.c -o main -O2 -march=haswell`

На самом деле, clang не является самодостаточным компилятором, а только так называемым C-frontend, то есть *интерфейсом компилятора* языка C. Clang не генерирует машинный код самостоятельно, вместо этого он формирует промежуточный код для кроссплатформенной виртуальной машины LLVM (Low Level Virtual Machine). Полученный промежуточный код затем транслируется в машинный код целевой архитектуры. Поэтому собственно оптимизация – это задача транслятора LLVM.

Программная модель LLVM изначально поддерживает векторные операции. Полное справочное описание автоматической векторизации при трансляции LLVM-кода приведено в [46]. В частности, там указано, как задавать ширину вектора для автоматической векторизации (так, флаги `-mllvm -force-vector-width=8` принудительно задают 8-элементные векторы).

Clang поддерживает специальные указания компилятору относительно ширины векторов в том или ином цикле; это уже не совсем *автоматическая* векторизация, но такие подсказки со стороны программиста помогают, в конечном итоге, получить от компилятора более эффективный код. Пример указаний по векторизации:

```
#pragma clang loop vectorize_width(2) interleave_count(2)
for(...) {
  ...
}
```

Одними из самых эффективных компиляторов для платформы x86 (x86-64) являются Intel C++ Compiler Classic (ICC) [47] и Intel oneAPI DPC++/C++ Compiler [48]. Подробное руководство по использованию средств автоматической векторизации для компилятора ICC представлено в [49]. Автовекторизация доступна на уровне оптимизации `-O2` и выше. Предусмотрены указания компилятору непосредственно в коде программы, относящиеся к тем или иным циклам, например `#pragma vector always` – векторизовать цикл, `#pragma novector` – не векторизовать цикл. Целевая аппаратная архитектура и

соответствующие наборы векторных инструкций в командной строке компилятора ICC (Linux) задаются опцией `-march=<target set>`, например `-march=core-avx2`, `-march=haswell` и т.п.

Лекция 8

5.5 Аппаратура и программные интерфейсы для организации вычислений общего назначения на основе GPU

Аппаратура графических процессоров (GPU) изначально проектировалась для выполнения параллельных вычислений, так как в полигональной трехмерной графике сцены формируются из множества полигонов (обычно, треугольников), которые подвергаются геометрическим преобразованиям, освещаются, проецируются, обрабатываются по каким-либо алгоритмам независимо (параллельно). После растеризации полигонов выполняется обработка полученных пикселей, также в параллельном режиме.

Приблизительно в начале 2000-х годов энтузиасты выдвинули идею организации *неграфических* вычислений на GPU. Несмотря на высокую вычислительную мощность видеокарт, использовать их процессоры для вычислений *общего назначения* (General Purpose GPU, GPGPU) в те годы было затруднительно, так как единственными API для доступа к функциям GPU были графические API Direct3D и OpenGL.

Достаточно удобно можно было проводить только операции из категории *map* (преобразование входного массива значений в выходной массив такого же размера), для этого применялись пиксельные шейдеры (то есть программы, предназначенные для вычисления цвета отдельных пикселей). Выполнение операций из категории *reduce* (преобразование входного массива значений в выходной массив меньшего размера, в пределе в одно или несколько чисел) требовало многопроходного рендеринга с последовательной установкой все меньших и меньших по размеру страниц вывода, а также чтением на следующем проходе выходных буферов предыдущего прохода (так называемая схема *ping-pong*).

В 2006 году компания NVIDIA, один из главных производителей видеокарт, выпустила фреймворк CUDA (Compute Unified Device Architecture), предназначенный именно для организации вычислений общего назначения на GPU. Программы CUDA разрабатываются на специальном диалекте языка C/C++. CUDA предоставляет библиотеки функций, содержащие реализации полезных алгоритмов (линейная алгебра, быстрое преобразование Фурье, обработка графов и других), качественно оптимизированные и тонко настроенные под аппаратную архитектуру GPU NVIDIA.

В ответ на появление фреймворка CUDA, предназначенного исключительно для GPU NVIDIA, в 2008-2009 гг. появился открытый кроссплатформенный стандарт OpenCL (Open Computing Language), на основе которого можно создавать реализации драйверов и среды исполнения для GPU любых производителей.

Таким образом, приблизительно к рубежу 2010-х годов прикладное ПО получило возможность эффективно использовать ресурсы GPU для организации массово-параллельной обработки данных. Однако прогресс производительности оказался неравномерным: какие-то приложения имели программную архитектуру и алгоритмы, хорошо подходящие для ускорения на GPU, а другие программы в силу разных причин не смогли (или даже не пытались) получить от GPU значительную выгоду.

Несмотря на то, что CUDA и OpenCL ставят целью *упростить* программирование GPU и взаимодействие между GPU и CPU при решении неграфических задач, различия в аппаратной архитектуре между типичными GPU и многоядерными CPU слишком велики для организации по-настоящему эффективных вычислений на основе только опыта последовательного программирования, или только опыта многопоточного SMP-программирования, без скрупулезного учета аппаратной специфики того и другого класса устройств. Говорить об *автоматической* трансляции последовательных программ с целью адаптации их для выполнения на GPU (по аналогии с достаточно успешно осуществляемой автовекторизацией на основе SIMD-инструкций CPU) на данный момент (конец 2021 г.) пока не приходится.

В любом случае, большинству прикладных программ требуется глубокая переработка кодовой базы для получения того скачка производительности, который *теоретически* может обеспечиваться графическими процессорами; так по состоянию на 2021 г. пиковая производительность CPU Xeon до 1.7 TFLOPS [50], пиковая производительность игровых GPU до 36 TFLOPS [51], профессиональных multi-GPU систем – до 5 petaFlops на специальных задачах из категории machine learning [52]. Понятно, что эти цифры недостижимы при практических вычислениях, но они демонстрируют масштаб различий в производительности.

Массово-параллельная обработка данных на GPU (рис. 5.5.1) относится к так называемой *гетерогенной* (неоднородной) модели вычислений [29]. Неоднородность связана с тем, что в алгоритмах в явном виде выделяются части, выполняемые на многоядерном центральном процессоре (SMP-системе с поддержкой векторных инструкций в каждом потоке), и части, выполняемые на графическом процессоре (массово-параллельном ускорителе). Эти части неравноправны: основная (управляющая) система на основе центрального процессора в терминологии гетерогенных вычислений называется *host*, а дополнительные массово-параллельные ускорители обозначаются как *device(s)*, их может быть несколько, различного типа. Соответственно, RAM, доступная центральному процессору, обозначается как *host memory*, а память ускорителя – *device memory*.

Код для CPU и для GPU может выполняться *одновременно*, так что степень параллелизма в гетерогенных системах теоретически увеличивается как сумма возможностей одного и другого типа процессоров. Физическая память у центрального и графического процессоров разная, хотя есть ускорители (как правило, с невысокой производительностью), которые не имеют собственной физической памяти и обращаются к данным, расположенным в общей оперативной памяти системы. В то время, когда и CPU и GPU загружены работой, подсистема ввода-вывода с поддержкой технологии DMA (Direct Memory Access) может осуществлять асинхронные операции ввода-вывода, автоматически заполняя массивы

входных данных путем автоматического переноса данных из внешних устройств (диск, сеть) в оперативную память, а также отправляя на внешние устройства результаты предыдущих операций.

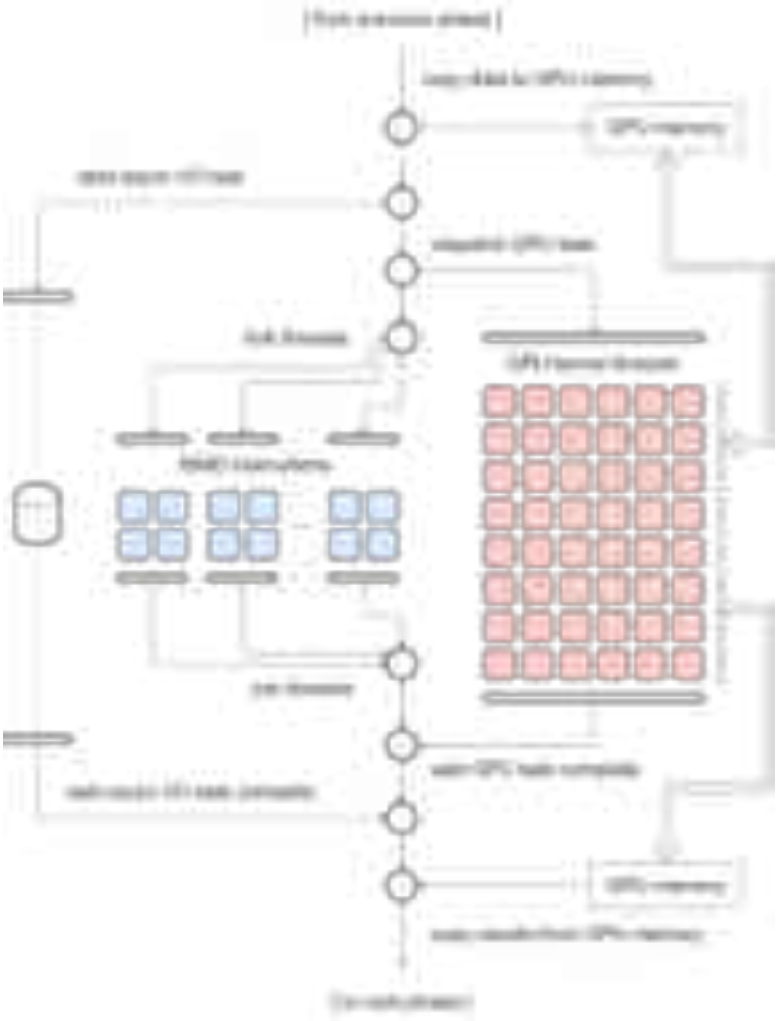


Рис. 5.5.1. Схема гетерогенной массово-параллельной обработки данных

Основным фактором, которым обусловлено стремление задействовать в программах вычислительные ресурсы GPU, является очень высокая пропускная способность обработки данных, достижимая *при определенных условиях*. Эти условия можно описать так называемой парадигмой *потоковой обработки* (stream processing). Потоковая обработка предполагает наличие достаточно большого массива (потока) данных, над каждым элементом которого выполняется некоторая совокупность операций. Операции описываются так называемыми *ядрами* (kernel function или compute kernel) – вычислительными процедурами, вызываемыми *параллельно и независимо* для всех элементов потока данных.

В сущности, такая программная модель была изначально характерна для графических процедур трехмерной полигональной графики: источником потока является множество атрибутов вершин треугольников, на которые разбита сцена, далее эти атрибуты (позиция, базовый цвет материала, текстурные координаты и др.) поступают на блок обработки вершин (позиционирование в мировом пространстве, настройка текстурирования и др.), далее производится проецирование треугольников на плоскость экрана, отсечение, растеризация, затем обработка полученного после растеризации множества фрагментов (пикселей) и

выдача окончательного цвета точки изображения. Вычислительные ядра не взаимодействуют друг с другом во время исполнения, поэтому, скажем, во время вычисления цвета отдельного пикселя нет возможности “узнать”, как происходит обработка соседних пикселей. Специальная управляющая программно-аппаратная среда вызывает вычислительные ядра в параллельных потоках в порядке, зависящем от фактической ситуации готовности операндов и доступности ALU.

В компьютерной графике применяются операции над векторами небольшого размера (2, 3, 4 элемента). Например, две координаты (*x*, *y*) пикселей в растровом изображении, три составляющих (*nx*, *ny*, *nz*) вектора нормали, три элемента цвета (*r*, *g*, *b*), четырехэлементные позиции точек в трехмерном пространстве, записанные в однородных координатах (*x*, *y*, *z*, *w*), четыре элемента при кодировании цветных пикселей с прозрачностью (*r*, *g*, *b*, *a*) и др. Необходимы также матричные вычисления с матрицами небольшого размера, обычно 4x4. Высокая точность в компьютерной графике требуется редко; например, цвет, выводимый на мониторы, обычно имеет 8 бит на каждый цветовой канал, так что весь пиксель, включая прозрачность, можно кодировать четырехбайтовым значением, а вычисления с вещественными числами при геометрических преобразованиях вершин трехмерных объектов и при проецировании сцен на экран вполне могут обходиться 16-битной арифметикой.

Таким образом, GPU как *графический ускоритель* изначально не нуждался в векторах длиной больше 4. Специфика, присущая графическим вычислениям – широкое использование операций с вещественными значениями, представленными в формате с плавающей точкой *одинарной* точности (тип single, или float, 32 бит) и *половинной* точности (half, 16 бит) – до сих пор учитывается при оптимизации GPGPU-процедур. Под 32-битные вещественные операнды адаптирована аппаратная структура ALU в GPU. При переходе к вычислениям общего назначения на GPU необходимо учитывать, что вычисления с вещественными операндами *двойной* точности (тип double, 64 бит), а также с целыми числами (тип int, 32 бит) могут выполняться значительно медленнее. С эволюцией аппаратуры разница в скорости обработки данных разных типов (int, float, double) постепенно снижается.

Параллельное программирование всегда требует учета особенностей вычислительной аппаратуры. При этом эволюция архитектуры GPU идет заметно быстрее, чем эволюция архитектуры CPU. Приемы и практики программирования с целью получения оптимальной производительности, применимые для одного поколения GPU, могут оказаться контрпродуктивными для другого поколения. Неизменным остается принцип маскирования задержек памяти за счет максимизации числа потоков: чем больше потоков поступает на выполнение, тем выше вероятность того, что в любой момент найдется поток, для которого все операнды уже поступили из памяти в регистры, и он готов к выполнению вычислительных инструкций. В связи с этим в GPU применяется вычислительная модель SIMT. Потоки запускаются группами; между потоками в составе одной группы можно обеспечить эффективную синхронизацию и взаимодействие (барьеры, быстрый обмен данными).

GPU не подходит для решения таких задач, в которых требуется *быстро* обработать *мало данных*; с этой целью рекомендуется использовать векторные инструкции CPU. В некоторых материалах по программированию GPU приводится такая аналогия: GPU – это многотонный

грузовик, который не слишком быстро едет, но в итоге за определенный промежуток времени перевозит гораздо больший груз; напротив, CPU – это спортивный легковой автомобиль, который имеет малую грузоподъемность, но является динамичным, маневренным, малоинерционным. Суть аналогии в том, что CPU лучше подходит для выполнения процедур управляющего характера (сложная логика выбора траекторий исполнения кода, запуск и остановка процедур, реакция на внешние события) с низкими задержками и высокой вариативностью действий (“маневренность”, частые “повороты”, “разгоны” и “торможения”), в то время как GPU идеален для процедур с невысокой расходимостью траекторий выполнения (однотипные вычислительные действия), но очень большим объемом данных, обрабатываемых одновременно и параллельно (“равномерное движение” со стабильно высокой нагрузкой).

GPU как правило оснащены небольшими объемами кэш-памяти, но очень большими пулами регистров [29]. Переключение потоков на GPU является очень быстрой операцией, так как при переключении потоков содержимое регистров не требуется сохранять в оперативной памяти. Но этот же фактор ограничивает общее число потоков, которые могут управляться планировщиком GPU и при этом находиться в разном состоянии – готовность, ожидание операндов и др.

Уникальной особенностью GPU является наличие так называемой *shared memory* – локальной памяти, разделяемой группой потоков. Shared memory – это своего рода кэш-память, напрямую доступная из программы (для CPU программ кэш-память прозрачна, программа не имеет полного контроля над ее содержимым). Таким образом небольшие резервы автоматического кэширования компенсируются наличием удобной локальной быстродействующей памяти с универсальным доступом. Главная цель shared memory – организация быстрого обмена данными между потоками. Доступ к памяти такого типа лишь немного медленнее доступа к регистрам, но регистры, используемые одним потоком, изолированы от регистров, используемых другими потоками, а shared memory доступна всем потокам группы. Для повышения пропускной способности чтения/записи физически эта память разбита на несколько банков, так что операции с разными банками могут выполняться одновременно. Время жизни данных в shared memory совпадает со временем жизни группы потоков, тогда как в глобальной (основной) динамической памяти ускорителя данные могут находиться сколь угодно долго.

Описание аппаратной архитектуры современного GPU (NVIDIA Ampere GA100) представлено в [53]. GA100 состоит из нескольких GPU-кластеров (GPU processing clusters, GPC), текстурных кластеров (texture processing clusters, TPC), потоковых мультипроцессоров (streaming multiprocessors, SM) и контроллеров памяти (HBM2 memory controllers).

С точки зрения прикладного программиста важно, что базовым вычислительным блоком GA100 является SM, потоковый мультипроцессор (рис. 5.5.2); рассматриваемый GPU содержит свыше 100 SM (до 128 в полной реализации), а каждый SM включает 64 блока выполнения операций с вещественными числами одинарной точности (FP32). Объем регистрового пула одного SM – 65536 32-битных значений. Объем локальной разделяемой памяти – до 164 Кб (конфигурируется). Максимальное число потоков, которым может управлять планировщик, в одном SM составляет 2048. Все SM функционируют независимо

друг от друга. Ориентируясь на эти цифры, можно приблизительно представить объемы эффективной загрузки GPU при решении практических задач.

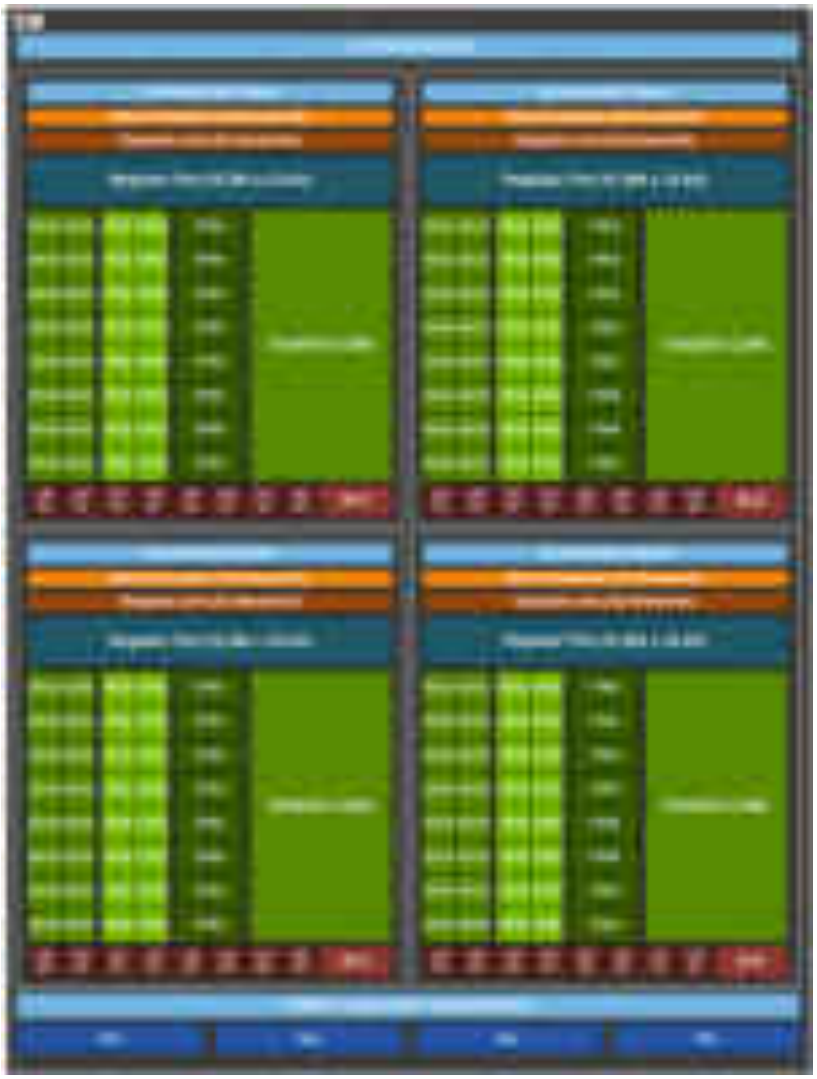


Рис. 5.5.2. Структура потокового мультипроцессора NVIDIA Ampere GA100
(источник изображения – [53])

5.6 Платформа NVIDIA CUDA

Платформа NVIDIA CUDA [3] в настоящее время стала основой *экосистемы* гетерогенного программирования (рис. 5.6.1); экосистема включает аппаратуру GPU (обязательно производства NVIDIA), драйвер, среду исполнения, набор инструментальных средств разработки, системные библиотеки математических подпрограмм и типовых алгоритмов, пользовательские приложения и библиотеки, написанные на языке C/C++, программные интерфейсы и средства связывания программ на скриптовых языках (в частности, на популярном в среде научных вычислений языке Python) с ядром и библиотеками CUDA, модули GPU-ускорения вычислений в популярных научных пакетах типа MATLAB, GNU Octave, TensorFlow и многих других.

Рис. 5.6.1. Экосистема CUDA (источник изображения – [57])

Краткое введение в технологию GPGPU и CUDA содержится в [55 – 58]. Полное справочное руководство по пакету средств разработки содержится в [3]. Установка системных средств CUDA для разных операционных систем подробно описана в руководстве [54].

После установки желательно проверить, как работают средства CUDA. CUDA использует собственный специализированный компилятор nvcc, который взаимодействует с компилятором общего назначения (по умолчанию gcc и g++ на Linux и cl.exe на Windows). Пример вывода версии компилятора nvcc:

```
$nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Oct_12_20:09:46_PDT_2020
Cuda compilation tools, release 11.1, V11.1.105
Build cuda_11.1.TC455_06.29190527_0
```

Команда nvidia-smi (NVIDIA System Management Interface) выводит информацию об аппаратуре и системной среде:

```
$nvidia-smi
Sat Dec 11 11:01:36 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   37C    P8    26W / 149W |      0MiB / 11441MiB |      0%      Default |
```

```
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Для получения всех преимуществ гетерогенного программирования полезно также знать информацию о центральном процессоре системы, особенно о количестве ядер и поддерживаемых векторных инструкциях. Под Linux характеристики процессора можно извлечь из файла `/proc/cpuinfo` (в примере вывод сокращен с помощью команды-фильтра grep):

```
$cat /proc/cpuinfo | grep -e "model name" -e processor -e flags
processor    : 0
model name   : Intel(R) Xeon(R) CPU @ 2.30GHz
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl
xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1
sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm
invpcid_single ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid
xsaveopt arat md_clear arch_capabilities
processor    : 1
model name   : Intel(R) Xeon(R) CPU @ 2.30GHz
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl
xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1
sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm
invpcid_single ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid
xsaveopt arat md_clear arch_capabilities
```

Как обычно, в начале работы рекомендуется собрать и запустить какую-нибудь простую программу, типа “Hello, World!”. Разумеется, такого рода программа должна содержать не только вывод строки приветствия, но и выполнение вычислительных действий, пусть и несложных, но актуальных для *гетерогенного* массово-параллельного программирования.

Ниже приведен пример программы, выполняющей сложение двух вещественных векторов длиной 100 млн. элементов типа float (около 400 мегабайт/вектор). В ней параллельное сложение векторов осуществляется в соответствии со схемой на рис. 5.5.1 (но без асинхронного ввода/вывода исходных данных). Работа выполняется частично на GPU средствами CUDA, и одновременно другая часть работы производится на многоядерном CPU средствами OpenMP. В программе измеряется общее время выполнения операции (от момента передачи исходных данных ускорителю до момента полной готовности результата).

Сложение векторов является задачей из категории *memory bound*, кроме того, пересылка данных между системной памятью и памятью ускорителя (несколько сотен мегабайт туда и мегабайт обратно) может занимать достаточно много времени. Фактически, генерация случайных чисел для заполнения исходных векторов занимает намного больше времени, чем последующее сложение этих чисел. Но рассматриваемый пример дает общее представление о

структуре гетерогенной программы. В нем также присутствуют зачатки оптимизации (генерация случайных чисел, настройка процедуры опроса состояния работы GPU – см. комментарии).

```
// заголовки CUDA
#include <cuda.h>
#include <cuda_runtime.h>
// прочие заголовки (для host-процедур)
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <immintrin.h>
#include <omp.h>

// kernel-процедура
//    (выполняется на GPU, запускается из host-программы)
//    a, b – векторы для суммирования,
//    c – вектор для результата;
//    n – размер векторов;
//    все векторы размещены в device-памяти
//    (предварительно a и b должны быть перенесены
//          из host-памяти в device-память;
//          после окончания вычислений вектор результата c
//          должен быть перенесен из device-памяти в host-память)
__global__ void vector_add(float* a, float* b, float* c) {
      // вычисление “плоского” индекса потока;
      // потоки запускаются блоками, каждый блок имеет трехмерный индекс,
      // в нашем случае блоки одномерные (см. далее код запуска);
      // blockDim.x – количество потоков в блоке,
      // blockIdx.x – индекс блока,
      // threadIdx.x – индекс потока внутри блока;
    int i = blockDim.x * blockIdx.x + threadIdx.x;
      // сложение элементов векторов по индексу,
      // соответствующему нашему потоку:
    c[i] = a[i] + b[i];
}

// host-процедура сложения векторов
void cpu_vector_add(const float* a, const float* b, float* c, int n) {
    #pragma omp parallel for simd
    for(int i = 0; i < n; ++i) {
        c[i] = a[i] + b[i];
    }
}

// процедура достаточно быстрой генерации случайных чисел
// с помощью процессорной инструкции RDRAND;
// (несмотря на применение intrinsic-функции,
//    время генерации исходных векторов
//    на порядок превышает время их сложения)
void generate_random_rdrand(float* buf, const int n) {
  const float r16max = 65536.0f;
  union {
    unsigned long long r;
    unsigned short r16[4];
```

```
  } random;

  #pragma omp parallel for
  for(int i = 0; i < n; i += 4) {
      // получаем 64 случайных бит
      //    (используется физический источник энтропии внутри CPU)
    _rdrand64_step(&random.r);
      // из 64-битного целого (4 unsigned short)
      // формируем 4 float-значения в диапазоне (-10, 10)
    buf[i] = (float)random.r16[0] / r16max;
    buf[i + 1] = -(float)random.r16[1] / r16max;
    buf[i + 2] = 10 * (float)random.r16[2] / r16max;
    buf[i + 3] = -10 * (float)random.r16[3] / r16max;
  }
}

// функции вычисления интервала времени между t1 и t2:
      // в наносекундах
long time_diff_ns(struct timespec t1, struct timespec t2) {
  time_t seconds = t2.tv_sec - t1.tv_sec;
  long nseconds = t2.tv_nsec - t1.tv_nsec;
  return (long)seconds * 1000000000L + nseconds;
}
      // в секундах
long time_diff_sec(struct timespec t1, struct timespec t2) {
  return time_diff_ns(t1, t2) / 1000000000L;
}
      // в миллисекундах
long time_diff_msec(struct timespec t1, struct timespec t2) {
  return time_diff_ns(t1, t2) / 1000000L;
}

int main(){
    // длина векторов
    int n = 100000000;

    // коэффициент для разделения работы на части между CPU и GPU
    double split_factor = 0.5f; // поровну

    // предварительное разбиение по количеству элементов векторов
    int cpu_count = (int)(split_factor * (double)n);
    int gpu_count = n - cpu_count;

    // конфигурация запуска kernel-процедуры:

    // количество потоков в одном блоке
    int threadsPerBlock = 256;
    // общее количество блоков (с запасом на кратность)
    int blocksPerGrid =
            (gpu_count + threadsPerBlock - 1) / threadsPerBlock;

    // точное количество элементов для обработки на GPU
    int device_n = blocksPerGrid * threadsPerBlock;

    // размер в байтах участков векторов для обработки на GPU
    size_t device_size = device_n * sizeof(float);
```

```
// размер в байтах всего вектора
size_t size = n * sizeof(float);

// инициализация host-памяти:

float* host_data;

// выделение host-памяти единым блоком:
// 3 вектора по n элементов (два операнда и результат)
host_data = (float*)aligned_alloc(32, 3 * size);

// указатель на начало результата в host-памяти
float* p_result = host_data + 2 * n;

// переменные для измерения времени
struct timespec t1;
struct timespec t2;

// “снимок” текущих показаний часов реального времени
// в момент начала генерации исходных данных в host-памяти
clock_gettime(CLOCK_REALTIME, &t1);
// заполнение двух векторов-операндов случайными числами
generate_random_rdrand(host_data, 2 * n);
// “снимок” текущих показаний часов реального времени
// в момент окончания генерации исходных данных в host-памяти
clock_gettime(CLOCK_REALTIME, &t2);
// вывод длительности времени, затраченного на генерацию
printf("\nReady RAND data in %ld ms\n", time_diff_msec(t1, t2));

// конфигурирование среды исполнения таким образом,
// чтобы во время опроса статуса выполнения kernel-процедуры
// системные потоки периодически отдавали центральный процессор
// рабочим потокам, обрабатывающим часть исходных данных на CPU
// (без этого рабочие CPU-потоки будут голодать,
//      а время CPU будет тратиться вхолостую на опрос GPU)
unsigned int flags;
cudaGetDeviceFlags(&flags);
flags |= cudaDeviceScheduleYield;
cudaSetDeviceFlags(flags);

// инициализация device-памяти:

float* device_a;
float* device_b;
float* device_result;

// “снимок” текущих показаний часов реального времени
// в момент начала переноса исходных данных в device-память
clock_gettime(CLOCK_REALTIME, &t1);

// выделение участков device-памяти
cudaMalloc((void**)&device_a, device_size);
cudaMalloc((void**)&device_b, device_size);
cudaMalloc((void**)&device_result, device_size);
```

```
    // перенос исходных данных из host-памяти в device-память
    cudaMemcpy(device_a, host_data, device_size, cudaMemcpyHostToDevice);
    cudaMemcpy(device_b, (void*)(host_data + n),
                         device_size, cudaMemcpyHostToDevice);

    // запуск kernel-процедуры (расширенный синтаксис nvcc!)
    vector_add<<<blocksPerGrid, threadsPerBlock>>>(device_a, device_b,
                                                   device_result);
    // kernel-процедура работает асинхронно;
    //      в это время можно часть работы  выполнить на CPU:
    cpu_vector_add(host_data + device_n, host_data + n + device_n,
                        p_result + device_n, n - device_n);

    // контроль ошибок CUDA (все может быть)
    cudaError_t err = cudaGetLastError();

    if(err) {
        printf("Error: %d\n", err);
    }
    else {
        // перенос результатов из device-памяти в host-память;
        // массив с содержит результат, доступный для CPU
        cudaMemcpy((void*)p_result,
                         device_result,
                               device_size,
                                     cudaMemcpyDeviceToHost);

        // "снимок" текущих показаний часов реального времени
        // в момент полной готовности результатов
        clock_gettime(CLOCK_REALTIME, &t2);

        // вывод нескольких начальных и конечных ячеек вектора-результата
        for(int i = 0; i < 10; ++i) {
            printf("%f ", p_result[i]);
        }
        printf("\n...\n");
        for(int i = n - 10; i < n; ++i) {
            printf("%f ", p_result[i]);
        }
        // вывод времени получения результата (CPU + GPU)
        printf("\nDone in %ld ms\n", time_diff_msec(t1, t2));
    }

    // освобождение ресурсов в device-памяти
    cudaFree(device_a);
    cudaFree(device_b);
    cudaFree(device_result);

    // освобождение ресурсов в host-памяти
    free(host_data);

    return 0;
}
```

Сборка исполняемого файла этой программы:

```
$nvcc add_vectors.cu -o addv -O3 -gencode arch=compute_37,code=sm_37 -Xcompiler
-fopenmp -Xcompiler -mrdrnd
```

Здесь параметры `-gencode arch=compute_37,code=sm_37` относятся к поддерживаемым аппаратурой моделям вычислений (таблицу соответствия моделей вычислений моделям GPU можно найти в [59]; для Tesla K80 compute capability = 3.7, эти значения следует установить при сборке программы). Поскольку nvcc для компиляции использует gcc, через командную строку параметры gcc предваряются опцией `-Xcompiler`.

На тестовой системе с виртуальными GPU и CPU (тест производился в облачной среде) получены следующие результаты:

```
$./addv
Ready RAND data in 2980 ms
0.305511  -0.977921  6.549988  -6.606140  1.427368  -0.456833  4.604034  -15.113220
0.873581 -0.532852
...
6.697388  -8.297424  0.555115  -1.513565  15.027771  -14.017639  0.732071  -1.282608
11.461029 -11.815643
Done in 288 ms
```

При измерении времени сложения векторов учитывалось и время передачи данных из host-памяти в device-память и обратно. При гетерогенном программировании следует минимизировать такие переносы, выполняя как можно большее количество арифметических операций над теми данными, которые доступны ускорителю. Очевидно, что задача сложения векторов хорошо подходит на роль программы “Hello, World!”, но ее решение именно гетерогенным способом не слишком эффективно.

Кстати, если выполнять сложение векторов с помощью функции `cublasSaxpy` из стандартной, хорошо оптимизированной библиотеки cuBLAS (CUDA Basic Linear Algebra Subroutine library), то на эту операцию уходит приблизительно столько же времени (в этом варианте программы для простоты кода все вычисления производились только на GPU):

```
$./addv_cublas
Ready RAND data in 3011 ms
1.320679  -0.667374  6.093140  -13.322296  0.653503  -1.141754  12.648468  -12.901611
1.481827 -0.758621
...
15.037384  -11.198578  0.954269  -0.588516  7.452087  -6.080322  0.932327  -0.256424
13.328552 -8.392029
Done in 390 ms
```

Так что наша начальная программа демонстрирует нормальную производительность.

Теперь следует подробнее остановиться на программной модели CUDA. Некоторые ее детали уже были затронуты в рассмотренном примере, но в силу простоты решаемой задачи не были использованы многие аспекты, важные для полноценного практического применения CUDA.

Потоки CUDA запускаются не по одному, а *блоками* (thread blocks). Блоки имеют трехмерную организацию (оси в этом пространстве обозначаются *x*, *y* и *z*), поэтому каждый отдельный поток (CUDA thread) в общем случае имеет трехмерный идентификатор. В kernel-процедурах предопределена трехмерная переменная `threadIdx` (координаты потока внутри блока), компоненты которой обозначаются `threadIdx.x`, `threadIdx.y`, `threadIdx.z`. Зачем нужна такая многомерность? Поток использует собственный идентификатор, чтобы выделить "свой" обрабатываемый элемент в общем массиве данных. Массово-параллельные вычисления обычно обрабатывают либо векторы (1D), либо матрицы (2D), либо объемы (3D), поэтому с помощью многомерного индекса удобно естественным образом идентифицировать элементы данных.

При запуске kernel-процедуры в программе указывается количество потоков в каждом блоке (все блоки одинаковы по размеру). В kernel-процедурах предопределена трехмерная переменная `blockDim` (размер блока), компоненты которой обозначаются `blockDim.x`, `blockDim.y`, `blockDim.z`.

Объем блока не может быть слишком большим, так как каждый блок потоков выполняется в рамках аппаратных возможностей одного потокового мультипроцессора (SM). Объемы регистрового пула и локальной разделяемой памяти в каждом SM ограничены, поэтому и на объем блока имеются ограничения – во всех версиях вычислительных возможностей CUDA-устройств (compute capabilities) допускается до 1024 потоков на блок. При этом максимальный размер блока по осям *x* и *y* – 1024, а по *z* – 64.

Для задачи обычно требуется больше одного блока потоков. Блоки объединяются в так называемую *сетку блоков* (grid of thread blocks, рис. 5.6.2). Этот агрегатный объект тоже трехмерный. Максимальная размерность сетки блоков по оси *x* составляет $(2^{31} – 1)$, по осям *y* и *z* – 65535. Для идентификации блока внутри сетки в kernel-процедурах предопределена трехмерная переменная `blockIdx` (индекс блока внутри сетки), компоненты которой обозначаются `blockIdx.x`, `blockIdx.y`, `blockIdx.z`. Общие размеры сетки блоков kernel-процедура может узнать из предопределенной трехмерной переменной `gridDim`, компоненты которой обозначаются `gridDim.x`, `gridDim.y`, `gridDim.z`.

Существует еще один, самый нижний уровень иерархии потоков CUDA – так называемые warp'ы (*warps*). Это слово в контексте CUDA обычно не переводится на русский язык. В английском языке оно имеет множество значений, но в терминологии, связанной с параллельными вычислениями, ближайший подходящий перевод – "основа (ткани)". Дело в том, что thread в английском языке означает "нить", поэтому терминология из ткацкого дела представляется очень оправданной. Так или иначе, планировщик GPU использует warp как базовую единицу параллельного исполнения. В CUDA размер warp'а – 32. В kernel-процедурах предопределена целочисленная переменная `warpSize`, ее значение в текущих версиях CUDA равно 32. В принципе, знать размер warp'а прикладной программе не обязательно, но учитывать специфику аппаратного планирования потоков необходимо при оптимизации производительности GPU-процедур.

Рис. 5.6.2. Трехмерная сетка блоков потоков CUDA

Компилятор nvcc, применяемый для разработки CUDA-программ на языке C/C++, поддерживает расширения языка для удобного описания kernel-процедур и конфигурации их запуска.

Поддерживаются следующие спецификаторы функций:

- `__global__` – функция запускается из host-среды и работает в device-среде; при определенных условиях такая функция может запускаться из device-среды; вызов функции с этим спецификатором происходит асинхронно, она возвращает управление до момента фактического окончания выполнения;
- `__device__` – функция запускается из device-среды и работает в device-среде;
- `__host__` – функция запускается из host-среды и работает в host-среде.

Спецификаторы `__global__` и `__host__`, а также `__global__` и `__device__` не могут применяться совместно. Спецификаторы `__device__` и `__host__` могут применяться совместно, тогда функция может исполняться или в device-среде, или в host-среде, в зависимости от аппаратных возможностей:

```
__host__ __device__ func() {
#if __CUDA_ARCH__ >= 800
   // Device code path for compute capability 8.x
#elif __CUDA_ARCH__ >= 700
   // Device code path for compute capability 7.x
#elif __CUDA_ARCH__ >= 600
   // Device code path for compute capability 6.x
#elif __CUDA_ARCH__ >= 500
```

```
   // Device code path for compute capability 5.x
#elif __CUDA_ARCH__ >= 300
   // Device code path for compute capability 3.x
#elif !defined(__CUDA_ARCH__)
   // Host code path
#endif
}
```

Поддерживаются следующие спецификаторы памяти в device-среде:

- `__device__` – переменная размещается в глобальной памяти ускорителя;
- `__constant__` – переменная размещается в глобальной памяти ускорителя и кэшируется в кэше констант;
- `__shared__` – переменная размещается в разделяемой памяти, доступной только потокам внутри одного блока;
- `__managed__` – переменная, имеющая характеристики и поведение объекта, размещенного в управляемой (managed) памяти;
- `__restrict__` – подсказка компилятору о том, что указатели не указывают на пересекающиеся участки памяти.

Поддерживаются встроенные векторные типы на основе стандартных скалярных типов (`char`, `uchar`, `int`, `uint`, `long`, `ulong`, `longlong`, `ulonglong`, `float`, `double`); количество элементов векторов – от одного до четырех (например, `ulong2`, `float4`). Каждый элемент вектора доступен по именам полей x, y, z, w, например:

```
float2 foo; foo.x = 0.5f; foo.y = 1.0f;
```

Определен макрос `make_<type name>`, например:

```
int3 foo_int = make_int3(1, 2, 3);
```

Для обозначения трехмерных размерностей используется тип `dim3` (векторный тип, основанный на `uint3`); элементы этого вектора, которые не получили явной инициализации, считаются равными 1.

Поддерживается расширенный синтаксис для запуска kernel-процедуры:

```
kernelName<<< GridSize, BlockSize, SMEMSize, Stream  >>> (arguments, ...);
```

Здесь `kernelName` – имя процедуры, `GridSize` (тип `dim3`) – размеры сетки блоков, `BlockSize` (тип `dim3`) – размеры блока потоков, `SMEMSize` (тип `uint`) – размер shared memory при динамической конфигурации (обычно это значение не указывается, разделяемая память конфигурируется по умолчанию), `Stream` (тип `uint`) – идентификатор потока, то есть последовательности CUDA-операций, таких как запуск kernel-процедур; поток с идентификатором 0 не требуется указывать. Аргументами могут быть указатели в device-памяти, а также числовые величины, передаваемые по значению. Количество аргументов – фиксированное.

Kernel-процедурам доступны функции барьеров памяти: `__threadfence_block()`, `__threadfence()`, `__threadfence_system()`. Эти функции необходимы для корректного обращения к данным в условиях совместной параллельной работы множества

потоков. Аппаратура ускорителя использует так называемую модель памяти со *слабым упорядочиванием* (weakly-ordered memory model), поэтому записываемые данные с точки зрения разных потоков могут появляться в разном порядке. Для разграничения этапов работы с памятью на “до” и “после” некоторого события используются барьеры (собственно, вызов барьерной функции и является разделительным событием).

Доступны также функции синхронизации выполнения инструкций в разных потоках внутри блока: `__syncthreads()`, `__syncwarp()` и другие. Система приостанавливает выполнение потоков блока (или warp’а в случае `__syncwarp`), пока траектории всех потоков блока или warp’а не подойдут к этому вызову. Типичное применение таких функций – запись данных разными потоками в разделяемую память, вызов sync-функции, гарантирующей, что все потоки выполнили запись, затем чтение и использование этих данных на новой фазе параллельного алгоритма.

Незаменимым средством при анализе программ является профайлер. CUDA предоставляет разные профайлеры; профайлер с интерфейсом командной строки называется nvprof [60]. На вход nvprof подается исполняемая программа, профайлер ее запускает и после завершения программы выводит статистику выполнения. Вот выдача из профайлера с нашей программой сложения векторов:

```
$nvprof ./addv
Ready RAND data in 2994 ms
==203== NVPROF is profiling process 203, command: ./addv
==203== Warning: Auto boost enabled on device 0. Profiling results may be inconsistent.
0.978729 -1.062241 10.305176 -7.728119 1.050613 -0.078659 13.838043 -10.490417 0.871262 -1.875427
...
13.186951 -6.784058 1.467712 -1.044159 8.685455 -4.961395 0.950790 -1.625641 7.636871 -7.384491
Done in 290 ms
==203== Profiling application: ./addv
==203== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   68.18%  128.98ms         1  128.98ms  128.98ms  128.98ms  [CUDA memcpy DtoH]
                   29.67%  56.128ms         2  28.064ms  27.886ms  28.242ms  [CUDA memcpy HtoD]
                    2.15%  4.0594ms         1  4.0594ms  4.0594ms  4.0594ms  vector_add4x(float4*, float4*, float4*)
      API calls:   45.26%  198.30ms         1  198.30ms  198.30ms  198.30ms  cudaSetDeviceFlags
                   42.54%  186.37ms         3  62.125ms  28.034ms  130.06ms  cudaMemcpy
                   11.70%  51.241ms         3  17.080ms  387.02us  25.427ms  cudaFree
                    0.31%  1.3665ms         3  455.50us  420.55us  481.86us  cudaMalloc
                    0.12%  531.00us         1  531.00us  531.00us  531.00us  cuDeviceTotalMem
                    0.05%  206.16us       101  2.0410us     167ns  79.592us  cuDeviceGetAttribute
                    0.01%  53.749us         1  53.749us  53.749us  53.749us  cudaLaunchKernel
                    0.01%  26.552us         1  26.552us  26.552us  26.552us  cuDeviceGetName
                    0.00%  5.4090us         1  5.4090us  5.4090us  5.4090us  cuDeviceGetPCIBusId
                    0.00%  4.6160us         1  4.6160us  4.6160us  4.6160us  cudaGetDeviceFlags
                    0.00%  2.9380us         1  2.9380us  2.9380us  2.9380us  cudaGetLastError
                    0.00%  1.7780us         3     592ns     225ns     997ns  cuDeviceGetCount
                    0.00%  1.3790us         2     689ns     294ns  1.0850us  cuDeviceGet
                    0.00%     299ns         1     299ns     299ns     299ns  cuDeviceGetUuid
```

Очень информативная выдача! Неожиданно, очень много времени (около 200 ms) выполняется функция `cudaSetDeviceFlags`. Однако убирать установку флага `cudaDeviceScheduleYield` из нашей гетерогенной программы не рекомендуется, так как у нас CPU тоже принимает участие в общей работе, и излишняя активность CUDA-среды в ожидании завершения kernel-процедуры будет мешать центральному процессору выполнять свое задание (проверено, без этого флага общее время вычислений увеличивается). Предсказуемо долго выполняется “вытягивание” данных из device-памяти в host-память. Перенос же данных “вниз”, из host-памяти в device-память, осуществляется

относительно быстро. Всего около 4 миллисекунд выполнялась kernel-процедура сложения векторов.

Приблизительно такие же результаты дает и профилирование сложение векторов с помощью функции `cublasSaxpy` из библиотеки cuBLAS:

```
$nvprof ./addvblas
Ready RAND data in 2967 ms
==284== NVPROF is profiling process 284, command: ./addvblas
==284== Warning: Auto boost enabled on device 0. Profiling results may be inconsistent.
==284== Warning: Profiling results might be incorrect with current version of nvcc compiler used to compile cuda app.
Compile with nvcc compiler 9.0 or later version to get correct profiling results. Ignore this warning if code is
already compiled with the recommended nvcc version
0.753815 -0.985062 13.715210 -11.112061 0.802185 -0.764664 17.216339 -6.895752 0.782959 -0.688171
...
8.277740 -11.916504 0.872360 -0.567245 11.441345 -8.517609 1.844559 -1.432709 13.002777 -9.148865
Done in 393 ms
==284== Profiling application: ./addvblas
==284== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   68.40%  265.87ms         1  265.87ms  265.87ms  265.87ms  [CUDA memcpy DtoH]
                   29.28%  113.82ms         3  37.940ms  2.4640us  57.341ms  [CUDA memcpy HtoD]
                    2.32%  8.9991ms         1  8.9991ms  8.9991ms  8.9991ms  void axpy_kernel_val<float,
float>(cublasAxpyParamsVal<float, float, float>)
      API calls:   67.75%  830.40ms         9  92.266ms  9.1600us  538.32ms  cudaFree
                   31.81%  389.90ms         4  97.475ms  31.919us  275.83ms  cudaMemcpy
                    0.22%  2.6482ms         7  378.31us  4.1340us  857.38us  cudaMalloc
                    0.13%  1.5703ms         3  523.42us  505.42us  558.84us  cuDeviceTotalMem
                    0.06%  752.71us       297  2.5340us     164ns  186.07us  cuDeviceGetAttribute
                    0.01%  124.95us         1  124.95us  124.95us  124.95us  cudaLaunchKernel
                    0.01%  91.177us         3  30.392us  27.770us  34.867us  cuDeviceGetName
                    0.00%  36.367us        18  2.0200us     477ns  21.577us  cudaEventDestroy
                    0.00%  20.853us         4  5.2130us  1.3600us  13.523us  cudaDeviceSynchronize
                    0.00%  19.845us        18  1.1020us     473ns  6.4790us  cudaEventCreateWithFlags
                    0.00%  6.6530us        12     554ns     370ns  1.7680us  cudaDeviceGetAttribute
                    0.00%  6.0220us         2  3.0110us  2.6010us  3.4210us  cuInit
                    0.00%  5.8060us         1  5.8060us  5.8060us  5.8060us  cuDeviceGetPCIBusId
                    0.00%  3.3720us         5     674ns     337ns  1.3770us  cuDeviceGetCount
                    0.00%  3.2300us         2  1.6150us     670ns  2.5600us  cudaGetLastError
                    0.00%  2.7660us         4     691ns     370ns  1.5050us  cuDeviceGet
                    0.00%  1.8920us         1  1.8920us  1.8920us  1.8920us  cudaGetDevice
                    0.00%  1.4250us         3     475ns     300ns     708ns  cuDeviceGetUuid
                    0.00%     933ns         2     466ns     363ns     570ns  cuDriverGetVersion
```

Здесь kernel-процедура выполняется 9 миллисекунд, но она в этой программе складывает на GPU векторы полностью, а не половину, как в предыдущей программе. Перенос массивов из device-памяти в host-память также предсказуемо выполняется в два раза дольше из-за линейного увеличения объемов данных.

5.7 Стандарт OpenCL

OpenCL (Open Computing Language) является открытым промышленным стандартом для разработки гетерогенного программного обеспечения [61], разрабатываемым и поддерживаемым некоммерческим консорциумом Khronos Group, в состав которого входят много крупных компаний, включая AMD, Apple, ARM, Intel, Nvidia, Sony Computer Entertainment и другие.

OpenCL определяет фреймворк параллельного программирования и включает язык (на основе C99), API, библиотеки, среду исполнения. Цель OpenCL – предоставить возможность реализации гетерогенного ПО, обладающего высокой *переносимость* (способностью без изменения программного кода работать в различных вычислительных средах), но при этом и высокой эффективностью. OpenCL-программы могут при необходимой поддержке со

стороны драйверов и среды исполнения объединять вычислительные возможности CPU, GPU, DSP (digital signal processors – цифровые сигнальные процессоры), FPGA (field-programmable gate arrays – программируемые вентильные матрицы).

OpenCL-платформа является гетерогенной средой (рис. 5.7.1), в которой host-система управляет в общем случае несколькими вычислительными устройствами – как правило, массово-параллельными ускорителями, но также и многоядерными CPU с поддержкой векторных инструкций. Каждое вычислительное устройство (OpenCL device) включает несколько вычислительных модулей (compute unit), а каждый модуль содержит в своем составе несколько процессорных элементов (processing element).

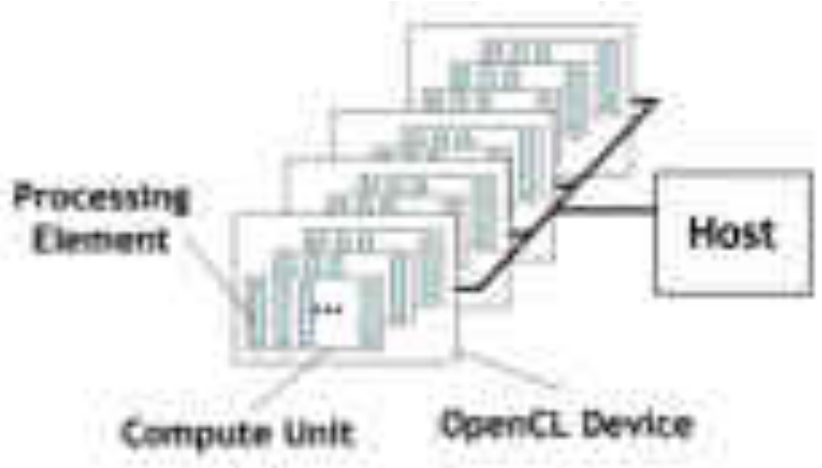


Рис. 5.7.1. Модель платформы OpenCL (источник изображения – [62])

Многомерная организация массово-параллельных потоков сходна с той, которая применяется в CUDA. Вообще, OpenCL и CUDA называют одни и те же объекты по-разному, но используют их, в сущности, одинаково, поэтому разработчикам не составляет значительных усилий адаптироваться, при необходимости, к альтернативной технологии. Краткий справочник [63] имеет компактное (на нескольких страницах) содержание, которое удобно использовать при повседневной работе с OpenCL.

В OpenCL отдельные потоки называются *рабочими элементами* (work-items), которые объединяются в *рабочие группы* (work-groups). Каждый поток имеет многомерный индекс (размерность – 1, 2 или 3). Пространство индексов в терминологии OpenCL называется NDRange. Все пространство делится на рабочие группы, рабочие элементы в рамках рабочей группы также имеют многомерные индексы (размерностью от 1 до 3). Внутри kernel-процедуры доступны следующие функции:

- `uint get_work_dim()` – получить размерность пространства индексов;
- `size_t get_global_size(uint dimindx)` – получить размер пространства по направлению, заданному `dimindx`;
- `size_t get_global_id(uint dimindx)` – получить глобальный индекс рабочего элемента по направлению, заданному `dimindx`;
- `size_t get_local_size(uint dimindx)` – получить размер рабочей группы по направлению, заданному `dimindx`;

- `size_t get_local_id(uint dimindx)` – получить локальный (в рамках рабочей группы) идентификатор рабочего элемента по направлению, заданному `dimindx`;
- `size_t get_num_groups(uint dimindx)` – получить количество рабочих групп по направлению, заданному `dimindx`;
- `size_t get_group_id(uint dimindx)` – получить идентификатор рабочей группы по направлению, заданному `dimindx`;
- `size_t get_global_linear_id()` – получить "плоский" (одномерный, линейный) глобальный идентификатор рабочего элемента;
- `size_t get_local_linear_id()` – получить "плоский" локальный (в рамках рабочей группы) идентификатор рабочего элемента.

Память на платформе OpenCL (рис. 5.7.2) имеет ту же организацию, что и в CUDA (разделяясь на host memory и device memory); терминология в отношении device-памяти частично другая: global memory, constant memory, local memory (соответствует shared memory в CUDA), private memory.

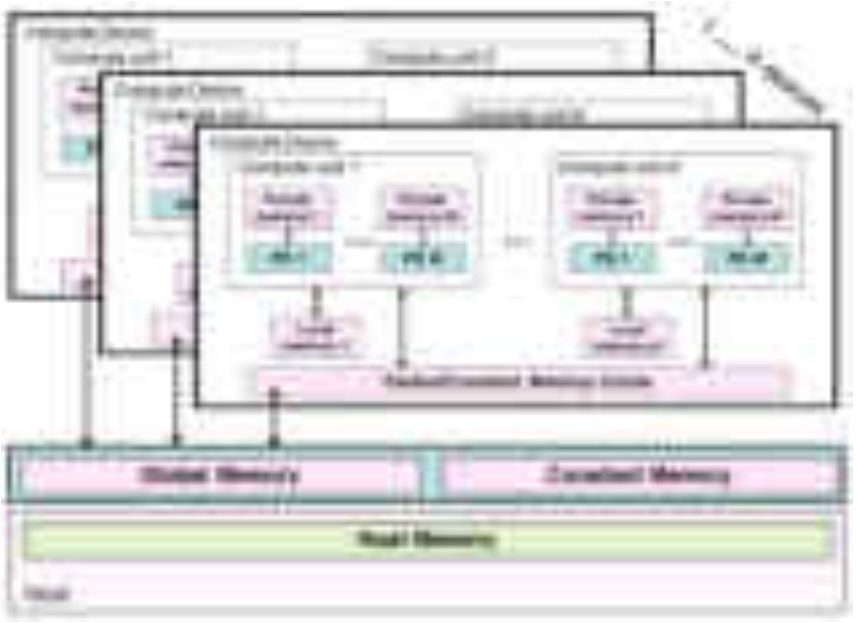


Рис. 5.7.2. Модель памяти OpenCL (источник изображения – [62])

Модель выполнения kernel-процедур в OpenCL представляется несколько более сложной, чем в CUDA, но на самом деле CUDA скрывает некоторые детали гетерогенного программирования, используя разумные умолчания. В OpenCL используются такие понятия как *платформа* (platform), *устройство* (device), *контекст* (context), *очередь команд* (command queue), *объект в памяти* (memory object, в частности – буфер), *программа* (program), *kernel-процедура* (kernel). Даже для написания простейшей OpenCL-программы, типа сложения двух векторов, придется вручную создавать все эти объекты. Пример такой программы приведен ниже; она во многом аналогична приведенной выше CUDA-программе.

```
// заголовок OpenCL
#include <CL/cl.h>
```

```
// прочие заголовки (для host-процедур)
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <immintrin.h>
#include <omp.h>

// kernel-процедура (записана в виде raw-строки C++11)
const char* kernel_add_source = R"(
__kernel void add(__global const float* a,
                  __global const float* b,
                  __global float* c) {
  int id = get_global_id(0); // используем одномерный идентификатор потока
  c[id] = a[id] + b[id];
}
)";

// host-процедура сложения векторов
void cpu_vector_add(const float* a, const float* b, float* c, int n) {
    #pragma omp parallel for simd
    for(int i = 0; i < n; ++i) {
        c[i] = a[i] + b[i];
    }
}

// процедура достаточно быстрой генерации случайных чисел
// с помощью процессорной инструкции RDRAND;
// (несмотря на применение intrinsic-функции,
//    время генерации исходных векторов
//    на порядок превышает время их сложения)
void generate_random_rdrand(float* buf, const int n) {
  const float r16max = 65536.0f;
  union {
    unsigned long long r;
    unsigned short r16[4];
  } random;

  #pragma omp parallel for
  for(int i = 0; i < n; i += 4) {
      // получаем 64 случайных бит
      //    (используется физический источник энтропии внутри CPU)
    _rdrand64_step(&random.r);
      // из 64-битного целого (4 unsigned short)
      // формируем 4 float-значения в диапазоне (-10, 10)
    buf[i] = (float)random.r16[0] / r16max;
    buf[i + 1] = -(float)random.r16[1] / r16max;
    buf[i + 2] = 10 * (float)random.r16[2] / r16max;
    buf[i + 3] = -10 * (float)random.r16[3] / r16max;
  }
}

// функции вычисления интервала времени между t1 и t2:
      // в наносекундах
long time_diff_ns(struct timespec t1, struct timespec t2) {
  time_t seconds = t2.tv_sec - t1.tv_sec;
  long nseconds = t2.tv_nsec - t1.tv_nsec;
```

```
  return (long)seconds * 1000000000L + nseconds;
}
      // в секундах
long time_diff_sec(struct timespec t1, struct timespec t2) {
  return time_diff_ns(t1, t2) / 1000000000L;
}
      // в миллисекундах
long time_diff_msec(struct timespec t1, struct timespec t2) {
  return time_diff_ns(t1, t2) / 1000000L;
}

int main() {
    // длина векторов
    const size_t n = 100000000;

    // размер векторов в байтах
    const size_t size = n * sizeof(float);

    // коэффициент для разделения работы на части между CPU и GPU
    double split_factor = 0.5f; // поровну

    // предварительное разбиение по количеству элементов векторов
    size_t cpu_count = (size_t)(split_factor * (double)n);
    size_t gpu_count = n - cpu_count;

    // конфигурация рабочих групп для GPU
    size_t gpu_local_work_size = 256;
    size_t gpu_groups_count =
      (gpu_count + gpu_local_work_size - 1) / gpu_local_work_size;

    // точное количество элементов для обработки на GPU
    size_t gpu_device_n = gpu_local_work_size * gpu_groups_count;

    // размер в байтах участков векторов для обработки на GPU
    size_t gpu_device_size = gpu_device_n * sizeof(float);

    // количество элементов векторов, оставшихся для обработки на CPU
    size_t cpu_device_n = n - gpu_device_n;

    // выделение host-памяти единым блоком:
    // 3 вектора по n элементов (два операнда и результат)
    float* host_data = (float*)aligned_alloc(32, 3 * size);

    // указатель на начало результата в host-памяти
    float* p_result = host_data + 2 * n;

    // переменные для измерения времени
    struct timespec t1;
    struct timespec t2;

    // “снимок” текущих показаний часов реального времени
    // в момент начала генерации исходных данных в host-памяти
    clock_gettime(CLOCK_REALTIME, &t1);
    // заполнение двух векторов-операндов случайными числами
    generate_random_rdrand(host_data, 2 * n);
    // “снимок” текущих показаний часов реального времени
```

```
    // в момент окончания генерации исходных данных в host-памяти
    clock_gettime(CLOCK_REALTIME, &t2);
    // вывод длительности времени, затраченного на генерацию
    printf("\nReady RAND data in %ld ms\n", time_diff_msec(t1, t2));

// работа с объектами среды исполнения OpenCL:

    // получаем идентификаторы платформ;
    // сначала необходимо узнать количество платформ:
    cl_uint platforms_count;
    clGetPlatformIDs(0, NULL, &platforms_count);
    // затем выделяем память под необходимое количество идентификаторов ...
    cl_platform_id* platforms = (cl_platform_id*)malloc(platforms_count);
    // ... и заполняем массив идентификаторов платформ
    clGetPlatformIDs(platforms_count, platforms, &platforms_count);

    // диагностический вывод названий платформ
    for(cl_uint i = 0; i < platforms_count; ++i) {
        char buf[100]; // для простоты используем статический буфер
        clGetPlatformInfo(platforms[i], CL_PLATFORM_NAME, 100, buf, NULL);
        printf("Platform #%d: %s\n", i, buf);
    }

    // запрашиваем вычислительное устройство (тип – GPU),
    // поддерживаемое основной платформой (первой в списке платформ)
    cl_device_id gpu_device;
    clGetDeviceIDs(platforms[0], CL_DEVICE_TYPE_GPU, 1, &gpu_device, NULL);

    // диагностический вывод названия устройства и версии OpenCL,
    // поддерживаемой для этого устройства
    size_t gpu_name_length = 0;
    clGetDeviceInfo(gpu_device, CL_DEVICE_NAME, 0, NULL, &gpu_name_length);
    char* gpu_device_name = (char*)malloc(gpu_name_length + 1);
    clGetDeviceInfo(gpu_device, CL_DEVICE_NAME, gpu_name_length,
                        gpu_device_name, NULL);
    size_t gpu_opencl_version_length = 0;
    clGetDeviceInfo(gpu_device, CL_DEVICE_VERSION, 0, NULL,
                        &gpu_opencl_version_length);
    char* gpu_opencl_version_name =
            (char*)malloc(gpu_opencl_version_length + 1);
    clGetDeviceInfo(gpu_device, CL_DEVICE_VERSION,
            gpu_opencl_version_length, gpu_opencl_version_name, NULL);

    printf("GPU: running on %s (%s)\n", gpu_device_name,
            gpu_opencl_version_name);

    // “снимок” текущих показаний часов реального времени
    // в момент начала создания контекста обработки данных
    clock_gettime(CLOCK_REALTIME, &t1);

    // создание OpenCL-контекста для GPU-устройства
    cl_context gpu_context = clCreateContext(NULL, 1, &gpu_device,
                                    NULL, NULL, NULL);

    // создание очереди команд для GPU-устройства в заданном контексте
    cl_command_queue gpu_command_queue =
```

```
        clCreateCommandQueueWithProperties(gpu_context,
              gpu_device, NULL, NULL);

// создание буферов в памяти GPU-устройства в заданном контексте
// и заполнение их исходными данными в момент создания
cl_mem gpu_d_a = clCreateBuffer(gpu_context,
    CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
        gpu_device_size, host_data, NULL);
cl_mem gpu_d_b = clCreateBuffer(gpu_context,
    CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
        gpu_device_size, host_data + n, NULL);
cl_mem gpu_d_c = clCreateBuffer(gpu_context,
        CL_MEM_WRITE_ONLY, gpu_device_size, NULL, NULL);

// создание OpenCL-программы в заданном контексте
// на основе исходного текста kernel-процедуры
cl_program gpu_program = clCreateProgramWithSource(gpu_context,
                        1, &kernel_add_source, NULL, NULL);
// компиляция и сборка программы для GPU-устройства
clBuildProgram(gpu_program, 1, &gpu_device, "", NULL, NULL);

// создание объекта kernel-процедуры (по ее имени в исходных текстах)
cl_kernel gpu_kernel = clCreateKernel(gpu_program, "add", NULL);

// установка аргументов kernel-процедуры
clSetKernelArg(gpu_kernel, 0, sizeof(cl_mem), &gpu_d_a);
clSetKernelArg(gpu_kernel, 1, sizeof(cl_mem), &gpu_d_b);
clSetKernelArg(gpu_kernel, 2, sizeof(cl_mem), &gpu_d_c);

// запуск kernel-процедуры на GPU-устройстве
// (задание помещается в очередь команд)
clEnqueueNDRangeKernel(gpu_command_queue, gpu_kernel,
                        1, NULL, &gpu_device_n, &gpu_local_work_size,
                            0, NULL, NULL);

// чтение буфера результатов
// (задание помещается в очередь команд)
clEnqueueReadBuffer(gpu_command_queue, gpu_d_c, CL_FALSE,
                        0, gpu_device_size, p_result, 0, NULL, NULL);

// передача очереди команд на устройство
clFlush(gpu_command_queue);

// запуск части работы на CPU
cpu_vector_add(host_data + gpu_device_n,
                host_data + n + gpu_device_n,
                    p_result + gpu_device_n, cpu_device_n);

// ожидание завершения выполнения заданий из очереди GPU
clFinish(gpu_command_queue);

// “снимок” текущих показаний часов реального времени
// в момент окончания обработки данных
clock_gettime(CLOCK_REALTIME, &t2);

// вывод нескольких начальных и конечных ячеек вектора-результата
```

```
    for(int i = 0; i < 10; ++i) {
        printf("%f ", p_result[i]);
    }
    printf("\n...\n");
    for(int i = n - 10; i < n; ++i) {
        printf("%f ", p_result[i]);
    }
    // вывод времени получения результата (CPU + GPU)
    printf("\nDone in %ld ms\n", time_diff_msec(t1, t2));

    // освобождение ресурсов OpenCL
    clReleaseKernel(gpu_kernel);
    clReleaseProgram(gpu_program);
    clReleaseMemObject(gpu_d_a);
    clReleaseMemObject(gpu_d_b);
    clReleaseMemObject(gpu_d_c);
    clReleaseCommandQueue(gpu_command_queue);
    clReleaseContext(gpu_context);
    clReleaseDevice(gpu_device);

    // освобождение ресурсов host-памяти
    free(gpu_opencl_version_name);
    free(gpu_device_name);
    free(platforms);
    free(host_data);

    return 0;
}
```

Программа компилируется с помощью следующей командной строки:

```
$g++ opencl-test2.cpp -o opencl-test2 -std=c++11 -O3 -lOpenCL -fopenmp -mrdrnd -mavx2 -m64
```

Вывод программы:

```
$./opencl-test2
Ready RAND data in 3000 ms
Platform #0: NVIDIA CUDA
GPU: running on Tesla K80 (OpenCL 1.2 CUDA)
1.804092 -0.597549 2.921448 -9.401855 0.462891 -0.697357 9.346924 -18.240967 1.579834 -0.513992
...
3.009949 -10.474854 0.444519 -0.452362 12.250061 -10.986633 1.169357 -0.744629 8.930969 -6.085510
Done in 643 ms
```

Здесь платформа OpenCL представлена средой исполнения NVIDIA CUDA. Кстати, профиль OpenCL 1.2 – достаточно старый (в настоящее время актуальны версии OpenCL 2.x … 3.0). Но и ускоритель Tesla K80 тоже новым не назовешь (хотя все равно это очень мощная и достаточно дорогостоящая вычислительная аппаратура). В любом случае, для теста не было выбора – такие ресурсы были предоставлены бесплатным облачным сервисом, который я использовал для подготовки этих материалов.

Источники информации

[1] Воеводин В.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ, 2006. – 112 с.

[2] Черняк Л. Архитектура фон Неймана, реконфигурируемые компьютерные системы и антимашина // Открытые системы №06, 2008

[3] Programming Guide : CUDA Toolkit Documentation
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[4] Asynchronous Data Copies using cuda::pipeline
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#memcpy_async_pipeline

[5] Барский А.Б. Архитектура параллельных вычислительных систем. 2-е изд. – М.: Интуит, 2016. — 298 с.

[6] Anant Agarwal : The Next Big Innovation in Microprocessors
https://www.sramanamitra.com/2007/08/20/the-next-big-innovation-in-microprocessors-anant-agarwal-part-1/

[7] Воеводин, В.В., Воеводин, Вл.В. Параллельные вычисления – СПб.: БХВ-Петербург, 2002. – 608 с.

[8] Programming Guide : CUDA Toolkit Documentation : Device Memory Accesses
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#device-memory-accesses

[9] Гергель В.П., Стронгин, Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие. – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184 с.

[10] Параллельная обработка данных : учеб. пособие для студ. вузов / А.О. Лацис. – М.: Издательский центр “Академия”, 2010. – 336 с.

[11] Daniel J. Sorin, Mark D. Hill, and David A. Wood, “A Primer on Memory Consistency and Cache Coherence” : Morgan & Claypool Publishers, 2011. – ISBN: 978-1-60845-565-2

[12] OpenMP : The OpenMP API specification for parallel programming
https://www.openmp.org

[13] GNU libgomp
https://gcc.gnu.org/onlinedocs/libgomp/

[14] IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)
https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html

[15] pthread.h(0p) – Linux manual page

https://man7.org/linux/man-pages/man0/pthread.h.0p.html

[16] C++ Thread support library
https://en.cppreference.com/w/cpp/thread

[17] C++ algorithms library
https://en.cppreference.com/w/cpp/algorithm

[18] C++ numerics library
https://en.cppreference.com/w/cpp/numeric

[19] C++ dynamic memory management
https://en.cppreference.com/w/cpp/memory

[20] C++ standard library header <execution>
https://en.cppreference.com/w/cpp/header/execution

[21] oneAPI Threading Building Blocks (oneTBB)
https://github.com/oneapi-src/oneTBB

[22] Class Thread (Java)
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Thread.html

[23] Interface Runnable (Java)
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Runnable.html

[24] Package java.util.concurrent
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/package-summary.html

[25] threading – Thread-based parallelism (Python 3)
https://docs.python.org/3/library/threading.html

[26] multiprocessing – Process-based parallelism (Python 3)
https://docs.python.org/3/library/multiprocessing.html

[27] concurrent.futures – Launching parallel tasks (Python 3)
https://docs.python.org/3/library/concurrent.futures.html

[28] multiprocessing.shared_memory – Provides shared memory for direct access across processes (Python 3)
https://docs.python.org/3/library/multiprocessing.shared_memory.html

[29] David B. Kirk, Wen-mei W. Hwu, “Programming Massively Parallel Processors : A Hands-on Approach”, 3rd Edition. – Morgan Kaufmann publications, Elsevier Inc., 2017. – 552 pp. – ISBN: 978-0-12-811986-0

[30] Масштабируемые процессоры Intel Xeon 3-го поколения
https://www.intel.ru/content/www/ru/ru/products/details/processors/xeon/scalable/platinum.html

[31] Supercomputer Fugaku
https://www.r-ccs.riken.jp/en/fugaku/about/
https://top500.org/system/179807/

[32] Yan Solihin, “Fundamentals of Parallel Computer Architecture: Multichip and Multicore Systems”. – Solihin Publishing & Consulting LLC, 2009. – 528 pp. – ISBN: 978-0-9841630-0-7

[33] MPI Forum
https://www.mpi-forum.org

[34] MPICH | High performance portable implementation of the Message Passing Interface (MPI)
https://www.mpich.org

[35] Open MPI : Open Source High Performance Computing
https://www.open-mpi.org/

[36] William Gropp, Ewing Lusk, Anthony Skjellum, “Using MPI : Portable Parallel Programming with the Message-Passing Interface”, 3rd Edition. – MIT Press, 2014. – 336 pp. – ISBN: 978-0-262-52739-2

[37] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI’04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS, pages 137–150, 2004.

[38] Apache Hadoop
https://hadoop.apache.org

[39] Tom White, “Hadoop: The Definitive Guide”, 4th Edition. – O'Reilly Media, Inc., 2015. – 756 pp. – ISBN: 978-1-491-90163-2

[40] Cray-1 Computer System Hardware Reference Manual 2240004, Rev C. – Cray Research, Inc., 1977.
http://bitsavers.trailing-edge.com/pdf/cray/CRAY-1/2240004C_CRAY-1_Hardware_Reference_Nov77.pdf

[41] Intel Intrinsics Guide : Intel
https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html

[42] Optimize Options (Using the GNU Compiler Collection (GCC))
https://gcc.gnu.org/onlinedocs/gcc-11.2.0/gcc/Optimize-Options.html

[43] Auto-vectorization in GCC

https://gcc.gnu.org/projects/tree-ssa/vectorization.html

[44] Clang: a C language family frontend for LLVM
https://clang.llvm.org/index.html

[45] Clang optimization levels
https://stackoverflow.com/questions/15548023/clang-optimization-levels

[46] Auto-Vectorization in LLVM
https://llvm.org/docs/Vectorizers.html

[47] Intel C++ Compiler Classic Developer Guide and Reference
https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top.html

[48] Intel oneAPI DPC++/C++ Compiler Developer Guide and Reference
https://www.intel.com/content/www/us/en/develop/documentation/oneapi-dpcpp-cpp-compiler-dev-guide-and-reference/top.html

[49] Automatic Vectorization : Intel C++ Compiler Classic Developer Guide and Reference
https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/optimization-and-programming-guide/vectorization/automatic-vectorization.html

[50] Export Compliance Metrics : App Metrics for Intel Microprocessors
https://www.intel.com/content/dam/support/us/en/documents/processors/APP-for-Intel-Xeon-Processors.pdf

[51] Видеокарты GeForce RTX 30
https://www.nvidia.com/ru-ru/geforce/graphics-cards/30-series/

[52] NVIDIA DGX A100 : The Universal System for AI Infrastructure
https://www.nvidia.com/en-us/data-center/dgx-a100/

[53] NVIDIA Ampere Architecture In-Depth
https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

[54] CUDA Quick Start Guide
https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html

[55] CUDA Refresher: Reviewing the Origins of GPU Computing
https://developer.nvidia.com/blog/cuda-refresher-reviewing-the-origins-of-gpu-computing/

[56] CUDA Refresher: Getting started with CUDA
https://developer.nvidia.com/blog/cuda-refresher-getting-started-with-cuda/

[57] CUDA Refresher: The GPU Computing Ecosystem

https://developer.nvidia.com/blog/cuda-refresher-the-gpu-computing-ecosystem/

[58] CUDA Refresher: The CUDA Programming Model
https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/

[59] GPUs | NVIDIA Developer
https://developer.nvidia.com/cuda-gpus

[60] Profiler :: CUDA Toolkit Documentation
https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvprof-overview

[61] Khronos OpenCL Registry – The Khronos Group Inc
https://www.khronos.org/registry/OpenCL/

[62] The OpenCL Specification
https://www.khronos.org/registry/OpenCL/specs/3.0-unified/html/OpenCL_API.html#_platform_model

[63] OpenCL 3.0 Reference Guide
https://www.khronos.org/files/opencl30-reference-guide.pdf

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.01 Методология научного познания

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Министерство образования и науки Российской Федерации

Ассоциация научных редакторов и издателей

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПОДГОТОВКЕ И ОФОРМЛЕНИЮ НАУЧНЫХ СТАТЕЙ В ЖУРНАЛАХ, ИНДЕКСИРУЕМЫХ В МЕЖДУНАРОДНЫХ НАУКОМЕТРИЧЕСКИХ БАЗАХ ДАННЫХ

Москва
2017

УДК 002.4
ББК 72
М 54

Авторы-составители:
О.В. Кириллова (Предисл., Введение, Гл. 1–5, Прил. 1), С.Л. Парфенова (Гл. 1, Прил. 2), Е.Г. Гришакина (Гл. 1, Прил. 2), Д.М. Кочетков (§ 2.1, Прил.1, 4), А.В. Кулешова (§ 2.2, Гл. 4), Е.М. Базанова (§ 3.1, 3.3), Е.Г. Доронина (§ 3.2), М.М. Зельдина ((§ 2.2, Гл. 5, Прил. 1, 5), К.А. Безроднова (Прил. 2), Е.А. Лягушкина (Прил. 2), М.А. Акоев (Прил. 3)

Под общей редакцией О.В. Кирилловой

Литературный редактор А.В. Кулешова
Технический редактор М.М. Зельдина

М 54 Методические рекомендации по подготовке и оформлению научных статей в журналах, индексируемых в международных наукометрических базах данных / Ассоциация научных редакторов и издателей; под общ. ред. О.В. Кирилловой. М, 2017. 144 с. (Прил.).

Методические рекомендации предназначены для студентов старших курсов, аспирантов, докторантов, готовящихся к опубликованию научных статей в зарубежных и российских научных журналах. В методических рекомендациях описываются основные этапы и требования к процессу подготовки к публикации результатов исследований; процесс отбора и оценки научного журнала для публикации научных статей; структура и оформление научной статьи; этические принципы и нормы научно-публикационного процесса; продвижение опубликованных статей, дан список использованных и рекомендуемых источников.

УДК 002.4
ББК 72

СОДЕРЖАНИЕ

Предисловие

Данные методические рекомендации разработаны в помощь авторам научных публикаций и содержат описание основных требований по подготовке научных статей для публикации в авторитетных журналах, индексируемых в международных наукометрических базах данных (МНБД) Web of Science и Scopus: дается краткая характеристика основных правил при подготовке рукописей к публикации; приводится основной инструментарий и инструкции по выбору целевых журналов с использованием МНБД и их приложений. Описывается структура научных статей и основные этические нормы и принципы, соблюдение которых обязательно при выполнении исследований и подготовке публикаций, а также – основные шаги по продвижению публикаций в международное научно-информационное пространство и включению результатов научных исследований в систему научных коммуникаций.

Рекомендации предусматривают требования к рукописям, подаваемым как в зарубежные, так и в российские журналы, индексируемые в МНБД.

К рекомендациям прилагаются Словарь терминов (Приложение 1), раскрывающий базовые понятия, использованные в рекомендациях, а также более подробные инструкции, позволяющие работать с каждым из ресурсов и их приложениями при отборе журналов (Приложение 4). В тексте рекомендаций даются ссылки на сайты, где размещены и актуализируются перечни российских и зарубежных журналов, включенных в МНБД Scopus и Web of Science, на другие ресурсы и методические материалы издательств, которые позволят авторам более подробно изучить данную тему. Дан список использованных и рекомендуемых источников.

Выделение определенных терминов в тексте рекомендаций *курсивом* означает, что они включены в Словарь терминов (Приложение 1), в котором даны разъяснения и уточнения этих понятий.

Рекомендации рассчитаны на начинающих авторов – аспирантов, молодых ученых, соискателей ученых степеней, и будут полезны другим научным работникам и специалистам, заинтересованным в успешной публикации и продвижении своих научных результатов в международные наукометрические системы и в целом – в международное научное пространство.

Введение

Научная публикация является неотъемлемой частью *научного исследования*, представляющей его промежуточный или конечный *научный результат*.

Публикация в *научном журнале* в современном мире *научных коммуникаций* играет двойную роль:

– является оперативным способом публикации и быстрого распространения информации о результатах оригинальных научных исследований авторов;

– является основным источником *библиометрических исследований* и оценки развития науки и достижений участников научного процесса – *авторов*, организаций, представляемых авторами, региона и страны в целом.

Публикация в виде научной статьи целесообразна, к ней будет проявлен интерес и с большей вероятностью будет процитирована, если она:

– представляет новые, оригинальные результаты или *методы исследований;*

– представляет рационализацию (уточнение или иную интерпретацию) опубликованных результатов;

– является обзором в области исследования или подведением итогов по определенной теме исследования;

– публикуется с целью расширения, но не повторения(!), знания в определенной, специфической области.

Публикация нецелесообразна, если работа представляет собой отчет, не имеющий научного результата; содержит устаревшую информацию; представляет собой дублирование ранее опубликованных работ или ошибочные, не применимые заключения.

Научная публикация в современном мире научных коммуникаций не имеет ценности, если ее никто не прочитал, не использовал и не процитировал. Поэтому очень важно представить научному сообществу результаты качественного научного исследования в авторитетном зарубежном или российском журнале, *индексируемом* в *международных наукометрических базах данных (МНБД) (*или – *«глобальных индексах цитирования»)*. Качественная *научная статья* в журнале, имеющем *библиометрические показатели*, с большей вероятностью привлечет внимание российских и зарубежных ученых и получит высокие показатели цитируемости.

Ни одна научная работа не может быть начата без предварительного изучения и анализа исследований, проводившихся ранее по выбранной теме или имеющих место в настоящее время. Результаты такого анализа публикаций по теме исследования отражается как непосредственно в научных статьях, так и публикуются в виде самостоятельных *систематических обзоров.*

Таким образом, если ученый хочет, чтобы его статья была опубликована в авторитетном международном журнале, он должен:

– подготовить качественную научную публикацию с качественными *заглавием, аннотацией (абстрактом, авторским резюме)* и *ключевыми словами;*

– выбрать *целевой научный журнал (target journal)*, соответствующий тематике и уровню представляемой статьи и, желательно, индексируемый или готовящийся (для российских журналов) к индексированию в МНБД;

– пройти *рецензирование*, обеспечиваемое журналом на уровне, достаточном для международного научного издания;

– оформить рукопись в соответствии с требованиями журнала;

– соблюсти при подготовке и подаче рукописи требования *этических норм*.

Публикация результатов своих исследований в авторитетном зарубежном журнале, представленном в МНБД, открывает для молодого ученого широкие перспективы карьерного роста, так как:

– подготовка к публикации в качественном журнале повышает научную квалификацию как ученого;

– повышает научный статус молодого ученого в научном сообществе, сначала – в своей стране, затем, в случае успеха, – за рубежом;

– улучшает «видимость» («visibility») и «доступность» («availability») научных разработок путем попадания публикаций в МНБД, что дает возможность анализировать оценку международным научным сообществом выполненных и опубликованных результатов научного исследования авторов;

– расширяет поле научной деятельности благодаря знакомству с зарубежными коллегами, заинтересовавшимися опубликованными результатами исследований, неформальному взаимодействию с ними, получению международных проектов, грантов, подготовке совместных публикации (*коллабораций*), и как результат – полноценному включению в систему научных коммуникаций на международном уровне.

В прагматическом плане публикация результатов научных исследований в авторитетных научных изданиях, индексируемых в МНБД, влечет за собой:

– повышение оценок результативности научной деятельности и материальное поощрение от организации, с которой аффилирован ученый, его карьерный рост;

– повышение рейтинга по наукометрическим показателям организации – университета, научного учреждения, компании, с которой аффилированы авторы;

– расширение присутствия страны в международном научном сообществе, укрепление позиций страны в целом.

Таким образом, усилия по подготовке к публикации результатов своих исследований в рейтинговых журналах, индексируемых в МНБД, безусловно, будут оправданы, способствуя повышению статуса авторов как ученых, их дальнейшему профессиональному и карьерному росту.

Глава 1. Основные этапы и требования к процессу подготовки к публикации результатов исследований

Проведение оригинального научного исследования в любой области науки требует значительных усилий и времени. Иногда необходимы годы, чтобы завершить работу, но это не означает, что полученные в процессе результаты не могут быть опубликованы до окончания исследования, на его промежуточных этапах.

В целом публикационный процесс можно представить следующим образом (рисунок 1).

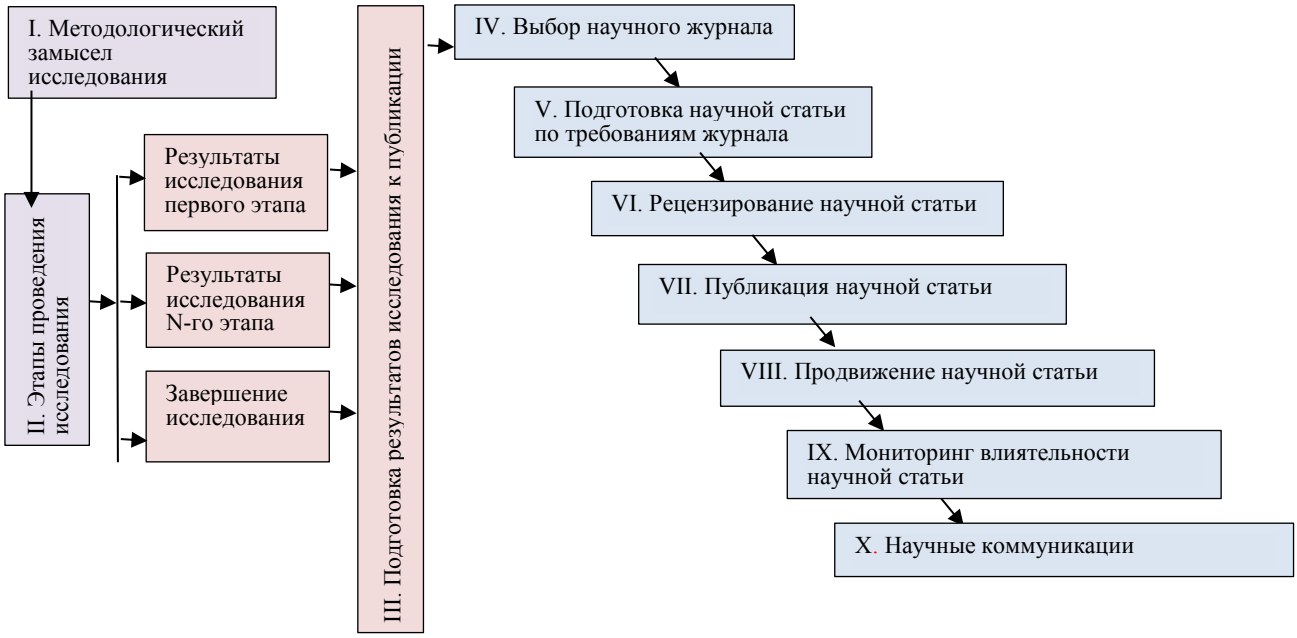


Рисунок 1 – Этапы публикационного процесса

I. Методологический замысел исследования

На стадии замысла формируется гипотеза исследования, логически определяющая порядок его проведения, основные этапы и предполагаемые результаты.

II. Этапы проведения исследования

Процесс проведения научного исследования на каждом этапе завершается результатами, которые должны быть представлены профессиональному сообществу в форме научных статей или других *типов научных публикаций*.

III. Подготовка результатов исследования к публикации

Стадия подготовки результатов исследования к публикации тесно связана с документированием научных результатов на каждом этапе исследования. Важно иметь четкое представление о способах интерпретации результатов исследования и определиться с типом научной публикации (*оригинальная научная статья*, *обзорная статья, краткое сообщение* и др.).

IV. Выбор целевого научного журнала

В идеале выбор научного(-ых) журнала(-ов) как источника(-ов) своих публикаций должен начинаться еще на этапе обзора и анализа мировых и отечественных достижений в предметной области намечаемого исследования и ранее – в процессе постоянной работы с профильной литературой по теме интересов и исследований ученого. Ученый должен публиковать свои работы в тех журналах, которые постоянно читает сам с целью отслеживания результатов мировых исследований по своей тематике. Такой подход помогает достичь конечной цели публикации – статья находит своего читателя.

Если читаешь журнал, то знаешь: его тематику; его авторов и организации, в которых эти авторы работают; правила оформления статей, списков литературы и научного аппарата; читаешь статьи, близкие по теме, которые можешь затем использовать и цитировать в своей работе. Подготовка *систематического обзора* литературы по теме исследования способствует формированию понимания, какие журналы являются целевыми для публикации собственных результатов.

Если такой перечень журналов заранее не сформирован, необходимо обратиться к ресурсам, позволяющим оценить и отобрать целевые журналы (см. Главу 2.1). При отборе важно ориентироваться не только на библиометрические показатели и соответствие тематики журнала основной предметной составляющей статьи, но и уметь оценить и исключить из рассмотрения *недобросовестные журналы*, нарушающие *этические нормы*. Особенно внимательно необходимо относиться к *журналам открытого доступа*, существующим за счет оплаты авторами своих публикаций («золотая» модель открытого доступа, Gold Open Access). Подробнее о недобросовестных журналах изложено в Главе 2.2.

Рекомендуется выбрать несколько журналов, однако это не означает, что рукопись можно одновременно подавать во все или несколько журналов. *Одновременная подача* («*веерная рассылка*») *рукописи* в разные журналы считается серьезным нарушением этических норм и может повлечь за собой неприятные последствия: выявление авторитетными журналами *дублирования публикаций* может привести к отказу в приеме рукописей этих авторов, а также к *ретрагированию (отзыву*) уже опубликованных

продублированных статей. Только после получения информации из одного журнала с отказом в публикации можно подавать рукопись в другой журнал! При этом при следующей подаче целесообразно внести изменения в рукопись по замечаниям редакторов и рецензентов предыдущего журнала. О требованиях соблюдения этических норм авторами при подготовке публикации изложено в Главе 4.

V. Подготовка научной статьи по требованиям журнала

Каждый журнал предъявляет требования к подаваемым рукописям, излагаемые в *Инструкциях для авторов* [1]. Все требования являются обязательными к исполнению. Рукописи, не соответствующие требованиям журнала, возвращаются авторам без рассмотрения. Как правило, инструкции крупных издательств (*Elsevier*, *Springer, Nature, Wiley*, *Taylor&Francis*, *Oxford University Press (OUP)*, *Sage*, *Emerald, Cambridge University Press (CUP)* и др.) имеют в основе очень схожие требования для журналов внутри издательств, независимо от тематической области, к которой принадлежит журнал. Также многое совпадает при рассмотрении инструкций журналов разных издательств по одной тематической области. Нельзя пользоваться инструкцией любого журнала издательства при подготовке рукописи в конкретный журнал, но желательно знать общие требования, предъявляемые ко всем журналам уже на первой стадии подготовки статьи.

Многие редакторы крупных издательств понимают, что подготовка научной статьи по требованиям журналов требует от авторов больших временных и других затрат, и в случае отказа от приема статьи до или после рецензирования это может быть напрасно потраченное время, которое ученый мог бы использовать на дальнейшие исследования. Поэтому один из журналов Издательства Elsevier предложил программу, названную «Your Paper, Your Way» (YPYW) (https://www.elsevier.com/authors/journal-authors/your-paper-your-way). Суть программы заключается в освобождении авторов от требований оформления рукописей, *иллюстраций*, *списков литературы* по правилам журнала до завершения процесса рецензирования. В таком случае процесс подачи статьи значительно упрощается. Только после сообщения о приеме статьи авторы начинают работу по доработке материала по формальным требованиям журнала. К этой программе присоединились более 500 журналов этого издательства. При подаче статьи в журнал Издательства Elsevier необходимо изучить правила для авторов и определить, входит ли журнал в эту программу.

В зависимости от того, какой тип статьи вы выбрали, следует изучить требования журнала к этому типу с точки зрения объема статьи, количества рисунков и количества источников. Типичные требования для журналов Издательства Elsevier [2]:

– *оригинальная научная статья* (Full Article) – стандартный формат для завершенных научных исследований – 8–10 стр. (18–20 страниц машинописного текста через 1,5 интервала), 5–8 рисунков, 25–40 ссылок;

– *краткое сообщение* (Short Communications Article) – не более 2500 слов, не более 2-х рисунков или таблиц; минимум 8 ссылок;

– *обзорная статья* (Review Paper/Perspectives) – критическое обобщение какой-то исследовательской темы; от 10 и более страниц, от 5 и более рисунков, 80 ссылок.

Если вы решили направить в журнал обзор, сначала изучите внимательно информацию в инструкции для авторов, принимает ли журнал обзоры, подаваемые по инициативе авторов. Встречаются журналы, которые публикуют обзоры, написанные только по заказу журнала.

Если вы не уверены, заинтересуется ли редакция журнала темой статьи, предварительно направьте в журнал краткий запрос с описанием основных положений предлагаемой статьи.

В Приложении 2 приведена сводная таблица, в которой выделены основные требования к подаваемым рукописям в журналах ведущих зарубежных издательств по областям науки (естественные, инженерные, точные, гуманитарные, социальные, медицина и сельское хозяйство), выявленные на основе анализа нескольких (двух–трех) журналов каждого издательства.

При подаче рукописи в зарубежный журнал авторам необходимо быть готовыми к тому, что все авторитетные, крупные издательства и большинство отдельных зарубежных журналов принимают статьи только в online режиме с сайта журнала. Подача статьи через систему «*электронной редакции*» позволяет автору проследить за ее прохождением через весь редакционный процесс. Так, Издательство Elsevier знакомит авторов с системой EVISE (https://service.elsevier.com/app/home/supporthub/publishing/), Springer дает подробное описание публикационного процесса и подачи статьи через собственную электронную систему (https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/submission/1302). Сервис ScholarOne Manuscript / Manuscript Central (http://scholarone.com/, разработка Компании Thomson Reuters (с конца 2016 г. владелец - Компания Clarivate Analytics) используется при подаче рукописей более чем в 3400 журналов крупных коммерческих и университетских издательств, таких как Cambridge University Press, Oxford University Press, IEEE, IET, Emerald, Royal Society of Chemistry, Sage, Taylor&Francis и др. Многие зарубежные и российские журналы используют для подачи и дальнейшей работы с рукописью в режиме «электронной редакции» открытое программное обеспечение *Open Journal System* (OJS), разработанное в рамках проекта

Public Knowledge Project (PKP) канадского Simon Fraser University совместно с другими университетами Канады и США (https://pkp.sfu.ca/ojs/). Авторам необходимо осваивать новые Интернет-технологии работы с журналами, как зарубежными, так и российскими.

Если автору(ам) трудно подать статью самому, можно привлечь в качестве посредника специалиста, знакомого с этим процессом. Однако это не означает, что постороннему человеку, не относящемуся к авторскому коллективу, в дальнейшем можно поручить всю работу с рукописью и взаимодействию с редакцией и рецензентами по редактированию статьи, точно также как довериться в выборе журнала и подаче статьи в любые журналы, о которых авторы не имеют представления. К сожалению, многие фирмы-посредники пытаются оказывать именно такие услуги, что приводит затем к потере репутации как журналов, с которыми они работают, так и к потере репутации авторами, опубликовавшими свои статьи в таких журналах. Опасно и нарушает этику научных публикаций сотрудничество с фирмами, предлагающими полные услуги по работе со статьей – от выбора журнала до ее опубликования (как правило, без участия авторов; т.н. «публикация под ключ»). Однако использовать услуги проверенных фирм по научному редактированию и корректуре (услуги «copyediting» и «proofreading») специальных текстов на английском языке рекомендуется.

Прежде чем подавать выполненную по всем формальным правилам журнала рукопись, необходимо быть уверенным в качестве и полной готовности ее содержательной части. Поэтому рекомендуется:

– корректно сформировать круг соавторов, внесших свой вклад в исследование и готовых взять на себя ответственность за представленные результаты и выводы;

– оценить возможности своей работы, степень ее оригинальности, актуальности и новизны, завершенности, готовности к представлению международному сообществу;

– оценить методологию и методы работы, достоверность и объективность выводов, их воспроизводимость, теоретическое и/или практическое значение;

– проверить ясность изложения и структурированность материала, основательность и логичность изложенной аргументации;

– подготовить и проверить качество текста на языке журнала, как правило, – на английском, воспользовавшись услугами редакторов и специалистов по тематике статьи, которые являются носителями языка или обладают совершенным его знанием;

– оценить качество списка использованных источников, охват ими международного опыта по теме исследования, отражение всех ссылок в тексте статьи, их новизну и уместность;

– подготовить качественные метаданные: информативное *заглавие статьи*, полную, излагающую содержание статьи *аннотацию (абстракт, abstract*) и дополняющие ее *ключевые слова.*

Более подробно о требованиях к подготовке статей написано в Главе 3.

При подготовке научной статьи важно наличие у авторов понимания этических принципов и норм публикационного процесса, пренебрежение к которым может негативно отразиться не только на публикационной, но и научной карьере ученого. Научный проступок и нарушение публикационной этики может принимать различные формы, быть умышленным или неумышленным. К примерам неправомерных действий и нарушений можно отнести: исследовательские мошенничества, в том числе фальсификацию и фабрикацию – манипулирование своими и чужими исследовательскими данными; плагиат – представление чужой идеи как собственной; представление результатов исследований как «салями-нарезка» («salami slicing») – подмена одной значимой рукописи несколькими мелкими работами с целью увеличения числа публикаций; наличие не заявленного конфликта интересов, который мог помешать автору быть беспристрастным в своих выводах; одновременную подачу статьи в более чем один журнал и др. Подробнее о понятиях и необходимости соблюдения этических норм при публикации результатов изложено в Главе 4.

Чтобы статья была принята к рассмотрению и дошла до рецензирования – не была отклонена главным редактором или ответственным редактором/ секретарем журнала на первом этапе прохождения рукописи:

– выбирайте журнал, точно соответствующий тематике вашего исследования, изучите его цели и задачи, тематический охват (рубрикацию) – для редактора важно, чтобы публикация «соответствовала объявленным целям» журнала;

– оформляйте статью строго в соответствии с требованиями журнала, не проявляя самостоятельности и волюнтаризма, в т.ч. соблюдайте требования к объему статьи, не увеличивая его, и – к спискам литературы;

– в списки литературы (References) включайте иностранные источники, которые должны быть проработаны при подготовке статьи; безусловно, при этом могут быть исключения, особенно это касается гуманитарной тематики;

– объем списка цитируемой литературы должен быть достаточным с точки зрения журнала и тематики (необходимо знать средний показатель объема списка литературы по предметной области);

– списки литературы вашей статьи для редактора и рецензента – демонстрация вашей эрудиции, информированности о текущих исследованиях в данной области,

поэтому цитируемые публикации должны быть как можно более новые (но и не следует увеличивать их чрезмерно, без причины);

– не увлекайтесь ссылками на свои работы, однако, и не исключайте их совсем, если публикация является продолжением предыдущих публикаций, даже если они были опубликованы на другом (русском) языке. Ссылки на собственные публикации демонстрируют преемственность ваших исследований, однако они должны быть сделаны на доступные источники, желательно – на статьи из журналов и составлять не более 1/3 списка литературы;

– хорошо продумайте и подготовьте *Сопроводительное письмо* (*Cover Letter*), оно должно вызвать интерес редактора к статье;

– обязательно укажите фамилию *автора для переписки* (*Corresponding Author*);

– посылайте рукопись тому редактору, на которого указывает журнал (если указан редактор по региону, то посылать надо ему, а не главному редактору);

– направляйте в той форме и тем способом, как указывает журнал.

Редакторы журналов зарубежных издательств, они же, как правило, сами – опытные авторы, часто пишут рекомендации авторам и ведут в Интернете блоги на эту тему. Они описывают опыт работы с собственными и/или поступающими статьями, основные действия и правила при подготовке текстов, готовят *чек-листы* (*check-list*) [3], позволяющие не упустить важные моменты при подготовке рукописи к публикации. Для более детального изучения этой темы рекомендуем воспользоваться этими рекомендациями [4–16].

В Приложении 3 приводится чек-лист (check-list) – рекомендации пошаговой проверки корректности действий и готовности рукописи, которые можно использовать при подготовке статьи, прежде чем ее отправить в редакцию, составленный на основе рекомендаций зарубежных и российских редакторов и авторов.

Если статья направлена на рецензирование, это уже значительный успех!

VI. Рецензирование научной статьи

Статья может считаться научной публикацией только в случае, если она прошла процесс до публикационного, предварительного *рецензирования*. Равно и журнал может считаться научным только при условии, что в нем организован процесс рецензирования. Автор должен понимать, как проходит данный процесс, для этого на сайте или в инструкции для авторов журнала он доступно и прозрачно описывается. Если информации на сайте нет, или журнал просит автора подготовить рецензию самостоятельно, или рецензирование выполняется только главным редактором и членами редколлегии, качество такого журнала не может быть расценено достаточным для

публикации. Наличие внешнего рецензирования, то есть экспертизы рукописи независимыми от журнала учеными, не входящими в его редколлегию, – один из важных признаков качественного научного издания. Наиболее приемлемым считается, когда экспертиза рукописи проходит рецензирование как минимум двумя экспертами, например, одним членом редколлегии и одним внешним экспертом.

Тип рецензирования говорит об уровне журнала. Наиболее распространенные типы рецензирования в авторитетных журналах:

– **двойное слепое (анонимное) рецензирование** (double-blind peer-review) – рецензент и авторы не знают фамилии друг друга;

– **одностороннее слепое (анонимное) рецензирование**, иногда пишут только «слепое» (single-blind peer-review, или blind) – рецензент знает фамилии авторов, авторы не знают фамилию рецензента;

– **открытое рецензирование** (open peer-review) – фамилии рецензента и авторов известны обеим сторонам.

Рецензентами могут быть авторитетные ученые, работающие по тематике журнала. Задача рецензента – оценить достоверность, научный уровень, значимость и оригинальность статьи, ее соответствие тематическим направлениям журнала, этическим принципам и нормам научно-публикационного процесса. По итогам рецензирования автору может быть предложено доработать рукопись или продолжить работу над результатами исследования. Опираясь на рекомендации рецензентов, редколлегии научных журналов решают, принимать рукопись или отклонить ее.

Все авторитетные издательства и журналы советуют авторам не обижаться на критические замечания рецензентов, не принимать критику на свой счет, так как она не относятся к личностям авторов. Как правило, критические замечания рецензентов помогают довести представленный материал до более высокого уровня.

Если авторы получили рекомендации по внесению изменений и исправлений в рукопись, это необходимо сделать оперативно, в указанный в сопроводительном письме срок. В случае возникновения вопросов по замечаниям, необходимо установить контакт с рецензентом для выяснения не понятных аспектов критики.

Если рецензент дал рекомендацию доработать статью, важно сделать все или большую часть того, что рекомендует эксперт. Наличие положительной рецензии с просьбой доработать рукопись – признак того, что статья может быть принята и опубликована. Редакторы ведущих журналов высказывают большие сожаления, когда авторы, получив замечания рецензентов, не доводят рукопись до необходимого уровня, бросают работу с ней или посылают ее в другой журнал. Редакторам и внешним

рецензентам не хочется делать лишнюю работу бесплатно. Работа редакторов и рецензентов не оплачивается. Хотя некоторые издатели предоставляют скидки на подписку журналов, покупку книг и другие льготы в форме вознаграждения, редактирование и рецензирование рассматриваются как выражение профессиональной ответственности по отношению к дисциплине, а также как средство быть в курсе результатов научных работ задолго до их обнародования [17–22].

Главное – не сдаваться, не обижаться, не думать, что к вам придираются, потому что не хотят опубликовать! Если бы не хотели, не передавали бы на рецензию, а сразу бы отклонили.

При завершении «работы над ошибками» необходимо составить сопроводительное письмо, в котором описана работа по всем пунктам замечаний. Нельзя присылать текст с видимыми правками («красным» от правок). Текст должен быть чистым [12].

Если рукопись отклонена, это не означает, что с ней надо прекратить работу. Рекомендуется узнать причины отказа, исправить статью по замечаниям и направить ее в другой журнал. Нет автора, который бы не получал отклонение рукописи авторитетными журналами. В высокорейтинговых журналах отклонение составляет от 80 до 90% поступающих текстов.

Если статья направлена в авторитетный журнал с высоким *импакт-фактором* и получила отказ, после ее доработки по замечаниям рецензентов можно снизить планку и послать в менее рейтинговый журнал. Иногда высокорейтинговые журналы сами рекомендуют журнал, куда можно направить статью. Если проведена большая работа с текстом, важно не сдаваться и довести его до публикации!

VII. Опубликование научной статьи

Поздравляем, Ваша статья принята!

Однако от даты принятия статьи до ее публикации может пройти от одного до 12 месяцев. Это зависит, в основном, от портфеля журнала, объема самого журнала (число статей в год), его периодичности (количества выпусков в год). Эти характеристики важно учитывать при выборе журнала, оценив примерные сроки публикации в случае принятия статьи. Важно обращать внимание на указанные в статьях журнала сроки от поступления статьи до ее принятия и публикации (Received, Accepted). Многие журналы указывают также дату получения статьи после рецензирования. Эти данные авторитетные издательства публикуют как в статьях, так и на сайтах журналов.

Многие авторитетные журналы и издательства, имеющие большие портфели статей и длительные сроки от приема статьи до ее публикации, оперативно публикуют принятые статьи, еще не имеющие точных выходных данных, в электронном виде на

своих сайтах (публикация называется «*Article in Press*»). Статьи, опубликованные предварительно on-line, МНБД размещают с этой маркировкой еще до выпуска печатного издания. Такой вариант электронной публикации не только дает авторам возможность оперативно представить полученные результаты международному сообществу, но и позволяет оперативно ссылаться на них другим ученым.

Перед публикацией статья проходит литературное редактирование, корректуру и техническую доработку, которые могут осуществляться как с участием, так и без участия автора. Обычно редактирование и корректура с участием автора проходит через обмен данными между ним и редакцией (издательством) по электронной почте или через редакционную систему журнала в режиме on-line.

Передача авторских прав. Автору статьи принадлежат следующие права: исключительное право на статью; право авторства; право автора на имя; право на неприкосновенность статьи; право на обнародование статьи.

Исключительные права на статью включают: публикацию, воспроизведение, тиражирование статьи, импорт оригинала или экземпляров статьи в целях распространения; перевод или другая переработка статьи; доведение статьи до широкой аудитории.

Исключительные права могут быть переданы автором на основании договора.

Для того чтобы журнал мог использовать Вашу статью, на основании российского законодательства до издания статьи необходимо подписать с редакцией/ издательством лицензионный (авторский) договор (соглашение). По лицензионному договору автор предоставляет издательству/редакции право использования статьи в установленных договором пределах. При подписании лицензионного договора автор сохраняет за собой право авторства, а редакция/издательство получает исключительное право на публикацию, воспроизведение, тиражирование бумажных и электронных копий статьи в течение всего срока, определенного лицензионным договором. В случае, если срок не определен, по умолчанию он составляет 5 лет.

Читатели получают доступ к статьям на условиях, которые совместно определят автор и редакция журнала. Условия могут быть описаны в редакционной политике на сайте журнала, если автор подписывает стандартное соглашение или присоединяется к действующему соглашению путем принятия оферты.

Договор – это права и обязанности автора и редакции журнала. Без договора читатели не могут получить доступ к статье. Отнеситесь к договору внимательно, он может допускать ограничения прав авторов на использование статьи в будущем.

Исключительные права авторов на статью фиксируются указанием знака Copyright © на титульной странице статьи.

Журналы, выходящие на международный уровень, для оповещения о правах авторов, издательства и читателей на распространение и использование публикаций журнала принимают одну из лицензий Creative Commons, указав знак CC, аббревиатур BY (Атрибуция/Attribution) и NC (Non Commercial), ND (No Derivatives), SA (Share Alike) на сайте и на издательской странице журнала, иногда – на титульной странице каждой статьи. Различные сочетания этих аббревиатур означают определенные права некоммерческого и коммерческого использования опубликованных материалов, всего шесть лицензий (https://creativecommons.org/licenses/). Например,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs

VIII. Продвижение научной статьи

После публикации в научном журнале жизнь научной статьи только начинается. Имея результаты качественного опубликованного исследования, автор(ы) имеет(ют) возможность не только повысить свою репутацию в научном сообществе, но и обеспечить дальнейшее развитие своей научной карьеры: привлекать в соавторы и соисполнители зарубежных коллег (создавать и участвовать в «*коллаборациях*», *collaborations*), принимать участие в новых проектах, получать гранты, выступать на конференциях в качестве приглашенных докладчиков и т.д.. Чем больше авторы используют возможности сделать доступными мировому научному сообществу результаты своих исследований, тем больше вероятности, что их работы будут признаны и процитированы. В Главе 5 изложены основные инструменты и методы продвижения публикаций и в целом результатов научных исследований [23].

IX. Мониторинг «влиятельности» научной статьи

Мониторинг «влиятельности» опубликованной статьи или ее препринта (в т.ч. в случае запрета издательством распространять конечный, опубликованный вариант статьи) в научном мире производится через изучение ее цитирования в МНБД, профессиональных и публичных социальных сетях, других информационных системах, индексирующих журналы. Важными являются показатели использования (обращения к аннотациям, открытия и скачивания статей) с сайтов издательств, агрегаторов ресурсов, архивов и других информационных систем. Для этого используются количественные методы анализа и формулы, индикаторы и метрики (показатели) «влиятельности» статей [24–27].

1. Основные статистические данные и вычисляемые на их основе показатели WoS, Scopus и других систем, учитывающих ссылки на включенные в эти системы публикации, препринты и другие издания:

- суммарное число публикаций автора;
- суммарное число цитирований публикаций (ссылок на публикации), включая *самоцитирование.* К этому показателю часто неправильно применяют термин «индекс цитирования». «Индексом цитирования» называются сами базы данных цитирования. Иногда этот показатель называют «*индекс цитируемости*» (http://www.spsl.nsc.ru/win/isitr/str_33h.html), однако и этот термин нельзя назвать корректным; суммарное число цитирований, исключая самоцитирование;
- среднее число ссылок на одну статью автора;
- среднее число ссылок в год или за другой период;
- *индекс Хирша* автора за весь или за любой установленный период.

2. Основные библиометрические показатели журналов, вычисляемые ежегодно по WoS и Scopus:

«*Импакт-фактор*» («impact-factor», «фактор влияния», IF) журнала; необходимо понимать, что несмотря на простоту подсчета «импакт-фактора» и использование его формулы в других системах, термин «импакт-фактор» является «брендом» WoS и не должен использоваться ни в какой другой информационной системе; использование его другими производителями ресурсов вводит в заблуждение пользователей и считается нарушением этических норм. В научных коммуникациях только импакт-фактор журналов, вычисленный на основе данных WoS, считается единственно корректным и используется во всех рейтингах и отчетах о научной деятельности. Поэтому авторы должны с осторожностью относиться к предложениям публиковать статьи в журналах, имеющих «импакт-фактор» в других системах, например, «CiteFactor», «Global Impact Factor», «Journal Impact Factor» и других. Джеффри Билл (Jeffrey Beal), американский библиотекарь из Университета Колорадо Денвер, который с 2009 по 2016 гг. вел свой блог недобросовестных издателей, журналов и других компаний, назвал такие компании компаниями, дающими некорректные метрики – «Misleading Metrics Companies», (https://scholarlyoa.com/other-pages/misleading-metrics/). В связи с закрытием сайта Дж. Билла посмотреть полные списки изданий, издательств и других компаний не представляется возможным.

Группа основных библиометрических индикаторов Scopus: *SJR, SNIP, CiteScore* (https://journalmetrics.scopus.com). CiteScore – новый индикатор, принятый Scopus в конце 2016 г., заменивший индикатор IPP (Impact per Paper). Индикатор CiteScore подобен

импакт-фактору, однако, в отличие от него, он рассчитывается на трехлетнем периоде цитирования, охватывает все типы публикаций (материалы конференций, статьи, обзоры, письма, редакционные статьи и т.д) и все типы цитирующих документов, включенных в Scopus (труды конференций, книги, продолжающиеся издания). Кроме того, введен также динамический показатель CiteScore Tracker для текущего года, который обновляется ежемесячно и демонстрирует актуальную продуктивность издания (http://elsevierscience.ru/news/398/citescore-novye-zhurnalnye-metriki-v-scopus).

3. *Альтметрики (альтернативные метрики*): методы *наукометрии,* использующие сети профессионального общения и сотрудничества ученых, созданные как альтернатива импакт-фактору и авторским показателям ввиду их ограничений (хронологические рамки, требование присутствия журнала в определенных индексах цитирования, тематика научного исследования и др.). Практически это метрики использования публикаций, страниц сайтов и т.п. в Интернет. К таким альтернативным показателям относится количество: скачиваний материалов и упоминаний в социальных сетях, новостях и блогах; просмотров; комментариев; цитат и др. Альтернативные показатели рассчитываются в общедоступных наукометрических ресурсах и базах данных, академических социальных сетях: *Google Scholar, ResearchGate, Mendeley, Zotero, Publish or Perish, Plum Analytics* и др. [28]

X. Научные коммуникации

Успешно выполненные научные исследования и опубликованные по их результатам тексты включаются в процесс научных коммуникаций. Научные коммуникации – система продвижения сформулированных научных идей, подтвержденных теоретическими и экспериментальными исследованиями внутри научного сообщества, включения их в процесс распространение научных знаний об окружающей действительности посредством различных каналов, средств, форм и институтов коммуникации. Научные коммуникации – совокупность видов профессионального общения в научном сообществе, один из главных механизмов развития науки, способа осуществления взаимодействия исследователей и экспертизы полученных результатов (http://terme.ru/termin/nauchnaja-kommunikacija.html) [24, 29].

Глава 2. Научные издания в международных наукометрических базах данных. Оценка и отбор научных журналов для публикации научных статей

2.1. Основные ресурсы, предназначенные для отбора целевых журналов

Международные наукометрические базы данных (МНБД), или «*глобальные индексы (указатели) цитирования»* в современном научном мире играют важную роль как основные источники информации:

– о наиболее значимых достижениях мировой науки и технологий, без изучения которых в настоящее время невозможно начать ни одно новое научное исследование;

– о наиболее авторитетных периодических и других изданиях, являющихся основными источниками распространения знания о наиболее важных достижениях науки и технологий.

Публикации, включенные в эти базы данных, не только получают быстрое распространение и, если заслуживают, то – признание международного сообщества, но и служат источниками библиометрических/ наукометрических исследований развития науки и технологий [24–27, 30].

Для выбора целевого журнала можно воспользоваться:

1) поиском по МНБД по тематическим запросам, составленным по ключевым словам готовящейся статьи;

2) перечнями журналов, индексируемых в МНБД, а также перечнями исключенных журналов;

3) специализированной базой данных Journal Citation Reports (JCR) на основе данных WoS;

4) дополнительными открытыми специальными инструментами поиска и анализа журналов (*Scimagojr.com, Journal Finder, Journal Metrics, Springer Journal Selector, Edanz Journal Selector* и др.);

5) тематическими поисками по метаданным статей или по предметным рубрикам платформ крупнейших издательств (http://sciencedirect.com, http://link.springer.com, http://www.nature.com/search/advanced и т.д.).

МНБД *Web of Science (WoS)* Компании *Clarivate Analytics* (до середины 2016 г. WoS принадлежала Компании *Thomson Reuters*) и *Scopus* Издательства *Elsevier* являются основными полноценными информационными системами, рассматриваемыми в качестве

инструментов для анализа продуктивности и успешности научной деятельности стран, организаций и отдельных ученых в мировом масштабе.

Основой оценки служат количественные (статистические) методы анализа цитирования этих публикаций. Наукометрические или библиометрические исследования построены на этих инструментах, а научные области, изучающие научные документальные потоки количественными методами, называются *библиометрия* и *наукометрия* [24–27, 30].

Каждый автор, строящий свою карьеру, заинтересован в том, чтобы подготовленная им статья дошла до международной читательской аудитории, была прочитала и процитирована в других статьях из журналов, индексируемых в МНБД. Поэтому важно выбрать для ее публикации правильный журнал, имеющий соответствующую тематике статьи целевую международную читательскую аудиторию. При этом необходимо знать основные индикаторы (библиометрические показатели), демонстрирующие авторитетность журнала.

МНБД Web of Science включает на сегодняшний день восемь баз данных, из них четыре индексируют журналы: *Science Citation Index Expanded* (SCIE), *Social Science Citation Index* (SSCI), *Arts and Humanity Citation Index* (A&HCI) и *Emerging Sources Citation Index* (ESCI) (http://ip-science.thomsonreuters.com/mjl/). Данные обновляются еженедельно. Известный всему миру основной библиометрический индикатор – *импакт-фактор («impact-factor»)* считается только для журналов, включенных в две первые из четырех БД (SCIE и SSCI). Для гуманитарных журналов, включенных в A&HCI, импакт-фактор не рассчитывается, равно как и для новой ESCI, запущенной в ноябре 2015 г. В совокупности во все четыре БД WoS CC в настоящее время включены 17 тыс. журналов, однако для выбора целевого издания интересны первые три БД, включающие 12,5 тыс. журналов. Эти журналы считаются наиболее авторитетными и, в основном, имеют импакт-фактор. Практически все журналы этих трех главных БД, за исключением небольшого числа изданий (не более пятидесяти), входят в Scopus. Классификатор Web of Science Subject Categories охватывает более 250 предметных рубрик (http://wokinfo.com/citationconnection/?utm_source=false&utm_medium=false&utm_campaign=false) [25].

МНБД Scopus представляет собой единый, не делимый, универсальный по тематике информационный массив, охватывающий все отрасли науки и технологий. Классификатор Scopus – *ASJC (All Science Journals Classification)* включает 27 кодов – основных тематических разделов, всего – 334 раздела и подраздела. Для оценки журналов Scopus использует «корзину метрик» [31], в которой основными библиометрическими

индикаторами считаются указанные ранее *SJR, SNIP* и *CiteScore (*https://journalmetrics.scopus.com). Scopus в настоящее время индексирует 22 тыс. журналов от 5 тыс. издательств мира, обновляется ежедневно (https://www.elsevier.com/solutions/scopus/content) [32–33].

Обе базы данных относятся к библиографическим/реферативным базам данных, так как не включают полные тексты индексируемых документов (статей из журналов, конференций, глав из книг и монографии), а только их *метаданные*: библиографические описания (заглавия статей, фамилии авторов, сведения об источнике публикации), авторские резюме (абстракты, аннотации, рефераты), ключевые слова и DOI. От традиционных реферативных баз данных их отличает наличие списков литературы, а также другой информации из статей (адресные данные авторов, информацию о финансировании и т.д.). В совокупности эти данные позволяют изучать цитирование и получать другую информацию для наукометрических исследований.

МНБД Scopus и WoS – закрытые ресурсы, распространяемые по подписке. Поэтому иногда трудно быстро ими воспользоваться для выбора журнала. В то же время для поиска журналов по ключевым словам статьи все-таки желательно найти возможность сделать поиски по этим БД. Результаты позволят точнее определить основные журналы по теме исследования, а также изучить мировые достижения в выбранной области исследования. В Приложении 4 представлена Инструкция по работе с ресурсами по выбору целевых журналов.

При поиске журнала, в первую очередь, рекомендуется воспользоваться дополнительными ресурсами и инструментами этих баз данных, доступными как по подписке, так и бесплатно. Ниже кратко охарактеризованы основные из этих ресурсов.

Journal Citation Reports (JCR) – источник сведений об импакт-факторе и других метриках журналов, включенных в две основные БД WoS – SCIE и SSCI, доступен по подписке; работа с JCR позволяет выбрать журналы по определенным предметным рубрикам и отсортировать их по различным показателям, в т.ч. ранжировать по убыванию импакт-фактора журнала и другим показателям [25–26].

Master Journal List Clarivate Analytics (http://ip-science.thomsonreuters.com/mjl/), включающий информацию о всех журналах, индексируемых во всех базах данных на платформе баз данных Clarivate Analytics (ранее – на платформе Web of Science Thomson Reuters), всего на платформе 24 базы данных, и только четыре относятся к Web of Science. По данному ресурсу можно проверять наличие журнала в БД WoS, а также в других реферативных базах данных этой компании, доступен бесплатно.

Scopus Source List (https://www.elsevier.com/solutions/scopus/content) – Excel-файл, включающий перечни журналов и других изданий (конференций, книг), индексируемых в Scopus, доступен бесплатно. В него входит также лист с кодами классификации ASJC. Лист с перечнем журналов, включенных в Scopus, содержит коды предметных рубрик классификации, по которым можно отсортировать и выбрать журналы определенного тематического раздела. При выборе журналов необходимо исключить из рассмотрения издания, которые уже не индексируются в Scopus: сделать сортировку по столбцу F – «Active or Inactive» и удалить из своего рабочего файла журналы с признаком «Inactive». Перечень включает показатели журналов (SJR, SNIP, CiteScore) за три года, что позволяет не только ранжировать отобранные журналы по этим метрикам, но и анализировать в динамике изменения показателей. Перечень журналов обновляется ежеквартально.

Список российских журналов, индексируемых в Scopus (http://elsevierscience.ru/products/scopus/) – размещен и постоянно актуализируется на российском сайте Издательства Elsevier. По данным на конец 2016 г. в Scopus индексировались более 410 журналов, более 70-ти из них были приняты в 2016 г. В списке указаны издательства, тематика, временной охват индексирования журналов.

Scopus Discontinued Sources List – перечень журналов, индексирование которых в Scopus прекращено или прервано. Список размещается на том же информационном сайте Scopus (Content Coverage – https://www.elsevier.com/solutions/scopus/content), доступен бесплатно. Этот список важен для определения журналов, которые уже не индексируются в Scopus, в том числе по причинам нарушения этических норм. Список содержит данные об издательствах журналов, а также о последних отраженных выпусках, проиндексированных в БД. По списку можно выявить издательства, журналы которых в нем наиболее часто фигурируют, и стараться с ними не работать, даже если выбранные журналы этих издательств еще индексируются в Scopus. Необходимо иметь в виду, что список исключенных журналов постоянно пополняется. Однако, для того, чтобы журнал оказался в него включен, идет достаточно долгая проверка, поэтому важно уметь самостоятельно определять недобросовестные журналы (см. § 2.2).

Scimagojr.com, SCImago Journal & Country Rank (http://scimagojr.com/) – общедоступный портал, созданный испанской группой Scimago (Университет Гранады) на основе данных Scopus. Включает анализ журналов и научные показатели стран в сравнительном аспекте или индивидуально по каждому журналу или стране. Система позволяет выгружать списки журналов по 27 основным тематическим областям и 313 конкретным предметным категориям, или по стране. Списки журналов выгружаются в Excel файлы в порядке ранжирования по разным параметрам журналов, выбранным перед

выгрузкой (по SJR, индексу Хирша). В списках указываются данные по объему журнала (общее количество статей в год), среднему количеству ссылок в списках литературы, издательству и другая полезная для анализа и выбора издания информация. Более подробно о работе с Scimago см. Инструкцию по работе с ресурсами по выбору целевых журналов (Приложение 4).

Journal Finder (http://journalfinder.elsevier.com/) – бесплатная система поиска целевого журнала по заглавию, абстракту, ключевым словам статьи, предлагаемая издательством Elsevier.

Платформы журналов крупнейших издательств Elsevier (http://www.sciencedirect.com/), Springer (http://link.springer.com/), Wiley (http://onlinelibrary.wiley.com/) и др. На всех платформах этих и других издательств предоставляется возможность отобрать журналы по предметным областям. Если журнал включен в Scopus и/или WoS, он размещает на своем сайте индикаторы журнала.

2.2. Критерии и определение недобросовестных журналов

В условиях роста требований к числу публикаций при отчетности по результатам научной деятельности и при получении проектов, грантов и т.д., есть большая вероятность выбрать для публикации недобросовестный журнал. Вероятность ошибиться в выборе журнала существует также в случаях, когда авторы пользуются услугами компаний-посредников, предлагающих публикацию «под ключ» – от выбора журнала до публикации статьи. Такие компании, предоставляя полный комплекс услуг по публикации, редко работают с авторитетными журналами. Поэтому рекомендуем авторам учиться самим отличать добросовестные журналы от недобросовестных, пользоваться МНБД и дополнительными к ним приложениями, а также работать с сайтами, специально созданными для информирования авторов о таких журналах [34–35].

Недобросовестные издания часто называют «хищническими», «хищниками», «паразитами», «мусорными». Прежде всего эти эпитеты относятся к журналам, существующим на бизнес-модели «золотого» открытого доступа (Gold Open Access), построенной на оплате авторами и организациями, с которыми аффилированы авторы, услуг по опубликованию своих статей. В условиях высокого спроса на публикации в связи с ростом требований к числу публикаций, этот бизнес стал очень успешным. Зарубежные и российские организации – ассоциации, комитеты, общества редакторов и издателей, информационные компании и библиотеки активно борются за очищение научного информационного пространства от недобросовестных издательств, отдельных журналов и других бизнес-компаний, паразитирующих на науке. К таким организациям, прежде

всего, относятся *Committee on Publicaiton Ethics (COPE), Directory of Open Access Journals (DOAJ), Open Access Publishers Association (OASPA), Council of Science Editors (CSE), International Committee of Medical Journal Editors (ICJME), World Association of Medical Editors (WAME), European Association of Science Editors and Publishers (EASE), Ассоциация научных редакторов и издателей (АНРИ)* и др. Основной организацией является Комитет по публикационной (издательской) этике (Committee of Publication Ethics, COPE, http://publicationethics.org). Кодексы и стандарты COPE по этике для авторов, редакторов, издателей, рецензентов, учреждений являются основополагающими. В членах COPE – более 10 тыс. членов – редакторов, издателей, ученых. Если авторы столкнулись с недобросовестной политикой редакции/ издательства журнала, являющегося членом COPE, они могут обратиться за помощью непосредственно в COPE. В России авторы могут обращаться в Совет по этике АНРИ, http://rasep.ru.

Большую популярность в борьбе с хищническими издательствами и изданиями открытого доступа приобрела работа американского библиотекаря из Библиотеки Аурария при Университете Колорадо Денвер Джеффри Билла (Jeffrey Beall, https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BB%D0%BB,_%D0%94%D0%B6%D0%B5%D1%84%D1%84%D1%80%D0%B8). Его сайт https://scholarlyoa.com, где были размещены списки издательств, журналов и компаний, некорректно использующих наукометрические метрики, в частности, «импакт-фактор», был одним из основных ресурсов, которым пользовались многие ученые, библиотекари, информационные специалисты и другая заинтересованная аудитория с целью определения качества журнала. Хотя в настоящее время сайт закрыт (https://debunkingdenialism.com/2017/01/16/what-happened-to-jeffrey-bealls-list-of-allegedly-predatory-publishers/), но списки журналов и издательств (Beall's Lists) сохранены в архиве и их можно посмотреть по этим ссылкам: List of publishers: https://web.archive.org/web/20170112125427/https:/scholarlyoa.com/publishers/; List of standalone journals: https://web.archive.org/web/20170111172309/https:/scholarlyoa.com/individual-journals/. В Приложении 5 приведен перечень критериев, которыми пользовался Дж. Билл, разработанные им с учетом кодексов и стандартов COPE.

В чем отличие добросовестных журналов от недобросовестных?

Добросовестные научные журналы содействуют тому, чтобы все участники процесса публичного представления научных результатов соблюдали правила и нормы поведения (этические принципы), позволяющие сохранить целостность и достоверность научного знания.

Недобросовестные издания используют науку и ученых как средство заработка, они делают ложь нормой, научное знание на страницах таких журналов мутирует, опубликованные результаты исследований могут быть не достоверны, не воспроизводимы, содержать в себе элементы плагиата, фальсификации и фабрикации, дублирования уже опубликованных материалов. Как правило, такие издания не проводят должного рецензирования. Это ведет к утрате доверия к науке и научным знаниям. Публикации в таких журналах могут отрицательно влиять на репутацию автора со стороны потенциальных соавторов и организаций, выделяющих финансирование на проведение научных исследований.

Для журналов, включенных в МНБД, основными явными признаками недобросовестности, указывающими на отсутствие рецензирования и направленность на получение прибыли, в первую очередь является резкий рост годового объема журнала – числа публикаций в год. В этом случае авторы должны сразу понимать, что публиковать статью в таком журнале не стоит: есть прямая опасность, что журнал будет исключен из МНБД.

Другими явными признаками журнала – претендента на исключение из МНБД – являются высокое самоцитирование журнала (превышение ссылок на свои публикации в сравнении над ссылками на журнал из других источников) и договорное цитирование (журнал цитируют только несколько определенных журналов). Такие характеристики также должны насторожить авторов. Нарушение этих показателей для WoS является основанием для прекращения, сначала на год, включения журнала в Journal Citation Reports (JCR).

Scopus ведет постоянную работу по выявлению недобросовестных журналов, индексируемых в этой МНБД. Однако исключение журналов из базы данных не происходит моментально после получения сведений о наличии признаков нарушения этики. Журнал получает предупреждение о выявленных нарушениях, его отражение может быть сначала прервано. Только после подтверждения нарушений индексирование журнала прекращается, и журнал включается в соответствующий список исключенных журналов (Scopus Discontinued Sources List на информационном сайте Scopus**,** см. § 2.1).

Проблему недобросовестных журналов и задачу по их выявлению решают некоторые российские университеты. Большую работу ведет Высшая школа экономики (НИУ «ВШЭ») по составлению «черных» и «белых» списков журналов. Основная цель – регулирование процесса установления надбавок в оценки качества статей по источникам их публикации [36]. Разработано «Положение о Списке журналов и издательств, публикации в которых не учитываются при назначении академических надбавок …», в

котором приведен перечень критериев при отборе недобросовестных журналов [37]. На сайте ВШЭ создан раздел, посвященный оценке журналов и публикаций (https://scientometrics.hse.ru/evaluation), на котором размещено данное Положение. Список журналов публично не доступен.

Признаки недобросовестных изданий, которые могут относиться ко всем, и зарубежным, и российским журналам, и не только открытого доступа, разработаны Советом по этике Ассоциации научных редакторов и издателей (АНРИ, http://rasep.ru) на основе российского и международного опыта. К основным относятся следующие признаки:

- В журнале нет рецензирования (это можно определить, например, по критически быстрому сроку прохождения статьи). Журнал просит автора самостоятельно подготовить или получить от коллег рецензии на свой текст;
- Журнал рассылает спам с предложением опубликоваться в кратчайшие сроки (2–3 дня, неделю и т.п.), что указывает также на отсутствие рецензирования и редактирования;
- Журнал указывает о себе недостоверную информацию (например, о включении в базы данных Scopus и Web of Science), приводит ложные индексы цитирования, несуществующие или несущественные показатели, не указывает ISSN;
- Журнал публикует тексты по очень широкому кругу научных дисциплин (всеобъемлющий охват тематических областей);
- На сайте нет достаточной информации для авторов, не раскрыта редакционная политика издания и публикационная этика, положение о рецензировании и платных услугах (если таковые есть);
- Многие статьи в журнале имеют критически низкий объем, около 3–4 страниц. Журнал требует поделить полноценную статью на несколько небольших статей;
- Критически высокий объем текстов в номере (от 40 и более статей);
- Журнал публикует материалы заочных конференций;
- Журнал предлагает повысить научный уровень статьи силами редакции («публикация под ключ»). Такая практика подменяет работу ученых, вводит в заблуждение относительно авторства материала, способствует нарушению целостности научных исследований;
- Журнал заявляет на сайте, что принимает тексты любого качества. Это косвенно указывает на неизбежность фабрикаций и фальсификаций при доведении текстов до приемлемого уровня;

- Журнал постфактум требует плату за публикацию, изначально не предоставив на сайте информацию о платных услугах;
- Журнал скрывает имена и фамилии своих сотрудников, экспертов, членов редколлегии. Если издание ведет добросовестную деятельность, они должны быть раскрыты, чтобы авторы могли убедиться в компетентности этих людей;
- Редакция предлагает агентские услуги, например, по подготовке платных рецензий (сюда не относятся легальные услуги, такие как перевод, редактирование или техническая подготовка рукописи);
- Журнал занимается продажей соавторства;
- Журнал предлагает услуги по манипуляции с цитированием, увеличение наукометрических показателей, включая избыточное самоцитирование.

Обнаружение таких признаков недобросовестности журналов, индексируемых в МНБД, также может быть причиной их исключения из этих ресурсов.

До и после выбора журнала рекомендуется проверять себя на предмет правильности принимаемого решения, задавая себе такие вопросы (http://thinkchecksubmit.org/):

– Известен ли этот журнал Вам или Вашим коллегам?

– Читали ли Вы когда-либо его статьи?

– Легко ли найти ранее опубликованные в этом журнале статьи?

– Легко ли идентифицировать издателя журнала и найти контактную информацию: есть ли название издательства на сайте журнала, можно ли связаться с издательством по телефону, электронной или обычной почте?

– Есть ли четкое указание, какой тип рецензирования использует журнал?

– Индексируется ли журнал в базах данных, которыми вы пользуетесь?

– Вам понятно, какие услуги придется оплатить? Есть ли на сайте журнала разъяснение, что включено в авторские сборы и в какой момент надо будет их оплатить?

– Вы знаете или слышали ли раньше о членах редакционной коллегии журнала?

– Упоминают ли члены редколлегии о своей работе в журнале на своих сайтах?

Перед отправкой рукописи вы должны быть уверены, что выбранный вами журнал ценится в вашей области, и поэтому повысит вашу репутацию и даст вашей работе больше шансов быть процитированной, а публикация в подходящем журнале даст вам возможность продвинуться по карьерной лестнице. Ваша статья должна быть проиндексирована, архивирована и легко доступна при поиске. Вы должны ожидать

повышения своего профессионального уровня, когда ваша работа будет отрецензирована и отредактирована.

Глава 3. Структура и оформление научной статьи

3.1. Общепринятые требования к структуре научной статьи

В настоящее время в международном научном сообществе сложилось четкое представление о том, что такое *научная статья* – письменный и опубликованный в рецензируемом научном журнале отчет, описывающий результаты оригинального экспериментального исследования, и удовлетворяющий определенным критериям. Научная статья об оригинальных экспериментальных исследованиях, как правило, написана в соответствии с общепринятым форматом – IMRaD (Introduction, Methods, Results, and Discussion). Иногда к аббревиатуре IMRaD добавляется буква A, обозначающая Abstract (Аннотация), получается AIMRaD. Если статья посвящена теоретическому исследованию, то раздел Methods (Методы) заменяется на Theoretical Basis (Теоретические основы) [38–42] .

Научные публикации в формате IMRaD впервые появились на страницах научных журналов в конце XIX в. В настоящее время этот формат научных статей стал универсальным стандартом, добровольно принятым большинством зарубежных и отечественных журналов. Тенденция к унификации структуры научных публикаций результатов оригинальных исследований стала особенно сильной с 1972 г., когда Национальный американский институт стандартов одобрил и рекомендовал всем научным журналам формат IMRaD, в основе которого лежит очень простая логика. Каждый раздел статьи отвечает на определенные вопросы. Первый – какой проблеме посвящено исследование? Ответ должен содержаться во Введении (Introduction). Следующий вопрос – как изучалась проблема? На него отвечает раздел Методы (Methods). Каковы основные находки или даже открытия? Ответ на этот вопрос содержится в разделе Результаты (Results). Что означают полученные результаты? Ответ – в разделе Обсуждение (Discussion). Кроме того, любая статья начинается с Заглавия (Title), за которым следуют сведения об авторах, включая место их работы, адреса, место выполнения представленного исследования. Затем следует Аннотация (Abstract), в которую входит характеристика основной темы, проблемы, объекта, цели исследования, ценность его результатов, а так же практическое значение итогов работы. В зависимости от требований научного журнала, Аннотация может быть структурированной либо

неструктурированной. Структурированная аннотация повторяет логику исследования и обычно имеет подзаголовки по той же структуре IMRaD: введение (Introduction), цель (Aims), методы (Methods), результаты (Results), заключение (Conclusion). Графическая аннотация представляет результаты исследования в визуальной форме (формулы, рисунок, график). После Аннотации следуют Ключевые слова (Keywords) и Основные положения (Highlights), освещающие наиболее важные результаты исследования. Далее начинается сама статья. В конце статьи, после раздела Обсуждение (Discussion), помещаются Благодарности (Acknowledgements) и Список Литературы (References). При необходимости и/или по требованию журнала, автор статьи может разместить Дополнительные материалы (Supplementary Materials) (Таблица1).

Таблица 1 – Структура научной статьи

<table>
<tr><td rowspan="11">Метаданные</td><td colspan="3">Заголовок</td><td colspan="3">Title</td></tr>
<tr><td rowspan="2" colspan="2">Сведения об авторах</td><td>Имя О. Фамилия</td><td rowspan="2" colspan="2">Information about authors</td><td>First Name Surname</td></tr>
<tr><td>Аффилиация</td><td>Affiliation</td></tr>
<tr><td colspan="3">Аннотация</td><td colspan="3">Abstract</td></tr>
<tr><td colspan="2">Неструктурванная</td><td>Структурированная</td><td colspan="2">Unstructured</td><td>Structured</td></tr>
<tr><td>Описательная</td><td>Информативная</td><td></td><td>Descriptive</td><td>Informative</td><td></td></tr>
<tr><td colspan="3">Графическая аннотация</td><td colspan="3">Graphical abstract</td></tr>
<tr><td colspan="3">Ключевые слова</td><td colspan="3">Keywords</td></tr>
<tr><td colspan="3">Основные положения</td><td colspan="3">Highlights</td></tr>
<tr><td rowspan="6">Текст статьи</td><td colspan="3">Введение</td><td colspan="3">**I**ntroduction</td></tr>
<tr><td colspan="3">Методы (Теоретические основы)</td><td colspan="3">**M**aterials and Methods (Theoretical basis)</td></tr>
<tr><td colspan="3">Результаты</td><td colspan="3">**R**esults</td></tr>
<tr><td colspan="3">Обсуждение</td><td colspan="3">**D**iscussion</td></tr>
<tr><td colspan="3">Заключение</td><td colspan="3">Conclusion</td></tr>
<tr><td colspan="3">Благодарности</td><td colspan="3">Acknowledgments</td></tr>
<tr><td>Метаданные</td><td colspan="3">Библиографический список</td><td colspan="3">References</td></tr>
</table>

В теле научной статьи должны быть ссылки на другие научные работы (библиографический список, внутритекстовые ссылки). Кроме того, в научной статье используются иллюстрации, описанные в тексте (таблицы, графики, схемы, диаграммы, рисунки, схематические чертежи, фотографии).

Исключения из правил оформления научных статей немногочисленны и не так существенны. Изредка (если методическая часть исследования занимает центральное место, например, когда в эксперименте намеренно использовалось несколько методов) можно объединить Методы и Результаты в один раздел – Эксперименты (Experimental). Среди немногих исключений отметим журнал Cell, в статьях которого с недавнего времени раздел Методы (Methods) стоит на последнем месте после раздела Обсуждение (Discussion). Такая организация статьи не отрицает общепринятого формата, а просто располагает разделы в ином порядке.

Структура IMRaD характерна для статей, посвященных оригинальным исследованиям. Значительное разнообразие в организации статей наблюдается в журналах, публикующих материалы описательного характера, такие как отчеты об

экспедициях, описания отдельных клинических случаев в медицине и т.д. IMRaD формат не используется для обзорных статей. Структура научной статьи подчиняется логике изложения материала и требованиям журнала, в котором планируется размещение данной публикации. Поэтому прежде чем приступать к написанию научной статьи, изучите правила для авторов (Guides for Authors) в выбранном журнале.

Рассмотрим особенности составных элементов научной статьи и основные требования, которые необходимо соблюдать при работе над ними. При этом необходимо иметь в виду, что метаданные статей: заглавие (Title); ФИО авторов (Byline); аффилиация (Affiliation); аннотация (Abstract); ключевые слова (Keywords); благодарности (Acknowledgements); списки литературы (References), обрабатываются (размечаются) в МНБД автоматически. Учитываются также данные сносок, если в них включены латинизированные библиографические ссылки, отсутствующие в списках литературы. Поэтому все перечисленные данные необходимо представлять в том порядке и по правилам, которые позволят их корректно обработать. Недостающие сведения об авторах (полные данные о ФИО, адрес организации и т.п.) могут быть взяты из раздела Информация об авторах/ Information about authors. Особое внимание необходимо обращать на представление фамилий авторов, аффилиации и списков литературы. Ошибки в библиографическом описании и выходных данных ссылок в списках литературы не позволяют правильно устанавливать связи между публикацией, включенной в МНБД, и ссылкой на нее. Цитирование в МНБД автоматически устанавливается по фамилии первого автора, названию журнала, году, номеру, страницам статьи (от-до). Только одна неправильно указанная буква или цифра может быть причиной потери ссылки, и исправить ее будет очень трудно.

Заглавие статьи (Title)

Максимальная длина заглавия статьи – 10–12 слов. Очень длинные заглавия, как и очень короткие трудно воспринимаются читателями.

Заглавие статьи должно быть: информативным, лаконичным, соответствовать научному стилю текста, содержать основные ключевые слова, характеризующие тему (предмет) исследования и содержание работы. Заглавие должно легко восприниматься читателями и поисковыми системами.

При переводе заглавия статьи на английский язык недопустимо использовать транслитерацию с русского языка на латиницу, кроме непереводимых названий собственных имен, приборов и др. объектов; также не используется жаргон, известный только русскоговорящим специалистам. Нежелательно использовать аббревиатуру и формулы.

Фамилии авторов (Byline)

В соответствии с принципами научной этики, авторами статьи могут являться те, и только те, кто сделал реальный вклад в исследование, отвечал за содержание рукописи, а также принимал участие в ее подготовке. Все правила, регламентирующие порядок упоминания авторов и определение авторства, согласовываются на начальных этапах подготовки текста.

Очередность упоминания авторов в большинстве случаев напрямую зависит от их вклада в выполненную работу. К примеру, в некоторых отраслях науки первым указывается автор, внесший наибольший вклад, остальные перечисляются по мере убывания их заслуг. Иногда первым указывается автор, выполнивший больше рутинной работы над статьей, а автор, руководивший исследованием, упоминается последним. Менее всего распространен вариант алфавитного перечисления авторов.

При формировании перечня авторов необходимо соблюдать этические нормы соавторства, разработанные COPE (Committee on Publishing Ethics, http://publicationethics.org) (см. Главу 4).

Первоначально выбранный вариант написания фамилии необходимо использовать всегда, во всех статьях. Не соглашайтесь ее менять по предложению журнала, желающего привести все метаданные статей к единой системе транслитерации. Для англоязычных метаданных важно соблюдать вариант написания сведений об авторе в последовательности: полное имя, инициал отчества, фамилия (Anna V. Ivanova). Такое написание также важно сохранять в англоязычных метаданных русскоязычных журналов. Не соглашайтесь, когда редакция опускает в написании ФИО инициала отчества, что свойственно для гуманитарных журналов. Этот инициал особенно важен при создании профилей авторов в МНБД, позволяя точно идентифицировать автора и не допускать ошибок при вливании в существующие профили. Отсутствие инициала отчества может быть причиной «потери» статьи для профиля автора.

Полное представление фамилии, имени и отчества в варианте, когда отчество стоит последним, является причиной ошибок в разметке данных статьи, которая производится в МНБД автоматически. В таких случаях профиль может быть создан на отчество (Vladimirovich, Mikhailovich).

При латинизации фамилии рекомендуется использовать вариант стандарта транслитерации для англоязычных систем (не немецкий и не французский), чаще применяется транслитерация стандарта BSI [1] . Можно воспользоваться системой

[1] BSI – Британский Институт Стандартов (British Standards Institution).

транслитерации на сайте http://translit.ru, при этом необходимо выбрать вариант стандарта, например, BSI.

Аффилиация (Affiliation)

В *Аффилиации* могут указываться названия и адреса (минимум – город, страна) как места основной работы автора, так и других организаций, к которым автор(ы) имел(и) отношение в период проведения исследования – например, организации, где проводились исследования в рамках конкретного проекта, или организация, с которой автор связан определенными обязательствами, относящимися к теме исследования. Таким образом, в аффилиации можно указывать несколько организаций.

При указании аффилиации необходимо придерживаться следующих общих рекомендаций [43]:

1) при выборе названия и адреса организации на английском языке предпочтительно использовать название и адрес, принятые уставом организации; чаще всего они указываются на сайте организации;

2) полный вариант аффилиации включает в себя почтовый адрес организации, название города, почтовый индекс, название страны. При написании адреса на английском языке необходимо следовать англоязычным правилам и указывать данные в следующей последовательности (учитывая знаки препинания): номер дома улица, город почтовый индекс, страна;

3) самый короткий приемлемый вариант аффилиации содержит названия организации, города и страны;

4) если в названии организации есть название города, в любом случае в адресных данных необходимо указывать город;

5) название организации и название ведомства следует приводить через запятую в именительном падеже, иначе статья может быть учтена только один раз и, вероятнее всего, отнесена к ведомству;

6) необходимо придерживаться унифицированного названия организации, как правило, зафиксированного в уставе организации и представленного на ее англоязычном сайте;

7) в англоязычной аффилиации не рекомендуется писать приставки, определяющие статус организации, например: «Федеральное государственное бюджетное научное учреждение» (Federal State Budgetary Institution of Science), «Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования», или аббревиатуру этой части названия (FGBNU, FGBOU VPO);

8) все составляющие аффилиации, в том числе названия факультетов, институтов внутри вузов, институтов Российской академии наук (РАН), федеральных исследовательских центров ФАНО и т.п., должны быть разделены между собой запятыми и пробелами;

9) в аффилиации необходимо давать полное название организации, без сокращений или аббревиатур; аббревиатура организации может быть указана после ее полного названия;

10) личные имена, включенные в название организации, на английском языке пишутся перед основным названием организации, а не после него. Инициалы фамилий можно указывать, но можно и опускать. Неприемлемо писать в названии организации с именем – … «named after».

Аннотация (Abstract)

Аннотация (абстракт, реферат, авторское резюме) включает характеристику основной темы, проблемы объекта, цели исследования, основные методы, результаты исследования и главные выводы. В аннотации необходимо указать, что нового несет в себе научная статья в сравнении с другими, родственными по тематике и целевому назначению [44–46]. Аннотация должна быть:

- информативной (не содержать общих слов);
- оригинальной (не быть калькой русскоязычной аннотации с дословным переводом);
- содержательной (отражать основное содержание статьи и результаты исследований);
- структурированной (следовать логике описания результатов в статье);
- «англоязычной» (быть написанной качественным английским языком);
- компактной (укладываться в объем от 150 до 250 слов).

Описательная неструктурированная аннотация содержит ключевые направления статьи, цель, данные, но, как правило, не детализирует методы, результаты и выводы.

Информационная неструктурированная аннотация информирует читателя об основных положениях статьи, кратко сообщает исходные данные, цель, методы, результаты, выводы и область применения результатов исследования.

Графическая аннотация отражает основные результаты исследования, представленного в научной статье, в виде единого графического изображения.

В зависимости от тематики различные издательства могут интерпретировать структуру статьи и аннотации. Например, издательство Emerald рекомендует для статей в журналах по экономике и бизнесу такую структуру аннотаций объемом до 250 слов:

Задача (Введение) – Причины/цели написания исследовательской работы;

Модель (Материалы и методы) – Методология/ как это было выполнено/ область исследования;

Выводы – Обсуждение/результаты;

Рамки исследования/возможность последующего использования результатов научной работы (если применимо) – Исключения/следующие шаги;

Практическое значение (если применимо) – Применение на практике/Что дальше?

Социальные последствия (если применимо) – Влияние на общество/политику;

Оригинальность/ценность – Кто сможет извлечь пользу из этой работы и что в ней нового?

Аннотация готовится после завершения статьи, когда текст написан полностью. Удобно писать структурированную аннотацию по структурированной статье, выбирая из каждого раздела самые важные сведения, которые в совокупности составят полное представление о содержании материала и позволят найти статью по основным терминам, включенным в аннотацию (вместе с заглавием и ключевыми словами).

В аннотацию не допускается включать ссылки на источники из полного текста, а также аббревиатуры, которые раскрываются только в полном тексте. Аббревиатуры и сокращения в аннотации должны быть раскрыты.

Необходимо понимать, что аннотация является основным и первоначальным источником информации о статье. Включенная вместе с другими метаданными во все ресурсы и в Интернет (на сайте журнала, в МНБД, в других информационных системах и т.д.), она «живет своей жизнью», отдельно от статьи, равно как и сама статья может существовать отдельно от журнала. Поэтому важно, чтобы аннотация вместе с другими метаданными давала возможность найти статью по более полному набору данных и ключевых слов (терминов, понятий), характеризирующих ее содержание.

Ключевые слова (Keywords)

Ключевые слова, составляющие семантическое ядро статьи, являются перечнем основных понятий и категорий, служащих для описания исследуемой проблемы. Эти слова служат ориентиром для читателя и используются для поиска статей в электронных базах, поэтому должны отражать дисциплину (область науки, в рамках которой написана статья), тему, цель и объект исследования.

В качестве ключевых слов могут использоваться как одиночные слова, так и словосочетания в единственном числе и именительном падеже. Рекомендуемое количество ключевых слов — 5–7 на русском и английском языках, количество слов внутри ключевой фразы — не более трех.

Основные принципы подбора ключевых слов:

- применяйте базовые (общеупотребимые) термины вместе со специальными;
- не используйте слишком сложные слова, слова в кавычках, слова с запятыми;
- каждое ключевое слово — это самостоятельный элемент, они должны иметь собственное значение.

Основные положения (Highlights)

Отражают ключевые результаты исследования, основное содержание статьи, изложенные тезисно и оформленные в виде 3–5 пунктов маркированного списка.

Введение (Introduction)

Введение – важная часть статьи, от его содержания зависит дальнейший интерес читателя к тексту. Введение должно «захватить», заинтересовать читателя.

В этом разделе описываются общая тема исследования, цели и задачи планируемой работы, теоретическая и практическая значимость, приводятся наиболее известные и авторитетные публикации по изучаемой теме, обозначаются нерешенные проблемы. Данный раздел должен содержать обоснование необходимости и актуальности исследования. Информация во Введении должна быть организована по принципу «от общего к частному».

Введение, как правило, состоит из четырех подразделов:

1. Описание проблемы, с которой связано исследование;
2. Обзор литературы, связанной с исследованием;
3. Описание белых пятен в проблеме или того, что еще не сделано;
4. Формулирование цели и задач исследования.

В первом подразделе необходимо представить, частью какой более широкой проблемы является представляемое исследование.

Второй подраздел посвящен обзору того, что и как было сделано другими исследователями в данной области. Основная часть подраздела содержит описание того, что опубликовано в статьях и книгах исследователей (и ваших собственных в том числе), если Вы на них опираетесь в представляемом исследовании.

В третьем подразделе вы показываете читателю, что обзор литературы закончен, и описываете важную область, в которой:

- исследования еще не проводились никем, потому что этот аспект проблемы был не замечен, пропущен или игнорирован;
- имеются противоречия или конфликты между результатами разных исследователей, гипотезами, выводами;
- необходимо продолжить или расширить исследования, так как их было недостаточно.

В четвертом подразделе формулируются цели и задачи исследования, которые зачастую переформулируются не один раз по мере того, как пишутся последующие разделы статьи: результаты и обсуждение.

Помимо вышеперечисленных подразделов, во Введении можно дать оценку важности проведенного исследования и кратко описать структуру публикации.

Методы и Материалы (Methods and Materials)

В этом разделе в деталях описываются методы, которые использовались для получения результатов. Обычно сначала дается общая схема экспериментов/исследования, затем они представляются настолько подробно и с таким количеством деталей, чтобы любой компетентный специалист мог воспроизвести их, пользуясь лишь текстом статьи.

При использовании стандартных методов и процедур лучше сделать ссылки на соответствующие источники, не забывая описать модификации стандартных методов, если таковые имелись. Если же используется собственный новый метод, который еще нигде ранее не публиковался, важно дать все необходимые детали. Если ранее метод был опубликован в известном журнале, можно ограничиться ссылкой. Однако рекомендуется полностью представить метод в рукописи, если ранее он был опубликован в малоизвестном журнале и не на английском языке.

Если в работе использованы химические или биохимические методы, то перечислите, какие реагенты и соединения применялись в эксперименте и какой степени чистоты они должны быть, за исключением стандартных лабораторных реактивов. Приведите химические названия и формулы соединений, которые являются новыми или нестандартными. Описывайте устройство использованных приборов и аппаратов только если они нестандартные или отсутствуют в продаже, либо вы их изготовили сами. Избегайте прямого указания торговых названий приборов и реактивов, хотя давать в скобках название компании-производителя и номер модели вполне допустимо. Для химических соединений используйте международные патентованные названия. Укажите, какие опасности имеют место при проведении данных экспериментов.

В исследовании биологического характера аккуратно идентифицируйте виды

растений, животных и микроорганизмов в соответствии с требованиями журнала. Если Вы имели дело с людьми, то обычно журналы требуют включить фразу об информированном согласии людей на участие в исследовании. Процедуры обычно описывают в хронологическом порядке.

В теоретической работе в разделе Theoretical Basis приводят математические выкладки с такой степенью подробности, чтобы можно было легко воспроизвести их и проверить правильность полученных результатов. Включите все необходимые данные, формулы, уравнения, назовите, какие преобразования над ними совершались. Если подробное описание математических преобразований занимает слишком много места, то можно привести их в приложении к статье.

Статистические процедуры представляются очень кратко, поскольку в большинстве случаев используются либо хорошо известные способы статистического анализа, либо их модификации. Стандартные статистические процедуры просто называются, ссылка на источник нужна только, если используются необычные или модифицированные методы.

Результаты (Results)

В этом разделе представлены экспериментальные или теоретические данные, полученные в ходе исследования. Результаты даются в обработанном варианте: в виде таблиц, графиков, организационных или структурных диаграмм, уравнений, фотографий, рисунков. В этом разделе приводятся только факты. Их интерпретацию, сопоставление с данными других исследователей оставьте для раздела Обсуждение. Если было получено много похожих зависимостей, представляемых в виде графиков, то приведите только один типичный график, а данные об имеющихся количественных отличиях между ними, представьте в таблице.

Существует три способа представления результатов:

• текст (вербальное представление);

• таблицы (полувербальное представление);

• рисунки: диаграммы, графики, изображения (визуальное представление).

Все три способа представления результатов количественного исследования (текст, таблицы и рисунки) должны дополнять, а не повторять друг друга. Каждый график, каждая таблица должны быть представлены и описаны в тексте. Обычно текстовое описание графиков также состоит из трех элементов. Первый указывает, что именно представлено в виде графика, и где это можно найти в статье. Второй описывает наиболее важные черты этого графика, а третий уже комментирует. Обычно текстовое описание графиков также состоит из трех элементов.

В руководстве для авторов журналы подробно описывают, как должны выглядеть иллюстрации: размер, оформление, а также формат, в котором они должны быть представлены.

Обсуждение (Discussion)

Раздел Обсуждение содержит интерпретацию полученных результатов исследования, предположения о полученных фактах, сравнение полученных собственных результатов с результатами других авторов. В Обсуждении вы двигаетесь от специфической информации разделов Методы и Результаты к более общей интерпретации результатов. В разделе можно:

- перечислить основные результаты, независимо от того, поддерживают или опровергают они проверяемую гипотезу, находятся в согласии или в противоречии с данными других исследователей;
- обобщить результаты;
- сравнить результаты с данными других исследователей;
- привести возможные объяснения сходства и противоречий с другими исследованиями;
- напомнить о цели и гипотезе исследования;
- обсудить соответствуют ли полученные результаты гипотезе исследования;
- указать на ограничения исследования и обобщения его результатов;
- предложить практическое применение;
- предложить направление для будущих исследований.

Заключение (Conclusion)

Заключение содержит главные идеи основного текста статьи. Эту часть раздела надо тщательно отредактировать, чтобы не повторять формулировок, приведенных в предыдущих разделах. Желательно сравнить полученные результаты с теми, которые планировалось получить, а также показать их новизну и практическую значимость, прописать ограничения, с которыми столкнулись в ходе работы. В конце приводятся выводы и рекомендации, определяются основные направления дальнейших исследований в данной области.

Благодарности (Acknowledgements)

В данном разделе принято выражать признательность коллегам, которые оказывали помощь в выполнении исследования или высказывали критические замечания в адрес вашей статьи. Однако прежде чем выразить благодарность, необходимо заручиться согласием тех, кого планируете поблагодарить.

Если вы использовали в работе нестандартное оборудование и материалы, то можно также перечислить, на каком и чьем специальном оборудовании выполнялись эксперименты, а также перечислить источники всех других специальных материалов и объектов исследования (культур, животных).

Необходимо выразить благодарность за финансовую поддержку исследования организациям и фондам, т.е. написать за счет каких грантов, контрактов, стипендий удалось провести исследование (This work was supported by the Russian Foundation for Basic Research, project no. 94-02-04253a).

Список использованных источников (References)

«Списки литературы – сырье для анализа цитирования» (*«Reference lists are the raw material for carrying out citation analyses»*, так сказал когда-то Юджин Гарфилд, создатель Институт научной информации США и Web of Science (Science Citation Index)) [47–48].

Научная статья должна содержать ссылки на информацию, полученную из конкретного источника (внутритекстовые ссылки), а также библиографический список этих источников в конце статьи. Списки литературы позволяют:

– признавать и использовать идеи других авторов, избежав обвинений в плагиате;

– читателю быстро найти источники материалов, на которые ссылается автор, для ознакомления с ними, и чтобы убедиться в достоверности данных из этих источников;

– демонстрировать масштаб и глубину исследования (цитирование своих предыдущих публикаций).

Задача авторов представить ссылки в списке литературы так, чтобы можно было:

– установить связку между публикацией в МНБД и ссылкой на нее;

– понять смысл ссылки англоязычному пользователю.

Цитирование в тексте статьи и списки литературы выполняются точно по требованиям журналов, однако необходимо понимать важность работы с этой частью статьи и знать основные правила работы с источниками. Изучая требования журнала к спискам литературы, необходимо обладать общей культурой цитирования (см. § 3.2).

В списки литературы включаются только источники, использованные при подготовке статьи. На все источники в тексте должны быть даны ссылки. Список цитируемых источников наряду с заглавием, аннотацией и введением относится к основным частям статьи, по которым редакторы и читатели определяют к ней свой первичный интерес.

При написании научных статей и обзоров используют различные стили цитирования источников в текстах статей и порядок расположения ссылок в списке литературы. Стили дают рекомендации по расположению и оформлению ссылок в тексте

публикаций и в списках литературы. В международных журналах каждой научной дисциплиной отдается предпочтение определенным стилям Однако, основными стилями, лежащими также в основе и других стилей, являются, так называемые, «Ванкуверский» (Vancouver Style, цифра по порядку следования ссылки в тексте, список литературы по порядку этих цифр) и «Гарвардский» (Harvard Style, «фамилия первого автора – год выхода» ссылки в тексте, алфавит – в списке литературы) (см. § 3.2).

Оформление библиографического списка в российских изданиях регламентируется государственными стандартами на библиографические описания. В русскоязычных журналах для составления списков литературы, как правило, используются ГОСТы на библиографические описания (ГОСТ 7.1–2003; ГОСТ Р 7.0.5–2008; ГОСТ 7.82–2001). Необходимо при этом учитывать, что ГОСТы носят рекомендательный характер, поэтому в русскоязычных журналах чаще используются зарубежные стандарты, более приемлемые при обработке списков литературы для баз данных цитирования, в т.ч. и в РИНЦ.

Выбранный вами стиль не должен отличаться от используемого в журнале, в который вы намерены подать рукопись. Следует заметить, что основным различием того или иного стиля являются принципы оформления именно библиографического описания. На сайте Zotero Style Repository (https://www.zotero.org/styles) приведены библиографические стили более 8 тыс. журналов. Этой информацией можно воспользоваться при ознакомлении с журналом, если он включен в этот перечень.

3.2. Культура цитирования и основные требования к использованию источников, цитированию и составлению списков литературы

Появление новых идей и открытий является отражением научного прогресса. Именно цитаты в научных работах связывают воедино концепции, технологии и достижения, которые определяют научные направления исследований. ***Цитирование*** – это заимствование фрагментов текстов (формул, иллюстраций, таблиц и других элементов) автором в своей работе из других источников с обязательным указанием источника, в том числе, информации об авторах, названии работы, выходных данных журнала/издательства и т.д. Цитирование является обязательным компонентом любой научной работы и одним из важных средств научной коммуникации. Цитирование:

– отсылает читателя к первоисточнику и позволяет подробно ознакомиться с основополагающими идеями научной работы;

– цитаты усиливают научную работу, предоставляя поддержку авторитетных ученых;

– качество и количество ссылок отражает качество и глубину исследования;

– не все источники дают достоверную информацию, что можно отразить при цитировании, предложив более точные или интересные идеи.

Авторы обязаны соблюдать этические, моральные и правовые нормы при цитировании. Читатель должен быть четко информирован о том, что является оригинальным материалом, а что переработанным из других источников. Ссылки на первоисточники дают возможность найти соответствующие источники, проверить достоверность цитирования, получить необходимую информацию. Использование библиографических ссылок в научных работах обязательно и употребляется в следующих случаях:

– при цитировании фрагментов текста, формул, формулировок, идей, таблиц, иллюстраций;

– при заимствовании положений, формул, формулировок, идей, таблиц, иллюстраций и т.п. не в виде цитаты;

– при перефразированном, недословном воспроизведении фрагмента чужого текста;

– при анализе в тексте содержания других публикаций;

– при необходимости отсылки читателя к другим публикациям, где обсуждаемый материал дан более полно.

Отсутствие ссылки ведет к нарушению авторских прав, поэтому ссылка на первоначальные источники является единственным легитимным способом использования чужих материалов. Ссылка на первоначальные источники помогает подчеркнуть оригинальность вашей собственной работы. Но необходимо помнить, что не меньшее внимание уделяется качеству цитируемых источников. Основным требованием к приводимым в научной работе источникам является их авторитетность и соответствие исследуемой тематике. Поэтому необходимо обращать внимание на научную квалификацию авторов, авторитетность журнала, в котором опубликована статья, год издания. При проведении анализа научной проблемы необходимо показать знакомство с классическими трудами, сославшись в работе на соответствующие источники. О наиболее известных научных трудах в исследуемой области можно получить информацию в справочной и учебной литературе, в библиографиях других научных статей и монографий.

В научных работах выделяют следующие виды цитирования:

Прямое цитирование

Прямое цитирование – это дословное воспроизведение отрывка из чужого текста.

Общие требования к прямому цитированию:

1. Текст цитаты заключается в кавычки и приводится в той грамматической форме, в какой он дан в источнике, с сохранением особенностей авторского написания.

2. Цитирование должно быть полным, без произвольного сокращения цитируемого текста и без искажений мысли автора.

3. Требования к форматированию длинных цитат, различаются в зависимости от стиля цитирования. В целом, если цитируемый материал занимает более трех строк, то необходимо придерживаться следующих правил:

– изменить шрифт на меньший (в документе, в котором основной текст имеет шрифт размером 12 пт, необходимо использовать шрифт в 10 пт);

– двойной отступ слева от страницы для всех строчек цитаты;

– не использовать кавычки для всей цитаты – сделанных графических изменений (изменение шрифта, двойной отступов и т.д.) достаточно, для того, чтобы указать, что материал копируется.

4. При цитировании каждая цитата должна сопровождаться ссылкой на источник, библиографическое описание которого приводится в соответствии с требованиями к оформлению списка использованных источников. Необходимо помнить, что обилие прямых цитат на каждой странице, следование цитат друг за другом без должного авторского анализа производит впечатление несамостоятельности работы. Поэтому при цитировании необходимо предоставлять материал, строго соответствующий идеям научной работы. Можно изменить формулировку или слова цитаты с целью ее сокращения, но при этом не должен меняться смысл. В этом случае используются специальные символы редактирования: при сокращении цитаты – многоточие, при добавлении поясняющих слов в прямую цитату – они заключаются в квадратные скобки.

Парафраз или пересказ

Кроме полных цитат, в научной работе широко распространен такой вид цитирования, как *парафраз*. Парафраз используется в случаях, когда необходимо представить краткое изложение объемной теоретической концепции или обобщенную информацию при ссылке на несколько авторов или источников информации.

Шесть шагов для эффективного написания парафраза (https://owl.english.purdue.edu/owl/resource/619/1/):

– перечитать первоначальный источник, пока не станет ясен его полный смысл;

– отложить оригинал в сторону, и написать свой пересказ;

– написать ключевые слова вашего пересказа;

– сопоставить с оригиналом, чтобы убедиться, что пересказ точно выражает идею и всю необходимую информацию источника;

– использовать кавычки для идентификации любого уникального термина, который заимствуется из источника;

– записать выходные данные источника для включения материала в работу.

Примеры правильного написания парафраз можно найти на сайте https://owl.english.purdue.edu/owl/resource/619/1/

Наряду с цитатами и парафразом, можно также выделить *резюмирование*. Данные виды цитирований представляют основные инструменты для интеграции чужих материалов и источников в вашу научную работу. При выборе вида цитирования необходимо отталкиваться от дисциплины и типа научной работы. Например, в обзоре литературы почти всегда используется резюмирование. Научные очерки, напротив, полагаются на все три инструмента. Парафраз и резюмирование незаменимы в научных работах, потому что они позволяют включать идеи других людей, важные для вашей научной работы, использовать достижения и подходы других исследователей без дословного цитирования. Важно четко понимать, какие именно мысли из цитируемого источника важны для *вашей* аргументации. Хотя пересказ и резюмирование предпочтительнее прямой цитаты, не стоит слишком сильно увлекаться ими, ваши идеи – важнее всего.

Цитирование по вторичным источникам

Цитирование по вторичным источникам возможно только на этапе знакомства с темой и проблематикой исследования, а также для определения понятийного аппарата работы. Все цитаты, которые используются подобным образом, должны быть тщательно выверены по первичным источникам. Также нужно быть уверенным в том, что во вторичном источнике не было допущено ошибок.

Случаи, в которых возможно цитирование по вторичному источнику:

– первоисточник утерян или недоступен (например, находится в закрытых архивах или библиотеках);

– первоисточник написан на сложном для перевода языке;

– текст цитаты известен по записи слов их автора в воспоминаниях других лиц;

– цитата приводится для иллюстрации хода мыслей и аргументации автора.

Кроме явных ссылок, указанных в списке литературы, существуют *неформальное цитирование* и *скрытое цитирование*. Скрытое цитирование состоит в использовании идей без прямой ссылки на ее автора, но с возможностью идентификации первоисточника через цепочку цитирований. В истории науки есть много примеров, когда концептуальные

статьи цитируют реже, чем работы, модифицировавшие их. Неформальное цитирование состоит в указании источника информации в тексте работы без включения его в список литературы. Например, в тексте даны только фамилии и инициалы авторов или использованы эпонимы, например, геометрия Лобачевского, распределение Вейбула–Гнеденко, принцип Беллмана–Заде и т.п. Часто используются термины без связи с фамилией автора, например, «метод наименьших квадратов или задача о Кенигсберских мостах».

Самоцитирование

Ранее опубликованные исследования автора могут являться источником цитаты. Такой вид цитирования позволит избежать дублирования информации и самоплагиата, а также поможет направить заинтересованного читателя к предыдущим и связанным работам. Необходимо помнить, что цитирование собственных работ должно быть уместным и обоснованным, дополнять научную работу и следовать ее задачам. Стремление искусственно завысить данные цитирования собственных работ может привести к обратному результату. Собственные цитаты должны быть оформлены по всем правилам цитирования.

Взаимное цитирование

Исследования показывают, что ученые, ссылающиеся на работу своих коллег, вероятнее всего найдут свою собственную работу в их ссылках. Этот эффект популярен и позволяет «накручивать» ссылки на статьи отдельных авторов и журналов. «Существует до смешного тесная взаимосвязь между количеством цитирований и количеством ссылок, – пишет Г. Вебстер, психолог из Университета Флориды в Гейнсвилле, занимающийся исследованиями *природы*, – если вы хотите получить больше цитируемости, ссылайтесь на большее количество авторов» (http://www.nature.com/news/2010/100813/full/news.2010.406.html).

Как не совершать ошибки при цитировании?

Несмотря на лаконичность и однозначность правил цитирования, периодически авторы научных работ допускают ошибки. В Таблице 2 представлены основные рекомендации по цитированию при написании научной работы.

Таблица 2 – Рекомендации по цитированию

Виды копирования	Этичность	Как избежать ошибок
Прямое цитирование	Дословное копирование приемлемо, если вы ссылаетесь на источник и ставите кавычки вокруг скопированного текста.	■ указывайте источники, которые использовались во время написания работы ■ убедитесь, что вы полно и правильно процитировали оригинал ■ используйте кавычки при копировании «слово-в-слово» и ссылки
Частичное копирование	Как правило, касается графиков, методов/методик, таблиц или рисунков из чужих материалов. Они должны быть процитированы.	■ используйте ссылки на оригинал
Парафраз	Пересказ приемлем, если вы правильно ссылаетесь на источник и точно передаете смысл и идеи источника материала.	■ убедитесь, что вы понимаете первоначальную идею автора ■ никогда не копируйте и не вставляйте слова, которые вы не в полной мере поняли ■ сравните ваш пересказ с источником, чтобы убедиться, что вы сохранили предполагаемый смысл и основную идею источника

Цитирование в зависимости от типа научной работы и области исследования

Правила цитирования зависят от типа работы (научная статья, монография, депонирование и другие) и того, как используется заимствованный материал.

Во-первых, необходимо определить важность источника для научной статьи. Если источник занимает центральное место в работе, необходимо дать о нем информацию в отдельном предложении, указав фамилию автора, приводя его важность и основные идеи. Также можно дать справку по автору(ам), если есть основания полагать, что читатель не знает его(их). В противном случае, можно использовать вводный список литературы или сноски.

Во-вторых, существуют различные формы цитирования для различных областей науки: в социальных науках будет использована одна форма цитирования, в естественных науках – другая. Для определения наилучшей формы необходимо проконсультироваться с научным руководителем.

Цитирование в естественных, технических и медицинских науках

Научные достижения в естественных, технических и медицинских науках в большинстве случаев не зависят от их точной формулировки: теории, теоремы, исследования, экспериментальные результаты и т.д., как правило, цитируются косвенно (парафраз). В тексте указываются короткие цитаты в цифровой («Ванкуверской») или в «автор-год» («Гарвардской») системах. Сносок обычно не делается. Прямое цитирование является редким и должно использоваться только тогда, когда значение имеет точная

формулировка. Данные из цитат не всегда принято помещать в кавычки, например, если вы «слово-в-слово» цитируете формулу или математическую теорему.

При цитировании, в тексте работы вы можете упомянуть автора.

Примеры:

Из исследования К. Гаусса и Л. Эйлера [7] известно, что ...

Поскольку решения этого уравнения всегда ограничены [8, теорема 1.7], то отсюда следует, что ...

Как уже упоминалось, необходимо как можно более точно указать информацию об источнике, т.е. вставить номера страниц, формул или другую нумерацию (как в последнем примере).

Цитирование в социальных и экономических науках

Пересказ/парафраз распространены в области гуманитарных и социальных наук. Важно правильно переформулировать исходный материал, а не просто изменить несколько слов в процитированном отрывке.

Пересказ/парафраз следует использовать с осторожностью, когда воспроизводится точная формулировка, аргументация или мнение. Можно также использовать и прямые цитаты в научной работе с помощью форматирования, например, отступов (влево или вправо) или курсивом. В этом случае не нужно использовать кавычки. Необходимо воспроизвести цитату точно и показать сокращения многоточием.

Для выделения цитат в тексте существует две основные системы, являющиеся общими для гуманитарных и социальных наук: 1) короткая цитата с использованием системы автор-год; 2) сноска.

Зарубежные библиографические стили (стандарты на библиографические описания)

При оформлении списков литературы в зарубежные журналы, а также в российские журналы, включенные в МНБД, в части списков литературы, представляемых на латинице (References), используются различные международные библиографические стили (иногда называются «стандартами на библиографическое описание»). Стили могут отличаться для журналов различных дисциплин. Разработчиками таких стандартов, как правило, являются американские научные общества и ассоциации, они же – крупные издательства, ведущие американские университеты или специальные библиотеки. В Таблице 3 приведен перечень таких стилей. Scopus включает в опцию «Create bibliography» («составить библиографический список») 10 вариантов стилей. В Таблице 3 стили, используемые в Scopus, отмечены звездочкой (*).

Таблица 3 – Основные международные стили оформления статей

Область научных исследований	**Стиль цитирования и библиографических описаний**
Биомедицина	Vancouver Style*
Биология, сельское хозяйство	Council of Biology Editors (CBE) Style*
Геология	GSA (Geological Society of America) Style
Гуманитарные науки, междисциплинарные исследования	MLA (Modern Language Association)*
Гуманитарные науки, Искусство	MHRA (Modern Humanities Research Association)
Гуманитарные науки Социальные науки	Harvard Citation Style (Harvard Referencing)*
Гуманитарные, естественные, социальные, исторические науки	Chicago (Turabian) Style (CMOS)*
Математика	AMS (American Mathematical Society) Style
Машиностроение	ASME (American Society of Mechanical Engineers) Style
Медицина	AMA (American Medical Association) Style
Медицина	NLM (National Library of Medicine) Style*
Социальные науки, Психология	APA (American Psychological Association) Style*
Политические науки	APSA (American Political Science Association) Style
Сельскохозяйственные науки, Биоинженерия	ASABE (American Society of Agricultural and Biological Engineers) Style
Социологические науки	ASA (American Sociological Association) Style
Управление персоналом, финансы и бухгалтерия	AMA (American Management Association) Style
Физика	AIP (American Institute of Physics) Style
Химия, Физика	ACS (American Chemical Society) Style
Электроника и информатика	IEEE (Institute of Electrical and Electronics Engineers, Inc) Style
Юриспруденция	ALWD (Association of Legal Writing Directors) Style

* – Стили, включенные в Scopus в качестве рекомендуемых при составлении в системе списков литературы

Правила оформления всех описанных выше систем (автор-год, сноска и др.) приводятся в стандартизированных правилах стилей цитирования, которые можно разделить на общие для:

– гуманитарных наук,

– технических, естественных и медицинских наук,

– социальных наук.

Некоторые стили, такие как Chicago Style, достаточно гибки. Другие, такие как MLA и APA, задают форматы в контексте единой системы цитирования. Различные стили оформления задают порядок появления, например, даты публикации, названия и номера страниц, использование курсива, круглых скобок, кавычек и т.д., характерные для данного стиля.

Гуманитарные науки

- Chicago Style (CMOS) –стандарт широко используется в истории и экономике, а также в некоторых социальных науках. Краткое описание данного стиля приведено здесь: http://www.chicagomanualofstyle.org/tools_citationguide.html
- Harvard referencing представляет собой особый вид *вводных ссылок*. Данный стандарт рекомендован Британским институтом стандартов (British Standard Institute) и Ассоциацией современного языка. Harvard включает краткое упоминание «автор-дата» (Smith, 2000), вставляется после цитируемого текста в скобках и полной ссылкой на первоисточник в конце статьи.
- MLA стиль разработан Ассоциацией современного языка и наиболее часто используется в гуманитарных науках и некоторых междисциплинарных исследованиях. Этот стиль использует вводные ссылки с автором и страницей (Smith 395) или автором, коротким названием и номером страницы (Smith, *Contingencies* 42). https://www.mla.org/MLA-Style
- Стиль MHRA разработан Гуманитарной Ассоциацией современных Исследований (Modern Humanities Research Association) и наиболее широко используется в искусстве и гуманитарных науках в Великобритании и Соединенных Штатах. Он похож на стиль MLA, но имеет некоторые отличия. Например, стиль MHRA использует сноски, ссылающиеся на цитату полностью, а также предоставляет библиографию. Некоторые читатели сочтут целесообразным, что сноски обеспечивают полное название, вместо укороченных ссылок, так что им не нужно обращаться к списку библиографии в конце текста во время чтения. http://www.mhra.org.uk/series/MSG

Технические, естественные и медицинские науки

- Американское химическое общество применяет стиль ACS , часто используется в химии и физике. В ACS ссылки пронумерованы в тексте и в списке литературы, номера повторяются по всему тексту по мере необходимости.
- Стиль AMS-LaTeX, разработанный Американским математическим обществом (AMS), как правило, реализуется с помощью BibTeX инструмента в LaTeX среде. Скобки с инициалами автора и год вставляются в текст в начале ссылки, например, [AB90]. http://www.ams.org/publications/authors/tex/amslatex
- Система Vancouver, рекомендованная Council of Science Editors (CSE), используется в медицинских научных работах и исследованиях. https://www.councilscienceeditors.org/
- Стиль Института инженеров по электротехнике и электронике (IEEE), или IEEE стиль, замыкает порядковый номер цитирования в квадратные скобки, похож на

Vancouver.
http://www.ieee.org/publications_standards/publications/journmag/IEEE_style_manual.pdf

Социальные науки

- Стиль Американской психологической ассоциации (стиль APA) наиболее часто используется в социальных науках. Он похож на стиль Harvard, хотя может иметь две формы: *название цитаты*, в которых фамилии авторов появляются в тексте и год издания, и *автор – дата цитаты*, в которых все фамилии авторов и год издания указывается в скобках. В обоих случаях в тексте цитаты указывается ссылка на алфавитный список источников в конце статьи. http://www.apastyle.org/
- Стиль ASA из Американской социологической ассоциации является одним из основных стилей, используемых в социологических публикациях. https://owl.english.purdue.edu/owl/resource/583/1/

Рекомендации к составлению списка литературы в журналы, индексируемые в МНБД

Чтобы ссылка «нашла» свою публикацию в МНБД, а также и в РИНЦ, т.е. цитирование было учтено:

– необходимо в список литературы включать только полное описание источника (статьи из журнала, доклада конференции, главы из книги) с обязательным указанием первой и последней страницы публикации (страница первая – страница последняя); ссылки на конкретные цитируемые страницы указываются во внутри-текстовых ссылках, например:

Для русскоязычных статей («Список литературы»):

1. Дедов И.И., Шестакова М.В., Викулова О.К. Государственный регистр сахарного диабета в Российской Федерации: статус 2014 г. и перспективы развития // Сахарный диабет. – 2015. – Т. 18. – N 3. – С. 5–22. DOI: 10.14341/DM201535-22

Для списков литературы на латинице («References»):

1. Dedov II, Shestakova MV, Vikulova OK. National register of diabetes mellitus in Russian Federation. Diabetes mellitus. 2015;18(3):5-22. (In Russ). DOI: 10.14341/DM201535-22

Внутри текста:

«Ванкуверский» стиль: [1] – цитирование всей статьи; [1, с. 15] или [1, p. 15] – при цитировании конкретных данных на конкретной странице;

«Гарвардский» стиль (список литературы не нумерован): (Дедов И.И. и др., 2015) – цитирование всей статьи; (Дедов И.И. и др., 2015, с. 15) или (Dedov II e.a., 2015, p. 15) – при цитировании конкретных данных на конкретной странице.

– в списке литературы не рекомендуется повторять один и тот же источник (статью, монографию) несколько раз, указывая в каждом описании разные страницы («Там же, с.

34» или «Ibid, p. 34»). Такое дублирование вводит в заблуждение о количестве использованных источников и не позволяет связать ссылки и публикацию, включенную в МНБД (цитирование (связка) устанавливается автоматически по полным выходным данным);

– если журнал дает рекомендации по цитированию своих статей (Для цитирования/ For citation), следует брать информацию из этой части. При отсутствии таких рекомендаций, необходимо использовать выходные данные статей, размещаемые в колонтитулах или в других местах в теле статьи (название журнала, год, том, номер, страницы). Использование этих данных позволит избежать ошибок при цитировании;

– если цитируемая статья имеет DOI, необходимо указывать его после описания цитируемой статьи;

– нежелательно включать в списки литературы анонимные источники и нормативные документы (постановления, законы, инструкции и т.д.), которые никогда не будут проиндексированы в базах данных цитирования, предпочтительно их цитировать непосредственно в тексте или во внутритекстовых сносках;

– нежелательно использовать в списках литературы труднодоступные, неопубликованные, малотиражные, а также локальные, популярные и образовательные источники: авторефераты диссертаций и диссертации, газеты, неопубликованные отчеты, учебные пособия и учебники;

– из авторефератов и диссертаций желательно использовать и цитировать опубликованные источники; особенно это касается статей из журналов, которые должны быть доступны всем, в т.ч. зарубежному читателю;

– не допускается делать произвольные сокращения названий журналов. Если в журнале отсутствует указание на сокращенное название, рекомендуется проверить наличие рекомендуемых сокращений в списке, разработанном ISSN центром: List of Title World Abbreviations (LTWA) http://www.issn.org/services/online-services/access-to-the-ltwa/; если нет рекомендаций по сокращению слов, следует давать их полное написание; крайне не рекомендуется давать названия журналов аббревиатурой, если она не предусмотрена журналом.

В дополнение к сказанному отметим, что при составлении списков литературы на латинице (для статей в зарубежные журналы или в российские, индексируемые в МНБД) необходимо:

– в список литературы включать данные из англоязычных метаданных статьи, которые размещены одним блоком на титульной странице или в конце статьи: ФИО авторов на латинице; заглавие статьи на английском языке; название журнала на латинице

(транслитерация, если нет информации об использовании журналом англоязычного названия); выходные данные (год, том, номер страницы «от-до»); указание на язык статьи, если она представлена на русском языке (In Russ.); DOI статьи (при наличии) или URL при отсутствии DOI, если есть доступ к статье;

– если в соответствии с используемым в журнале стилем выходные данные статей приводятся словами (том, номер и страницы), их надо указывать сокращениями английских слов, а не транслитерацией: vol., No (no, N) или #, pp. Также это касается выходных данных для книг (город, страницы): Moscow, 2015, 230 p. Транслитерируется только собственное название издательства, но слово «издательство» пишется в сокращении на английском (Nauka Publ.);

– не допускается делать только транслитерацию описания, без перевода заглавия статьи или названия книги на английский язык: описание должно быть прочитано иностранным читателем, он должен понять его смысл; также при цитировании русскоязычных книг, материалов конференций, анонимных и других источников, о которых заведомо известно, что они никогда не будут включены в МНБД, рекомендуется делать перевод их названий без его транслитерации, указывая после описания на язык издания (In Russ.);

– описание книги может быть дано в двух вариантах: 1) название книги приводится на латинице (транслитерация) с указанием параллельного перевода названия на английский, в квадратных скобках после транслитерированного; 2) дается перевод названия книги на английский с указанием в конце описания языка книги (In Russ.);

Если книга представляет собой перевод иностранной монографии, желательно давать два варианта описания книги в одной ссылке: сначала оригинальное, затем – переводное (с указанием «Rus. Ed.:»). Если данные об оригинальном издании отсутствуют, необходимо при описании переводного найти правильное написание фамилий иностранных авторов, не транслитерировать их произвольно;

– при заимствовании для цитирования источников из других цитируемых статей, что допускается в крайнем случае при отсутствии возможности доступа к источнику (см. выше), необходимо проверять существование источников и корректность их данных, делая поиски в Интернете, на платформе издательства и в библиотеке. Если данные об источнике в сети отсутствуют, или встречаются только вторичные ссылки, включать такой источник в список литературы не рекомендуется.

Если статья публикуется на русском языке в журнале, включенном в МНБД, необходимо учитывать и выполнять требования журнала к составлению списков литературы в двух вариантах: на кириллице (если есть русскоязычные источники) и

латинице (русскоязычные источники должны быть представлены на латинице). При этом в англоязычном варианте списка литературы (References) должен быть использован один из принятых в МНБД библиографических стилей или подобный (российский ГОСТ в МНБД обрабатывается некорректно) [49–50].

Списки литературы являются одной из основных частей статьи, которую проверяют ответственный секретарь журнала при приеме рукописи. Секретарь журнала Издательства Elsevier на это тратит 75% своего времени [12], так как от качества ссылок зависит отношение читателей к журналу и авторитет журнала в научном сообществе, определяемом по показателям его цитирования в МНБД. Плохое качество списка литературы может стать причиной отказа от приема рукописи к дальнейшему рассмотрению. К «плохому качеству» с точки зрения редакторов и МНБД, прежде всего, относятся:

– наличие ошибок в описании: пропуск первого автора; перестановка порядка фамилий авторов; не соблюдение пунктуации принятого стиля; не полные выходные сведения (отсутствие указания на страницы статьи); не правильное указание первой и последней страницы статьи и т.д.;

– преобладание ссылок на малоизвестные источники, отсутствующие в МНБД, на недоступные источники (с точки зрения редакторов); наличие источника в институтском репозитории для англоязычного редактора не является основанием считать его доступным, особенно если в описаниях не сделаны ссылки на сайты с полными текстами;

– преобладание русскоязычных ссылок на источники, отсутствующие в МНБД; в зависимости от тематики, их доля может быть разной, однако даже при публикации результатов региональных гуманитарных и социальных исследований необходимо проверять наличие в МНБД источников по теме; это могут быть как российские, так и зарубежные источники.

Инструменты для подготовки списков литературы

Стили определяют не только правила оформления ссылок в тексте, но и правила составления библиографических описаний на все виды документов, которые могут быть отражены в списке литературы. Важно, чтобы в списках литературы соблюдались все правила описания источников в соответствии с принятым стилем (стандартом на библиографическое описание). Корректно оформить цитирование в тексте статьи позволяют как сервисы внутри МНБД, так и специально разработанные программы управления ссылками.

Опция «Create Bibliography» в Scopus. Этой опцией Scopus предлагает автоматически создать и сохранить (выгрузить) библиографический список найденных и отобранных из БД источников по одному из десяти предлагаемых в этой БД стилей.

Основными программами, используемыми для цели управления библиографическими ссылками, в настоящее время являются **Mendeley** (поддерживается Издательством Elsevier) и **EndNote** (разработка Компании Thomson Reuters, с 2016 г. поддерживается компанией Clarivate Analytics).

Программы управления ссылками значительно облегчают цитирование литературы в тексте рукописи, позволяя:

– брать уже готовое библиографическое описание нужной работы с сайта журнала или из библиографической БД открытого или платного доступа;

– создавать собственную библиотеку цитируемой литературы;

– изменять стиль цитирования, использованный в рукописи, на любой другой, использовать алфавитный или последовательный порядок, создавать собственный стиль цитирования или загрузить нужный с сайта журнала;

– автоматически изменять нумерацию ссылок в тексте и нумерацию источников в списке литературы при добавлении или удалении ссылок, избавляя автора от трудоемкой работы ручных правок и перепроверки всего списка;

– обращаться к аннотации цитируемого источника и к его полному тексту (при наличии) [26].

Программы полностью интегрированы в текстовый редактор Microsoft Word и позволяют включать, исключать и редактировать ссылки в процессе работы над текстом публикации.

3.3. Особенности написания научных статей на английском языке

Отличительная черта научного стиля – академическое изложение, адресованное специалистам. Признаки научного стиля – точность передаваемой информации, убедительность аргументации, логическая последовательность изложения, лаконичность, абстрактность, скрытая эмоциональность, авторитетность.

Написанная хорошим английским языком научная статья является ключом к успешной публикации и индексации в престижных библиографических базах данных. Для российских исследователей написание статьи на английском языке вызывает определенные сложности, объясняемые отчасти тем, что риторика научного текста, написанного на русском языке, значительно отличается от англосаксонской риторики.

Одно из основных правил английской риторики заключается в том, что вся ответственность за понимание текста статьи лежит на авторе. Автор так должен выстроить

текст статьи, каждого раздела, каждого параграфа, чтобы читателю не пришлось ломать голову над тем, что он имел в виду. Главная задача читателя – впитывать информацию, не прилагая усилий для понимания текста.

Англо-американская риторика подразумевает, что читатель не должен тратить свое драгоценное время, читая описание научной области и всех предыдущих исследований, дожидаясь, когда автор статьи сочтет нужным сообщить, какую именно проблему он будет освещать в рамках своей исследовательской статьи. Автор не может произвольно использовать слова или идеи другого ученого без объяснений, почему он выбрал именно эти слова и эти идеи, почему та или иная теория описана именно в этом абзаце, а не тремя страницами позже. В академических англоязычных работах последовательность изложения материала всегда линейна. Авторы уделяют особое почтение читателям и стараются максимально показывать свою эрудицию, используя минимальное количество слов.

Специфика английского и русского языков такова, что буквальный перевод невозможен [51]. Структура, риторика и выбор лексики родного языка не должны проявляться в содержании статьи, которую исследователь пишет на английском. Для того, чтобы готовить свою научную статью на английском, нужно очень хорошо представлять себе лексико-синтаксические особенности английского научного текста.

Для неопытного человека, впервые приступившего к написанию научной статьи на английском языке, подобная задача представляет трудности. Возможно, поэтому многие научно-педагогические работники и исследователи предпочитают сначала написать статью на родном языке, а затем перевести ее на английский. При этом ими выполняется двойная работа: 1) по созданию и шлифовке текста научной статьи и 2) его переводу на английский язык (зачастую с сохранением грамматических и лексических особенностей русского языка, снижающих качество текста). Русское предложение, как правило, содержит много вводных слов, которые без ущерба для понимания можно опустить при переводе, поскольку английские предложения строятся более эргономично. А.Л. Пумпянский, известный советский переводчик технической литературы и автор книг по техническому переводу, указывал, что хороший перевод статьи на английский язык должен быть короче русского оригинала примерно на 10% [52].

Для овладения навыками стилистически корректного выбора необходимых лексических единиц из синонимических рядов; умения адекватно и максимально полно передавать свою мысль средствами английского языка; корректно и логично строить высказывания, – можно воспользоваться зарубежными и отечественными учебными изданиями, обучающими написанию научных статей на английском языке. В списке

рекомендованной литературы приведены ссылки на эти пособия, которые рекомендуем изучить [38–42, 53–65].

Глава 4. Этические принципы и нормы научно-публикационного процесса. Недобросовестные практики, существующие в современной научно-публикационной среде

Публикация – завершающая стадия научного исследования и ответственный этап работы для всех участников исследования. От научных публикаций ожидают предоставления детального и достоверного изложения результатов исследования. Поскольку публикации формируют основу не только для новых исследований, но и для практического применения результатов, они влияют на научное сообщество и, косвенно, на все общество в целом. Поэтому исследователи обязаны гарантировать, что их публикации являются честными, ясными, точными, полными и взвешенными, они не должны допускать введение читателей в заблуждение, выборочного или двусмысленного изложения фактов. Редакторы журналов также несут ответственность за добросовестность научных публикаций в соответствии с принятыми в их издании правилами [66–68].

Международным стандартом для авторов является «Ответственный подход к публикации научно-исследовательских работ» (Responsible research publication: international standards for authors), разработанный членами COPE и принятый на Второй Всемирной конференции по целостности исследований в Сингапуре в 2010 г. [69]. Из этого документа следует, что авторы должны обеспечивать ответственный подход к публикации научно-исследовательских работ, что означает:

1. Надежность и основательность:

– публикуемое исследование должно быть проведено в соответствии с этическими и юридическими нормами;

– публикуемое исследование должно быть качественно и тщательно выполненным;

– исследователи должны использовать соответствующие методы анализа и представления данных (при необходимости обращаться за консультацией к специалисту в этой области);

– авторы несут коллективную ответственность за свою работу и содержание публикации. Исследователи должны тщательно проверять свои публикации на всех стадиях, чтобы гарантировать, что все их методы и результаты изложены точно. Авторы должны тщательно проверять все расчеты, качество представления данных, формируемую ими документацию и доказательства.

2. Честность:

– исследователи должны представлять результаты честно, без фабрикаций, фальсификаций или недобросовестного манипулирования данными. Редактирование публикуемых изображений (например, микроснимков, рентгенограмм, снимков электрофореза) не должно создавать вероятности введения читателя в заблуждение;

– исследователи должны стремиться описывать свои методы и представлять открытия ясно и однозначно, следовать правилам изложения научных работ; публикации должны предоставлять достаточную информацию для того, чтобы другие исследователи могли повторить проведенные эксперименты;

– отчеты об исследованиях должны быть полными. В них не должна опускаться информация о необъяснимых фактах, противоречивых данных, и данных, противоречащих теориям или гипотезам авторов или спонсоров исследования;

– спонсоры исследований не должны иметь права накладывать вето на публикации результатов, неблагоприятно представляющих их продукцию или положение. Исследователи не должны заключать соглашений, позволяющих спонсорам запрещать или контролировать публикации результатов (кроме исключительных случаев, например, если исследование признано секретным на правительственном уровне);

– авторы должны сразу же уведомлять редактора в случае обнаружения ошибки в любой поданной ими на публикацию, принятой для публикации или уже опубликованной работе. Авторы должны сотрудничать с редакторами при необходимости правки, сокращения или изъятия работы;

– цитаты и ссылки на другие работы должны быть точными и аккуратно оформленными;

– авторы не должны копировать из других публикаций ссылки на работы, с которыми они сами не ознакомились.

3. Взвешенность:

– новые результаты должны быть представлены в контексте предыдущих исследований. Работы других ученых должны быть подобающим образом отражены. Обзор и выводы из существующих исследований должны быть полными, сбалансированными и включать сведения вне зависимости от того, поддерживают они гипотезы и толкования автора публикации или нет. В журналах должно проводиться четкое разграничение между научными статьями и колонками редактора и статьями, представляющими субъективную точку зрения;

– все ограничения проведения исследования должны быть отражены в публикации.

4. Оригинальность:

– авторы должны соблюдать требования к публикациям относительно того, что предлагаемая работа является оригинальной и не была ранее опубликована нигде ни на каком языке. Работа не может быть направлена одновременно в несколько изданий, кроме случаев, когда издатели соглашаются на совместное издание. Если статья издается совместно, этот факт должен быть известен читателям;

– должны соблюдаться конвенции и законодательство в отношении авторских прав. Материалы, защищенные авторским правом (например, таблицы, цифры или крупные цитаты), могут воспроизводиться только с разрешения их владельцев;

– в публикации следует ссылаться на имеющие к ней отношение предыдущие работы, как других исследователей, так и самого автора, делать это правильно и точно. Во всех возможных случаях должна быть указана ссылка на первоисточник;

– необходимо указывать авторство данных, текста, рисунков и идей, которые автор получил из других источников, они не должны представляться как принадлежащие автору публикации. Прямые цитаты из работ других исследователей должны выделяться кавычками и соответствующей ссылкой;

– авторы должны уведомлять издателей, если предлагаемые ими к публикации данные ранее публиковались где-либо, если какие-либо интерпретации этих данных направлены в другие издательства. В этом случае авторы должны предоставить копии таких публикаций или работ, отправленных на рассмотрение в другие журналы;

– различные публикации, возникающие в результате работы над одним исследовательским проектом, должны четко идентифицироваться как таковые и должны содержать ссылки на первоначальные работы. Переводы и адаптации для различных аудиторий должны быть четко обозначены, иметь ссылки на первоисточник, соблюдать соответствующие конвенции об авторском праве и правила получения разрешений на использование. В случае сомнений, авторы должны попросить и получить разрешение издателя первоисточника.

5. Прозрачность:

– все источники финансирования исследований, включая прямую и косвенную финансовую поддержку, предоставление оборудования/материалов, иные виды поддержки (например, помощь специалистов по статистической обработке данных или технических писателей) должны быть указаны;

– авторы должны предоставлять информацию о степени участия спонсора исследования (если таковой есть) в подготовке проекта, выполнении, анализе, интерпретации результатов и подготовке отчета об исследованиях;

– авторы должны предоставить информацию о финансовых и нефинансовых интересах и отношениях, которые могли бы повлиять на интерпретацию их открытий, а также информацию, существенную для издателей, рецензентов и читателей. Это включает любые отношения автора с журналом, например, если издатели публикуют свои собственные исследования в собственном журнале. Кроме того, авторы должны следовать требованиям журнала и учреждения по вопросам раскрытия конкурирующих интересов.

6. Авторство и ссылки на источники:

– исследовательская литература содержит не только информацию об открытиях, но и о том, кто эти открытия совершил. Следовательно, авторство научных публикаций должно точно отражать вклад отдельных лиц в исследовательскую работу и написание отчета о ней;

– в случаях, когда люди, сделавшие основной вклад, перечислены как авторы, а те, чей вклад в исследование или написание работы был менее существенен или носил чисто технический характер, указаны в разделе выражения благодарности, критерии авторства должны быть согласованы в начале проекта. Критерии авторства в определенной сфере исследований должны быть согласованы, опубликованы и постоянно применяться исследовательскими центрами, профессиональными и академическими сообществами и спонсорами. Хотя редакторам журналов следует развивать и публиковать критерии авторства, от них не стоит ожидать разрешения споров по данному вопросу. Ответственность за правильное определение авторства лежит на самих авторах, действующих в соответствии с правилами, принятыми в их учреждении. Научные учреждения должны развивать и поддерживать справедливые стандарты определения авторства и выражения признательности. Такие учреждения должны решать споры по вопросам авторства, обеспечивая при этом соблюдение процедуры;

– исследователи должны гарантировать, что только те лица, которые соответствуют критериям авторства (то есть внесли значительный вклад в работу), считаются авторами, и что заслуживающие авторства исследователи не будут исключены из этого списка;

– научные учреждения и редакторы научных изданий должны внедрять практику предотвращения *гостевого, подарочного или безымянного авторства*.

Гостевые авторы – авторы, не соответствующие принятым критериям авторства, но внесенные в список благодаря их званию, репутации или предполагаемому влиянию;
Подарочные авторы –авторы, не соответствующие принятым критериям авторства, но внесенные в список авторов благодаря личным отношениям или за плату;
Безымянные авторы –авторы, соответствующие критериям авторства, но не указанные в списке авторов

– все авторы должны дать согласие на внесение в список авторов и должны одобрить направленную на публикацию и отредактированную версию работы. Любые изменения в списке авторов должны быть одобрены всеми авторами, включая тех, кто исключен из списка. Ответственный автор выступает контактным лицом между издателем и другими авторами. Он должен информировать соавторов и привлекать их к принятию решений по вопросам публикации (например, в случае ответа на комментарии рецензентов);

– авторы не должны вводить читателей в заблуждение, публикуя благодарности людям, которые фактически не привлекались к работе и не оказывали поддержку [70–71].

7. Отчетность и ответственность:

– все авторы должны прочитать и хорошо знать представляемую к публикации работу, гарантировать, что эта работа соответствует принципам, изложенным в данном руководстве. В большинстве случаев на авторов накладывается совместная ответственность за добросовестность исследования и достоверность данных в отчете по нему. Однако, если авторы принимают ответственность только за отдельные сегменты публикуемого материала, это должно быть указано;

– авторы должны работать вместе с редакторами или издателями для скорейшего исправления своих работ в случае обнаружения в них ошибок или упущений после публикации;

– авторы должны придерживаться соответствующих конвенций, требований и постановлений, чтобы их материалы, реагенты, программное обеспечение или наборы данных были доступны для других исследователей, которые их запросят. Исследователи, научные учреждения и спонсоры должны иметь четкую политику для рассмотрения таких запросов. Авторы обязаны следовать определенным стандартам журналов. Если предлагается выражение признательности за предоставленные материалы, не уместно требование указать себя в числе авторов (например, в качестве условия для предоставления материалов);

– авторы должны соответствующим образом отвечать на комментарии после публикации, а также на публикуемую корреспонденцию. Они должны попытаться ответить на вопросы рецензентов и предоставить необходимые пояснения и дополнительную информацию, если таковая потребуется.

8. Соблюдение соглашений относительно рецензирования коллегами (peer-review) и публикации:

– авторы должны выполнять требования издателей о том, что работа не должна одновременно предлагаться для публикации более чем в одно издание;

– авторы должны сообщать редактору, если они отказываются от рецензирования [другими экспертами] их работы или не готовы отвечать на комментарии рецензента после получения условного согласия на публикацию;

– авторы должны ответить на вопросы рецензента профессионально и в кратчайшие сроки;

– авторы должны с уважением отнестись к запросу издателя на ограничение публикаций в СМИ и не должны позволять сообщать о своих открытиях в СМИ, если их статья была принята к публикации (но еще не опубликована) в научном издании. Авторы и их научные учреждения должны поддерживать связь и взаимодействовать с издателями для координирования деятельности со СМИ (например, пресс-релизы или пресс-конференции) в связи с публикацией. Пресс-релизы должны точно отражать содержание работы и не должны включать в себя данные, выходящие за пределы результатов исследования.

9. Ответственное отражение результатов исследований с участием людей или животных:

– соответствующие одобрения, лицензии и регистрации должны быть получены до начала исследований, информация об этом должна быть включена в отчет об исследовании (например, одобрение экспертного совета, организации, комитета по исследовательской этике, разрешение национальных лицензирующих властей на использование животных);

– по запросу редактора авторы должны предоставить свидетельство, что исследование, описанное в работе, получило необходимые разрешения и проводилось этично (например, копии одобрений, лицензий, формы согласия участников);

– исследователи не должны публиковать или распространять идентифицируемые личные данные человека, собранные в ходе исследования без его согласия (или согласия его представителей). Исследователи должны помнить, что многие научные журналы в данное время находятся в свободном доступе в сети Интернет, и должны иметь в виду риск причинения вреда или морального ущерба нецелевой аудитории (например, участникам исследований или их семьям, которые могут узнать себя в изложении ситуационных исследований, описаниях, изображениях или родословных);

– методология статистического анализа данных должна быть определена в начале исследований, план анализа данных для получения предварительных результатов должен быть подготовлен заранее, их следует придерживаться. Вторичный или апостериорный анализ нужно четко отличать от первичного и анализа, указанного в плане;

– исследователи должны публиковать все значимые результаты исследований, которые важны для понимания. В частности, этической нормой является публикация результатов всех клинических испытаний. Публикация неуспешных исследований или экспериментов, которые отвергают гипотезу, может избавить других от потери времени и ресурсов на осуществление схожих проектов. Если результаты незначительных и не дающих статистически значимых результатов исследований могут быть объединены для получения более полезной информации (например, путем мета-анализа), такие данные должны быть опубликованы;

– авторы должны по запросу предоставлять редакторам журналов протоколы исследований (например, клинических испытаний), чтобы рецензенты и редакторы могли сравнить отчет об исследовании с протоколом, убедиться, что оно было проведено в соответствии с планом, и никакие важные детали не были опущены. Исследователи должны следовать соответствующим правилам регистрации клинических испытаний и включать регистрационный номер испытаний во все публикации, связанные с этими испытаниями.

Принципы этического поведения должны соблюдать все участники научно-публикационного процесса. Для этого COPE (http://publicationethics.org) и другие организации и ассоциации разрабатывают стандарты не только для авторов, но и для редакторов, рецензентов, издателей и учреждений, с которыми аффилированы авторы. В том числе разрабатываются стандарты, предусматривающие конкретные шаги в случае обнаружения недобросовестных практик участников этого процесса [66–67]. Российское редакционно-издательское сообщество только недавно приступило к решению этих вопросов. Совет по этике Ассоциация научных редакторов и издателей (АНРИ) начал работу по разработке документов и анализу выполнения этических норм авторами, редакторами, издателями. На сайте АНРИ на страницах Совета по этике научных публикаций (http://rasep.ru/sovet-po-etike) и на сайте Академии АНРИ представлен ряд важных документов по проблемам соблюдения этических норм для всех участников научно-публикационного процесса, в т.ч. опубликована Декларация «Этические принципы научных публикаций», принятая членами АНРИ в мае 2016 г. (http://rasep.ru/sovet-po-etike/deklaratsiya). Однако недобросовестные практики в научной среде получили широкое распространение и требуют совершенствования мер противодействия неэтичному поведению исследователей.

Недобросовестные практики, существующие в современной научно-публикационной среде, включают в себя все виды преднамеренного обмана на любом этапе процесса (заявка – исследование – публикация), а также такие исключительные

случаи халатности, когда встает вопрос о профессиональной репутации. К недобросовестным практикам относятся, но не ограничиваются ими, следующие пункты: 1) фабрикация и/или фальсификация научных результатов; 2) плагиат данных, идей или фрагментов статей; 3) намеренный отбор или замалчивание результатов в публикации, когда эти результаты имеют отношение к выводам; 4) ложное использование статистических или других методов; 5) намеренная или халатная небрежность в сокрытии деталей методики; 6) ложное информирование об авторстве (приписное почетное авторство, невидимое авторство (отсутствие указания на участие молодых исследователей)); 7) ложное представление результатов других исследователей (ложное (фиктивное) цитирование); 8) недопустимый повтор публикации (самоплагиат и дублирующие публикации); 9) ненадлежащее обращение с объектами исследования; 10) сговоры с целью искусственного повышения цитирования.

Стандартного определения недобросовестного поведения исследователей не существует, с появлением новых научных методов появляются новые дефиниции, но в целом недобросовестное поведение затрагивает десять пунктов, указанных выше.

Понятие «недобросовестное поведение исследователей» применимо к любым действиям, включающим ненадлежащее обращение с объектами изучения или намеренное манипулирование научной информацией, при котором она перестает отражать наблюдаемые явления и теряет достоверность. Можно использовать следующее определение недобросовестного поведения: ***«Поведение исследователя, преднамеренное или нет, не соответствующее этическим и научным стандартам***».

Понятия «недобросовестность» и «мошенничество» являются центральными в определении недобросовестного поведения исследователей, но не каждое причинение вреда объекту исследования обязательно является результатом недобросовестного поведения исследователей. К недобросовестному поведению не относится ошибка или нефальсифицированные противоречия в плане, проведении, интерпретации или оценке в методах исследования и результатах. Однако редакторы и прочие участники публикационного процесса должны рассматривать недобросовестное поведение исследователей в тех случаях, когда вред является результатом научной деятельности, не отвечающей этическим нормам или ставшей прямым результатом безответственного поведения исследователя. Работа низкого качества не приравнивается к недобросовестному поведению, если только исследователи не использовали методы низкого качества с целью фальсификации или без учета вреда, который мог быть причинен объектам исследований.

Фабрикация – подделка данных (либо результатов) исследования. Исследование может не проводиться вовсе и существовать лишь в виде статьи. Фабрикация является придумыванием данных или результатов, записью и сообщением о них.

Фальсификация – манипулирование данными, оборудованием, исследовательскими процессами, изменением/пропуском данных с целью получения «необходимого» результата.

Пиратство и плагиат – использование чужих научных достижений, идей, процессов, результатов или слов без указания ссылки на их автора/авторов. Под **пиратством** подразумевается несанкционированное воспроизведение или использование идей, данных или методов, разработанных другими, без соответствующего разрешения или уведомления. В этом типе недобросовестного поведения ключевую роль играет обман. Цель нарушителя – нечестная подача идей или методов как своих собственных. Плагиат – форма пиратства, включающая в себя несанкционированное использование или близкую имитацию языка (рисунков, изображений или таблиц), а также мыслей других людей и их представление в качестве собственной оригинальной работы.

Самоплагиат (автоплагиат) – использование частей своих предыдущих работ без какой-либо переработки для «клонирования» публикаций. Плагиат в целом включает использование материалов, созданных другими, однако может применяться к случаям дублирования собственных работ исследователя, которые были опубликованы прежде, без ссылки на них (иногда это называют самоплагиатом или повторной публикацией).

Компиляция – составление «своего» труда из фрагментов других исследований, не содействующего приращению научного знания.

Ненадлежащее обращение с объектами исследования. Исследователи несут обязательства перед объектами изучения. Эти обязательства применимы в тех случаях, когда объектами являются люди и животные, при том, что изучаться может как организм в целом, так и его образцы. Если в качестве объектов исследования выступают люди, открытой формой недобросовестного поведения ученых являются несоблюдение принципов Хельсинской декларации[2], стандартов международных социологических, психологических и иных профильных ассоциаций, неполучение одобрения соответствующих организаций и государственных комитетов по экспериментам на людях и несоблюдение их требований. Для исследователей, изучающих животных, несоблюдение институциональных или государственных рекомендаций по уходу за лабораторными животными и их использованию также является открытой формой

[2] World Medical Association Declaration of Helsinki: ethical principles for medical research involving human subjects // JAMA. 2000. P. 284. P. 3043–3045.

недобросовестного поведения. К ненадлежащему обращению с объектами исследования относится:

– ненадлежащее обращение с лабораторными животными;

– подвергание объектов исследования риску причинения физического и психологического ущерба без информирования о возможном вреде;

– подвергание объектов исследования (или окружающей среды) риску причинения вреда, вызванного несоответствием практики и протоколов определенным и (или) утвержденным стандартам;

– нарушение конфиденциальности информации о людях, являющихся объектами исследования, без их согласия.

Цифровые изображения и недобросовестное научное поведение

Можно выделить два типа недобросовестного поведения при работе с цифровыми изображениями: неподобающее манипулирование и мошенническое манипулирование. Неподобающие манипуляции включают приспособление изображений к требованиям журнала и не затрагивают интерпретации данных. Мошеннические манипуляции включают корректировку изображений, которая влияет на интерпретацию данных. Многие манипуляции представляют собой неприемлемые изменения оригинальных данных, могут указывать на недобросовестность поведения и считаться мошенническими. Здесь под мошенничеством понимается фальсификация или фабрикация изображений; оно не заключает в себе правовых критериев причинения ущерба третьей стороне в случае использования сфальсифицированных изображений.

К признакам **неэтичного поведения в области научных публикаций** также относятся:

1) требование к авторам самостоятельно предоставлять рецензии на собственные статьи, а также договорное и псевдорецензирование. Данная практика подразумевает отсутствие рецензирования в журнале;

2) предложение агентских услуг. Оказание таких услуг авторам как «публикация под ключ», переписка с редакцией от лица автора, доработка агентом статей по рекомендациям рецензента, подготовка платных рецензий;

3) продажа соавторства, подарочное соавторство, изменение состава авторов. Указание в числе авторов лиц, не внесших интеллектуальный вклад в исследование, является нарушением авторских прав и норм этики, поскольку не только вводит в заблуждение читателей, но и расценивается как мошенничество;

4) публикация материалов «*заочных научных конференций*». Поскольку практика таких конференций напрямую связана с махинациями и мошенничеством в сфере науки,

публикация материалов несуществующих конференций расценивается как неэтичная, содействующая распространению псевдонаучных текстов;

5) передача редакторами текстов статей в другие журналы без согласования с авторами. Публикация статьи в журнале, который не был согласован с автором, является нарушением интересов автора;

6) передача редакторами или рецензентами материалов авторов третьим лицам. Передача присланных в редакцию материалов статей третьим лицам, кроме рецензентов и сотрудников редакции, является нарушением авторских прав и принципа конфиденциальности редакционных процессов;

7) манипуляции с цитированием. Искусственное увеличение наукометрических индексов, избыточное самоцитирование и дружественное цитирование, нерелевантные ссылки вводят в заблуждение читателей и интерпретируются как мошенничество;

8) плагиат, фальсификации и фабрикации. Редакция добросовестно работает с текстами статей, предотвращая на страницах своих изданий появление недобросовестных научных публикаций, содержащих плагиат, фальсификацию и фабрикацию данных;

9) веерная рассылка одного и того же текста статьи в несколько научных журналов;

10) самоцитирование и дублирующие публикации. Авторы с целью повышения числа научных публикаций и наукометрических показателей рассылают один и тот же текст под разными названиями, маскируя материал под новый, по разным изданиям. Приращения научного знания такого рода публикации не дают.

Глава 5. Продвижение опубликованных статей: системы идентификации авторов и публикаций, профессиональные сети, базы данных, архивы, репозитории

О продвижении своей статьи автор(-ы) должны думать еще до ее публикации. Этому, в первую очередь, способствует правильный выбор журнала, доступ к которому имеет мировое научное сообщество, а также – использование различных идентификаторов, относящихся как к самой статье, так и к фамилиям авторов. В первую очередь это: уникальный идентификатор статьи *DOI*, уникальный идентификатор автора *ORCID* (http://orcid.org) и идентификатор *ResearcherID* (http://www.researcherid.com). Владельцы МНБД разрабатывают дополнительные инструменты, в т.ч. сервисы для авторов по управлению своими публикациями и работы с библиографической

информацией. К таким системам можно отнести *Mendeley* и *EndNote.* Более подробно об идентификаторах авторов и публикаций рекомендуем прочитать на сайтах систем и в материалах зарубежных и российских специалистов [72–73].

Важное значение для продвижения имеет репутация журнала, в котором опубликована статья, его присутствие и видимость в научном пространстве, политика в отношении распространения опубликованных в журнале статей. Такая информация публикуется, как правило, в разделе «Для авторов», многие издательства и журналы размещают информацию о своей политике на сайте *SHERPA/Romeo* (http://www.sherpa.ac.uk/romeo/index.php). Автору могут быть предложены следующие варианты: возможность архивировать финальную версию статьи; архивировать только препринты (версию рукописи до прохождения рецензирования и рецензирования редакцией/издательством); полный запрет на архивирование любой версии статьи. Всю интересующую информацию необходимо проверять на официальном сайте выбранного журнала и/или уточнять в редакции. Около 70% журналов, входящих в SHERPA/Romeo, разрешают размещать препринты статей в открытом доступе в институциональных репозиториях либо на сайтах авторов.

Как правило, продвижение публикаций подразумевает активность ученых/ авторов в социальных и профессиональных сетях, и в Интернете в целом, поставку статей и препринтов в открытые архивы и другие информационные ресурсы, включение опубликованных результатов исследований в систему научных коммуникаций. Открытых систем, способствующих продвижению публикаций много. Зарубежные издательства и компании, создающие и поддерживающие научные информационные ресурсы, публикуют справочные материалы о том, каким образом автор может продвигать свое исследование после публикации результатов [23]. Также этой теме уделяют большое внимание российские информационные специалисты [26].

Персональный веб-сайт, страница и/или блог

Персональный веб-сайт позволяет ученому представить информацию о себе (CV) и своих работах, дать возможность доступа к полным текстам статей, дополнить ранее опубликованные материалы новыми комментариями, инициировать дискуссии и привлекать коллег к участию в них. К персональным сайтам ученых часто обращаются журналисты, когда нужно получить комментарий, консультацию или разъяснение по профильному вопросу, что также работает на имидж ученого и позволяет довести информацию о его работе до более широкой аудитории.

Блог может находиться на персональном сайте или на отдельной платформе, предназначаться для широкой или узкой аудитории.

1. Присвоение идентификатора DOI статьям

Уникальный идентификатор цифрового объекта *DOI* (Digital Object Identifier) обеспечивает способ постоянной идентификации объекта, которым чаще всего бывает электронный документ, в нашем случае – публикация. Обязательным условием является наличие метаданных объекта на определенном сайте с фиксированным, неизменным адресом (*URL*), тогда как сам объект (статья на сайте) может отсутствовать. DOI дает возможность безошибочно определять: библиографические данные, постоянное местонахождение публикации, точно ее цитировать. Все DOI (описания публикаций и их постоянный URL) регистрируются в системе CrossRef. Через нее происходит связь ссылки по DOI с самим документом или его описанием. Этот цифровой идентификатор используется практически всеми ведущими зарубежными издательствами и журналами. Наличие DOI, который присваивает статье издательство, в дальнейшем позволяет точно цитировать его в списках литературы и связывать саму публикацию в МНБД со ссылками на нее. Необходимо понимать, что DOI присваивается статье только один раз и всегда имеет только актуальный адрес URL. Указание DOI на других сайтах будет считаться цитированием статьи с данными основного сайта, на который ведет DOI. Поэтому, когда авторам какие-то системы (электронные библиотеки, другие электронные платформы) предлагают присвоить статье (или книге, или главе в монографии), которая раньше не имела DOI, этот идентификатор, необходимо понимать, что после этого этот сайт (URL) становится для статьи основным. Это не всегда желательно, т.к. качество сайтов (например, только русский язык сайта) может ограничивать возможности доступа к статье иностранных читателей. Подробнее о DOI можно ознакомиться в статье [74].

2. Регистрация в системе ORCID

Регистрация автора в системе *ORCID (**O**pen **R**esearcher and **C**ontributor **ID,*** http://orcid.org) позволяет однозначно идентифицировать автора и не спутать его с однофамильцами. Важно не только завести профиль, но и максимально подробно заполнить его: указать место учебы и работы, научные достижения, проекты, составить перечень опубликованных статей. Профиль ORCID необходимо указывать во всех публикациях, давать ссылки на персональном веб-сайте и в блоге. Идентификатор автора ORCID включается в профили авторов в Scopus и позволяет дополнять профиль автора в ORCID данными о публикациях и их цитировании из Scopus. Наличие DOI статьи упрощает этот процесс: ORCID интегрирован с систему CrossRef, позволяя автоматически добавлять новые статьи авторов с указанными DOI в профиль автора в ORCID. Наличие ORCID также дает редакции возможность получить необходимую информацию о публикационной активности автора, направляющего свою рукопись в журнал. Поэтому

отредактированный авторский профиль ученого в ORCID может играть определенную роль еще в процессе принятия редакционного решения об одобрении рукописи.

3. Профессиональные социальные сети

ResearchGate. Профессиональная социальная сеть *ResearchGate* (http://www.researchgate.net) позволяет авторам загружать свои статьи в профили, вести блоги, подписываться на обновления персональных страниц, создавать собственные темы и т.д. Участники сети могут выкладывать предпоследний вариант рукописи, уже принятый к печати, но еще не оформленный издательством, либо публиковать только библиографические данные, система позволяет запросить полный текст у автора, если это чужая статья. Так образуется цитирование в этой системе. Большинство издательств при составлении договоров с авторами разрешают ограниченное предоставление авторских копий коллегам с целью обмена научной информацией [26].

Google Scholar. Создание авторского профиля в профессиональной сети *Google Scholar* («*Google Академии*», http://scholar.google.ru) с включением в него своих публикаций позволяет авторам отслеживать их цитирование в сети. Регистрация в системе дает автору возможность отредактировать данные о проиндексированных в системе публикациях и добавить отсутствующие.

Academia.edu. Профессиональная социальная сеть *Academia.edu,* как и другие научные сети, позволяет создавать профиль, добавлять препринты и публикации, отслеживать их цитирование в этой сети.

4. Институциональные репозитории

Многие университеты и научные организации создают свои *институциональные репозитории*. При наличии разрешения издательства на использование публикации или ее препринта в открытом доступе, этот ресурс является одним из первых, куда ученому, работающему в организации, создавшей этот репозиторий, следует размещать свои тексты. Самым актуальным каталогом репозиториев является система OpenDOAR (http://opendoar.org).

5. Открытые электронные архивы препринтов и научных публикаций

Открытые электронные архивы научных препринтов и публикаций – информационные системы, включающие в себя научные документы, как правило, по определенным тематическим областям. К таким ресурсам относятся, в первую очередь, *arXiv.org* (архив препринтов и публикаций по физике, математике, астрономии, информатике и биологии), *RePEc* (**Re**search **P**apers in **Ec**onomics, http://repec.org), *bioRxiv* (http://biorxiv.org/). Эти ресурсы очень популярны среди специалистов и, соответственно,

препринты или статьи, размещенные в них, быстро находят своего читателя и получают возможность быть процитированными.

6. Публикация наборов исходных данных и иллюстраций

Одной из таких систем, позволяющих сохранять «сырые данные» из своих публикаций является репозиторий иллюстраций FigShare (https://figshare.com/).

7. Социальные медиа

Относительная простота использования, широкая аудитория, возможность быстрого контакта и отслеживание реакции пользователей на статью (перепосты и лайки) делают *Facebook* и *Twitter* полезными и эффективными каналами для распространения опубликованных результатов. В зарубежных странах еще одним полезным инструментом называют социальную сеть *LinkedIn*, доступ к которому в настоящее время в России заблокирован.

8. Системы управления источниками литературы

Системы (программы) управления ссылками *Mendeley, CiteULike, Zotero, F1000 Workspace* также предлагают возможность обмениваться информацией с коллегами, делиться ссылками на статьи и получать оперативные отклики. Основная задача этих систем – облегчить авторам работу с библиографическими источниками (см. Главу 3).

10. Регистрация ученого в качестве рецензента

Участие в работе международных журналов в качестве рецензента повышает авторитет ученого, позволяя ему первым узнавать о новейших работах, использовать свои знания при рецензировании статей других ученых. Регистрация ученых в качестве рецензентов в системе *Publons* (https://publons.com) способствует их выходу на международный уровень. Каждому ученому дается возможность зарегистрироваться в качестве рецензента и указать, с каким журналом он сотрудничает или сотрудничал ранее.

11. Использование средств массовой информации

Важность работы ученого со средствами массовой информации часто недооценивают, хотя коммуникация с обществом и реакция общества на проводимые исследования не менее важна, чем признание заслуг профессиональной среде. Помочь в общении с журналистами (подготовка пресс-релиза, интервью и другие материалы) могут пиар-отделы вузов и институтов.

Использованные и рекомендуемые источники

1. Обновление инструкции для авторов научных журналов: Методические материалы. Пер. с англ. под ред. А. Ю. Гаспаряна, О. В. Кирилловой. Пер. с англ. А. В. Бажанова. СПб.: Сев.-Зап. ин-т упр. фил. РАНХиГС, 2015. 48 с. Адрес доступа: http://conf.neicon.ru/materials/15-Domestic0515/Instruction-0515.pdf

2. Якшонок Г.П. Публикация международного уровня: рекомендация по подготовке [Презентация, видео]. Адрес доступа: http://conf.neicon.ru/index.php/science/index/pages/view/domestic-video?video=yakshonak

3. Shaikh A.A. 7 steps to publishing in a scientific journal: Before you hit «submit», here's a checklist (and pitfalls to avoid) // Elsevier Connect. Posted on 4 April 2016. Адрес доступа: https://www.elsevier.com/connect/7-steps-to-publishing-in-a-scientific-journal

4. Быкова М. Мастер-класс Марины Быковой «Как опубликовать научную статью в зарубежном журнале?». Адрес доступа: http://iphras.ru/page13795185.htm

5. Дембовски М. (Oxford University Press) Как опубликовать статью в международном журнале = How to get published with international journal [Видеозапись выступления на 2-й Международной научно-практической конференции «Научное издание международного уровня: проблемы, решения, подготовка и включение в индексы цитирования и реферативные базы данных»]. Адрес доступа: http://conf.neicon.ru/index.php/science/index/pages/view/domestic-video?video=dembowski

6. Blocken B. 10 tips for writing a truly terrible article. In this fun but informative post, Editor Bert Blocken highlights some of the major mistakes early career researchers make when preparing and submitting a manuscript to a scientific journal // Elsevier Connect. January 11, 2017. Адрес доступа: https://www.elsevier.com/authors-update/story/publishing-tips/10-tips-for-writing-a-truly-terrible-journal-article

7. Borja A. Six things to do before writing your manuscript. In this new series – «How to Prepare a Manuscript for International Journals» – a seasoned editor gives advice to boost your chances of acceptance // Elsevier Connect. Posted on 12 May 2014. Адрес доступа: https://www.elsevier.com/connect/six-things-to-do-before-writing-your-manuscript

8. Borja A. 11 steps to structuring a science paper editors will take seriously. A seasoned editor gives advice to get your work published in an international journal // Elsevier Connect. Posted on 24 June 2014. Адрес доступа: https://www.elsevier.com/connect/11-steps-to-structuring-a-science-paper-editors-will-take-seriously

9. Dash M. Where should I publish my scholarly research article? // Journal of Microbiology & Experimentation. 2014. Vol. 1. No. 4. Art. 00022. Адрес доступа: http://medcraveonline.com/JMEN/JMEN-01-00022.pdf

10. How to get published. Preparing your manuscript / Elsevier Publishing Campus. Адрес доступа: https://www.publishingcampus.elsevier.com/pages/18/preparing-your-manuscript.html

11. Information for journal article authors / Springer. Адрес доступа: https://www.springer.com/gp/authors-editors/journal-author

12. Kerr A. Confessions of a managing editor (or 6 reasons I'm returning your manuscript): Things you do – innocently, of course – to drive your science editor crazy // Elsevier Connect. Posted on 23 July 2014. Адрес доступа: https://www.elsevier.com/connect/confessions-of-a-managing-editor-or-6-reasons-im-returning-your-manuscript: Перевод блога: Признание ответственного редактора (или 6 причин, по которым я возвращаю рукописи). Адрес доступа: http://academy.rasep.ru/dopy/31-razvitie-kompetentsij-avtorov-po-podgotovke-nauchnykh-publikatsij/napisanie-stati-na-anglijskom-yazyke/307-priznanie-otvetstvennogo-redaktora-ili-6-prichin-po-kotorym-ya-vozvrashchayu-rukopisi

13. Shaikh A.A. 5 secrets to surviving (and thriving) a PhD program: A PhD candidate shares the lessons he's learned preparing his dissertation and publishing research along the way // Elsevier Connect. Posted on 25 June 2015. Адрес доступа: https://www.elsevier.com/connect/5-secrets-to-surviving-and-progressing-in-a-phd-program

14. Snyde S. 4 ways to win an editor's heart. Useful tips from an editor that might just help you get published // Elsevier Connect. Author Update. Posted on 27 January 2016. Адрес доступа: https://www.elsevier.com/authors-update/story/career-tips-and-advice/4-ways-to-win-an-editors-heart

15. Training. Advice. Live Discussion. Networks. Free online lectures. Interactive training courses. Expert advice. Resources to support you in publishing your world-class book or journal article. Certificates to recognize your efforts / Elsevier. Адрес доступа: https://www.publishingcampus.elsevier.com/

16. Wiley Journal Authors . Адрес доступа: https://authorservices.wiley.com/author-resources/Journal-Authors/index.html

17. What is peer review? / Elsevier. Reviewers. Адрес доступа: https://www.elsevier.com/reviewers/what-is-peer-review

18. What is peer review? Wiley author services. Адрес доступа: https://authorservices.wiley.com/Reviewers/journal-reviewers/what-is-peer-review/index.html

19. Peer review process and editorial decision making at journals // Editage Insights. 2013. Nov 4. Адрес доступа: http://www.editage.com/insights/peer-review-process-and-editorial-decision-making-at-journals

20. What to expect during peer review / Taylor&Francis. Адрес доступа: http://authorservices.taylorandfrancis.com/what-to-expect-during-peer-review/

21. Evaluating Information Sources: What Is A Peer-Reviewed Article? / Lloyd Sealy Library. Адрес доступа: http://guides.lib.jjay.cuny.edu/c.php?g=288333&p=1922599

22. Murji K. Peer review – a view from the social sciences [Презентация; Видео; текст доклада] // Научное издание международного уровня – 2016: Решение проблем издательской этики, рецензирования и подготовки публикаций. Труды 5-й Междунар. научю-практ. конф., Москва, 17–20 мая, 2016. С. 44–49. Адрес доступа: http://elibrary.ru/download/elibrary_26546526_98826882.pdf; http://conf.neicon.ru/index.php/science/domestic0516/pages/view/domestic0516-video?video=/17/17_Murji

23. Sharing and promoting your article / Elsevier. Адрес доступа: https://www.elsevier.com/authors/journal-authors/submit-your-paper/sharing-and-promoting-your-article

24. Наукометрия и экспертиза в управлении наукой: Сб. тр. / ИПУ РАН. М., 2013. 568 с. (Управление большими системами. Вып. 44). Адрес доступа: http://ubs.mtas.ru/archive/index.php?SECTION_ID=685

25. Руководство по наукометрии: Индикаторы развития науки и технологий/ Акоев М.А., Маркусова В.А., Москвалева О.В., Писляков В.В. Екатеринбург: Изд-во Урал. ун-та, 2014. 250 с.

26. Мазов Н.А., Гуреев В.Н. Подготовка публикации к изданию: информационно-библиографический минимум (по наукам о Земле) / Под ред. М.И. Эпова. 2-е изд., испр. и доп. Новосибирск: ИНГГ СО РАН, 2015. 157 с.

27. Писляков В.В. Библиометрические индикаторы: Практикум / НФПК. М.: Инфра-М, 2014. 60 с.

28. Локтев А. Новые подходы к оценке значимости научных исследований на основе альтметрик и их влияние на рейтинговые показатели / Elsevier [Презентация]. Дата: 26.05.2016. Адрес доступа: lib.rudn.ru/file/altmetrics.ppt.

29. Коммуникации в современной науке / Под ред. Мирского Э.М., Садовского В.Н. М.: Прогресс, 1976. 433 с.

30. Geisler E. Metrics of Science and Technology. Westport, Connecticut – London: Quarum Books, 2000. 175 p. Available at: https://books.google.bg/books?id=UD6t9TxQZbIC&pg=PA171&lpg=PA171&dq=Garfield,+E.,+Citation+Indexing+--+Its+Theory+and+Application+in+Science,+Technology,+and+Humanities,+New+York:+John+Wiley+%26+Sons,+1979.&source=bl&ots=U0bgbVMtLh&sig=yMLo-PeVf6Ob5xVK3saHhzQj7wc&hl=ru&sa=X&ved=0ahUKEwj9pJXFtsvRAhUiP5oKHZBpBNQQ6AEIRjAF#v=onepage&q&f=false

31. Колледж Л., Джеймс К. «Корзина метрик» — лучшее средство для оценки авторитета журнала // Научный редактор и издатель. 2016. Т. 1. № 1–4. С. 25–31. Адрес доступа: http://www.scieditor.ru/jour/article/view/17

32. Scopus Content Coverage Guide / Elsevier. Updated Jan. 2016. Адрес доступа: https://www.elsevier.com/__data/assets/pdf_file/0007/69451/scopus_content_coverage_guide.pdf

33. Scopus. Краткое руководство / Elsevier. Адрес доступа: http://elsevierscience.ru/files/pdf/Scopus_Quick_Reference_Guide_Russian_v2.pdf

34. Gasparyan A. Yu. Choosing the Target Journal: Do Authors Need a Comprehensive Approach? // J Korean Med Sci. 2013. No. 28. P. 1117–1119. DOI: 10.3346/jkms.2013.28.8.1117. Адрес доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3744695/pdf/jkms-28-1117.pdf

35. How to choose a target journal / Springer. Адрес доступа: https://www.springer.com/gp/authors-editors/journal-author/how-to-choose-a-target-journal/1396

36. Еникеева А., Стерлигов И. Псевдонаучные журналы: выявление и борьба с ними (опыт НИУ ВШЭ) // Научное издание международного уровня – 2016: Решение проблем издательской этики, рецензирования и подготовки публикаций. Труды 5-й Междунар научю-практ. конф., Москва, 17–20 мая, 2016. [Презентация, видео]. Адрес доступа: http://conf.neicon.ru/materials/15-Domestic0516/20160518-10-Sterligov.pdf

37. Положение о Списке журналов и издательств, публикации в которых не учитываются при назначении академических надбавок и в оценке публикационной активности научных работников НИУ ВШЭ. Утв. 04.09.2015. Адрес доступа:

https://www.hse.ru/data/2015/09/29/1074562819/%D0%9F%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BE%20%D0%A1%D0%BF%D0%B8%D1%81%D0%BA%D0%B5%20%D0%B8%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B9.pdf

38. Как написать и опубликовать статью в международном научном журнале / Сост. Свидерская И.В., Кратасюк В.В. Красноярск: Сиб. федерал. ун-т, 2011. 52 с. Адрес доступа: http://index.petrsu.ru/files/Kak_napisat_i_opublikovat_statyu.pdf

39. Попова Н.Г., Коптяева Н.Н. Академическое письмо: статья IMRAD: Учеб. пособие для аспирантов и науч. сотрудников естественнонаучных специальностей. Екатеринбург: ИФиП УрО РАН, 2015. 161 с.

40. Day R.A. How to Write and Publish a Scientific paper. Cambridge University Press, UK, 1989.

41. Cargill M., O'Connor P. Writing scientific research Articles: strategies and steps. Blackwell Publishers, UK, 2009.

42. Glasman-Deal H. Science Research Writing for Non-Native Speakers of English. Imperial College London, UK, 2010.

43. Кириллова О.В. Значение и основные требования к представлению аффилиации авторов в научных публикациях // Научный редактор и издатель. 2016. Т.1. № 1–4. С. 25–31. Адрес доступа: http://www.scieditor.ru/jour/article/view/18

44. Кириллова О.В. Редакционная подготовка научных журналов по международным стандартам. Рекомендации эксперта БД Scopus. М., 2013. 90 с. Адрес доступа: http://academy.rasep.ru/images/documents/1_1kirillovaredprep_2013.pdf

45. Абрамов Е.Г. Какой должна быть аннотация к научной статье? // Научная периодика: проблемы и решения. 2012. № 3. С. 4–6. URL: http://cyberleninka.ru/article/n/kakoy-dolzhna-byt-annotatsiya-k-nauchnoy-statie

46. Церео К. Как я могу сделать аннотацию к своей статье более эффективной? // Научный редактор и издатель. 2016. Т. 1. № 1–4. С. 43–45. Адрес доступа: http://www.scieditor.ru/jour/article/view/19/9

47. Garfield E. Citation Analysis as a Tool in Journal Evaluation // Science. 1972. Vol. 178. No. 4060. P. 471–479. Адрес доступа: http://www.garfield.library.upenn.edu/essays/v1p527y1962-73.pdf

48. Olensky M. How is Bibliographic Data Accuracy Assessed? Proc. of 17th International Conference on Science and Technology Indicators, Montréal: Science-Metrix and OST, 2012. P. 628–639. Адрес доступа: Canada.http://2012.sticonference.org/Proceedings/vol2/Olensky_Bibliographic_628.pdf

49. Кириллова О.В. О культуре цитирования: цели, задачи и факторы риска допускаемых ошибок [Презентация, Видео] // 3-я международная научно-практическая конференция «Научное издательство международного уровня - 2014: повышение качества и расширение присутствия в мировых информационных ресурсах» 19–21 мая 2014 г., Финансовый университет при Правительстве РФ, г. Москва. Адрес доступа: http://conf.neicon.ru/materials/07-domestic2014/140520-Kirillova.pdf

50. Кравченко С.А. Стандарты и стили библиографических ссылок [Презентация]. Адрес доступа: http://repo.knmu.edu.ua/bitstream/123456789/12659/3/%D0%9A%D1%80%D0%B0%D0%B2%D1%87%D0%B5%D0%BD%D0%BA%D0%BE%20%D0%A1.%D0%90._%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D1%8B%20%D0%B8%20%D1%81%D1%82%D0%B8%D0%BB%D0%B8.pdf

51. Базанова Е.М. Научная публикация: писать на английском языке или переводить // Научный редактор и издатель. 2016. Т. 1. № 1–4. С. 17–24. Адрес доступа: http://www.scieditor.ru/jour/article/view/16/6

52. Пумпянский А.Л. Чтение и перевод английской научной и технической литературы. Книга по требованию, 2012. ISBN 978-5-458-32415-1.

53. Колесникова Н.И. От конспекта к диссертации. М.: Флинта, 2004.

54. Короткина И.Б. Академическое письмо: процесс, продукт и практика: Учеб. пособие для вузов. М.: Изд-во Юрайт, 2015. 295 с. (Образовательный процесс).

55. Alenkina T. Academic Writing in the Sciences: Theory and Practice. M.: МФТИ, 2015. 276 с.

56. Boardman C. A. J. Writing to Communicate 2: Paragraphs and Essays. 3rd ed. / C. A. Boardman, J. Frydenberg. New York: Pearson, Longman, 2008.

57. Brandon L. Paragraphs and Essays with Integrated Readings. 11th ed / L. Brandon, K. Brandon. Boston, MA: Wadsworth, Cengage Learning, 2011.

58. Galko F.D. Better Writing Right Now! Using Words to Your Advantage. New York: Learning Express, LLC, 2002.

59. Hacker D. & Sommers N.A. Writer's Reference. 7th ed. / D. Hacker, N. A. Sommers. Boston: Bedford/St.Martin's, 2011.

60. Leki I. Academic Writing: Exploring Processes and Strategies. 2nd ed. Cambridge: Cambridge University Press, 1999.

61. McCormack J. Extended Writing and Research Skills / J. McCormack, J. Slaght. Reading: Garnet Education, 2012.

62. Morley J. University Writing Course / J. Morley, P. Doyle, I. Pople. Berkshire: Express Publishers, 2007.

63. Murray R. Writing for Academic Journals. Maldenhead: Open University Press, 2005.

64. Oshima A., Hogue A. Writing Academic English / A. Oshima, A. Hogue. New York: Pearson, 2006.

65. Smalzer W.R. Write to Be Read: Reading, Reflection, and Writing. Cambridge: Cambridge University Press, 1996.

66. Подготовка и издание научного журнала. Международная практика по этике редактирования, рецензирования, издания и авторства научных публикаций: Руководства Комитета по этике научных публикаций (Committee on Publication Ethics – COPE) и Издательства Elsevier: Сб. переводов. М., 2013. 140 с. Адрес доступа: http://academy.rasep.ru/files/documents/_____3_1.pdf

67. Белая книга Совета научных редакторов о соблюдении принципов целостности публикаций в научных журналах. Обновленная версия 2012 г. / Комитет по редакционной политике (2011–2012); пер. с англ. В. Н. Гуреева; под ред. Н. А. Мазова. Екатеринбург: Изд-во Урал. ун-та, 2016. 132 с. (Библиотека научного редактора и издателя). DOI: 10.15826/B978-5-7996-1742-4. Адрес доступа: http://academy.rasep.ru/all-materials/549-belaya-kniga-soveta-nauchnykh-redaktorov-o-soblyudenii-printsipov-tselostnosti-publikatsij-v-nauchnykh-zhurnalakh-obnovlennaya-versiya-2012-g

68. Publishing Ethics Resource Kit (PERK) for editors: The Publishing Ethics Resource Kit (PERK) is an online resource to support journal editors in handling publishing ethics allegations // Elsevier Connect. Адрес доступа: https://www.elsevier.com/editors/perk

69. Уэйджер Э., Кляйнерт С. Ответственный подход к публикации научно-исследовательских работ: международные стандарты для авторов // Подготовка и издание научного журнала. Международная практика по этике редактирования, рецензирования, издания и авторства научных публикаций: Руководства Комитета по этике научных публикаций (Committee on Publication Ethics – COPE) и Издательства Elsevier: Сб. переводов. М., 2013. С. 77–86. Адрес доступа: http://academy.rasep.ru/files/documents/_____3_1.pdf

70. Альберт Т., Уэйджер Э. Как разрешать споры об авторстве: руководство для новых исследователей // Подготовка и издание научного журнала. Международная практика по этике редактирования, рецензирования, издания и авторства научных публикаций: Руководства Комитета по этике научных публикаций (Committee on Publication Ethics – COPE) и Издательства Elsevier: Сб. переводов. М., 2013. С. 87–96. Адрес доступа: http://academy.rasep.ru/files/documents/_____3_1.pdf

71. Gasparyan A.Yu. Ethical issues in science editing: Authorship [Презентация]. Адрес доступа: http://academy.rasep.ru/images/meropriyatiya/webinar_03102014/Authorship-May-2014-AYGasparyan.pdf

72. Мазов Н.А., Гуреев В.Н. Роль единых идентификаторов в информационно-библиографических системах [проблемы идентификации метаданных научных публикаций, книг, журналов, авторов и организаций] // Научно-техническая информация. Сер. 1. 2014. № 9. С. 32–37. Адрес доступа: http://www.ipgg.sbras.ru/ru/science/publications/publ-rol-edinykh-identifikatorov-v-informatsionoo-bibliograficheskikh-2014-041873

73. Скалабан А., Юрик А. Системы авторской идентификации как инструменты повышения видимости научных публикаций в Интернете // Системный анализ и прикладная информатика. 2015. № 4. С. 4–10. Адрес доступа: http://rep.bntu.by/handle/data/20855

74. Викулин А.С., Дементов А.В., Митрофанов М. И., Скалабан А.В. DOI в современной научной коммуникации // Университетская книга. 2016. Декабрь. С. 56–61. Адрес доступа: http://elpub.ru/images/files/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F_DOI.pdf

ПРИЛОЖЕНИЕ 1

СЛОВАРЬ ТЕРМИНОВ (ГЛОССАРИЙ)

A

Abstract (см. *Авторское резюме, Аннотация, Реферат*)

Academia.edu – платформа для ученых. Позволяет распространять среди участников сети свои статьи, принимать участие в обсуждениях работ, поддерживать связь с коллегами. Официальный сайт: https://www.academia.edu.

Acknowledgments (см. *Благодарности*)

Affiliation (см. *Аффилиация*)

Arts & Humanities Citation Index (A&HCI) – Индекс (указатель) цитирования по гуманитарным наукам и искусству семейства баз данных Web of Science Core Collection. Индексирует около 1 800 журналов, а также избранные статьи из 250 других журналов, с 1975 г. по настоящее время. Для журналов, включенных в A&HCI, импакт-фактор не рассчитывается. Список журналов: http://ip-science.thomsonreuters.com/mjl/publist_ah.pdf. Найти конкретный журнал и посмотреть его данные можно здесь*:* http://www.thomsonscientific.com/cgi-bin/jrnlst/jloptions.cgi?PC=H&utm_source=false/en/products-services/scholarly-scientific-research/scholarly-search-and-discovery/arts-humanities-citation-index.htmlutm_medium&utm_medium=false&utm_campaign=false, а также непосредственно в базе данных.

Author Guidelines (см. *Правила для авторов*)

Authorship (см. *Авторство*)

Altmetrics (см. *Альтметрики*)

***ArXiv* –** крупнейший в мире архив электронных публикаций научных статей и их препринтов по физике, математике, астрономии, информатике и биологии. Доступ к архиву бесплатный. Официальный сайт: https://arxiv.org/

B

Bibliometrics (см. *Библиометрия*)

BioRxiv – репозиторий научных статей по биологии. Официальный сайт: http://biorxiv.org/

Blind Peer-Review (см. *Слепое рецензирование*)

Beall, Jeffrey (см. *Билл, Джеффри*)

Byline – фамилия, имя и отчество автора(-ов).

C

Centre for Science and Technology Studies (CWTS) – Центр изучения науки и технологий. Занимается изучением динамики научных исследований и их связей с технологиями, инновациями и бизнесом Лейденского университета (University of Leiden). Является разработчиком метрик ***SNIP***, ***IPP*** и ***CiteScore***, рассчитываемых по данным ***Scopus*** (http://www.journalindicators.com/indicators). Сайт CWTS: https://www.cwts.nl

CiteScore – библиометрический показатель (метрика), используемая международной наукометрической базой данных (МНБД) ***Scopus***. ***CiteScore*** вошел в «корзину метрик» Scopus в конце 2016 г. и заменил предыдущий показатель ***IPP***. CiteScore, показывает среднее цитирование публикаций издания за 3-летний период в публикациях исследуемого года, следующего за трехлетним периодом. Рассчитывается аналогично ***импакт-фактору*** WoS, но отличается окном цитирования (трехлетнее по сравнению с

двухлетним периодом классического импакт-фактора) и охватом типов публикаций, учитываемых при подсчете цитирования (кроме статей и обзоров из журналов, учитываются другие типы публикаций, включаемые в Scopus: письма, заметки, редакционные статьи, труды конференций и другие документы). CiteScore позволяет отслеживать показатели в динамике: показатель CiteScore для текущего года обновляется каждый месяц (CiteScore Tracker).
О CiteScore см. https://www.elsevier.com/editors-update/story/journal-metrics/citescore-a-new-metric-to-help-you-choose-the-right-journal; https://www.cwts.nl/blog?article=n-q2y254

CiteULike – сервис для ученых. Позволяет пользователям сохранять и обмениваться ссылками на научные работы. Официальный сайт: http://www.citeulike.org/

Clarivate Analytics – новая компания, образовавшаяся в результате продажи интеллектуальной собственности Компании Thomson Reuters группе компаний ONEX. В результате с октября 2016 г. Clarivate Analuytics владеет и управляет коллекцией ведущих подписных ресурсов, ориентированных на научные исследования, патентную аналитику и нормативные стандарты, фармацевтическую и биотехнологическую разведку, защиту товарных знаков, защиту домена бренда и управление интеллектуальной собственностью. В их число входят: ***Web of Science***, Cortellis, Thomson Innovation, Derwent World Patents Index, CompuMark, MarkMonitor и Techstreet. Официальный сайт: http://clarivate.com/

Collaboration (см. *Коллаборация*)

Committee on Publication Ethics (COPE) – международная организация (ассоциация), базирующаяся в Великобритании, объединяющая более 10000 членов – редакторов, издателей научных журналов мира, занимающаяся вопросами разработки и продвижения этических норм и правил в научную редакционно-издательскую сферу. Разработанные COPE совместно с другими ассоциациями и владельцами ресурсов этические кодексы и стандарты являются основными руководствами для всех научных изданий. Официальный сайт: http://publicationethics.org

Conference Proceedings Citation Index – Science (CPCI-S) – указатель цитирования материалов конференций по точным, естественным и техническим наукам семейства МНБД ***Web of Science Core Collection***. Охватывает период с 1990 г. по настоящее время. Доступ платный. (http://wokinfo.com/products_tools/multidisciplinary/webofscience/cpci/?utm_source=false/en/products-services/scholarly-scientific-research/scholarly-search-and-discovery/conference-proceedings-citation-index.htmlutm_medium&utm_medium=false&utm_campaign=fals)

Conference Proceedings Citation Index – Social Science & Humanities (CPCI-SSH) – указатель цитирования материалов конференций по гуманитарным наукам семейства МНБД Web of Science Core Collection. Охватывает период с 1990 г. по настоящее время. Доступ платный.
(http://wokinfo.com/products_tools/multidisciplinary/webofscience/cpci/?utm_source=false/en/products-services/scholarly-scientific-research/scholarly-search-and-discovery/conference-proceedings-citation-index.htmlutm_medium&utm_medium=false&utm_campaign=fals).

COPE (см. *Committee on Publication Ethics*)

Corresponding Author (см. *Контактный автор*)

Cover Letter (см. *Сопроводительное письмо*)

Current Contents (в настоящее время **Current Contents Connect®**) – база данных на платформе ***Web of Science***. Включает содержание, аннотации и библиографическую информацию из недавно опубликованных выпусков ведущих научных журналов. Ранее принадлежала компании ***Thomson Reuters***, в настоящее время владелец базы данных – компания ***Clarivate Analytics*** (https://en.wikipedia.org/wiki/Current_Contents).

Current Contents Connect (см. *Current Contents*)

Creative Commons – некоммерческая организация, создатель свободных и несвободных публичных лицензий. Официальный сайт: https://creativecommons.org/

D

Digital Object Indentifier (DOI) – уникальный числовой код, который присваивается цифровому документу. Каждая журнальная статья, публикуемая онлайн, должна иметь номер DOI. Этот номер может использоваться для цитирования, если статья была опубликована онлайн раньше, чем в печати, и до получения выходных данных: тома, номеров страниц, номера выпуска.

E

Edanz Journal Selector – сервис для авторов, позволяющий подобрать журнал для публикации по заданным параметрам. Официальный сайт: https://www.edanzediting.com

Eigenfactor (Айгенфактор) – метрика базы Web of Science. Учитывает не только количество цитирований, но и значимость цитирующего источника. Также использует данные по стоимости научной периодики. Данные доступны на ***Journal Citation Reports (JCR)***, а также на официальном сайте проекта (http://www.eigenfactor.org/about.php).

eLIBRARY.RU – электронная библиотека научных публикаций, крупнейшая в России. Библиотека интегрирована с **Российским индексом научного цитирования (РИНЦ) –** бесплатным общедоступным инструментом измерения и анализа публикационной активности ученых и организаций, созданным по заказу Минобрнауки РФ (http://elibrary.ru/elibrary_about.asp).

Elsevier – международное издательство научной литературы, крупнейшее в мире. Основано в 1880 г. За эти годы компания прошла путь от небольшого голландского издательского дома, посвященного в основном классической науке, до международного мультимедийного издательского гиганта с более чем 20 тыс. продуктов для образования, науки и медицины. Компания берет свое название от House of Elzevir, голландского семейного издательского дома, созданного братьями Эльзевир в 1580 г. Наиболее известные продукты компании – ***Scopus*** и ***ScienceDirect*** (https://www.elsevier.com/about/company-information).

Emerald (Emerald Publishing Group) – международное издательство научной и практической литературы. Компания была основана в 1967 г. Сейчас под управлением Emerald находится широкий спектр цифровой продукции, среди которой портфолио из примерно 300 журналов, более 2,5 тыс. книг и свыше 450 учебных кейсов (http://emeraldgrouppublishing.com/about/index.htm).

Emerging Sources Citation Index (ESCI) – указатель цитированной литературы базы Web of Science. Охватывает литературу из развивающихся источников. Содержит журналы, которые отвечают критериям базы Core Collection, но пока не имеют достаточных показателей цитирования.

Ethics (см. Этика)

F

F1000 (Faculty of 1000) – компания, которая ведет издательскую деятельность и разрабатывает продукты для ученых в области медицинских наук и наук о жизни. Владеет издательством F1000 Research, персонализированным сервисом по поиску научных статей F1000 Prime, библиографическим менеджером F1000 Workspace. Официальный сайт: http://f1000.com/

FigShare – репозиторий, предназначенный для публикации и хранения наборов данных, иллюстраций к научных статьям, видеозаписей. Официальный сайт: https://figshare.com/

G

Garfield, Eugene (см. Гарфилд, Юджин)

G-индекс – одна из вариаций ***индекса Хирша*** на основе подсчета высокоцитируемых публикаций. Множество публикаций имеет g-индекс, равный g, если g является

наивысшим рангом, таким, что первые g публикаций вместе имеют по крайней мере g^2 цитирований. Во всех случаях g≥h *(Бредихин С.В. Методы библиометрии и рынок электронной научной периодики / Бредихин С.В., Кузнецов. А.Ю. Новосибирск: Новосибирск: ИВМиМГ СО РАН, НЭИКОН, 2012. С. 161)*.

Google Scholar (Академия Google). Google Scholar обеспечивает широкий доступ к поиску академической литературы. Здесь можно осуществлять поиск по множеству дисциплин и источников. Кроме того, сервис предоставляет возможность размещения своих работ для их распространения среди академических кругов (https://scholar.google.ru).

H

Hirsch Index, h-index (см. *Индекс Хирша*)

I

I-индекс – индикатор измерения публикационной активности научной организации, одна из разновидностей ***индекса Хирша***. Организация имеет i-индекс равный i, если i ученых из данной организации имеют индекс Хирша не менее i *(Kosmulski M. I. A bibliometric index // Forum Akademickie. 2006. № 11. P. 31)*.

Impact Factor (см. *Импакт-фактор*)

Impact per Publication (IPP) – метрика Scopus. Использовалась до введения CiteScore. Рассчитывалась также как импакт-фактор Web of Science, но по трехлетнему окну цитирования в отличие от двухлетнего окна, принятого при расчете классического импакт-фактора. Метрика была разработана ***Leiden University's Centre for Science & Technology Studies (CWTS)*** *(*http://www.journalmetrics.com/ipp.php*)*.

Invited Submission (см. *Заказная статья*)

IMRAD – акроним для Introduction, Methods, Results And Discussion (Введение, Материалы и Методы, Результаты и Обсуждение), описывает принятую в большинстве журналов структуру научной статьи.

ISSN (International Standard Serial Number) – международный стандартный номер сериальных изданий, идентификатор сериальных и других периодических и продолжающихся изданий.

J

Journal (см. *Журнал*)

Journal Citation Reports (JCR) – аналитический продукт компании ***Thomson Reuters*** (с окт. 2016 г. – компании ***Clarivate Analytics***). Содержит измеримые статистические показатели для критической оценки научных журналов на основе данных цитирования. Изучая пристатейные списки литературы, система анализирует влияние (impact) в пределах журнала или предметной категории, показывает взаимосвязи между цитируемыми и цитирующими журналами. Инструмент работает по направлениям *«science»* и *«social science»*. Является основным источником сведений об импакт-факторе журналов, рассчитываемом по данным журналов, индексируемых в ***Web of Science Core Collection*** (SCI и SSCI). Другого источника этого показателя не существует.

Journal Finder – сервис для авторов. Позволяет подобрать журнал для публикации статьи по заданным параметрам. Поиск производится по журналам издательства ***Elsevier***. Официальный сайт: http://journalfinder.elsevier.com/

Journal Impact factor (JIF) (см. *Импакт-фактор журнала*)

L

LaTeX – это высококачественная система верстки; она включает в себя функции, предназначенные для производства технической и научно-технической документации.

LaTeX де-факто является стандартом научных коммуникаций и публикации научных документов. LaTeX доступен в качестве бесплатного программного обеспечения'

M

Mendeley – научная социальная сеть с функцией библиографического менеджера. Официальный сайт: https://www.mendeley.com/

O

Open Journal System – система для создания сайта научного журнала с открытым исходным кодом. Как правило, используется в журналах открытого доступа. Официальный сайт: https://openjournalsystems.com/

ORCID (Open Researcher and Contributor ID) – уникальный идентификатор ученого, формирующийся автоматически при регистрации в системе на сайте http://orcid.org и служащий основным идентификатором ученых как авторов публикаций, получателей грантов и т.д.

P

Peer Review (см. *Рецензирование*)

Percentile (см. *Перцентиль*)

Publish or Perish – программа для поиска и анализа научной литературы.

Publons – сервис для рецензентов. Официальный сайт: https://publons.com/home/

Q

Quartile (см. *Квартиль*)

R

ResearchGate – социальная сеть для ученых. Официальный сайт: https://www.researchgate.net

RePEc – интернет-проект, посвященный систематизации исследовательских работ в области экономики (http://repec.org/).

Retraction (см. *Отзыв статьи (Ретрагирование)*)

Researcher ID – цифровой идентификатор автора. Официальный сайт: http://www.researcherid.com/

Russian Science Citation Index (RSCI) – Индекс цитирования российской науки, совместный проект научной электронной библиотеки ***eLIBRARY*** и компании **Thomson Reuters** (с 2016 г. – компании ***Clarivate Analytics***). ***RSCI*** доступен как на платформе ***eLIBRARY***, так и на ***Web of Science***).

S

Science Citation Index (SCI) – база данных цитирований научной литературы. Создана в 1950-е годы Юджином Гарфилдом (США). В настоящее время название изменено на ***Science Citation Index Expanded (SCIE)***. В русскоязычной литературе часто встречается термин «Указатель цитированной литературы» (см. напр. Маркусова В.А.). Отметим, что данный термин лучше отражает суть понятия, так как изначально SCI не выполнял функции оценки, а лишь служил целям поиска информации. SCI содержит информацию по точным, техническим, естественным и медицинским наукам (то, что в англоязычной традиции называется science). Охватывает примерно 8 500 основных журналов по 164 научным дисциплинам. На регулярной основе SCI стал выпускаться с 1964 г. ***Институтом научной информации (ISI, Institute for Scientific Information)***, созданным тем же ***Ю. Гарфилдом***. В 1973 г. появляется ***Social Science Citation Index (SSCI, Указатель цитированной литературы по социальным наукам)***, а в 1978 г. база данных

была дополнена ***Arts&Humanities Citation Index (A&HCI, Указатель цитированной литературы по гуманитарным наукам)***. Все вместе они составляют **Web of Science Core Collection**. (*Маркусова В.А. К 50-летию Science Citation Index: История и развитие наукометрии / Маркусова В.А. // Руководство по наукометрии: индикаторы развития науки и технологии / Акоев М.А. и др. Екатеринбург: Изд-воУралю ун-та, 2014. 250 с.; Garfield E. Citation indexes for science // Science. № 122(3159). P.108–111).*

ScienceDirect – полнотекстовая база данных компании ***Elsevier***. Включает контент практически по всем отраслям научного знания. На сегодняшний день содержит более 13 млн публикаций из более чем 2,5 тыс. журналов, а также 33 тыс. книг от издательства ***Elsevier*** и его партнеров. Официальный сайт: http://www.sciencedirect.com/

Scientometrics (см. *Наукометрия*)

Scimagojr.com (SJR) – открытый ресурс (платформа), построенный на данных МНБД Scopus. Разработка испанской группы SciMago. Позволяет анализировать журналы и ранжировать их по различным библиометрическим показателям, включая h-index, SJR, средний показатель цитирования статей за разный период времени и др. Система позволяет выгружать в Excel файлы ранжированные списки по предметным областям и категориям тематического классификатора ASJC Scopus, включающего 334 раздела и подраздела.

SCImago Journal Rank (SJR) – метрика ***Scimago***. Основана на идее, что не все цитирования одинаковы. SJR измеряет влияние научных журналов, которое выражается одновременно в количестве полученных цитирований и важности или престижности журнала, откуда идут данные цитирования (http://www.journalmetrics.com/sjr.php)

Scopus – реферативная и наукометрическая МНБД компании ***Elsevier***, ***индекс цитирования***.

Охватывает международных исследований в области науки, технологии, медицины, и, в меньшей степени, гуманитарных наук. Обновляется еженедельно. Scopus содержит ссылки и аннотации из более чем 22 тыс. рецензируемых журналов от более чем 5 000 международных издательств. Scopus является реферативной базой данных. «Реферативная» означает, что данная база содержит библиографические описания публикаций, включая аннотации и ключевые слова. При этом она не содержит полных текстов, но часто содержит ссылки на них. Адрес ресурса: http://scopus.com. Информационный сайт Scopus: https://www.elsevier.com/solutions/scopus/content

Self-citation (см. *Самоцитирование*)

SHERPA/Romeo – база данных политик журналов и издательств по распространению препринтов и постпринтов. Официальный сайт: http://www.sherpa.ac.uk/romeo

Social Sciences Citation Index (SSCI) – указатель цитированной литературы ***Web of Scienece*** по социальным наукам. Включает библиографические данные, цитаты и аннотации значимых статей англоязычных авторов из 3 000 журналов, охватывающих 55 дисциплин. Также содержит индивидуально отобранные релевантные публикации из более 3 500 ведущих мировых научно-технических журналов.

Source Normalized Impact per Paper (SNIP) – метрика ***Scopus***. SNIP измеряет контекстуальное влияние публикации с помощью взвешивания цитирований на основе общего количества цитирований в предметной категории. Цитирование имеет большую значимость в предметной категории с меньшим показателем цитирования, и наоборот. Рассчитывается как отношение цитирований статьи к потенциальному количеству цитирований в предметной области. Это позволяет проводить прямое сравнение источников из разных предметных категорий. Потенциал цитирования варьируется не только между предметными категориями – группами журналов в одной области исследования – или дисциплинами (например, журналы по математике, механике и социальным наукам имеют, как правило, меньше цитирований чем журналы в категории «Науки о жизни»), но и между различными журналами в пределах одной предметной категории. Например, фундаментальные журналы имеют тенденцию иметь больше

цитирований, чем прикладные или клинические журналы, а журналы, охватывающие развивающиеся отрасли, имеют больше цитирований, чем периодика по классическим предметам или журналы более общего профиля. SNIP нивелирует данные различия *(http://www.journalmetrics.com/snip.php)*.

Springer Journal Suggester – сервис для поиска научных журналов по заданным параметрам. Ведет поиск по журналам издательства Springer. Официальный сайт: http://journalsuggester.springer.com/

Springer Nature – издательская компания, одна из крупнейших в мире. Образована в 2015 г. в результате слияния компаний Springer Science+Business Media и Georg von Holtzbrinck Publishing Group's Nature Publishing Group, Palgrave Macmillan и Macmillan Education. Официальный сайт: http://www.springernature.com

Submission Checklist – чек-лист для подачи статьи на рассмотрение.

Submitting Author/ Submitting Agent – автор, подающий статью в журнал, или агент (посредник), подающий статью за автора.

Structured Abstract (см. *Структурированная аннотация/реферат*)

T

Target Journal (см. *Целевой научный журнал*)

Title (см. *Заглавие статьи*, название рукописи)

Title Page (см. *Титульная страница статьи*)

Thomson Reuters – компания – поставщик интеллектуальной информации для бизнеса и профессионалов в различных отраслях. До октября 2016 г. – собственник продуктов ***Web of Science*, *Journal Citation Reports*, *InCites*** и др. С 2016 г. собственником этих продуктов является компания ***Clarivate Analytics***.

U

URL – единый указатель ресурсов URL (Uniform Resource Locator) представляет собой стандартизированную запись ресурса в сети Интернет и на сегодняшний день является наиболее широко распространённой системой идентификации для ссылок на сетевые документы. Идентичные сетевые документы часто можно обнаружить в различных местах, но у различных копий одного и того же документа будут различные URL.

W

Webometrics (см. *Вебометрика*)

Web of Science (WoS) – онлайн-платформа, на которой размещены различные реферативные и наукометрические базы данных компании ***Clarivate Analytics*** (до окт. 2016 г. принадлежала Компании Thomson Reuters). Часто употребляется в значении семейства МНБД ***Web of Science Core Collection***.

Web of Science Core Collection (WoS CC) – семейство МНБД компании ***Clarivate Analytics*** (до окт. 2016 г. принадлежало компании ***Thomson Reuters***), размещенное на платформе ***Web of Science***, индексы цитирования. Включает 9 баз данных, в том числе 4 БД, индексирующих периодические издания: ***Science Citation Index Expanded (SCIE)***, ***Social Sciences Citation Index (SSCI)***, ***Arts & Humanities Citation Index (A&HCI)***, ***Emerging Sources Citation Index (ESCI)***; 2 БД, индексирующих материалы конференций: ***Conference Proceedings Citation Index – Science (CPCI-S)***; ***Conference Proceedings Citation Index – Social Sciences and Humanities (CPCI-SSH)***; 2 БД, индексирующих книги – Book Citation Index – Science (BKCI-S), Book Citation Index – Social Sciences & Humanities (BKCI-SSH); 1 БД по химии – Reaction Citation Index.

Z

Zotero – библиографический менеджер. Помогает собирать, систематизировать и анализировать исследования. Сохраняет основные поля описания публикации: автор, название и др. и может экспортировать эту информацию в качестве отформатированных библиографических ссылок в заданном пользователем виде. Официальный сайт: https://www.zotero.org/about/

А

Автор (Author) – физическое лицо, трудом которого создано произведение.

Авторство (Authorship) – понятие, определяющее принадлежность произведения или изобретения конкретному лицу или группе лиц. Определить авторство бывает достаточно сложно. Существуют критерии авторства, вклада каждого участника в процесс подготовки публикации, разработанные авторитетными зарубежными ассоциациями (COPE, ICMJE, APA, NIH и др.). Авторам необходимо знать и учитывать эти требования при составлении списка авторов, порядка их следования и т.п. (http://rasep.ru/sovet-po-etike/kodeksy-i-knigi/136-otvetstvennyj-podkhod-k-publikatsii-nauchno-issledovatelskikh-rabot-mezhdunarodnye-standarty-dlya-avtorov).

Аннотация (авторское резюме, реферат, Abstract) – краткое содержание статьи. Часто включается в поисковые системы баз данных, чтобы читатель могут отобрать релевантные запросу материалы. Может быть структурированным или неструктурированным в зависимости от требований журнала. Большинство типов статей должны сопровождаться аннотацией/рефератом.

Альтметрики (Altmetrics) – методы наукометрии, использующие сети профессионального общения и сотрудничества ученых, созданные как альтернатива импакт-фактору и авторским показателям ввиду их ограничений (хронологические рамки, требование присутствия журнала в определенных индексах цитирования, тематика научного исследования и др.). Альтметрики основаны на подсчете скачиваний документа, упоминаний его в социальных сетях (Facebook, Twitter) и специальных сервисах для ученых (ResearchGate, Mendeley).

Аффилиация (Affiliation) – данные о местонахождении/месте работы авторов, включающие названия организаций. Относится к метаданным публикации.

Б

Библиометрия (Bibliometry) – научная дисциплина и род прикладной деятельности по количественному анализу научной коммуникации (сетей публикаций и цитирований). Является частью наукометрии.

Библиометрические/ наукометрические индикаторы (метрики, Bibliometric/ Scientometric indicators, Metrics) – количественные показатели, характеризующие те или иные аспекты научных публикаций и их множеств. Основными библиометрическими показателями являются количество публикаций ученого, организации, журнала; абсолютная и нормализованная цитируемость публикаций; ***импакт-фактор*** (WoS); ***индекс Хирша*** (WoS, Scopus, любой индекс цитирования); ***CiteScore*** (Scopus), ***SJR*** (Scopus), ***SNIP*** (Scopus).

Библиографическая ссылка (Citation, Bibliography) – совокупность библиографических сведений о цитируемом, рассматриваемом или упоминаемом в тексте документе, необходимых и достаточных для общей характеристики, идентификации и поиска документа.

Билл, Джеффри (Beall, Jeffrey) – библиотекарь и ассоциированный профессор в Университете Колорадо, Денвер. Является известным критиком изданий открытого доступа, особенно известен своим блогом *Scholarly Open Access,* который мониторит

«хищнические журналы открытого доступа» («predatory open access publishing», термин ввел в оборот сам Билл).

Благодарности *(*Acknowledgements) – раздел, который автор включает в статью, чтобы выразить признательность или поблагодарить людей, которые внесли вклад в подготовку статьи, но не являются ее соавторами. В этом же разделе размещается информации о финансовой поддержке и грантах. Раздел «Благодарности» не может использоваться для упоминания неформального вклада друзей или родственников.

Безымянный автор – автор, соответствующий критериям авторства, но не указанный в списке авторов.

В

Введение (Introduction) – раздел статьи, в котором, как правило, содержится информация о том, какой проблеме посвящено исследование.

Вводная ссылка – особый вид ссылки на источник, размещенный в основном тексте статьи сразу за отсылкой или цитатой. Как правило, указывается в круглых скобках и содержит упоминание фамилии автора и года публикации.

Вебометрика (Webometrics) – изучение количественных аспектов создания и использования информационных ресурсов, структур и веб-технологий, основанное на библиометрическом и инфометрическом подходах. Вебометрика изучает веб-страницы как документы *(Björneborn, L., & Ingwersen, P. Toward a basic framework for webometrics / Björneborn L., Ingwersen P. // Journal of the American Society for Information Science and Technology. 2004. № 55(14). PP. 1216–1227. DOI:10.1002/asi.20077).*

Веерная рассылка (см. *Одновременная подача*)

Виды научных изданий (Source types) – законченные и оформленные издания, содержащие научный контент, прошедший оценку экспертным сообществом (процесс рецензирования) и изданные отдельными книгами, продолжающимися или периодически выходящими выпусками. К основным видам научных изданий относятся научные периодические издания (журналы, ежегодники), продолжающиеся научные издания (продолжающиеся сборники статей, книжные серии), отдельные сборники научных статей, монографии, труды научных конференций и других научных форумов. Виды научных изданий могут распространяться в различных форматах – печатном, на переносных электронных или микро-носителях (CD/DVD/HD/микрофиши), и в электронном виде через Интернет (online режиме).

Г

Гарфилд, Юджин (**Garfield, Eugene**), родился 16 сентября 1925 г. – признанный основатель и лидер наукометрического сообщества. Изобретатель множества инновационных библиометрических продуктов, среди которых **Current Contents**, ***Science Citation Index***, ***Journal Citation Reports***. Он же является автором импакт-фактора (http://www.garfield.library.upenn.edu).

Гостевой автор – автор, не соответствующий критериям авторства, но внесенный в список авторов благодаря их званию, репутации или предполагаемому влиянию.

Д

Дата публикации (Publication Date) – дата, когда рукопись стала доступна в печатном виде или онлайн. Некоторые базы данных считают дату публикации онлайн официальной датой публикации.

Двойное слепое рецензирование (Double-blinded Peer-Review) (см. также ***Рецензирование***) – процесс рецензирования, при котором автор и рецензент не знают имен друг друга. Этот метод используется во многих журналах для предупреждения предвзятой оценки рецензента в отношении некоторых авторов. Имя рецензента

скрывают, чтобы гарантировать ему возможность свободно выражать свою позицию по отношению к статье, не опасаясь возмездия со стороны автора.

Доработка с существенными исправлениями (Major Revision) – решение редакции, которое требует внести существенные структурные изменения в рукопись, прежде чем она может быть допущена до публикации

Доработка с несущественными исправлениями (Minor Revision) – решение редакции, требующее от автора внесения минимальных исправлений в рукопись (исправление орфографических ошибок или внесение изменений в форматирование), прежде чем она будет допущена к публикации.

Ж

Журнал (Journal, Magazine) – периодическое издание по определенной теме. Журналы различаются по размеру тиража и объему представленных материалов и охватывают все вопросы, изучаемые в академических и исследовательских, а также профессиональных областях. В английском языке существует четкое разграничение понятий: Journal – научный журнал; Magazine – отраслевой, научно-популярный журнал. Отраслевой (научно-производственный, специализированный) журнал имеет также перевод Trade Journal.

Журнал открытого доступа (Open Access Journal) – журнал, финансовая модель которого не предполагает взимания платы за доступ с читателей или с представляющих их интересы учреждений (автор определения – Ларс Бьорнсхауг).

З

Заглавие статьи (Title) – главная часть метаданных, формулирующая основную тему публикации и размещаемая в самом начале статьи на титульной странице.

Заказная статья (Invited Submission, Invited Article) – рукопись, написанная по заказу редактора. Автора, как правило, выбирают по наличию авторитета в научной области. Заказывают обычно обзоры, чтобы закрыть пробелы в информации в определенной области, или работу конкретного автора, чтобы привлечь внимание к конкретному небольшому журналу, в котором будет опубликована такая статья.

Заключение рецензента (Reviewer Conclusion) – подробный ответ автору по его статье. Как правило, включает замечания и комментарии рецензента, редактора и в некоторых случаях главного редактора.

Заочная конференция – не существующие мероприятия. Под таким названием отдельные компании и просто частные лица приглашают авторов к публикации в сборнике трудов так называемой конференции за определенную плату, привлекая скоростью публикации и оперативным размещением в РИНЦ. Определить статус «заочной», то есть отсутствующей конференции, можно легко. Если даже обозначено место проведения, но нет регистрации участников, программы конференции, запланированных мероприятий, сайта конференции, а есть только сбор статей в сборник, это уже говорит о том, что конференция является фиктивной. Короткие сроки между «объявлением» конференции (1–2 месяца) и ее проведением также являются признаком того, что такой конференции в природе нет и быть не может.

И

Иллюстрация (Illustration, Figure)– графическое изображение, поясняющее текст.

Импакт-фактор (ИФ, Journal Impact Factor, JIF) – библиометрический показатель (метрика), доступный на платформе Web of Science, наиболее известная и востребованная метрика научных журналов; присваивается журналам, включаемым в две МНБД WoS CC – SCIE и SSCI. Классическим импакт-фактором считается показатель, рассчитываемый по двухлетнему периоду цитирования, предшествующему году исследования публикаций.

Например, импакт-фактор журнала 2015 г. (JIF-2015) вычислен следующим образом: 2015=A/B, где: A – число цитирований в течение 2015 г. статей и других типов публикаций, опубликованных в данном журнале в 2013–2014 гг., в журналах, индексируемых в этих БД; B – число статей и других типов публикаций, опубликованных в данном журнале за эти же два года (2013–2014 гг.). Также рассчитывается 5-летний импакт-фактор. Полные данные по журналам с импакт-фактором публикуются ежегодно в ***Journal Citation Reports***.

Индекс Херфиндаля-Хиршмана (Herfindal-Hirshman Index, hhi-индекс) – библиометрический показатель (метрика). Рассчитывается как сумма квадратов процентных долей количества статей, опубликованных различными организациями, по отношению к общему количеству статей в журнале в текущем году, в которых организация идентифицирована. Традиционно используется в экономике для расчета степени концентрации определенной отрасли. Чем больше различных организаций, авторы из которых публикуются в журнале, и чем равномернее распределены между ними публикации, тем меньше величина этого показателя. Максимальное значение равно 10000 и достигается, когда в журнале публикуются авторы только из одной организации (http://elibrary.ru/title_profile.asp?id=7350).

Индекс Хирша (Hirsch Index, h-index) – библиометрический показатель (метрика), используется для измерения производительности ученого. Ученый имеет h-индекс равный h, если h из его Np публикаций имеют, по крайней мере, h цитирований каждая, а остальные (Np-h) имеют количество цитирований не больше h. Множество всех публикаций, удовлетворяющих данному критерию, называют h-ядром (Hirsch core).

Индексирование (Индексация, Indexing) – 1. Процесс извлечение данных из публикаций для пополнения базы данных для последующего поиска информации. 2. В информационном поиске – процесс описания документов и запросов в терминах информационно-поискового языка. По результатам индексирования каждому документу назначается набор ключевых слов, отражающих его смысловое содержание.

Индекс цитирования (указатель цитирования, Citation Index) – библиографическая/ реферативная база данных научных публикаций, индексирующая ссылки, указанные в пристатейных списках этих публикаций и предоставляющая количественные показатели этих ссылок. Термин также употребляется в значении «совокупная цитируемость публикаций» (автора, организации и т.д.).

Институт научной информации США (Institute for Scientific Information, ISI) – компания, созданная Ю. Гарфилдом в 1960 г. ISI является разработчиком и первым владельцем наукометрических баз данных ***Science Citation Index***, ***Social Science Citation Index*** и ***Arts & Humanities Citation Index***, а также приложения к ним – ***Journal Citation Reports***, которые затем вошли в коллекцию баз данных ***Web of Science Core Collection.***

Институциональный репозиторий (Institutional repository) – электронный архив для длительного хранения, накопления и обеспечения долговременного и надежного открытого доступа к результатам научных исследований, проводимых в учреждении.

Инструкции для авторов (Author Guidelines), см. также ***Правила для авторов*** **–** набор руководств, которые помогают автору подготовить статью в соответствии с требованиями стиля журнала. Инструкции для авторов также включают важную информацию о нюансах подачи рукописи.

Инфометрия (Infometrics) – научная дисциплина, предметом которой являются измерения количественных характеристик информации. В состав инфометрии входят наукометрия, библиометрия, вебометрика *(Nacke O. Informetrie: Ein neuer name für eine neue disziplin // Nachrichten für Dokumentation. 1979. № 30 (6). PP. 219–226).*

Информационная модель науки – модель, которая рассматривает науку как «самоорганизующуюся систему, управляющуюся своими информационными потоками.

Развитие науки изучается как развитие ее информационных потоков» (автор определения – В.В. Налимов) (*Налимов, В.В. Наукометрия / В.В. Налимов, З.М. Мульченко. М.: Наука, 1969. 192 с.).*

Исправленная рукопись (Revision/ Revised Manuscript) – после первой экспертной оценки от автора может потребоваться внести изменения и исправления в рукопись, чтобы журнал мог принять ее к публикации. Рекомендованные исправления могут касаться как синтаксиса и грамматики, так и быть более серьезными: изменить дизайн исследования, обновить или добавить данные и т.д.

К

Квартиль (Quartile) – четвертая часть всей совокупности данных выборки, представленной в виде информационного ряда (в статистике). Используется в МНБД при ранжировании журналов по ***импакт-фактору*** (WoS) или ***SJR*** (по «корзине метрик») Scopus.

КиберЛенинка – научная электронная библиотека, построенная на парадигме открытой науки (Open Science), включает более 1 млн статей из российских журналов. Официальный сайт: http://cyberleninka.ru/

Ключевые слова (Keywords) – основные термины, характеризующие содержание статьи или другой публикации. Вносятся авторами и относятся к метаданным публикации. Ключевые слова используются для поиска в базах данных документов по определенной теме.

Коллаборация (Collaboration, сотрудничество), см. также **Международная коллаборация –** совместная деятельность (процесс), в какой-либо сфере, двух и более человек или организаций для достижения общих целей, при которой происходит обмен знаниями, обучение и достижение согласия (консенсуса).

Комитет по публикационной этике *(см. Committee on Publication Ethics, COPE*)

Комментарии для автора (Comments to Author) – комментарии, сделанные рецензентом или ответственным редактором во время экспертизы, включают пометки и вопросы, требующие внимания автора.

Контактный автор (Corresponding Author) – автор, указанный в статье как лицо, к которому можно обращаться по любым вопросам, связанным со статьей. Контактный автор, как правило, уполномочен корректировать финальную версию статьи, вести переписку и работать с выпускающим редактором. Ранее контактный автор отправлял оттиски статьи по требованию, но эта практика уходит в прошлое.

Конфликт интересов (Conflict of Interest, см. также ***Положение о конфликте интересов***) – потенциальные интересы, направление работы или связь с другим объектом, которые могут повлиять или исказить решение автора, рецензента или редактора. Для биомедицинских журналов, в частности, авторы часто связаны рабочими отношениями с компаниями, чьи продукты обсуждаются в статье. Также известен конкурирующий интерес. В областях, где наличие конфликта интересов может иметь значение, большинство журналов требуют, чтобы авторы указали на наличие конфликта интересов.

Краткое сообщение – жанр материала, публикуемого в научном журнале. Как правило, содержит информацию о новых исследованиях.

М

Международная коллаборация (International Collaboration) – сотрудничество ученых двух или более стран для совместного выполнения научных исследований и публикации результатов в соавторстве. Признано, что публикации международных коллабораций ученых получают значительно больше цитирования, чем публикации авторов одной страны.

Международные наукометрические базы данных (Глобальные индексы цитирования, МНБД) **–** базы данных, индексирующие метаданные научных и прочих публикаций, в т.ч.

их ссылки друг на друга, используются для поиска информации, служат источниками библиометрических/наукометрических исследований. К числу индексируемых метаданных относятся ***абстракты***, ***аффилиации***, ***ключевые слова***, ***DOI***, ***списки литературы***, сведения о грантовой поддержке, авторские идентификаторы типа ***ORCID*** и т.д. Основными МНБД, универсальными по охвату тематических направлений, являются ***Web of Science Core Collection*** компании ***Clarivate Analytics*** и ***Scopus*** издательства ***Elsevier***. Существует также большое количество отраслевых МНБД (MathSciNet в математике, Astrophysics Data System в астрономии и т.д.), однако они не обладают таким широким спектром аналитических инструментов, как две основных МНБД.

Метаданные (статьи, издания, Metadata) – краткая информация о статье (книге, трудах конференций и т.д.), включаемая в библиографические/ реферативные базы данных и другие поисковые системы и позволяющая проводить отбор документов и их анализ. К метаданным научной статьи (издания) относятся: заглавие статьи, фамилии авторов, аннотация, ключевые слова, сведения об источники издания (выходные данные статьи), DOI. В индексах цитирования к метаданным можно отнести другие сведения: места работы авторов (аффилиацию), списки литературы, сведения о финансировании.

Методы (Methods) – раздел статьи, который содержит информацию о методах сбора данных. . Методы должны быть описаны таким образом, чтобы дать возможность другим коллегам их использовать для получения того же результата, который описан в статье.

Метрика (Metrics, *см. Библиометрический/ наукометрические индикаторы)*

Множественные публикации (Multiple Submissions) – также известны как «salami slicing», автор искусственно увеличивает количество своих публикаций, разделяя результаты, которые могли бы быть опубликованы в одной статье, на несколько статей. Часто такие документы содержат и другие нарушения публикационной этики.

Н

Название рукописи, название статьи (Title) (см. *Заглавие статьи*)

Нарушение этики (Misconduct, Ethics Violations) – поведение, которое определяется как неэтичное. Как правило, нарушения касаются поведения автора (плагиат, сомнительное авторство, сокрытие необходимой информации), но иногда это определение применяют и для характеристики поведения рецензента (например, согласие на рецензирование статьи, в отношении которой есть конфликт интересов).

Наукометрия (Scientometrics) – область знания, занимающаяся изучением науки количественными методами. Термин введен в 1969 г. В.В. Налимовым. Определение основано на кибернетическом подходе и информационной модели науки (*Налимов В.В. Наукометрия / В.В. Налимов, З.М. Мульченко. М.: Наука, 1969. 192 с.).*

Научная публикация (Scientific publication) – опубликованная в научном издании часть или полные результаты научного исследования, представляющая его промежуточный или конечный научный результат.

Научная статья (Article) – опубликованное в составном научном издании (периодическом или продолжающемся издании, сборнике статей) авторское произведение, описывающее результаты промежуточного или законченного оригинального научного исследования (первичная научная статья) или посвященное рассмотрению ранее опубликованных научных статей, связанных общей темой (систематический обзор).

Научное издание (Scientific publication) – издание, публикующее результаты теоретических и/или экспериментальных научных исследований.

Научные коммуникации (Scientific communication) – система продвижения сформулированных научных идей, подтвержденных теоретическими и экспериментальными исследованиями внутри научного сообщества, включения их в

процесс распространение научных знаний об окружающей действительности посредством различных каналов, средств, форм и институтов коммуникации. Научные коммуникации – совокупность видов профессионального общения в научном сообществе, один из главных механизмов развития науки, способа осуществления взаимодействия исследователей и экспертизы полученных результатов.

Научный журнал (Journal, Academic/ Scholarly Journal, Scientific Journal) – периодическое издание, публикующее результаты теоретических или экспериментальных научных исследований, по определенной тематике и для определенной читательской аудитории.

Недобросовестные журналы (Questionable journals, Predatory Journals)– журналы, нарушающие этические нормы.

Неформальное цитирование (Informal citations) – указание источника информации в тексте работы без включения его в список литературы.

О

Обзорная научная статья (Review, Survey) – тип статьи (иногда называют систематическим обзором, мета-анализом), который включает в себя изучение большого объема литературы и выводы относительно ранее опубликованных материалов.

Одновременная подача рукописи/ «веерная рассылка» (Duplicate Submission) – множественная подача на рассмотрение одной и той же статьи в один и тот же промежуток времени. Одновременная подача относится к признакам неэтичного поведения авторов и в случае параллельной публикации ведет к ретракции статей журналами, заинтересованными в оригинальном материале и заботящихся о своей репутации.

Оригинальная научная статья (Original article) – тип статьи, публикующей промежуточные или окончательные результаты проведенного научного исследования.

Основные положения (Highlights) – раздел статьи в печатной и/или электронной версии журнала. Отражает ключевые результаты исследования, основное содержание статьи, изложенные тезисно и оформленные в виде 3–5 пунктов маркированного списка.

Отзыв статьи (Retraction) – решение редакции или авторов. Редакторы прибегают к отзыву статьи в случае если нарушены этические нормы. При ретрагировании статья остается в журнале, но в информации о статье, в файле, в базах данных указывается, что статья отозвана. Причиной для отзыва статьи может быть дублирующая публикация, обнаруженный плагиат, серьезные ошибки в исследовании и другие нарушения. Автор в случае признания своей вины также может быть инициатором отзыва статьи. Авторы могут отозвать статью также в случае обнаружения серьезных ошибок в опубликованных результатах и по другим причинам, связанным с выполнением исследования.

Отказ (Reject) – решение редакции журнала не публиковать рукопись. Обычно это решение свидетельствует о том, что рукопись не соответствует минимальным критериям для возможности публикации.

Открытое рецензирование (Open Peer-Review) (см. также ***Рецензирование)*** – 1. Тип рецензирования, когда автор и рецензент в процессе рецензирования знают фамилии друг друга. 2. Содержание рецензий находится в открытом доступе (публичное рецензирование, public peer-review). 3. Открытое приглашение (Open-invitation) подразумевает возможность для любого человека оставить комментарий к рукописи и предложить свое решение редактору. Является альтернативой традиционному рецензированию, в отличие от него не является замкнутым процессом, в котором журнал сам выбирает рецензента для статьи. Иногда проводится после издания публикации как «пост-публикационное рецензирование» (https://en.wikipedia.org/wiki/Scholarly_peer_review).

Открытый доступ (Open Access) – возможность получить доступ к рукописи бесплатно и без технических ограничений. Некоторые журналы публикуют контент в открытом

доступе, и при этом взимают плату с авторов. Некоторые журналы делают контент бесплатным по прошествии определенного времени (с «эмбарго», такой журнал не считается журналом открытого доступа). Некоторые финансирующие организации требуют от авторов, чтобы все материалы, опубликованные с результатами профинансированных исследований, находились в открытом доступе.

Отстранение автора (Ban of Author) – лишение автора возможности подачи статьи в конкретный журнал из-за этических нарушений на определенный период времени.

П

Парафраз (Paraphrase) – 1. Краткое изложение объемной теоретической концепции или обобщенную информацию при ссылке на несколько авторов или источников информации. 2. Пересказ, изложение текста своими словами.

Периодическое издание (Periodical, Serial) – издание, выходящее через определенные промежутки времени, как правило, с постоянным для каждого года числом номеров (выпусков), не повторяющимися по содержанию, однотипно оформленными, нумерованными и (или) датированными выпусками, имеющими одинаковое заглавие.

Перцентиль (Percentile) – мера статистического измерения, которая показывает значение, в которое попадает данный процент наблюдений в группе. В ***Journal Citation Reports (JCR)*** перцентиль показывает, сколько процентов журналов (например, в предметной категории) имеют более низкий ***импакт-фактор*** *(*https://en.wikipedia.org/wiki/Percentile*).*

Письмо об отказе (Rejection Letter) – письмо, в котором редакция журнала сообщает о своем решении не публиковать рукопись. Некоторые журналы используют письмо об отказе, чтобы информировать авторов о том, что они не должны заново подавать рукопись на рассмотрение; некоторые журналы предлагают внести изменения в статью и после обновления материала подать статью на рассмотрение еще раз.

Письмо о принятии к публикации (Acceptance Letter) – письмо-подтверждение о принятии к публикации, которое отправляется из редакции журнала автору статьи. Письмо может содержать официальные формы, которые необходимо заполнить (авторский договор, заказ публикации цветных изображений в статье, заказ дополнительных экземпляров). Письмо также может содержать информацию о том, что автор должен будет сделать в будущем (например, вовремя вернуть корректуру рукописи).

Письмо о решении (Decision Letter) – уведомление автора о судьбе его статьи. Как правило, решения могут быть следующими: Принять; Принять с незначительными доработками; Отправить на доработку; Отказать; Отказать и рекомендовать подать статью на рассмотрение после устранения всех нарушений.

Письмо редактору (Letter to the Editor) – письмо, в котором автор перечисляет достоинства или недостатки ранее опубликованной статьи, вступает в полемику с другим автором. Некоторые журналы публикуют такие письма в качестве статьи.

Плагиат (Plagiarism) – акт присвоения чужой работы и позиционирование ее как своей собственной. Журналы периодически получают рукописи, некоторые части которых были скопированы из других работ. Плагиат является очень серьезным этическим преступлением. Авторы могут столкнуться с серьезными дисциплинарными взысканиями, если плагиат будет обнаружен.

Подарочный автор (Gift author) – автор, не соответствующий принятым критериям авторства, но внесенный в список литературы благодаря личным отношениям или за плату.

Положение о конфликте интересов (Conflict of Interest Statement) – раздел статьи. Многие журналы требуют от всех авторов указывать на любой потенциальный конфликт интересов. Авторов просят объявить любые финансовые и личные отношения к третьим

лицам, чьи интересы могут положительно или отрицательно влиять на содержание статьи, даже если автор считает, что никакого влияния не должно быть. В своих руководствах ведущие издательства подробно описывают, что подразумевается под «конфликтом интересов» (см. подробнее: https://www.elsevier.com/__data/assets/pdf_file/0010/92476/ETHICS_COI02.pdf; http://www.springer.com/cda/content/document/cda_downloaddocument/481_Guideline_Conflict+of+Interest_Ethik+in+der+Medizin.pdf?SGWID=0-0-45-1342723-p1085184). Сокрытие имеющегося конфликта интересов рассматривается как нарушение норм этики публикации (http://icmje.org/recommendations/browse/roles-and-responsibilities/author-responsibilities--conflicts-of-interest.html).

Правила для авторов (Author Guidelines, также **Руководство для авторов**, **Инструкция для авторов) –** требования для авторов, в которых описаны детали написания и оформления рукописи по правилам, принятым журналом. Более расширенные варианты инструкций могут также включать информацию о статистической отчетности, методологии, публикационной этике и качественных рисунках, пригодных для печати.

Прайс, Дерек Джон де Солла (Price, Derek John de Solla) (22.01.1922 – 03.09.1983) – известный физик и историк науки, считающийся одним из основателей ***Наукометрии (Scientometrics)***.

Препринт – научное издание, содержащее материалы предварительного характера, опубликованные до выхода в свет издания, в котором они могут быть помещены.

Приложение (Appendix) – дополнительный материал (например, большие наборы данных в табличной форме, примеры вопросников), который обычно слишком велик по объему для размещения в теле статьи и прерывает повествование. Приложения обычно размещаются в конце статьи или публикуются только онлайн.

Принять к публикации (Accept) – финальное решение о публикации статьи, принятое журналом. Принятие к публикации означает, что редакторы и рецензенты начинает работать над статьей для ее последующей публикации в журнале.

Прямое цитирование – дословное воспроизведение отрывка из чужого текста.

Р

Рабочее название (Working Title) – название, которое используется при подготовке черновика рукописи. Название меняется или становится более формализованным после окончания работы над статьей.

Разрешение на повторное использование (Permission) – юридический термин. Если в рукописи публикуются рисунки, таблицы или иные материалы, ранее опубликованные, автор последующей публикации должен получить разрешение на повторное использование у владельца авторских прав. Как правило, издатель ранее опубликованной работы может дать разрешение на воспроизведение материала, но при этом может попросить, чтобы при повторном использовании материала стояла ссылка на первоисточник.

Ретрагирование (Retraction) (см. ***Отзыв статьи)***

Реферат (авторское резюме, аннотация, абстракт, Abstract) – краткое точное изложение содержания документа, включающее основные фактические сведения и выводы, без дополнительной интерпретации или критических замечаний автора реферата.

Рецензент (Reviewer) – эксперт в предметной области, приглашенный или назначенный редактором для оценки рукописи. Задача рецензента – оценить уровень статьи, ее сильные и слабые стороны, указать на ошибки и неточности, предложить для исследования дополнительный материал, если автор упустил его в процессе работы, а также рекомендовать или не рекомендовать материал к публикации.

Рецензирование (Peer-Review) – процесс оценки рукописи экспертами в определенной предметной области. В некоторых случаях рецензенты знают имена авторов и авторы

знают имена рецензентов (открытое рецензирование). При двойном слепом рецензировании имя автора не называется рецензенту. При одностороннем слепом рецензировании рецензент знает, кто автор статьи. В обоих случаях имя рецензента не известно автору.

Рецензия (Review) – оценка принятой рукописи, часто коллегами или экспертами в предметной области, для оценки ее сильных и слабых сторон.

Российский индекс научного цитирования (РИНЦ) – национальная информационно-аналитическая система, аккумулирующая более 6 млн публикаций российских авторов, а также информацию о цитировании этих публикаций из более 4500 российских журналов.

Рукопись (Manuscript) – объединение текста, таблиц и графических файлов; результат научной деятельности, подготовленный к публикации.

С

Самоцитирование (Self-citation) – цитирование собственных работ.

Систематический обзор (Systematic Review) – 1. Тип обзора литературы который собирает и критически анализирует большой охват исследований по определенной теме или большое число публикаций. Сделать обзор существующих исследований часто быстрее и дешевле, чем приступить к новому исследованию (https://en.wikipedia.org/wiki/Systematic_review). 2. Научное исследование ряда опубликованных отдельных однородных оригинальных исследований с целью их критического анализа и оценки. Систематический обзор проводится с использованием методологии, позволяющей исключить случайные и систематические ошибки, обобщающей и интерпретирующей входные данные. В систематическом обзоре используются стандартизированные методы отбора и проверки результатов исследований (например, мета-анализ). Систематические обзоры используются в медицине и в ряде других наук, где целесообразен общий методологический анализ оценка опубликованных данных по конкретным темам. (https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BE%D0%B1%D0%B7%D0%BE%D1%80).

Слепое (анонимное) рецензирование (Blinded) – принятый тип рецензирования, который подразумевает, что имена рецензента и автора скрыты друг от друга. Разновидности слепого рецензирования: одностороннее слепое (рецензент знает имя автора, автор не знает имени рецензента), двойное слепое – имя рецензента неизвестно автору, имя автора неизвестно рецензенту.Это подразумевает, что при направлении на рецензирование редакция должна удалить имена авторов, удаляется также идентифицирующая автора информация (информация об организации автора, информация о финансовой поддержке, номера\шифры клинических исследований, адреса проведения исследований и т.д.). Переписка в таком случае между рецензентом и автором происходит через почту редакции или через редакционную online систему.

Сноски (Footnotes) – дополнительный текст, который публикуется внизу страницы, в котором, как правило, размещается более подробная информация или комментарий автора, редактора или переводчика. В некоторых случаях в сносках публикуются ссылки на процитированные работы.

Сопроводительное письмо (Cover Letter) – письмо, которое автор прикладывает к подаваемой статье в выбранный журнал. В сопроводительном письме автор может объяснить, почему считает выбранный журнал подходящим для публикации; попросить редактора ускорить процесс рассмотрения; добавить обязательную информацию – например, согласие пациента, подтверждение корректности данных и т.д. Советы по подготовке Cover Letters: https://www.springer.com/gp/authors-editors/authorandreviewertutorials/submitting-to-a-journal-and-peer-review/cover-letters/10285574.

Список литературы (Пристатейная библиография, Библиографический список, Библиография, References) – раздел рукописи, статьи, другой публикации, в котором перечислены опубликованные источники, процитированные в основном тексте рукописи.
Стиль цитирования (Reference Style) – синтаксис, в котором отформатирована ссылка.
Структурированная аннотация/реферат (Structured Abstract) – структурированная аннотация/реферат представляет собой краткое содержание статьи, разделенное на несколько информационных блоков. Как правило, они повторяют структуру самой статьи – Введение, Материалы и Методы, Результаты, Выводы и т.д.

Т

Титульная страница статьи (Title Page) – начальная часть статьи. Содержит: ***заглавие статьи***, список авторов и контактную информацию ***контактного автора***. Некоторые журналы могут просить включать следующую информацию: текущий заголовок, ***положение о конфликте интересов***, заявление о финансовой поддержке, ***благодарности***, количество слов, таблиц и рисунков.
Тип рецензирования (Peer-Review type, см. *Рецензирование)*
Типы научных публикаций (Document types) – определение опубликованному научному тексту в зависимости от его содержания, назначения, источника публикации и целевой аудитории. Основным признаком научной публикации является ее научный контент (промежуточные или конечные результаты исследований, обзор своих результатов и других исследований по теме и т.п.) и, как правило, рецензирование (исключение составляют препринты, публикуемые с целью оперативного информирования коллег о полученных результатах и признания авторского приоритета на полученные результаты). Нет стандартизованного перечня типов научных публикаций. К основным типам можно отнести следующие: научная статья (оригинальная или аналитический, систематический обзор), краткие сообщения, научные заметки, опубликованные доклады на конференциях, препринты и пост-принты (http://help-infoscience.epfl.ch/deposit/document-types). В одно издание (журнал) может быть включено несколько типов научных публикаций. Не все типы публикаций, включаемые в журнал, относятся к научным (персоналии, информационные и рекламные материалы не являются научными материалами). Следует отличать типы научных публикаций и ***виды научных изданий (Source types)***. Тип научной публикации и вид издания являются определяющими характеристиками при отборе документов и изданий в МНБД и другие реферативные базы данных научных публикаций.

У

Управляющий редактор (Managing Editor) – наемный работник в редакции, в область ответственности которого входит, как правило, управление поступающими рукописями на всех этапах подготовки к публикации, рецензирования и отправки в издательство для редакционной подготовки.

Ц

Целевой научный журнал (Target Journal) – журнал, соответствующий тематике и уровню представляемой статьи.
Цитирование (Citedness, Citation, Quotation) – заимствование фрагментов текстов (формул, иллюстраций, таблиц и других элементов) автором в своей работе из других источников с обязательным указанием источника, в том числе, информации об авторах, названии работы, выходных данных журнала/издательства и т.д. Цитирование является обязательным компонентом любой научной работы и одним из важных средств научной коммуникации.
Цифровой идентификатор объекта (см. *Digital Object Identifier*, *DOI*)

Ч

Чек-лист для подачи статьи на рассмотрение (Submission Checklist) – редакционный документ. Некоторые журналы настаивают на том, чтобы автор полностью подготавливал рукопись к подаче статьи на рассмотрение по стандартам журнала. В чек-листах обычно прописывают перечень необходимых сведений и данных, которые автор должен направить редактору вместе со статьей.

Черновик (Draft) – любая версия рукописи до ее финального варианта.

Чистая копия (Чистовик) (Clean Copy) – версия принятой рукописи, в которую внесены все предложенные изменения и исправления.

Э

Электронная версия печатного издания – электронное издание, в основном воспроизводящее соответствующее печатное издание (расположение текста на страницах, иллюстрации, ссылки, примечания и т.п.).

Электронное научное издание (Electronic Scientific Publication) – электронное издание, содержащее сведения о теоретических и (или) экспериментальных исследованиях, а также научно подготовленные к публикации памятники культуры и исторические документы. Возможности электронного научного издания шире, чем возможности научного печатного издания. Так, издатель имеет возможность добавлять к статьям дополнительные материалы (видеоаннотации, наборы данных), использовать альтметрики для оценки востребованности журнала, давать доступ к изданию неограниченному кругу лиц в один момент времени, индексировать журнал в поисковых системах и базах данных и т.д.

Этика (Ethics) – научная дисциплина. В приложении к публикационному процессу охватывает ряд вопросов, которые возникают в связи с сомнениями в достоверности результатов или идей, представленных в статье, а также в корректности действий авторов, редакторов и рецензентов. Публикационная этика охватывает широкий круг вопросов, начиная от споров об авторстве, неполного раскрытия конфликта интересов, до более серьезных проблем, таких как фальсификация или плагиат.

ПРИЛОЖЕНИЕ 2

Основные требования к подаваемым рукописям в журналах ведущих зарубежных издательств по областям науки

ГУМАНИТАРНЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	SAGE	SPRINGER
Аннотация	Краткая аннотация 100–150 слов, должна содержать цель исследования, основные результаты и выводы.	Не более 220 слов	100–150 слов	150–250 слов, должна содержать цель исследования, основные результаты и выводы.
Ключевые слова	Максимум от 6 до 10 ключевых слов	Ограничений не установлено	5–10 ключевых слов	4–6 ключевых слов
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.1.2, ...), 1.2 и т.д.* *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения).	Рукопись должна быть представлена в формате А4, двойной интервал, поля 3 см. Объем статьи – максимум 10000 слов, включая сноски. Абзацы должны иметь отступ или четко обозначены. Требования к структуре статьи не установлены.	Объем статьи между 4000 и 7000 слов. Тематические исследования должны быть максимум 6500 слов. Текст должен быть структурирован, с четкой иерархией заголовков и подзаголовков. Требования к структуре статьи не установлены.	Структурируйте текст, используя 3-х уровневую (и более) систему рубрик. – Здесь понимается, что статья не должна идти сплошным тестом, а разделена на подзаголовки от трех и более. Требования к структуре статьи не установлены.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	SAGE	SPRINGER
	Результаты (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как A, B и т.д. У формул и уравнений в приложениях отдельная нумерация.			Для выделения отдельных структурных частей использовать курсив. Пронумеровывать страницы (автоматически с помощью функции Word, не самостоятельно). Файл в формате DOCX (Word 2007 или выше) или DOC (для более ранних версий Word).
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы:	Авторы должны предоставить электронные версии рисунков в отдельном файле. Подписи рисунков отдельно (не на самом рисунке). Иллюстрации нумеруются в соответствии с их последовательностью в тексте.	Иллюстрации могут быть цветными. Необходимо предоставить рисунки с высоким разрешением в отдельном файле.	Рисунки, созданные в MS Office, являются приемлемыми. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Шрифт *Helvetica* или *Arial*.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	SAGE	SPRINGER
	TIFF, JPEG, PDF. Подписи к рисункам отдельно (не на самом рисунке). Представить каждый рисунок в отдельном файле.			
Таблицы	Таблицы нумеруются в соответствии с их последовательностью в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты, описанные в статье. Не более 8 таблиц/рисунков в совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.	Таблицы нумеруются в соответствии с их последовательностью в тексте. У каждой таблицы должны быть подписи. Ограничение на количество таблиц не устанавливаются.	Таблицы необходимо предоставить в отдельном файле. Ограничение на количество таблиц не устанавливаются.	Таблицы нумеруются в соответствии с их последовательностью в тексте. Все таблицы должны быть пронумерованы арабскими цифрами. Для каждой таблицы должно быть указано название. Ограничение на количество таблиц не устанавливаются.
Благодарности	Благодарности людям, финансирующим организациям, их названия, номера грантов и т.д. должны быть помещены в отдельном разделе на титульной странице либо на последней странице. Названия финансирующих организаций должны быть написаны полностью.			
Список литературы	Список литературы должен включать только те работы, которые упоминаются в тексте и	Указать полные имена, а не инициалы, где возможно. В научных статьях список литературы должен содержать	Гарвардский стиль оформления списка литературы	Список литературы должен включать только те работы, которые упоминаются в тексте и

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	SAGE	SPRINGER
	которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте. Описания источников в списке следуют в алфавитном порядке по фамилии первого автора каждой работы.	значимые источники, в обзорных статьях – список литературы не ограничен.		которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте. Описания источников в списке следуют в алфавитном порядке по фамилии первого автора каждой работы.
Использованные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Brain and Language, https://www.elsevier.com/journals/brain-and-language/0093-934x/guide-for-authors 2. Language & Communication, https://www.elsevier.com/journals/language-and-communication/0271-5309/guide-for-authors	Author Resource Centre, http://www.mat.net.ua/yuliya-article/autor-mark-2015.htm (см. разделы Figures, Instructions to Authors, LaTex) 1. History Workshop Journal, https://academic.oup.com/hwj/pages/General_Instructions 2. Journal of Social History, https://academic.oup.com/jsh/pages/For_Authors	1. Arts and Humanities in Higher Education, https://uk.sagepub.com/en-gb/eur/journal/arts-and-humanities-higher-education#submission-guidelines 2. Discourse & Society, https://uk.sagepub.com/en-gb/eur/discourse-society/journal200873%20#submission-guidelines	1. Biology & Philosophy, http://www.springer.com/philosophy/epistemology+and+philosophy+of+science/journal/10539 2. Linguistics and Philosophy, http://www.mat.net.ua/yuliya-article/autor-mark-2015.htm

СОЦИАЛЬНЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Аннотация	До 150 слов, должна содержать цель исследования, основные результаты и выводы	Авторы должны предоставить структурированную аннотацию (4–7 подзаголовков): Цель (обязательно) методология/подход (обязательно) Выводы (обязательно) Ограничения исследования/последствия (если применимо) Практические последствия (если применимо) Социальные последствия (если применимо) Оригинальность/ценность (обязательно). Всего максимум 250 слов (включая ключевые слова и классификацию). Авторы должны избегать использования личных местоимений.	Не более 150 слов	150–250 слов, должна содержать цель исследования, основные результаты и выводы	До 150 слов, должна содержать введение в тему, методы, результаты, выводы
Ключевые слова	До 6 ключевых слов	До 12 ключевых слов	До 6 ключевых слов	4–6 ключевых слов	Ограничения не установлены

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.1.2, ...), 1.2 и т.д.* *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как A, B и т.д. У формул и уравнений в приложениях отдельная нумерация.	Объем статьи максимум 12000–13000 слов (5000–8000 слов), включая основной текст, список литературы и приложения. Каждый рисунок или таблица считается как 280 слов. Требования к структуре статьи не установлены.	Не более 35 печатных страниц (460 слов на странице, включая рисунки и таблицы) Требования к структуре статьи не установлены.	Структурируйте текст, используя 3-х уровневую (и более) систему рубрик. Здесь понимается, что статья не должна идти сплошным тестом, а разделена на подзаголовки от трех и более. Требования к структуре статьи не установлены. Для выделения отдельных структурных частей использовать курсив. Пронумеровывать страницы (автоматически с помощью функции Word,	Присланная статья должна быть не более 40 страниц, включать в себя: заголовок, аннотацию, текст (введение в тему, методы, результаты, выводы), сноски, ссылки на источники, таблицы и изображения.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
				не самостоятельно). Файл в формате DOCX (Word 2007 или выше) или DOC (для более ранних версий Word).	
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстраций нумеруются в соответствии с их последовательностью в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы: TIFF, JPEG, PDF. Подписи рисункам отдельно (не на самом рисунке). Представить каждый рисунок в отдельном файле.	Все рисунки (графики, диаграммы, рисунки, скриншоты, и фотографии) должны быть представлены в электронном виде в отдельном файле. Все рисунки должны быть качественными, читаемыми и пронумерованы в соответствии с последовательностью в тексте и пронумерованы арабскими цифрами.	Рисунки предоставить в отдельном файле с высоким разрешением, особенно если они построены в Excel или Power Point.	Рисунки, созданные в MS Office, являются приемлемыми. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Шрифт *Helvetica* или *Arial*.	Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Обязательны упоминания их в тексте. (Например: Как показано на рис. 1).

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Таблицы	Таблицы нумеруются в соответствии с их последовательностью в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты, описанные в статье. Не более 8 таблиц/рисунков в совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.	Таблицы должны быть предоставлены в отдельном файле. На каждую таблицу в тексте должно быть указание (Например: как показано в табл. 1…) Ограничение на количество таблиц не устанавливается.	Всегда указывать первоначальный источник, даже если это «расчеты авторов». Все таблицы должны иметь двойную линию вверху и внизу; внутри таблицы использовать одинарные линии, чтобы разграничить разделы, заголовки и т.д. Ограничение на количество таблиц не устанавливается.	Таблицы нумеруются в соответствии с их последовательностью в тексте. Все таблицы должны быть пронумерованы арабскими цифрами. Для каждой таблицы должно быть указано название. Ограничение на количество таблиц не устанавливается.	Таблицы нумеруются в соответствии с их последовательностью в тексте. Обязательны упоминания их в тексте. (Например: Как показано в табл. 1). Ограничение на количество таблиц не устанавливается.
Благодарности	Указать наименование организации, которая предоставила финансирование, номера грантов писать необязательно.	Авторы несут ответственность за указание источников внешнего финансирования научных исследований в своей статье. Авторы должны описать всех спонсоров.	Указать все свои источники финансирования, а также их отсутствие.	Должны быть размещены в отдельном разделе на титульном листе. Названия финансирующих организаций должны быть	Предоставить информацию на отдельном листе, который не включается в рукопись.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
				написаны полностью.	
Список литературы	Нет жестких требований по форматированию ссылок при подаче. Ссылки могут быть в любом одном формате. Настоятельно рекомендуется использование DOI при указании источника.	Гарвардский стиль оформления списка литературы	В научных статьях список литературы должен содержать значимые источники, в обзорных статьях – список литературы не ограничен. Если авторы хотят указать источники специализированной литературы, то могут разместить их в приложении.	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте. Описания источников в списке следуют в алфавитном порядке по фамилии первого автора каждой работы.	Нумерация идет в том порядке, в котором они впервые упоминаются в тексте.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	EMERALD	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Использован-ные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Accounting, Organizations and Society, https://www.elsevier.com/journals/accounting-organizations-and-society/0361-3682/guide-for-authors 2. European Economic Review, https://www.elsevier.com/journals/european-economic-review/0014-2921/guide-for-authors	1.Accounting, Auditing & Accountability Journal, http://www.emeraldgrouppublishing.com/products/journals/author_guidelines.htm2. International Journal of Sociology and Social Policy, http://www.emeraldgrouppublishing.com/products/journals/author_guidelines.htm?id=ijssp	1. Author Resource Centre, http://www.oxfordjournals.org/en/authors/index.html (см разделы Figures, Instructions to Authors, LaTex) 2. Economic Policy, https://academic.oup.com/economicpolicy/pages/General_Instructions 3. Socio-Economic Review, https://academic.oup.com/ser	1. Experimental Economics, http://www.oxfordjournals.org/en/authors/index.html 2. Journal of Economic Growth, http://www.springer.com/economics/growth/journal/10887	1. American Anthropologist, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1548-1433/homepage/ForAuthors.html 2. City and Community, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1540-6040/homepage/ForAuthors.html

ТОЧНЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Аннотация	До 150 слов, должна содержать цель исследования, основные результаты и выводы	До 150 слов, должна содержать ведение в тему, краткое нетехническое изложение	Должна содержать цель исследования, основные результаты и выводы	150–250 слов, должна содержать цель исследования, основные результаты и выводы	Не более 250 слов, в научной и обзорной статье должна включать цель исследования,

		основных результатов, их практическое использование			основные результаты и выводы
Ключевые слова	3–6 ключевых слов	Ограничения не установлены	Ограничения не установлены	4–6 ключевых слов	Максимум 10 ключевых слов
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.1.2, ...), 1.2 и т.д.* *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как A, B и т.д. У формул и уравнений в приложениях отдельная нумерация.	Объем статьи 2000–3000 слов (не более 5–6 рисунков и таблиц) *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение	Требования к структуре статьи не установлены	Структурируйте текст, используя 3-х уровневую (и более) систему рубрик. – Здесь понимается, что статья не должна идти сплошным тестом, а разделена на подзаголовки от трех и более. Требования к структуре статьи не установлены. Для выделения отдельных структурных частей использовать курсив. Пронумеровывать страницы (автоматически с помощью функции Word, не самостоятельно).	*Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы).

		результатов работы).		Файл в формате DOCX (Word 2007 или выше) или DOC (для более ранних версий Word).	
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстраций нумеруются в соответствии с их последовательностью в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы: TIFF, JPEG, PDF. Подписи рисункам отдельно (не на самом рисунке). Представить каждый рисунок в отдельном файле.	Представить рисунки в отдельном файле. Используйте символ шрифт для греческих букв. Предпочтительны векторные файлы. Ограничение на количество рисунков не устанавливается.	Представить рисунки в отдельном файле. Количество иллюстраций должно быть сведено к минимуму. Подписи к рисункам должны быть перечислены на отдельном листе.	Рисунки, созданные в MS Office, являются приемлемыми. Иллюстрации нумеруются в соответствии с их последовательность ю в тексте. Шрифт *Helvetica* или Arial. Ограничение на количество рисунков не устанавливается.	Иллюстрации небольшие (5 × 5,5 см) с минимумом деталей. Избегать больших сложных схем. Представить рисунки в отдельном файле. Ограничение на количество рисунков не устанавливается.
Таблицы	Таблицы нумеруются в соответствии с их последовательностью в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты,	Представить таблицы с названиями в отдельном файле. Ограничение на количество	Все таблицы должны иметь название. Сноски могут быть использованы в таблицах, но не в тексте.	Таблицы нумеруются в соответствии с их последовательность ю в тексте. Все таблицы должны быть	Представить таблицы в отдельном файле. Ограничение на количество таблиц не устанавливается.

	описанные в статье. Не более 8 таблиц/рисунков в совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.	таблиц не устанавливается.	Ограничение на количество таблиц не устанавливается.	пронумерованы арабскими цифрами. Для каждой таблицы должно быть указано название. Ограничение на количество таблиц не устанавливается.	
Благодарности	Собирать благодарности в отдельном разделе в конце статьи перед списком литературы и поэтому не включать их на титульном листе, в качестве сноски в названии или иначе	Короткая благодарность. Могут быть указаны номера грантов.	Отдельно, в конце статьи	Благодарность людей, гранты, фонды и т. д. должны быть помещены в отдельном разделе на титульной странице. Имена финансирующих организации должны быть написаны полностью.	Не установлено ограничений
Список литературы	Список литературы представляется в конце статьи на отдельной странице.	Каждая ссылка должна быть отражена в списке литературы. Рекомендовано не более 50 источников.	Ссылки на книги, журнальные статьи, статьи в сборниках и конференции или семинара, процедуры и технические отчеты должны быть перечислены в конце статьи в номерном порядке.	Нумерация идет в том порядке, в котором они впервые упоминаются в тексте. Список всех авторов, когда шесть или меньше; когда семь или более, перечислить трех первых и добавить	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные

				др. Обозначить ссылки в тексте, таблицах и надписях арабскими цифрами (в скобках).	работы должны быть упомянуты в тексте. Ссылки записи в списке следует в алфавитном порядке по фамилии первого автора каждой работы.
Использован-ные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Applied Numerical Mathematics, https://www.elsevier.com/journals/applied-numerical-mathematics/0168-9274/guide-for-authors 2. Computers & Fluids, https://www.elsevier.com/journals/computers-and-fluids/0045-7930/guide-for-authors	1. Nature Nanotechnology, http://www.nature.com/nnano/authors/index.html	1. Author Resource Centre, http://www.oxfordjournals.org/en/authors/index.html (см. разделы Figures, Instructions to Authors, LaTex) 2. IMA Journal of Applied Mathematics, https://academic.oup.com/imamat/pages/General_Instructions 3. International Mathematics Research Notices, https://academic.oup.com/imrn/pages/General_Instructions	1. Mathematische Zeitschrift, http://www.springer.com/mathematics/journal/209 2. Numerische Mathematik, http://www.springer.com/mathematics/computational+science+%26+engineering/journal/211	1. Wiley-Blackwell House Style Guide, https://authorservices.wiley.com/asset/photos/House_style_guide_ROW4520101451415.pdf 2. Communications on Pure and Applied Mathematics, http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1097-0312/homepage/ForAuthors.html 3. Numerical Linear Algebra with Applications, http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-1506/homepage/ForAuthors.html -

ЕСТЕСТВЕННЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Аннотация	Краткая аннотация до 150 слов, должна содержать цель исследования, основные результаты и выводы	Общее введение в тему и краткое изложение ваших основных результатов и их последствий	100–250 слов	150–250 слов, должна содержать цель исследования, основные результаты и выводы	В научной и обзорной статье – не более 250 слов, должна содержать цель исследования, основные результаты и выводы
Ключевые слова	3–6 ключевых слов	Не установлено ограничений	Не более 5–6 ключевых слов	4–6 ключевых слов	Максимум 10 ключевых слов
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.1.2, ...), 1.2 и т.д.* *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими).	Объем статьи не более 3000 слов. *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования	Структура: Введение, материалы и методы (или экспериментальные), результаты (или наблюдения), обсуждения (или заключение) Технические требования: Объем статьи не более 5000 слов (примерно 15 страниц двойного интервала, 12 пт), исключая рисунки, таблицы и ссылки.	Структурируйте текст, используя 3-х уровневую (и более) систему рубрик. Здесь понимается, что статья не должна идти сплошным тестом, а разделена на подзаголовки от трех и более. Требований к структуре статьи не установлены. Для выделения отдельных структурных	*Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
	Выводы (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как A, B и т.д. У формул и уравнений в приложениях отдельная нумерация.	или обсуждение результатов работы).	Пронумеровать страницы.	частей использовать курсив. Пронумеровывать страницы (автоматически с помощью функции Word, не самостоятельно). Файл в формате DOCX (Word 2007 или выше) или DOC (для более ранних версий Word).	исследования или обсуждение результатов работы).
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы: TIFF, JPEG, PDF. Подписи к рисункам	Иллюстрации с высоким разрешением. Допустимые форматы: . PDF, PS, SVG, PSD, TIF, PNG и JPG. Рисунки предоставить в отдельном сопроводительном файле. Подпись менее 200 слов. Не более 6 рисунков и таблиц.	Рисунки должны быть читаемы в черно-белом цвете. Иллюстрации с высоким разрешением. Ограничения на количество рисунков не устанавливаются.	Рисунки, созданные в MS Office, являются приемлемыми. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Шрифт *Helvetica* или *Arial*. Ограничения на количество рисунков не устанавливаются.	Фотографии должны быть небольшие (5 × 5,5 см). Избегать большие сложные схемы. Рисунки и таблицы предоставить в отдельном сопроводительном файле. Ограничения на количество рисунков не устанавливаются.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
	отдельно (не на самом рисунке). Представить каждый рисунок в отдельном файле.				
Таблицы	Таблицы нумеруются в соответствии с их последовательностью в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты, описанные в статье. Не более 8 таблиц/рисунков в совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.	Шрифты *Arial* или *Helvetica* (как и во всей статье). Таблицы предоставить в отдельном сопроводительном файле. Не более 6 рисунков и таблиц.	Все таблицы должны поместиться на одной печатной странице и пронумерованы римскими цифрами. Ограничения на количество таблиц не устанавливаются.	Таблицы нумеруются в соответствии с их последовательностью в тексте. Все таблицы должны быть пронумерованы арабскими цифрами. Для каждой таблицы должно быть указано название. Ограничения на количество таблиц не устанавливаются.	Таблицы и рисунки предоставить в отдельном сопроводительном файле. Каждая таблица должна быть упомянута в тексте (Например: как показано в табл. 1…) Ограничения на количество таблиц не устанавливаются.
Благодарности	Благодарности людям, финансирующим организациям, их названия, номера грантов и т. д. должны быть помещены в отдельном разделе на титульной странице либо на последней странице. Названия финансирующих организаций должны быть написаны полностью.				
Список литературы	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы	Не более 50–70 ссылок. Ссылки на книги, журнальные статьи, статьи в сборниках	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или	Список литературы должен включать только те работы, которые упоминаются в тексте и которые

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
	приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте.	или приняты к публикации.	конференций или семинаров.	приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте. Описания источников в списке следует в алфавитном порядке по фамилии первого автора каждой работы.	были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте.
Использованные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Carbohydrate Polymers, https://www.elsevier.com/journals/carbohydrate-polymers/0144-8617/guide-for-authors 2. Inorganic Chemistry Communications, https://www.elsevier.com/journals/inorganic-chemistry-communications/1387-7003/guide-for-authors	1. Nature Cell Biology, http://www.nature.com/ncb/authors/index.html 2. Nature Chemistry, http://www.nature.com/nchem/authors/index.html	1. Author Resource Centre, http://www.oxfordjournals.org/en/authors/index.html (см разделы Figures, Instructions to Authors, LaTex) 2. Bioscience, https://academic.oup.com/bioscience/pages/General_Instructions 3. Chemical Senses, https://academic.oup.com/chemse/pages/General_Instructions	1. Journal of the American Oil Chemists' Society, http://www.springer.com/chemistry/industrial+chemistry+and+chemical+engineering/journal/11746?print_view=true&detailsPage=pltci_2595148 – 2. Microchimica Acta, http://www.springer.com/chemistry/physical+chemistry/journal/604	1. ChemMedChem, http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1860-7187/homepage/2452_authors.html 2. Electroanalysis, http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1521-4109/homepage/2049_guidelines.html

ИНЖЕНЕРНЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	EMERALD	SAGE
Аннотация	Авторы должны предоставить структурированную аннотацию (4–7 подзаголовков): Цель (обязательно) Методология/подход (обязательно) Выводы (обязательно) Ограничения исследования/последствия (если применимо) Практические последствия (если применимо) Социальные последствия (если применимо) Оригинальность/ценность (обязательно) Максимум 250 слов в общей сложности (включая ключевые слова). Авторы должны избегать использования личных местоимений.	Не должна превышать 200 слов
Ключевые слова	Максимальное количество ключевых слов составляет 12	Ограничения не установлены
Структура статьи	Объем статьи 1000–4000 слов (2000–4000), включая основной текст, список литературы и приложения. В рисунке или таблице можно использовать до 280 слов. Требования к структуре статьи не установлены.	Объем оригинальных и обзорных статей не должен превышать 5000 слов. Введение. Этот раздел должен быть кратким, без подзаголовков. Материалы и методы. Эта часть должна содержать достаточное описание, чтобы все процедуры можно было повторить. Она может быть разделена на подразделы, если описаны несколько методов. Результаты и обсуждение Выводы. Следует четко объяснить основные выводы из работы, подчеркнув ее важность и актуальность.
Рисунки	Все рисунки (графики, диаграммы, рисунки, скриншоты и фотографии) должны быть представлены в электронном виде в отдельном файле.	Все рисунки должны быть представлены в электронном виде в отдельном файле и иметь высокое разрешение. Максимум 10 иллюстраций.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	EMERALD	SAGE
	Все рисунки должны быть качественными, удобочитаемыми и пронумерованы в соответствии с последовательностью в тексте и арабскими цифрами. Не устанавливаются ограничения на количество рисунков.	
Таблицы	Таблицы должны быть предоставлены в отдельном файле. На каждую таблицу в тексте должно быть указание (Например: как показано в табл. 1…) Не устанавливаются ограничения на количество таблиц.	Таблицы должны быть предоставлены в отдельном файле. Не устанавливаются ограничения на количество таблиц.
Благодарности	Авторы несут ответственность за указание источников внешнего финансирования научных исследований в своей статье. Авторы должны описать всех спонсоров.	Все участники, не отвечающие критериям авторства, должны быть перечислены в этом разделе.
Список литературы	Гарвардский стиль оформления списка литературы	Ванкуверский стиль оформления списка литературы
Использованные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Aircraft Engineering and Aerospace Technology, http://www.emeraldgrouppublishing.com/products/journals/author_guidelines.htm?id=aeat 2. Assembly Automation, http://www.emeraldgrouppublishing.com/products/journals/author_guidelines.htm?id=aa	1. Advances in Mechanical Engineering, https://uk.sagepub.com/en-gb/eur/journal/advances-mechanical-engineering#submission-guidelines 2. The Journal of Strain Analysis for Engineering Design, https://uk.sagepub.com/en-gb/eur/journal/journal-strain-analysis-engineering-design#submission-guidelines

МЕДИЦИНСКИЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	NATURE PUBLISHING GROUP	OXFORD UNIVERSITY PRESS	SPRINGER	WILEY
Аннотация	До 150 слов, должна содержать цель исследования, основные результаты и выводы	Должна включать в научной статье 200–250 слов**,** в обзорной – 150 слов	Не более 250 слов. В аннотации нельзя использовать ссылки на источники.	150–250 слов, должна содержать цель исследования, основные результаты и выводы	Должна включать в научной и обзорной статье не более 250 слов, должна содержать цель исследования, основные результаты и выводы.
Ключевые слова	3–6 ключевых слов	3–8 ключевых слов	До 6 ключевых слов	4–6 ключевых слов	Максимум 10 ключевых слов
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.1.2, ...), 1.2 и т.д.* *Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только	*Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение	Не более 3500 слов. Сноски будут включены в общий объем статьи (их количество не должно превышать 40). Требования к структуре статьи не установлены.	Структурируйте текст, используя 3-х уровневую (и более) систему рубрик. Здесь понимается, что статья не должна идти сплошным тестом, а разделена на подзаголовки от трех и более. Требования к структуре статьи не установлены. Для выделения отдельных структурных	*Введение* (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования

	существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как А, В и т.д. У формул и уравнений в приложениях отдельная нумерация. Объем статьи не должен превышать 5000 слов.	результатов работы). Объем статьи не более 4500 слов без учета аннотации, списка литературы, рисунков и таблиц.		частей использовать курсив. Пронумеровывать страницы (автоматически с помощью функции Word, не самостоятельно). Файл в формате DOCX (Word 2007 или выше) или DOC (для более ранних версий Word).	или обсуждение результатов работы). Объем статьи не более 3500 слов. Не более 35 сносок.
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстраций нумеруются в соответствии с их последовательность	Максимальное число рисунков (таблиц) в научной статье – 5, в обзорной статье – 6–8.	Рисунки не ограничены по количеству, но в каждом допускается до 150 слов.	Рисунки, созданные в MS Office, являются приемлемыми. Иллюстрации нумеруются в соответствии с их последовательностью в	Рисунки черно-белые. Подписи к рисункам отдельно (не на самом рисунке). Желательно подпись – 100 символов.

	ю в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы: TIFF, JPEG, PDF. Подписи рисункам отдельно (не на самом рисунке). Представить каждый рисунок в отдельном файле.			тексте. Шрифты *Helvetica* или *Arial*. Не устанавливаются ограничения на количество рисунков.	
Таблицы	Таблицы нумеруются в соответствии с их последовательность ю в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты, описанные в статье. Не более 8 таблиц/рисунков в	Максимальное число рисунков (таблиц) в научной статье – 5, в обзорной статье – 6–8.	Таблицы не ограничены по количеству; в каждой допускается до 150 слов.	Таблицы нумеруются в соответствии с их последовательностью в тексте. Все таблицы должны быть пронумерованы арабскими цифрами. Для каждой таблицы должно быть указано название. Не устанавливаются ограничения на количество таблиц.	Шрифт *Calibri*. Максимум 5 таблиц и рисунков

	совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.				
Благодарности	В отдельном разделе в конце статьи перед списком литературы. Можно указать номер грантов.	Должны быть краткими и включать в себя все источники поддержки, в том числе спонсорство.	Выражение признательности людям, информация о грантах, фондах и т.д. должны быть размещены в отдельном разделе перед списком литературы.	Должны быть размещены в отдельном разделе на титульном листе. Названия финансирующих организаций должны быть написаны полностью.	Благодарности (включая финансирование) должны быть помещены в конце текста.
Прочее			В дополнение к аннотации, рукописи должны также включать резюме 40 слов главного пункта статьи.		
Список литературы	Список литературы представляется в конце статьи на отдельной странице.	Список литературы не должен превышать в научной статье – 100 источников, в обзорной статье – 150 источников.	Ссылки на книги, журнальные статьи, статьи в сборниках и конференции или семинара. Не более 50 источников.	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте. Ссылки записи в списке	Нумерация идет в том порядке, в котором они впервые упоминаются в тексте.

				следует в алфавитном порядке по фамилии первого автора каждой работы.	
Использован-ные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Clinics in Dermatology, https://www.elsevier.com/journals/clinics-in-dermatology/0738-081x/guide-for-authors 2. Journal of Pediatric Health Care, https://www.elsevier.com/journals/journal-of-pediatric-health-care/0891-5245/guide-for-authors	2. Blood Cancer Journal, http://www.nature.com/bcj/about/for_authors.html 3. Translational Psychiatry, http://www.nature.com/tpj/for_authors.html	1. Author Resource Centre, http://www.oxfordjournals.org/en/authors/index.html 2. Annals of Oncology, https://academic.oup.com/annonc/pages/General_Instructions 3. Carcinogenesis, https://academic.oup.com/carcin/pages/General_Instructions	1. Intensive Care Medicine, http://www.springer.com/medicine/critical+care+and+emergency+medicine/journal/134 2. European Journal of Nuclear Medicine and Molecular Imaging, https://academic.oup.com/chemse/pages/General_Instructions	1. Paediatric and Perinatal Epidemiology, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1365-3016/homepage/ForAuthors.html 2. Pediatric Allergy and Immunology, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1399-3038/homepage/ForAuthors.html

СЕЛЬСКОХОЗЯЙСТВЕННЫЕ НАУКИ

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	WILEY
Аннотация	Аннотация до 250 слов, должна содержать цель исследования, основные результаты и выводы	100–200 слов	До 300 слов. Введение в тему, методы, результаты, выводы.
Ключевые слова	6–10 ключевых слов	До 6 ключевых слов	До 8 ключевых слов
Структура статьи	*Разделы должны быть пронумерованы 1.1 (то 1.1.1, 1.2,...), 1.2 и т.д.*	20000 слов и более Шрифт *Times New Roman* (12 пт) или стиль аналогичного типа. Все заголовки и подзаголовки	Присланная статья должна быть не более 40 страниц, включать в себя: заголовок, аннотацию, текст, сноски, ссылки на источники, таблицы и изображения.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	WILEY
	Введение (цель работы, обзор литературы, краткое изложение результатов). *Материалы и методы* (описываются только существенные изменения). *Результаты* (результаты должны быть четкими и краткими). *Выводы* (основные выводы исследования или обсуждение результатов работы) *Приложения* Если есть больше чем одно приложение, они должны быть идентифицированы как A, B и т.д. У формул и уравнений в приложениях отдельная нумерация.	выравниваются по левому краю. Заголовки разделов жирным и курсивом. Двойной интервал, односторонняя печать. Использовать Руководство Chicago Style для оформления. Требования к структуре статьи не установлены.	Требования к структуре статьи не установлены.
Рисунки	Использовать следующие шрифты: *Arial*, *Courier*, *Times New Roman*. Иллюстраций нумеруются в соответствии с их последовательностью в тексте. Если иллюстрация создается в приложении Microsoft Office (Word, PowerPoint, Excel), оставить как есть в формате документа. Приемлемые форматы: TIFF, JPEG, PDF. Подписи рисункам отдельно (не на самом рисунке). Представить	Рисунки и таблицы на отдельном листе в конце статьи. Иллюстрации нумеруются в соответствии с их последовательностью в тексте. Подпись располагается под рисунком. Ограничения на количество рисунков не устанавливаются.	Представить рисунки в отдельном файле. Ограничения на количество рисунков не устанавливаются.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	WILEY
	каждый рисунок в отдельном файле.		
Таблицы	Таблицы нумеруются в соответствии с их последовательностью в тексте. Сноски к таблицам указываются под ними. Данные, представленные в таблицах, не должны дублировать результаты, описанные в статье. Не более 8 таблиц/рисунков в совокупности; любые дополнительные рисунки и таблицы могут быть включены в дополнительные данные.	Каждая таблица должна быть расположена в конце статьи на отдельном листе и подписана в соответствии с определенным отрывком в статье. Надпись располагается в верхней части таблицы. Ограничения на количество таблиц не устанавливаются.	Шрифт желательно *Arial* или *Helvetica*. Предоставить в отдельном файле (Word). Ограничения на количество таблиц не устанавливаются.
Благодарности	В отдельном разделе в конце статьи перед списком литературы. Указать список тех лиц, которые оказывали помощь в ходе исследования.	Указывать на титульном листе	Благодарность в конце текста
Список литературы	Список литературы должен включать только те работы, которые упоминаются в тексте и которые были опубликованы или приняты к публикации. Высказывания, мнения экспертов и неопубликованные работы должны быть упомянуты в тексте.	Указываются в конце статьи в алфавитном порядке.	Нумерация источников в том порядке, в котором они впервые упоминаются в тексте.

КАТЕГОРИЯ/ ИЗДАТЕЛЬСТВО	ELSEVIER	OXFORD UNIVERSITY PRESS	WILEY
Использованные источники (разделы для авторов издательств, инструкции для авторов в журналах)	1. Agricultural and Forest Meteorology, https://www.elsevier.com/journals/agricultural-and-forest-meteorology/0168-1923/guide-for-authors 2. Animal Behaviour, https://www.elsevier.com/journals/animal-behaviour/0003-3472/guide-for-authors	1. Author Resource Centre, http://www.oxfordjournals.org/en/authors/index.htm (см. разделы Figures, Instructions to Authors, LaTex) 2. American Journal of Agricultural Economics, https://academic.oup.com/ajae/pages/Author_Guidelines 3. Zoological Journal, https://academic.oup.com/zoolinnean/pages/General_Instructions	1. Agricultural and Forest Entomology, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1461-9563/homepage/ForAuthors.html 2. Evolutionary Applications, http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1752-4571/homepage/ForAuthors.html#section10

ПРИЛОЖЕНИЕ 3

Чек-лист подготовки публикации

Как читать чеклист?
Чек-лист читается и проверяется по пунктам в процессе выполнения научных исследований для предотвращения ошибок при написании и опубликовании научной работы. К каждому пункту предполагается написание методических указаний, раскрывающих способ выполнения работы. Чек-лист и методические указания не ставят целью показать, как выполнять научное исследование. Цель чек-листа – сформулировать в форме проверочных утверждений критерии выполнения практик, применяемых в научном исследовании.

Применимость
Пункты, отмеченные звездочкой *, должны быть уточнены в зависимости от сложившейся практики научной коммуникации в отдельных науках.

1. Сформулировано направление исследований.
2. Определено сообщество ученых и практиков, которым интересны результаты проводимых исследований.
3. Регулярно читаются и отслеживаются публикации из журналов и трудов конференций по теме исследований.
4. Определены работы, в которых описано возникновение темы исследований, и отслеживая цитирования которых, члены сообщества найдут вашу работу.
5. Определены обзоры и работы за последние 2–3 года с результатами по теме исследования.
6. Тема исследования четко сформулирована в форме research proposal. Написано введение для будущей публикации.
7. Выбраны признанные сообществом методы исследования.
8. Определены следующие шаги по развитию исследования.
9. Известны средства коммуникации, принятые в сообществе (конференции, научные журналы, семинары).
10. Намечены варианты изданий для публикации итогового варианта работы.
11. Определены требования к изложению результатов и полученные результаты соответствуют им.
12. Проведено обсуждение research proposal с научной группой внутри организации.
13. Определен объем раскрываемых сведений и принято решение по обнародованию презентации с неопубликованными результатами исследований (вопрос приоритетов и оригинальности).
14. Проведено обсуждение research proposal и первых результатов на научной конференции или семинаре, признанными сообществом.
15. Составлен список коллег, которым будет интересна итоговая публикация.
16. Предварительные результаты исследования опубликованы на странице автора (если решено опубликовать предварительный материал, презентацию или препринт).
17. Проведено обсуждение материалов публикации с будущими соавторами.
18. * Соавторы дали согласие на указание своего имени в качестве автора.

19. * Составлен итоговый список авторов публикации. Для каждого соавтора определена его роль в публикации (роли не публикуются, но предъявляются по первому требованию редактора).
20. * Определен порядок указания авторов в публикации.
21. * Для каждого автора указаны места работы, их порядок и каноническое написание организации (просим прислать соавторов и сверяем на сайте).
22. Изучены правила грантовых организаций по указанию грантов. Исключены конфликты по указанию нескольких источников финансирования.
23. Выбран тип публикации и написан черновой вариант текста публикации.
24. * Принято решение о публикации в открытом доступе и определены источники оплаты публикации открытого доступа (если это допускается правилами журнала).
25. * Наборы экспериментальных данных и лабораторный журнал подготовлен для предъявления в случае требования верифицировать результаты исследований.
26. Составлен и проверен полный перечень использованных источников в тексте и в ходе работы (использовано программное обеспечение для управления библиографией).
27. * Принято решение о публикации и дате публикации наборов экспериментальных данных и лабораторного журнала в data repository.
28. * Наборы экспериментальных данных и лабораторный журнал опубликованы в data repository.
29. Установлен и соблюдается порядок внесения правок в текст публикации.
30. Сформирована последовательность изданий, в которые будет подаваться публикация. Порядок изданий определяется на основании престижности журналов и значимости полученных результатов. Публикация в каждый момент времени подается только в один журнал.
31. Сформулировано итоговое название публикации, содержащее ключевые слова и отражающее полученный результат.
32. Изучены Руководства для Авторов (Guide for Authors) и получены шаблоны оформления публикаций для каждого журнала и текст публикации приведен в соответствие требованиям.
33. Составлен перечень требований к комплектности подаваемого материала.
34. Иллюстративный и сопроводительный материал публикации приведен в соответствии с требованием журнала.
35. Составлена аннотация, дающая точное и полное представление о содержании работы.
36. Составлен полный и точный список ключевых слов к публикации.
37. Пристатейная библиография отформатирована в соответствии со стилем, требуемым в журнале.
38. Сформирован раздел благодарностей (Acknowledgements).
39. Определен порядок подачи материала: способ подачи в редакцию и автор, ответственный за переписку с редактором. Автор, ответственный за переписку, указывается как автор по переписке (контактный автор, corresponding author), указывается одно место его работы.
40. Текст публикации вычитан на предмет отсутствия ошибок и приведен к требованиям, принятым в подаваемом журнале.

41. Публикация проверена на предмет соответствия требованиям к подаваемому в журнал материалу.
42. Проверено выполнение требований организации к публикации материалов: патентная проверка, экспортный контроль, уведомление о посылке публикации в печать.
43. Определены требования журнала по публикации авторской версии работы в репозиториях открытого доступа и отправки материалов коллегам.
44. Написано сопроводительное письмо (Cover Letter) редактору с пояснением, почему работа будет полезной и важной для журнала.
45. Публикация отправлена в журнал.
46. Отработаны замечания рецензентов и редактора, отметка об исправлении или выражение несогласия по каждому пункту доработки отправлены редактору.
47. В случае отказа публикация переоформлена по правилам следующего в очереди журнала и отправлена в него.
48. Авторская версия с внесенными исправлениями подготовлена для публикации в репозитории открытого доступа и рассылке коллегам.
49. Авторская версия опубликована в репозитории открытого доступа и отправлена коллегам в сроки и в соответствии с правилами издательства.

ПРИЛОЖЕНИЕ 4

Инструкция по работе с ресурсами по выбору целевых журналов

1. Отбор и анализ журналов по Scopus

Для того чтобы подобрать журнал для публикации в *Scopus*, необходимо выполнить следующий алгоритм:

1. Проводим стандартный тематический поиск по ключевым словам вашей работы (рисунок 1). Важно максимально точно подобрать англоязычные термины, которые отражают суть вашей статьи. Чтобы правильно составить поисковое предписание (запрос), важно предварительно изучить инструкцию (Search tips).
Выбираем опцию *Documents*.

Рисунок 1 – Поисковый запрос

По полученной выборке можно сделать постатейный анализ документов и отобрать наиболее цитируемые работы, которые можно будет использовать как для обзора литературы, так и для отбора наиболее авторитетных журналов по данным о цитировании в них статей по вашей тематике. Для отбора самых цитируемых работ необходимо использовать сортировку полученных документов по убыванию их цитирования (*Cited by*).

2. Приступаем к анализу журналов, включающих полученные в результате поиска публикации. Нажимаем на кнопку *«Анализ результатов поиска»*, которая находится над списком документов (рисунок 2).

Рисунок 2 – Результаты поиска

3. Получаем список журналов для дальнейшего анализа. Выбираем вкладку *Source*, на которой содержится количественная информация по источникам (рисунок 3). Таким образом, мы получаем список журналов, которые наиболее часто публикуют статьи по вашей тематике. Система позволяет получить до 160 журналов и одновременно провести сравнение до 10 журналов из перечня, журналы можно менять.

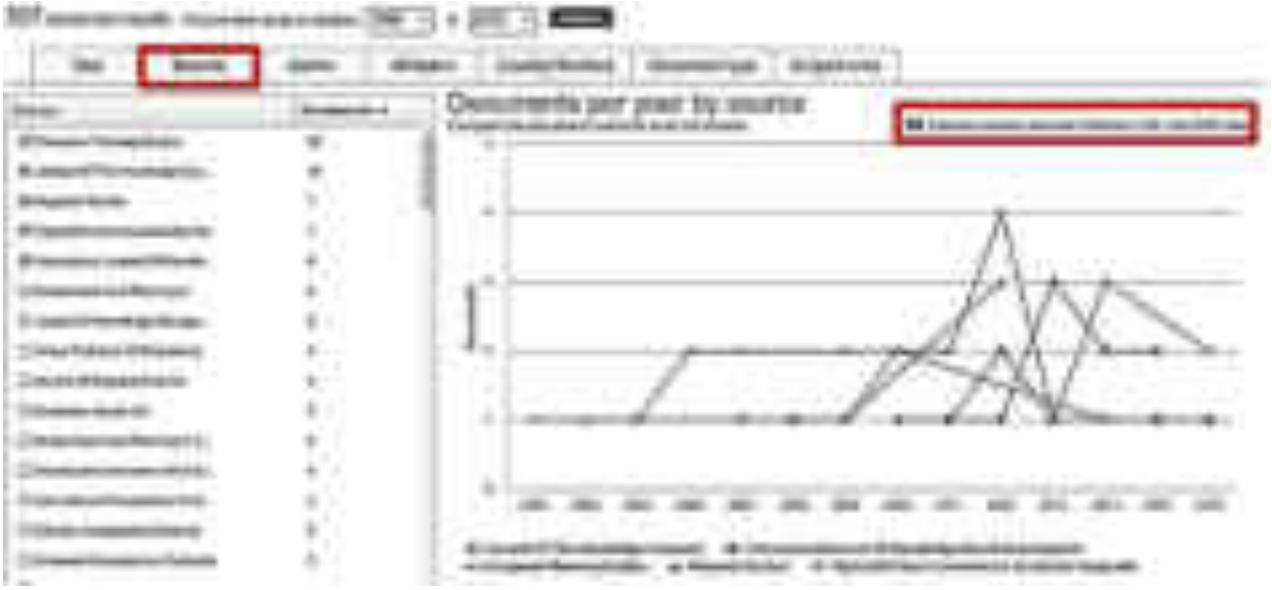


Рисунок 3 – Статистика по источникам

4. Сравниваем журналы по различным показателям. Нажав на ссылку *Compare sources* (рисунок 3), мы можем провести качественный анализ журналов, сравнив их по различным метрикам (рисунок 4). Все данные можно экспортировать в MS Excel.

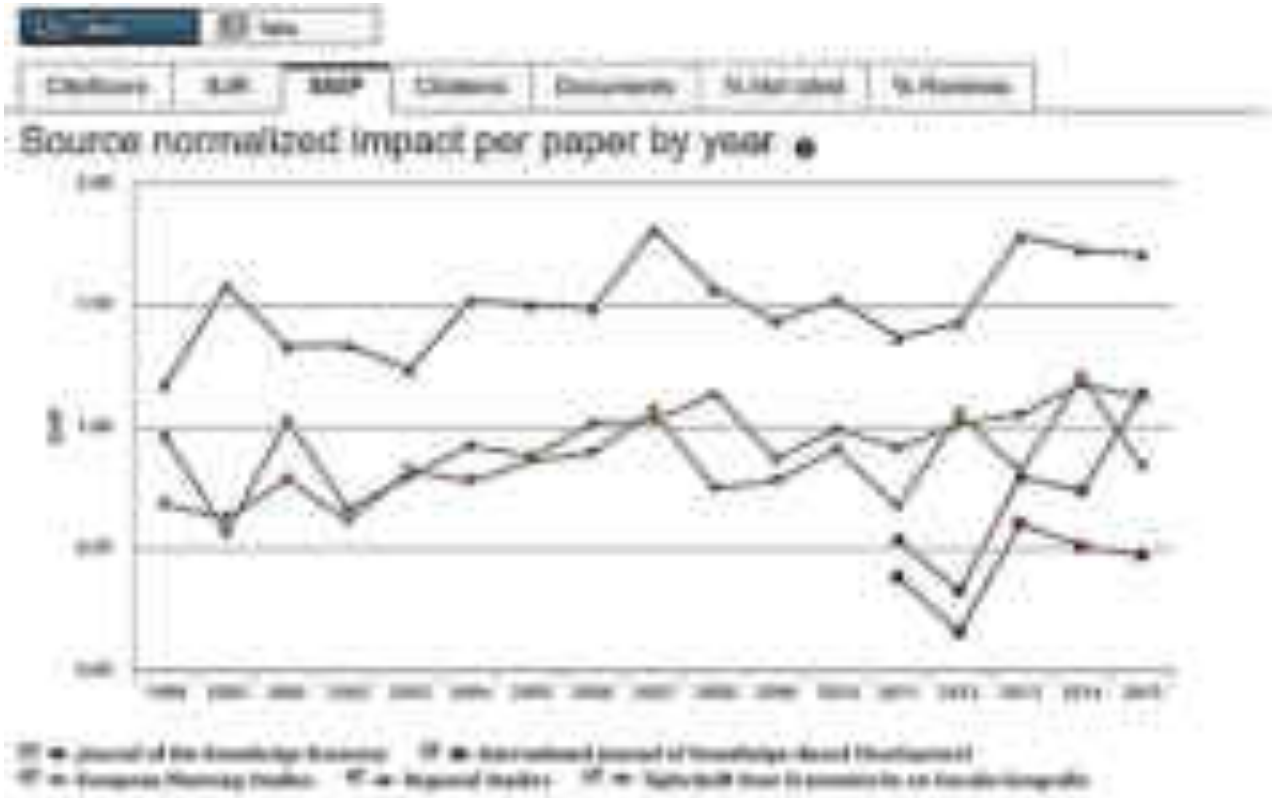


Рисунок 4 – Пример сравнения журналов по показателю SNIP

Elsevier предоставляет пользователям несколько метрик, которые позволяют провести всесторонний анализ журналов[3]:

- *CiteScore* считает цитаты из всех документов в течение года один на все документы, опубликованные в источнике в предыдущие три года (рисунок 5). Это обеспечивает более надежную и точную индикацию влияния журнала. Например, чтобы вычислить значение 2015 г., CiteScore считает цитирования, полученные в 2015 г., на документы, опубликованные в 2012, 2013 или 2014 гг. Это число делится на количество документов, индексированных в Scopus и опубликованных в 2012, 2013 и 2014 гг.

Рисунок 5 – CiteScore
Источник: https://journalmetrics.scopus.com

[3] Journal Metrics: https://journalmetrics.scopus.com

Рисунок 6 – Source Normalized Impact per Paper (SNIP)
Источник: презентация World-Class Publication от Elsevier

6. Дополнительные показатели при отборе журналов.
При выборе журналов для анализа важно учитывать другие, кроме статистики по числу полученных статей, показатели, в том числе:
– по стране публикации (Country| Territories). Необходимо учитывать, что, возможно, трудно будет опубликовать статью в журнале, который на текущий момент не опубликовал ни одной статьи российского автора при достаточно большом объеме.
– по объему журнала – среднему числу документов;
– по типу документов – возможно вы выбрали журнал, который публикует только обзоры;
– по составу авторов и т.д.
Все эти данные имеются в системе анализа выборки.
Для выбора правильного журнала для публикации важен комплексный анализ всех параметров издания!

2. Отбор и анализ журналов по Scimagojr.com

Scimagojr.com (SJR)[4] – открытый ресурс (платформа), построенный на данных МНБД Scopus. Разработка испанской группы SciMago. Позволяет анализировать журналы и ранжировать их по различным библиометрическим показателям, включая h-index, SJR, средний показатель цитирования статей за разный период времени и др. Система позволяет выгружать в MS Excel файлы ранжированные списки по предметным областям и категориям тематического классификатора ASJC Scopus, включающего 334 раздела и подраздела.
Показатели каждого журнала можно проследить в динамике за несколько лет. Данные по каждому изданию визуализируются графически.

1. Чтобы уточнить данные по конкретному изданию, воспользуемся строкой поиска на главной странице ресурса (ISSN, название, издатель). Для того чтобы просмотреть список журналов и отфильтровать его по параметрам, необходимо перейти по ссылке Journal Rankings (Journal Ranks). Мы можем отфильтровать открывшийся список по предметной области, категории, стране, типу источника и году (рисунок 7).

[4] Scimago Journal & Country Rank (SJR): http://www.scimagojr.com

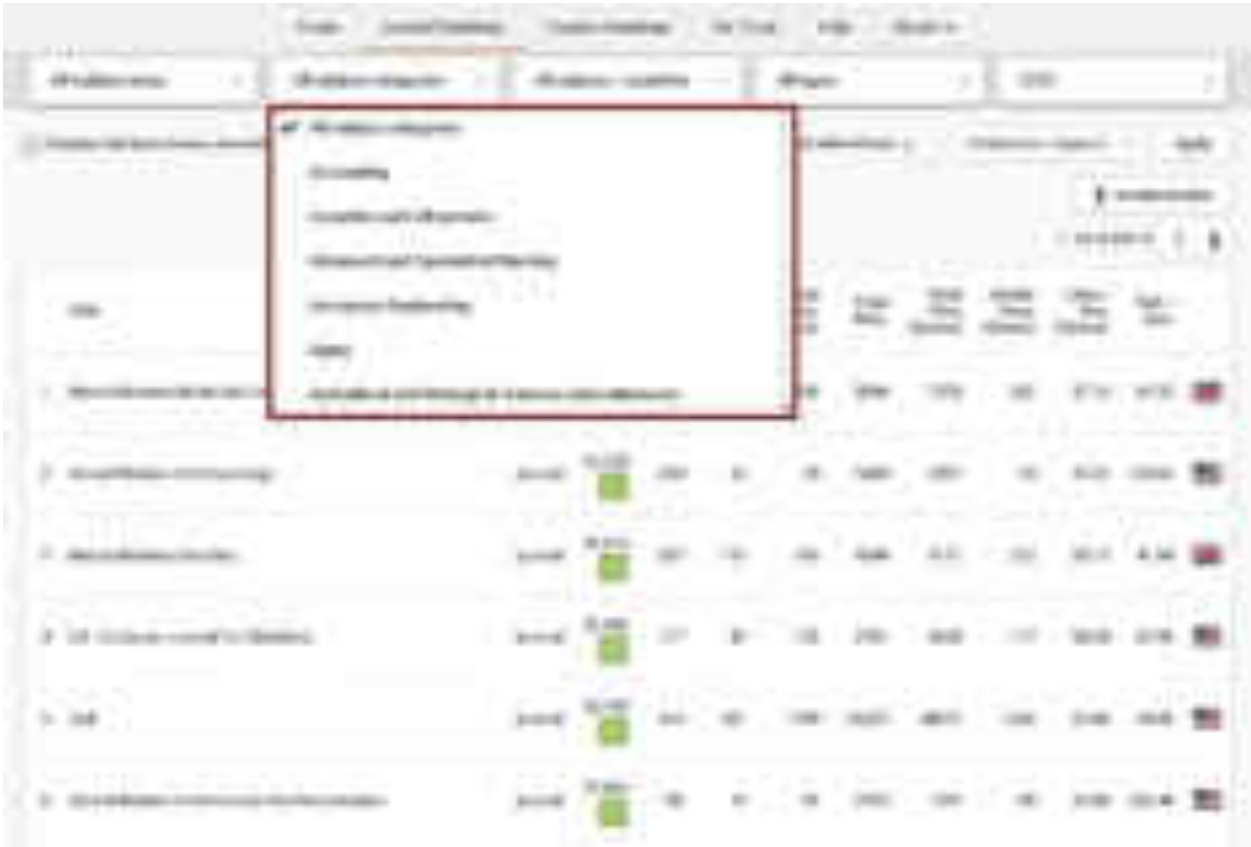


Рисунок 7 – Настройка списка в Scimago
Источник: http://www.scimagojr.com/journalrank.php

2. Названия журналов кликабельны, страница журнала содержит информацию о предметных категориях, издателе, цели издания, а также визуализацию основных параметров журнала.

3. Отбор и анализ журналов по Web of Science

База данных *Web of Science* от *Thomson Reuters* предполагает использование схожего алгоритма действий для отбора журнала для публикации.

1. Осуществляем тематический поиск, после чего нажимаем на вкладку *«Анализ результатов»*, которая находится в правой части экрана над результатами поиска (рисунок 8).

Рисунок 8 – Анализ результатов поиска в Web of Science

2. В открывшемся окне анализа результатов выбираем категорию *Source Titles*, количество записей для анализа (ТОП 10, 25, 50, 100, 250, 500) и нажимаем кнопку *Analyze* (рисунок 9).

Рисунок 9 – Настройки анализа

3. В открывшемся окне мы видим количество публикаций по каждому источнику (рисунок 10). Кроме того, мы можем просмотреть список публикаций, проставив галочку напротив названия источника и нажав *View Records* в верхней или нижней части окна.

Рисунок 10 – Количественный анализ публикаций по тематике в Web of Science

4. Названия журналов кликабельны, по ним доступна аналитическая информация из *Journal Citation Reports* (рисуноу 11). *Journal Citation Reports (JCR)*[5] – аналитический продукт компании *Thomson Reuters*. Содержит измеримые статистические показатели для критической оценки научных журналов на основе данных цитирования. Анализируя пристатейные ссылки литературы, система анализирует влияние (impact) в пределах журнала или предметной категории, показывает взаимосвязи между цитируемыми и цитирующими журналами. Журналы напрямую сравниваются по квартилям и с помощью трендового анализа. Инструмент работает по направлениям «science» и «social science».

Рисунок 11 – Информация из Journal Citation Reports

4. Отбор и анализ журналов по Journal Citation Reports

Можно воспользоваться ссылкой *Journal Citation Reports* в верхней части экрана, чтобы перейти непосредственно к JCR[6]. Ресурс представляет собой ранжированный список

[5] Thomson Reuters: http://ipscience.thomsonreuters.com/product/journal-citation-reports/?utm_source=false&utm_medium=false&utm_campaign=false

[6] Зависит от подписки научно-образовательной подписки.

журналов, который можно фильтровать по предметным категориям, странам, издательствам, коллекциям Web of Science (рисунок 12). Доступен поиск по ISSN, названию.

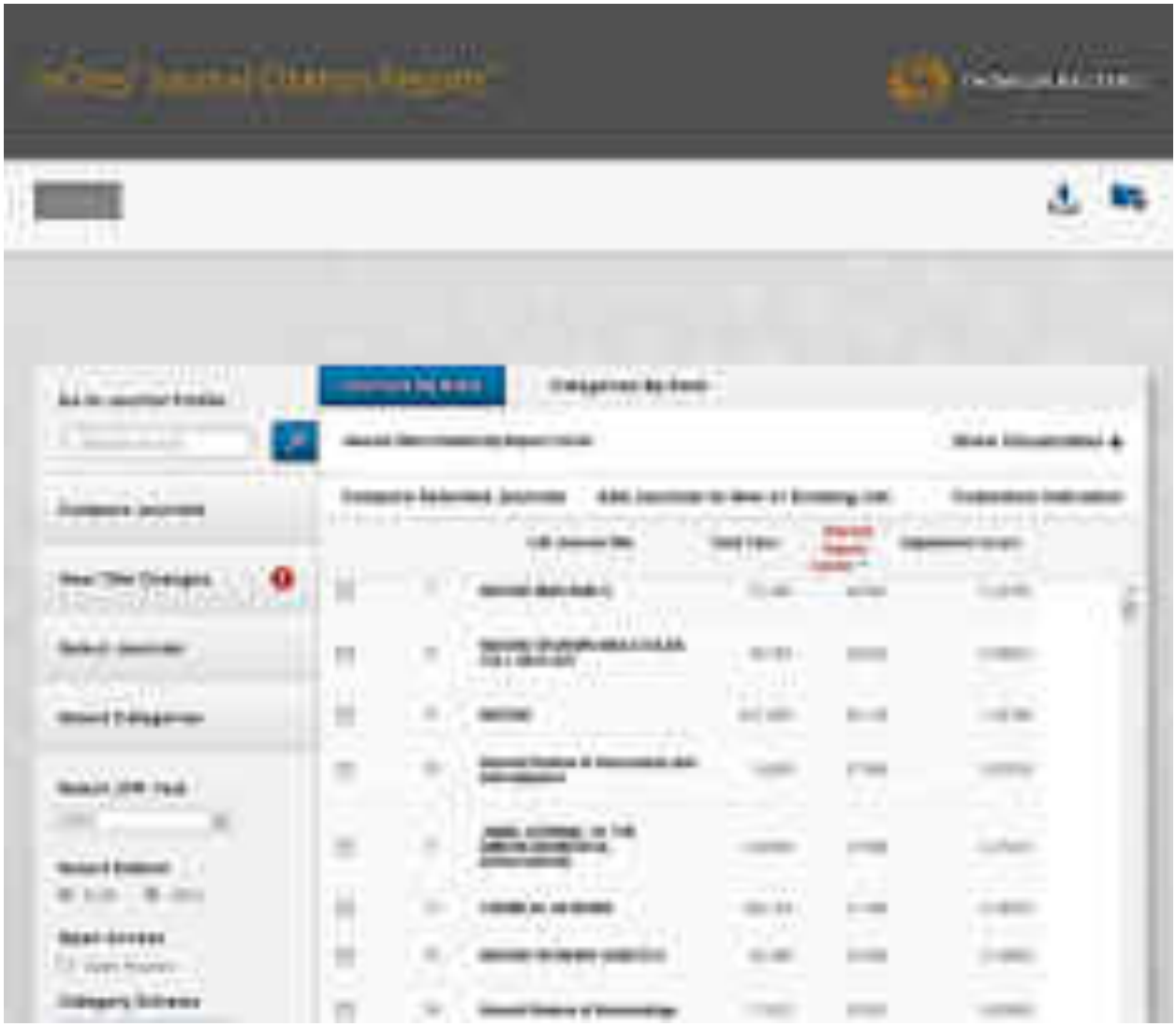


Рисунок 12 – Journal Citation Reports

Основной метрикой является *импакт-фактор журнала (Journal Impact factor (JIF))* – важнейший библиометрический показатель, введенный в оборот Ю. Гарфилдом. Показывает среднее количество цитирований одной статьи в журнале. Вычисляется по формуле IF = C / N, где C – количество цитирований, которое получил тот или иной журнал на статьи, опубликованные за определенный период времени; N – общее количество публикаций в журнале за тот же период[7]. Классический период для расчета импакт-фактора (публикационное окно) – 2 года. Также рассчитывается 5-летний импакт-фактор.

Квартиль (quartile) – в статистике: четвертая часть всей совокупности данных выборки, представленной в виде информационного ряда. В Journal Citation Reports (JCR) используется применительно к списку журналов, ранжированных по импакт-фактору. *Ранг* является местом журнала по показателям в общем списке журналов по предметной категории.

[7] Бредихин С. В. Анализ цитирования в библиометрии / Бредихин С.В., Кузнецов А.Ю., Щербакова Н.Г. Новосибирск: ИВМиМГ СО РАН; НЭИКОН, 2013. С. 122.

5. Отбор журналов по дополнительным бесплатным сервисам для авторов

5.1. Journal Finder издательства Elsevier

Среди ресурсов свободного доступа выделим *Journal Finder* издательства Elsevier[8]. Данный ресурс помогает подобрать журнал для публикации на основании ее названия и аннотации (рисунок 13).

Рисунок 13 – Elsevier Journal Finder

Система не просто ищет подходящие журналы, но и дает важную сопутствующую информацию (рисунок 14):

- Impact – импакт-фактор журнала;
- Acceptance – процент принятых статей;
- Editorial Times – время с момента отправки до принятия первого решения;
- Production Times – время с момента принятия до публикации онлайн;
- Open Access – возможность публикации в открытом доступе;
- Embargo Period – период эмбарго для традиционной издательской модели;
- Open Access Fee – плата за публикацию в открытом доступе;
- User License – информация о лицензировании контента.

[8] JournalFinder: http://journalfinder.elsevier.com

Рисунок 14 – Результаты подбора

Из результатов поиска доступен переход на сайт журнала, где содержится информация о его цели и задачах (Aims and Scope), кроме того, на сайте всегда есть «Руководство для авторов» (Guide for Authors) (рисунок 15). Прочтение данной информации является обязательным для окончательного выбора журнала.

Рисунок 15 – Сайт журнала

5.2. ***JournalGuide***[9] – это бесплатный инструмент, созданный группой разработчиков программного обеспечения, бывших исследователей и ветеранов публикаций на *Research Square*. Целью *JournalGuide* является приведение всех источников данных в одном месте, чтобы дать авторам простой способ выбрать лучший журнал для своих исследований. Источники данных включают основные наборы данных, общественные ресурсы, информацию, представленную непосредственно редакторами журналов, и даже реальный опыт публикаций, предоставленный авторами.

5.3. ***FindMyJournal***[10] является первым программным продуктом, который помогает исследователям выбрать наиболее подходящий журнал, чтобы опубликовать их рукопись. Он использует математический объективный алгоритм, чтобы составить список наиболее подходящих журналов для публикации. Поиск подходящего журнала вручную и сравнение его с тысячами доступных журналов в Интернете по объему и содержанию является процессом чрезвычайно трудозатратным. Но с *FindMyJournal* вы просто вводите критерии вашей публикации, отвечая на 11 вопросов, и видите 5 наиболее подходящих журналов менее чем за несколько секунд. *FindMyJournal* охватывает более 29000 журналов в области физики, биологических и медицинских наук с сотнями журналов в каждой подкатегории.

[9] JournalGuide: https://www.journalguide.com
[10] FindMyJournal: https://findmyjournal.com

ПРИЛОЖЕНИЕ 5

Критерии для определения хищных издательств открытого доступа

(разработаны Джеффри Биллом)

3 версия, 1 января 2015 г.

Критерии, представленные ниже, были разработаны в качестве основы для оценки недобросовестных издательств и журналов открытого доступа. Критерии основаны на двух документах Комитета по этике научных публикаций (Committee on Publication Ethics, COPE):

Code of Conduct for Journal Publishers / Свод правил для издателей научных журналов (http://publicationethics.org/resources/code-conduct)

Principles of Transparency and Best Practice in Scholarly Publishing / Принципы прозрачности и лучших практик в научно-издательском деле (http://publicationethics.org/resources/guidelines-new/principles-transparency-and-best-practice-scholarly-publishing)

Оценка научных издательств открытого доступа – это процесс, который включает в себя внимательное, осторожное, тщательное и зачастую скептическое рассмотрение контента, политики и веб-сайта издательства: связь с издателем при необходимости, прочтение отзывов авторов о работе с издательством, понимание того, использует ли издательство в своей работе недобросовестные практики, перечисленные ниже, которые, как известно, характерны для хищных издательств, рассмотрение любых дополнительных достоверных сведений об издательстве, получение, оценка и принятие во внимание обратной связи от авторов и от самих издательств.

Некоторые журналы, конечно, являются самостоятельными. Они издаются независимо от какого-либо многопрофильного издательства. Однако в большинстве случаев мы оцениваем журналы как часть портфеля многопрофильного издательства. Часто при их описании используется слово «fleet» (флот), означающее, что даже новое издательство внезапно запускает огромное количество новых журналов, от нескольких десятков до нескольких сотен названий сразу.

Практики, описанные ниже, предназначены для оценки как независимых журналов и издательств, так и для многопрофильных издательств и «fleet»-журналов в их портфеле.

Редактор и персонал

– Владелец издательства является редактором всех журналов, которые публикует организация.

– Ни один человек не указан в качестве редактора какого-либо конкретного журнала.

– В журнале невозможно найти информацию о редакционной коллегии.

– Не указана какая-либо информация, позволяющая отнести редактора, персонал редакции и/или членов редакционной коллегии к научной среде (например, аффилиация).

– Существуют доказательства, показывающие, что редактор и/или члены редакционной коллегии не проводят научную экспертизу и не могут считаться «издательским фильтром» в научной среде.

– Два или более журналов имеют дублирующие редакционные коллегии (например, один и тот же состав редакционной коллегии для более чем одного журнала).

– Журналы имеют недостаточное число членов редакционной коллегии (например, всего два или три члена), используют несуществующие редколлегии (придумали имена и регалии членов); указывают в списке редакционной коллегии людей, которые

не давали согласие на это, или указывают в списке членов видных ученых, освобождая их от какой-либо работы в журнале, за исключением использования их имен и/или фотографий.
– Географический охват редакционной коллегии отсутствует полностью или слишком незначительный, особенно для журналов, которые заявляют о своем международном охвате.
– Редакционная коллегия дискриминирует своих членов по гендерному признаку (например, исключает из состава женщин).

Управление журналом

Издатель…

– Демонстрирует отсутствие прозрачности в публикационном процессе.
– Не имеет политики или практики хранения цифровых копий, это означает, что если журнал прекратит деятельность, то весь контент исчезнет из сети Интернет.
– Начинает издавать много журналов сразу, зачастую используя один и тот же шаблон для создания домашней страницы каждого журнала.
– Публикует недостоверную информацию или скрывает информацию об авторских сборах, предлагая опубликовать статью на странице для авторов, а затем выставляя счет за услуги, о которых автор был не предупрежден заранее.
– Не позволяет поисковым системам индексировать опубликованный контент, таким образом делая его невидимым для поисковых систем академических индексов.
– Ограничивает доступ к pdf-файлам, что усложняет проверку на плагиат.

Целостность

– Название журнала не соответствует миссии журнала.
– Название журнала не отражает его происхождение (например, журнал со словами «Канадский» или «Швейцарский» в названии, но при полном отсутствии среди редакторов и членов редакционной коллегии людей, представляющих эти страны).
– В электронных письмах или на своем веб-сайте издательство указывает, что один или более журналов имеет импакт-фактор (рассчитываемый компанией Thomson Reuters (*с 2016 г. расчетом импакт-фактора занимается компания Clarivate Analytics*) или другие импакт-факторы, рассчитываемые фейковыми сервисами, или использует сомнительные метрики, которые вводят в заблуждение относительно реального положения журнала в международном пространстве (улучшают реальное положение вещей).
– Издательство рассылает спам-запросы для рецензирования научных статей специалистам, которые не работают в научной области, о которой идет речь в статье.
– Издательство утверждает, что журналы индексируются в признанных базах данных или утверждает, что журналы индексируются в других базах данных, хотя ресурсы или сервисы, в которые включен журнал, таковыми не являются.
– Издательство не использует достаточно ресурсов для предотвращения и устранения неправомерных действий автора, поэтому в журналах часто встречается плагиат, самоплагиат, манипуляции с изображениями и т.д.
– Издательство просит контактного автора предложить рецензента для присланной статьи, к которому впоследствии обращается без проверки качества его квалификации и реального существования (такая практика позволяет авторам участвовать в рецензировании собственных статей).

Другое

Издательство-хищник может…

– Повторно публиковать статью, ранее опубликованную в другом журнале, без разрешения и соблюдения условий лицензии.
– Использовать эпитеты в описании, например, «ведущее издательство», даже если само издательство образовано совсем недавно.

– Использовать авторитет западных стран, публикуя несуществующие почтовые адреса, например, США, чтобы привлечь авторов из развивающихся стран.
– Минимально проводить редактирование и корректуру статей, или не проводить их вовсе.
– Публиковать статьи, которые не являются научными, например, очевидно лженаучные статьи.
– Иметь форму «Свяжитесь с нами», которая включает только веб-форму или адрес электронной почты, само издательство при этом скрывает или не раскрывает свое местонахождение.

Низкие журнальные стандарты / практики

Следующие практики считают низкими журнальными стандартами, но при этом они не приравнивают издательство или журнал к категории хищников. Авторам следует уделить должное внимание этим пунктам до подачи рукописи в журнал.
– Издательство копирует «руководство для авторов» полностью или с минимальным редактированием у другого издательства.
– Издательство указывает недостаточную контактную информацию, включая контактную информацию, которая нечетко указывает местонахождение головного офиса или ложные сведения о местонахождении головного офиса (например, используя только почтовый адрес).
– Издательство выпускает журналы, тематика которых слишком широкая (например, «Журнал «Образование»), чтобы привлечь как можно больше статей и получить больше доходов от авторских сборов.
– Издательство публикует журналы, которые объединяют области, не связанные друг с другом (например, «Международный журнал бизнеса, гуманитарных и технических наук»).
– Издательство взимает плату за публикацию с авторов, но при этом требует от автора передать права на нее и закрепляет свои права на использование контента журнала. Или издательство требует передать право на статью при представлении рукописи.
– Издательство имеет некачественный веб-сайт, на котором есть неработающие ссылки, орфографические и грамматические ошибки.
– Издательство помещает на своем сайте изображения, на использование которых правообладатель не давал разрешения.
– Издательство чрезмерно увлекается спам-рассылками, чтобы привлечь новых авторов или членов редакционной коллегии.
– Редакция использует e-mail адреса, которые размещены на gmail.com, yahoo.com или на других бесплатных почтовых серверах.
– Издательство работает вразрез с государственной политикой лицензирования статей, или показывает свое непонимание тонкостей лицензирования статей в журналах открытого доступа, или дает противоречивую информацию о лицензировании.
– Издательство не публикует информацию о политике отзыва статьи или отзывает статьи без официального заявления об этом. Также издательство не публикует исправления или уточнения, или не имеет политики по этим вопросам.
– Издательство не использует такие стандартные идентификаторы, как ISSN или DOI, или использует их неправильно.
– Для названия издательства издатель использует такие слова, как «Центр», «Ассоциация», «Институт» и другие, при этом его организационная структура и миссия не соответствует этим понятиям.
– Издательство размещает настолько много навязчивой рекламы на своем сайте, что это мешает доступу к контенту.
– Издательство не является членом профильных ассоциаций или преднамеренно не следует принятым стандартам в отрасли.

– Издательство помещает ссылки на существующие признанные конференции и ассоциации на свой сайт в качестве подтверждения легитимности своей деятельности.
– Издательство заявляет о быстрой публикации и/или необычно быстром рецензировании.
– Существуют доказательства, которые показывают, что издательство на самом деле не проводит рецензирование добросовестно.
– Создается впечатление, что издательство озабочено только получением авторских сборов, но никак не развивает сервисы для читателей, или же выставляет счета для оплаты, не проводя предварительной оценки рукописей.
– Издательство так организует процесс публикации, что демонстрирует хищное предпринимательское поведение. Владелец может иметь опыт управления проектами, на сайте издательства могут присутствовать журналы по управлению и бизнесу, но при этом принципы деловой этики не соблюдаются.
– Издательство или его журналы не включены в стандартные базы периодических изданий или не представлены в каталогах и базах данных для библиотек.
– Издательство полностью копирует или использует созвучные названия журналов других издательств.
– Издательство размещает на своем веб-сайте текст, который описывает движение за открытый доступ, и затем навязывает себя как стремящегося к соблюдению всех ценностей и целей открытого доступа.
– Ни один из членов действующей в журнале редакционной коллегии никогда не публиковал статей в этом журнале.
– Отсутствует географический охват авторов статей (или он очень незначительный) в одном или многих журналов издательства, что является индикатором того, что журнал является наиболее легким путем для публикации авторов из одной страны или региона.
– Издательство имеет опцию «срочной публикации» за дополнительную плату. Как правило, ускоренное рассмотрение гарантирует публикацию, но правильность и благонадежность использованных в статье данных находятся под сомнением, поскольку их никто не проверяет.

Автор благодарит Билла Коэна (Bill Cohen) и доктора Майкла Фирмина (Michael Firmin) за помощь в работе над финальной и предшествующей версией этого документа.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.02 методы оптимизации

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДЫ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

Учебно-методическое пособие

Составитель М. Н. Служивый

Ульяновск
УлГТУ
2019

УДК 621.391 (076)
ББК 32я7
М 54

Рецензент
доктор технических наук,
заведующий кафедрой «Информационные системы и технологии» УлГУ,
профессор А. А. Смагин

Рекомендовано научно-методической комиссией
радиотехнического факультета в качестве учебно-методического пособия

М 54 **Методы моделирования и оптимизации** : учебно-методическое пособие / сост. М. Н. Служивый. – Ульяновск : УлГТУ, 2019. – 25 с.

Рассмотрены численные методы поиска экстремума функций одной переменной: метод перебора, метод деления отрезка пополам, метод Ньютона, метод касательных, а также метод градиентного спуска. Наряду с этим имеется задание по графическому решению задачи линейного программирования с двумя переменными. Каждое задание представлено в виде 20 вариантов.

Рекомендовано в качестве методических указаний при выполнении курсовой работы по дисциплине «Методы моделирования и оптимизации» у студентов направления 11.04.02 «Инфокоммуникационные технологии и системы связи».

Работа подготовлена на кафедре «Телекоммуникации».

УДК 621.391 (076)
ББК 32я7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учебно-методическое пособие содержит методические указания по выполнению курсовой работы по дисциплине «Методы моделирования и оптимизации».

В первой части пособия представлены 5 заданий по численным методам поиска экстремумов функции одной действительной переменной: метод перебора, метод деления отрезка пополам, метод Ньютона, метод касательных, а также метод градиентного спуска.

Вторая часть пособия включает одно задание по графическому решению задачи линейного программирования с двумя переменными.

Каждое задание представлено в виде 20 вариантов.

Значительная часть заданий из пп.1.1-1.5 заимствована из учебников [1-2], задания из п.2 из сборников задач [3-6].

1. ЧИСЛЕННЫЕ (ИТЕРАЦИОННЫЕ) МЕТОДЫ ПОИСКА ЭКСТРЕМУМА

Большую группу приближенных методов минимизации функций составляют прямые методы минимизации, основанные на вычислении только значений минимизируемой функции в некоторых точках и не использующие значений ее производных.

1.1. Метод перебора

Метод перебора является простейшим из прямых методов минимизации. Пусть $f(x) \in Q[a; b]$ и требуется найти какую-либо из точек минимума x^* функции $f(x)$ на отрезке $[a; b]$ с абсолютной погрешностью $\varepsilon > 0$. Разобьем $[a; b]$ на n равных частей точками деления $x_i = a + i(b-a)/n$, $i = 0, 1, 2, \ldots, n$, где $n \geq (b-a)/\varepsilon$. Вычислив значения $f(x)$ в этих точках, путем сравнения найдем точку x_m, для которой

$$f(x_m) = \min_{0 \leq i \leq n} f(x_i). \qquad (1)$$

Далее полагаем $x^* \approx x_m$, $f^* \approx f(x_m)$. При этом максимальная погрешность ε_n определения точки x^* равна $\varepsilon_n = (b-a)/n$.

Пример 1. Найти минимальное значение f^* и точку минимума x^* функции $f(x) = x^4 + 8x^3 - 6x^2 - 72x$ на отрезке [1.5; 2]. Точку x^* найти с погрешностью $\varepsilon = 0.05$.

Решение. $f(x) \in Q[1.5; 2]$, т. к. $f''(x) = 12x^2 + 48x - 12 > 0$ при $x \in [1.5; 2]$. Выбрав $n = \dfrac{2-1.5}{0.05} = 10$, вычислим значения $f(x_i)$, $x_i = 1.5 + i \cdot 0.05$, $i = 0, 1, 2, \ldots, 10$, поместив их в таблице 1.

Таблица 1

x_i	1.50	1.55	1.60	1.65	1.70	1.75	1.80	1.85	1.90	1.95
$f(x_i)$	–89.4	–90.2	–91.2	–91.8	–92.08	–92.12	–91.9	–91.4	–90.5	–89.4

Из таблицы 1 находим $x^* \approx 1.75$, $f^* \approx -92.12$.

Метод перебора, предполагающий предварительный выбор точек x_i, $i = 0, 1, 2, \ldots, n$, называется также пассивной стратегией поиска точки минимума x^*. На практике точки x_i выбираются заранее, когда удобно провести $n+1$ независимых экспериментов по измерению значений $f(x)$, а последовательное измерение этих значений трудоемко

или невозможно, например, ввиду нехватки времени. Однако использование уже полученной в предыдущих экспериментах информации о функции $f(x)$ для выбора очередной точки x_i измерения (вычисления) $f(x)$ приводит к более эффективному поиску точки x^*. Методы минимизации, в которых точки x_i определяются в процессе поиска точки минимума с помощью найденных ранее значений функции $f(x)$, называются последовательными методами.

Задание 1. Методом перебора найти точку минимума x^* функции $f(x)$ на отрезке $[a; b]$ с абсолютной погрешностью ε и минимум f^*. Результаты представить в виде таблицы 1.

№ вар.	$f(x)$	$[a; b]$	ε	x^*	f^*
1	2	3	4	5	6
1	$f(x)=x^2-2x+e^{-x}$	[1; 1.5]	0.05		
2	$f(x)=\operatorname{tg}x-2\sin x$	[0; π/4]	0.03		
3	$f(x)=\sqrt{1+x^2}+e^{-2x}$	[0; 1]	0.01		
4	$f(x)=x^4+4x^2-32x+1$	[1.5; 2]	0.05		
5	$f(x)=\frac{1}{7}x^7-x^3+\frac{1}{2}x^2-x$	[1; 1.5]	0.05		
6	$f(x)=x^3-3\sin x$	[0.5; 1]	0.05		
7	$f(x)=5x^2-8x^{5/4}-20x$	[3; 3.5]	0.02		
8	$f(x)=\frac{1}{3}x^3-5x-x\ln x$	[1.5; 2]	0.02		
9	$f(x)=x^4+2x^2+4x+1$	[–1; 0]	0.01		
10	$f(x)=-\left(x^5-5x^3+10x^2-5x\right)$	[–3; –2]	0.01		
11	$f(x)=x^2+3x\ln x-3x$	[0.5; 1]	0.05		
12	$f(x)=x^2-2x-2\cos x$	[0.2; 0.7]	0.01		
13	$f(x)=(x+1)^4-2x^2$	[–3; –2]	0.02		
14	$f(x)=3(5-x)^{4/3}+2x^2$	[1.5; 2]	0.025		
15	$f(x)=-x^3+3(1+x)\left[\ln(1+x)-1\right]$	[–0.5; 0.5]	0.05		
16	$f(x)=2+x^2+x^{2/3}-\ln\left(1+x^{2/3}\right)-2x\operatorname{arctg}\left(x^{1/3}\right)$	[0.5; 1]	0.025		
17	$f(x)=\frac{\cos x}{x^2}$	[7; 11]	0.01		

Окончание табл. 1

1	2	3	4	5	6
18	$f(x)=\dfrac{x}{1+x^2}$	[–2; 0.5]	0.03		
19	$f(x)=\dfrac{1}{(x-1)(x-2)(x-3)}$	[0.2; 2]	0.02		
20	$f(x)=x\mathrm{e}^{-x^2/2}$	[–2; 0.5]	0.025		

1.2. Метод деления отрезка пополам

Метод деления отрезка пополам является простейшим последовательным методом минимизации. Он позволяет для любой функции $f(x)\in Q[a;b]$ построить последовательность вложенных отрезков $[a;b]\supset[a_1;b_1]\supset\ldots\supset[a_{n-1};b_{n-1}]\supset[a_n;b_n]$, каждый из которых содержит хотя бы одну из точек минимума x^* функции $f(x)$. Пусть $\varepsilon>0$ – требуемая точность определения точки x^*. Выбрав $\delta\in(0;2\varepsilon)$, построим последовательности $\{a_n\}, \{b_n\}, \{x_1^{(n)}\}, \{x_2^{(n)}\}$, $n=0,1,\ldots,$ используя рекуррентные формулы $a_0=a$, $b_0=b$;

$$x_1^{(n-1)}=(a_{n-1}+b_{n-1}-\delta)/2, \qquad x_2^{(n-1)}=(a_{n-1}+b_{n-1}+\delta)/2; \tag{2}$$

$$a_n=a_{n-1},\ \ b_n=x_2^{(n-1)}, \text{ если } \ f\left(x_1^{(n-1)}\right)\le f\left(x_2^{(n-1)}\right),$$

$$a_n=x_1^{(n-1)},\ \ b_n=b_{n-1}, \text{ если } \ f\left(x_1^{(n-1)}\right)> f\left(x_2^{(n-1)}\right).$$

Переход от отрезка $[a_{n-1};b_{n-1}]$ к отрезку $[a_n;b_n]$ методом деления отрезка пополам представлен на рис. 1, если $f\left(x_1^{(n-1)}\right)\le f\left(x_2^{(n-1)}\right)$, и на рис. 2, если

$$f\left(x_1^{(n-1)}\right)> f\left(x_2^{(n-1)}\right).$$

Полагая $x^*\approx(a_n+b_n)/2$, находим x^* с абсолютной погрешностью, не превосходящей величины

$$\varepsilon_n=(b_n-a_n)/2=(b-a-\delta)/2^{n+1}+\delta/2, \text{ при } n\to\infty. \tag{3}$$

Используя условие $\varepsilon_n\le\varepsilon$, из (3) можно найти необходимое число шагов n для обеспечения требуемой точности ε. Однако на практике часто поступают иначе: определив границы отрезка $[a_n;b_n]$, вычисляют ε_n по формуле (3) и сравнивают с заданной точностью ε.

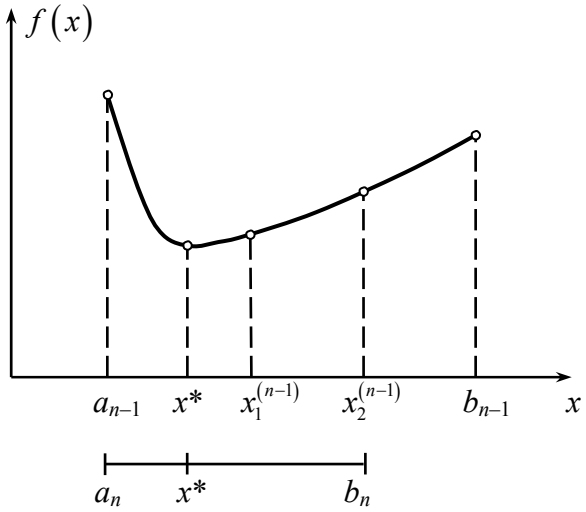


Рис. 1

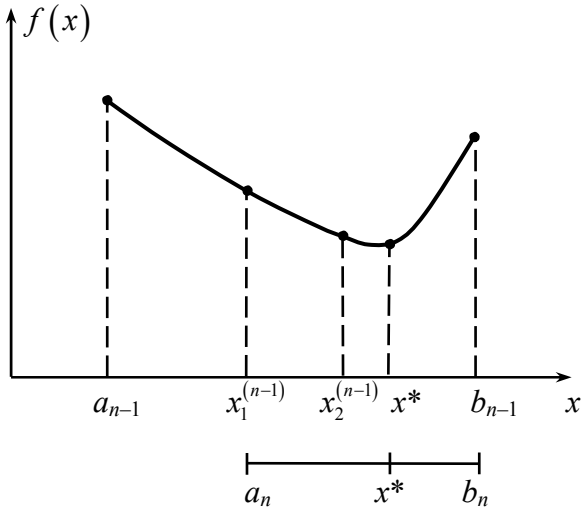


Рис. 2

Пример 2. Найти минимальное значение f^* и точку минимума x^* функции $f(x)=x^4+8x^3-6x^2-72x$ на отрезке $[1.5;\ 2]$. Точку x^* найти с погрешностью $\varepsilon=0.05$.

Решение. Положим $\delta=0.02<2\varepsilon=0.1$. Построим последовательность вложенных отрезков $[a_n;\ b_n]$ по формулам (2), записывая результаты вычислений в таблицу 2.

Таблица 2

n	a_n	b_n	$\varepsilon_n=\dfrac{b_n-a_n}{2}$	$x_1^{(n)}$	$x_2^{(n)}$	$f\left(x_1^{(n)}\right)$	$f\left(x_2^{(n)}\right)$	Примечание
0	1.5	2	0.25	1.74	1.76	–92.135	–92.096	$f\left(x_1^{(0)}\right)<f\left(x_2^{(0)}\right)$, $b_1=x_2^{(0)}$
1	1.5	1.76	0.13	1.62	1.64	–91.486	–91.696	$f\left(x_1^{(1)}\right)>f\left(x_2^{(1)}\right)$, $a_2=x_1^{(1)}$
2	1.62	1.76	0.07	1.68	1.70	–91.995	–92.084	$f\left(x_1^{(2)}\right)>f\left(x_2^{(2)}\right)$, $a_3=x_1^{(2)}$
3	1.68	1.76	0.04					$\varepsilon_3<\varepsilon$, точность достигнута

Следовательно, $x^*\approx 1.732$, $f^*\approx f(1.732)=-92.13$.

Для увеличения скорости сходимости метода величину $\delta\in(0;\ 2\varepsilon)$ целесообразно выбирать как можно меньшей, однако этот выбор ограничен снизу используемым количеством верных десятичных знаков

при задании аргумента x. В любом случае δ должно быть больше машинного нуля в ЭВМ.

Задание 2. Методом деления отрезка пополам найти точку минимума x^* функции $f(x)$ на отрезке $[a; b]$ с абсолютной погрешностью ε и минимум f^*. Результаты представить в виде таблицы 2.

№ вар.	$f(x)$	$[a; b]$	ε	x^*	f^*
1	$f(x) = x\sin x + 2\cos x$	[–5; –4]	0.02		
2	$f(x) = x^4 + 8x^3 - 6x^2 - 72x + 90$	[1.5; 2]	0.05		
3	$f(x) = x^6 + 3x^2 + 6x - 1$	[–1; 0]	0.1		
4	$f(x) = 10x\ln x - \frac{x^2}{2}$	[0; 0.5]	0.05		
5	$f(x) = x^2 + 2\left(x\lg\left(\frac{x}{e}\right) - 2\right)$	[0; 0.5]	0.01		
6	$f(x) = 3x^4 - 10x^3 + 21x^2 + 12x$	[–0.5; 0]	0.01		
7	$f(x) = 2x^2 - \frac{2x}{\ln 2}$	[0; 1]	0.02		
8	$f(x) = e^x - \frac{1}{3}x^3 + 2x$	[–1.5; –1]	0.01		
9	$f(x) = -\sqrt{20x - x^2} + 0.01\sin x$	[9; 11]	0.05		
10	$f(x) = (0.1x - 5)^8 + \cos(0.02x)$	[49; 51]	0.02		
11	$f(x) = \ln x + 0.1\sin(0.1x)$	[10; 12]	0.01		
12	$f(x) = (x - 0.9)^2 + (x - 1.1)^4$	[0.8; 1.2]	0.05		
13	$f(x) = (x - 1)^4 - 2x^2$	[2; 2.5]	0.02		
14	$f(x) = 3(4 - x)^{2/3} + x^2$	[0.6; 0.7]	0.025		
15	$f(x) = -x^3 - 4\ln(1 - x)$	[–1; 0]	0.05		
16	$f(x) = 2 + x^3 - 2x - \ln(3 + x^2)$	[0.5; 1]	0.025		
17	$f(x) = (x + 1)^2\sin(x - 1)$	[0; 1]	0.01		
18	$f(x) = 32x^2(x^2 - 1)^3$	[–1.5; 0]	0.03		
19	$f(x) = x - 2\arctan x$	[0.7; 1.5]	0.02		
20	$f(x) = \frac{1}{x} + 4x^2$	[0; 1.5]	0.05		

1.3. Метод касательных

Метод касательных применяется для минимизации выпуклых дифференцируемых функций. Функция $f(x)$ называется выпуклой на отрезке $[a; b]$, если

$$f\left[\alpha x' + (1-\alpha)x''\right] \le \alpha f(x') + (1-\alpha) f(x'') \tag{4}$$

для произвольных $x', x'' \in [a; b]$ и $\alpha \in [0; 1]$.

Проверка условия (4) почти всегда вызывает затруднения, поэтому на практике используют следующий критерий выпуклости: для того, чтобы дважды дифференцируемая на отрезке $[a; b]$ функция $f(x)$ была выпуклой на отрезке $[a; b]$, необходимо и достаточно, чтобы $f''(x) \ge 0$ при всех $x \in [a; b]$.

Опишем метод касательных. Пусть $f(x)$ – выпуклая дифференцируемая на отрезке $[a; b]$ функция, причем $f'(a) \cdot f'(b) < 0$. Построим последовательности $\{a_n\}$, $\{b_n\}$ и $\{c_n\}$, $n = 1, 2, \ldots$, в соответствии с рекуррентными соотношениями

$$a_0 = a, \ b_0 = b,$$

$$c_{n-1} = \frac{b_{n-1} f'(b_{n-1}) - a_{n-1} f'(a_{n-1}) + f(a_{n-1}) - f(b_{n-1})}{f'(b_{n-1}) - f'(a_{n-1})}, \tag{5}$$

$$a_n = a_{n-1}, \ b_n = c_{n-1} \ \text{ при } \ f'(c_{n-1}) \ge 0, \tag{6}$$

$$a_n = c_{n-1}, \ b_n = b_{n-1} \ \text{ при } \ f'(c_{n-1}) < 0.$$

После n шагов полагаем $x^* \approx c_n$, $f^* \approx f(c_n)$. Требуемая точность минимизации $f(x)$ считается достигнутой, если производная $f'(c_n)$ достаточно близка к нулю, т. е. $|f'(c_n)| \le \varepsilon$, где $\varepsilon > 0$ – заданное число, характеризующее точность.

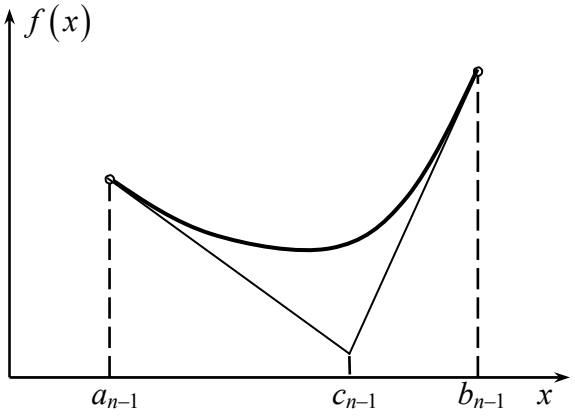


Рис. 3

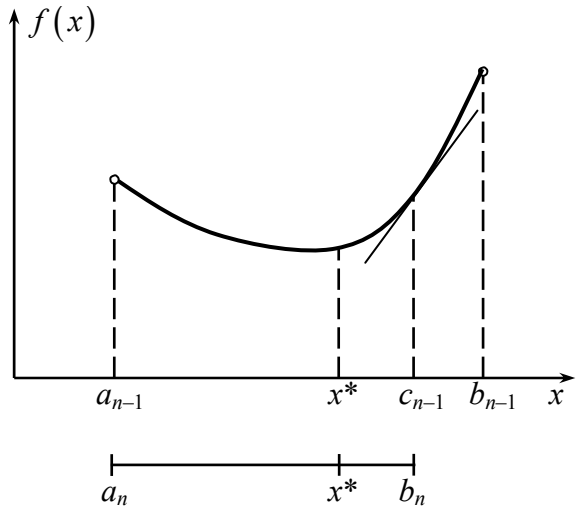


Рис. 4

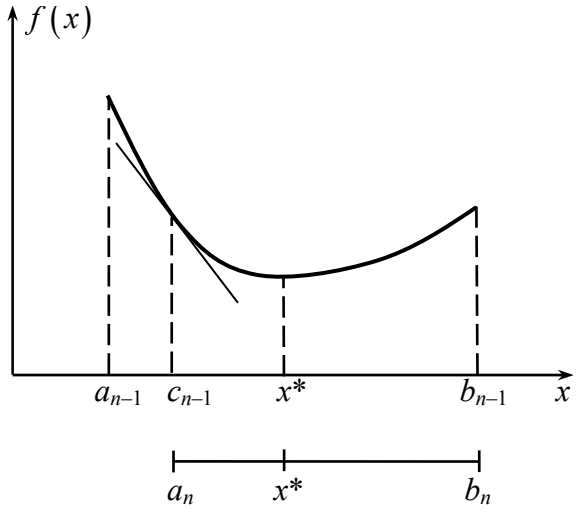


Рис. 5

Метод касательных имеет простой геометрический смысл: величина c_{n-1} из (5) – это абсцисса точки пересечения касательных к графику $f(x)$, проведенных в граничных точках отрезка $[a_{n-1}; b_{n-1}]$ (рис. 3). Рис. 4 и 5 поясняют формулы (5) для случаев $f'(c_{n-1}) \ge 0$ и $f'(c_{n-1}) < 0$ соответственно. Отрезок $[a_n; b_n]$ выбирается так, чтобы $x^* \in [a_n; b_n]$.

Если условие $f'(a) \cdot f'(b) < 0$ не выполняется, то

а) $x^* = a$ при $f'(a) > 0$, $f'(b) > 0$;

б) $x^* = b$ при $f'(a) < 0$, $f'(b) < 0$;

в) $x^* = a$, если $f'(a) = 0$, и $x^* = b$, если $f'(b) = 0$.

Пример 3. Убедиться, что функция $f(x) = x^2 + e^x$ выпукла на $[-1; 1]$, и минимизировать ее методом касательных с точностью $|f'(c_n)| \le 0.05$.

Решение. Так как $f''(x) = 2 + e^x > 0$, то $f(x)$ – выпуклая функция; кроме того, $f'(a) \cdot f'(b) < 0$. Проведем вычисления по формулам (5), (6), поместив результаты вычислений в таблицу 3.

Таблица 3

n	a_n	b_n	c_n	$f'(c_n)$	Примечание
0	–1	1	0.11586	1.35	$f'(c_0) > 0,\ b_1 = c_0$
1	–1	0.11586	–0.41637	–0.173	$f'(c_1) < 0,\ a_2 = c_1$
2	–0.41637	0.11586	–0.14313	0.58	$f'(c_2) > 0,\ b_3 = c_2$
3	–0.41637	–0.14313	–0.27806	0.2012	$f'(c_3) > 0,\ b_4 = c_3$
4	–0.41637	–0.27806	–0.34679	0.0134	$\lvert f'(c_4)\rvert < 0.05$, точность достигнута

Из таблицы 3 находим $x^* \approx c_3 = -0.34679$; $f^* \approx f(c_3) = 0.8272$.

Задание 3. Показать, что функция $f(x)$ является выпуклой на $[a; b]$, и минимизировать ее методом касательных с точностью $|f'(c_n)| \le \varepsilon$, т. е. найти точку минимума x^* и минимальное значение f^*. Представить результаты в виде таблицы 3.

№ вар.	$f(x)$	$[a; b]$	ε	x^*	f^*
1	2	3	4	5	6
1	$f(x) = x - \ln x$	[0.1; 2]	0.01		
2	$f(x) = x^2 - \sin x$	[0; π/2]	0.01		
3	$f(x) = x^4 + x^2 + x + 1$	[–1; 2]	0.01		
4	$f(x) = \frac{x^2}{2} - \cos x$	[–1; 2]	0.01		
5	$f(x) = \sqrt{1 + x^2} + e^{-2x}$	[0; 1]	0.01		
6	$f(x) = e^x + \frac{1}{x}$	[0.1; 2]	0.01		
7	$f(x) = (x - 4)^2 + \ln x$	[3; 5]	0.01		
8	$f(x) = x^4 + e^{-x}$	[0; 1]	0.01		
9	$f(x) = x^3 - \ln x$	[0; 1]	0.01		
10	$f(x) = \sin x + \ln x$	[4; 5]	0.01		
11	$f(x) = \sin x + \operatorname{arctg} x$	[–2; –1]	0.01		
12	$f(x) = 3e^x + \arccos x$	[–1; 0]	0.01		
13	$f(x) = 2e^x - \cos x + 3x^2$	[–1; 0]	0.01		
14	$f(x) = \sqrt{1 + e^{-2x}} + 2x^3$	[0; 1]	0.01		
15	$f(x) = \sqrt{1 + 2x} + 3x^2$	[–0.4; 0]	0.01		
16	$f(x) = e^{\sqrt{2x}} - 10 \sin x$	[1; 2]	0.01		
17	$f(x) = e^{x^2} - 5 \ln x$	[0.1; 1]	0.01		

Окончание табл. 3

1	2	3	4	5	6
18	$f(x)=\frac{x^3}{3-x^2}$	[–5; –2]	0.02		
19	$f(x)=\frac{x^3}{x-1}$	[–3; –0.5]	0.02		
20	$f(x)=\frac{x^4}{x^3-1}$	[1.2; 2.5]	0.02		

1.4. Метод Ньютона

Метод Ньютона использует первую и вторую производные функции $f(x)$ и при определенных условиях обеспечивает значительно более высокую скорость сходимости к точке минимума x^*, чем рассмотренные ранее методы минимизации.

Пусть $f(x)$ – выпуклая, дважды дифференцируемая на **R** функция. Выбрав начальное приближение x_0, построим последовательность

$$x_n = x_{n-1} - \frac{f'(x_{n-1})}{f''(x_{n-1})}, \quad n = 1, 2, \ldots \tag{7}$$

Считая неравенство $|f'(x_n)| \le \varepsilon$ (ε – достаточно малое число) условием достижения требуемой точности вычислений, положим $x^* \approx x_n$, $f^* \approx f(x_n)$.

При неудачном выборе x_0 последовательность (7) может расходиться. Если же точка x_0 достаточно близка к x^*, то эта последовательность сходится к x^* достаточно быстро.

Оценка скорости сходимости может быть сформулирована следующим образом. Пусть $f(x)$ – выпуклая, дважды дифференцируемая на **R** функция, причем $f''(x) \ge \mu > 0$ при всех $x \in \mathbf{R}$ и $f''(x)$ удовлетворяет условию Липшица на **R** с константой *L*. Тогда, если начальное приближение x_0 удовлетворяет условию

$$q = \frac{L}{2\mu^2}|f'(x)| < 1,$$

то последовательность (7) сходится к единственной точке минимума x^* функции $f(x)$ на **R**, причем

$$|x^* - x_n| \le \frac{2\mu}{L} q^{2^n},\ n = 0,1,\ldots$$

Метод Ньютона часто используется на завершающем этапе минимизации, когда точка минимума x^* грубо найдена другим, менее трудоемким методом и требуется найти x^* с большой точностью.

Пример 4. Методом Ньютона найти точку минимума x^* и минимальное значение f^* функции $f(x) = (x-2)^4 - \ln x$ с точностью $|f'(x_n)| \le 10^{-7}$.

Решение. Вычислим производные $f'(x) = 4(x-2)^3 - 1/x$, $f''(x) = 12(x-2)^2 + 1/x^2$. Выберем $x_0 = 3$ и проведем вычисления по формуле (13), записывая результаты в таблице 4.

Таблица 4

n	x_n	$f(x_n)$	$f'(x_n)$
0	3	–0.0986	3.67
1	2.6972477	–0.7558859	0.985
2	2.5322701	–0.8488508	0.208
3	2.4736906	–0.8553636	0.021
4	2.4663735	–0.8554408	0.0003
5	2.4662656	–0.8554408	$5\cdot10^{-8} < 10^{-7}$

Окончательно находим $x^* \approx = 2.4662656$; $f^* \approx = -0.8554408$.

Задание 4. Методом Ньютона найти точку минимума x^* и минимальное значение f^* функции $f(x)$ с точностью $|f'(x_n)| \le \varepsilon$. Представить результаты в виде таблицы 4.

№ вар.	$f(x)$	x_0	ε	x^*	f^*
1	2	3	4	5	6
1	$f(x) = x^2 + e^{-x}$	1	10^{-4}		
2	$f(x) = 2x + e^{-x}$	0.5	10^{-4}		
3	$f(x) = x^2 + x + \sin x$	1.5	10^{-4}		
4	$f(x) = x^2 - x + e^{-x}$	2	10^{-4}		
5	$f(x) = e^x + e^{-2x} + 2x$	–0.5	10^{-4}		
6	$f(x) = 2x^2 + x + \cos^2 x$	1	10^{-4}		
7	$f(x) = x - \ln x$	0.3	10^{-6}		

Окончание табл. 4

1	2	3	4	5	6
8	$f(x)=x^2-\sin x$	0.7	10^{-6}		
9	$f(x)=x^4+x^2+x+1$	0.5	10^{-6}		
10	$f(x)=\frac{x^2}{2}-\cos x$	1.2	10^{-6}		
11	$f(x)=\sqrt{1+x^2}+e^{-2x}$	1	10^{-6}		
12	$f(x)=e^x+\frac{1}{x}$	0.1	10^{-6}		
13	$f(x)=(x-4)^2+\ln x$	2.5	10^{-6}		
14	$f(x)=x^4+e^{-x}$	–0.7	10^{-6}		
15	$f(x)=x^3-\ln x$	1.5	10^{-5}		
16	$f(x)=\sin x+\ln x$	3.8	10^{-5}		
17	$f(x)=\sin x+\operatorname{arctg} x$	–1	10^{-5}		
18	$f(x)=\frac{x^3+2x^2+7x-3}{2x^2}$	1.3	10^{-4}		
19	$f(x)=\frac{x}{2}+\operatorname{arc\,cot} x$	0.4	10^{-4}		
20	$f(x)=\frac{(x^2-1)(x-2)}{x}$	0.7	10^{-4}		

1.5. Метод безусловной минимизации, основанный на вычислении первых производных функции

Постановка задачи минимизации функции n переменных $f(\mathbf{x})=f(x_1,x_2,\ldots,x_n)$ на множестве $U\subset\Re_n$ не отличается от постановки в одномерном случае. Если $U=\Re_n$, то говорят о безусловной минимизации функции $f(\mathbf{x})$.

Для решения задачи безусловной минимизации функции $f(\mathbf{x})$ наиболее часто применяют приближенные методы, в основе которых лежит вычисление производных $f(\mathbf{x})$ первого порядка. Такие методы обычно называют градиентными. В ряде других методов требуется вычисление не только первых, но и вторых производных функции $f(\mathbf{x})$.

Метод градиентного спуска. Пусть $f(\mathbf{x})$ – выпуклая дифференцируемая во всем пространстве $\Re_n$ функция и требуется найти ее точку минимума $\mathbf{x}^*$. Выбрав произвольное начальное приближение $\mathbf{x}^{(0)} \in \Re_n$, построим последовательность

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k f'\left(\mathbf{x}^{(k)}\right), \quad k = 0, 1, \ldots, \tag{8}$$

где величины α_k (параметрические шаги) выбираются достаточно малыми для того, чтобы выполнялось условие

$$f\left(\mathbf{x}^{(k+1)}\right) < f\left(\mathbf{x}^{(k)}\right), \ k = 0, 1, \ldots. \tag{9}$$

В качестве условия окончания вычислений обычно используется близость к нулю градиента $f'\left(\mathbf{x}^{(k)}\right)$, т. е. выполнение неравенств

$$\left|\frac{\partial f\left(\mathbf{x}^{(k)}\right)}{\partial x_i}\right| \le \varepsilon, \ i = 1, 2, \ldots, n, \qquad \text{или} \qquad \left\|f'\left(\mathbf{x}^{(k)}\right)\right\| = \sqrt{\sum_{i=1}^{n}\left[\frac{\partial f\left(\mathbf{x}^{(k)}\right)}{\partial x_i}\right]^2} \le \varepsilon$$

(ε – заданное достаточно малое число), после чего полагают $\mathbf{x}^* \approx \mathbf{x}^{(k)}$, $f^* \approx f\left(\mathbf{x}^{(k)}\right)$.

Если при некотором k условие (9) нарушается, то шаг α_k в (8) уменьшают (дробят) в заданное число раз до выполнения неравенства (9) и продолжают вычисления.

Пример 5. Минимизировать в $\Re_2$ функцию $f(\mathbf{x}) = f(x_1, x_2) = x_1^2 + 2x_2^2 + e^{x_1+x_2}$ методом градиентного спуска, завершив вычисления при $\left|\partial f\left(\mathbf{x}^{(k)}\right)\Big/\partial x_i\right| \le 0.05, \ i = 1, 2$.

Решение. Выбрав начальное приближение $\mathbf{x}^{(0)} = (0,0)$ и $\alpha_0 = 1$, построим последовательность (4), записывая результаты вычислений в таблице 5.

Вычислим производные

$$\partial f\left(\mathbf{x}^{(k)}\right)\Big/\partial x_1 = 2x_1 + e^{x_1+x_2}, \qquad \partial f\left(\mathbf{x}^{(k)}\right)\Big/\partial x_2 = 4x_2 + e^{x_1+x_2}$$

Таблица 5

k	$x_1^{(k)}$	$x_2^{(k)}$	$f\left(\mathbf{x}^{(k)}\right)$	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_1}$	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_2}$	α_k	Примечание
0	0 –1 0	0 –1 0	1 3.145 1	1 – 1	1 – 1	1 0.5	(5) нарушено уменьшаем α_0

	–0.5 0	–0.5 0	1.118 1	– 1	– 1	 0.25	в 2 раза
1	–0.25	–0.25	0.794	0.106	–0.393	0.25	(5) выполнено
2	–0.27663	–0.15163	0.774	0.0983	0.0451	0.25	(5) выполнено
3	–0.30126	–0.16291	0.772	0.0262	–0.023		точность достигнута

Таким образом, $\mathbf{x}^* \approx (-0.3012, -0.1629)$, $f^* \approx 0.7725$

Задание 5. Минимизировать в $\Re_2$ функцию $f(\mathbf{x}) = f(x_1, x_2)$ методом градиентного спуска, завершив вычисления при $\left|\partial f\left(\mathbf{x}^{(k)}\right)/\partial x_i\right| \le 0.001,\ i = 1, 2$.

№ вар.	$f(\mathbf{x})$	$\partial f(\mathbf{x})/\partial x_1$	$\partial f(\mathbf{x})/\partial x_2$	x^*	f^*
1	$f(\mathbf{x}) = 7x_1^2 + 2x_1x_2 + 5x_2^2 + x_1 - 10x_2$				
2	$f(\mathbf{x}) = 3x_1^2 - 3x_1x_2 + 4x_2^2 - 2x_1 + x_2$				
3	$f(\mathbf{x}) = x_1^2 + 4x_1x_2 + 17x_2^2 + 5x_2$				
4	$f(\mathbf{x}) = 5x_1^2 - 4x_1x_2 + 5x_2^2 - x_1 - x_2$				
5	$f(\mathbf{x}) = 4x_1^2 + 4x_1x_2 + 6x_2^2 - 17x_1$				
6	$f(\mathbf{x}) = 2x_1^2 - 2x_1x_2 + 3x_2^2 + x_1 - 3x_2$				
7	$f(\mathbf{x}) = 10x_1^2 + 3x_1x_2 + x_2^2 + 10x_2$				
8	$f(\mathbf{x}) = x_1^2 - 2x_1x_2 + 6x_2^2 + x_1 - x_2$				
9	$f(\mathbf{x}) = x_1^2 + 2x_2^2 + e^{x_1^2 + x_2^2} - x_1 + 2x_2$				
10	$f(\mathbf{x}) = \sqrt{x_1^2 + x_2^2 + 1} + \frac{1}{2}x_1 - \frac{1}{2}x_2$				
11	$f(\mathbf{x}) = x_1^4 + 2x_2^4 + x_1^2x_2^2 + 2x_1 + x_2$				
12	$f(\mathbf{x}) = x_1^2 + 3x_2^2 + \cos(x_1 + x_2)$				
13	$f(\mathbf{x}) = \sqrt{1 + 2x_1^2 + x_2^2} + e^{x_1^2 + 2x_2^2} - x_1 - x_2$				
14	$f(\mathbf{x}) = x_1 + 5x_2 + e^{x_1^2 + x_2^2}$				
15	$f(\mathbf{x}) = x_1^4 + x_2^4 + \sqrt{2 + x_1^2 + x_2^2} - 2x_1 + 3x_2$				
16	$f(\mathbf{x}) = 2x_1^2 + 3x_2^2 - 2\sin\left(\frac{x_1 - x_2}{2}\right) + x_2$				
17	$f(\mathbf{x}) = x_1^2 + e^{x_1^2 + x_2^2} + 4x_1 + 3x_2$				
18	$f(\mathbf{x}) = (x_1 - 4)^2 + (x_2 + 4)^2$				
19	$f(\mathbf{x}) = 4x_1^2 + 4x_1 + x_2^2 - 8x_2 + 5$				
20	$f(\mathbf{x}) = (\ln x_1) - x_2,\quad x_1 \ge 1$				

2. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Рассмотрим задачу линейного программирования с двумя переменными и получим ее решение графически [1]:

$$f(\mathbf{x}) = c_1 x_1 + c_2 x_2 \to \min, \tag{10}$$

$$a_{i1} x_1 + a_{i2} x_2 \le b_i, \quad i = 1, \ldots, m, \tag{11}$$

$$x_1 \ge 0, \;\; x_2 \ge 0. \tag{12}$$

На плоскости (x_1, x_2) любое из неравенств (11) определяет полуплоскость, лежащую по одну из сторон от прямой $a_{i1} x_1 + a_{i2} x_2 = b_i$. Для того чтобы определить расположение этой полуплоскости относительно граничной прямой, можно подставить координаты какой-либо точки (при $b_i \ne 0$ проще всего взять начало координат) в соответствующее неравенство (11) и проверить его выполнение.

Таким образом, допустимое множество U задачи (10)-(12) является пересечением первого квадранта $x_1 \ge 0, \; x_2 \ge 0$ и полуплоскостей, соответствующих неравенствам (11). Поэтому множество U представляет собой либо:

а) пустое множество, тогда задачи (10)-(12) не имеют решений из-за несовместности ограничений (11), (12);

б) многоугольник (рис. 6);

в) неограниченное многоугольное множество.

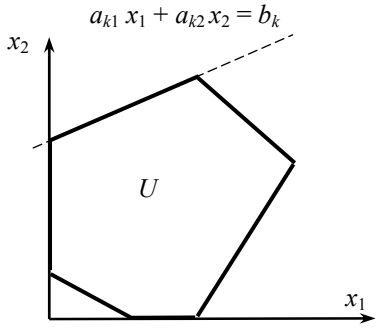


Рис. 6

Для решения задач (10)-(12) в случае $U \ne \varnothing$ рассмотрим семейство линий уровня функции $f(\mathbf{x})$ из (10)

$$c_1 x_1 + c_2 x_2 = C, \; C = \text{const}, \tag{13}$$

которые являются параллельными прямыми. Антиградиент $-f'(\mathbf{x}) = (-c_1, -c_2) = \mathbf{e}$ перпендикулярен прямым (13) и указывает направление убывания $f(\mathbf{x})$. Если перемещать параллельно самой себе

произвольную прямую (13), проходящую через допустимое множество U, в направлении **e** убывания $f(\mathbf{x})$ до тех пор, пока эта прямая будет иметь хотя бы одну общую точку с множеством U, то в своем крайнем положении указанная прямая пройдет через точку множества U, в которой целевая функция $f(\mathbf{x})$ принимает минимальное на U значение.

Пример 6. Используя графический метод, найти решение следующей задачи линейного программирования:

$f(\mathbf{x}) = -3x_1 - 2x_2 \to \min$

$x_1 + 2x_2 \le 7, \quad 2x_1 + x_2 \le 8,$

$x_2 \le 3, \;\; x_1 \ge 0, \;\; x_2 \ge 0.$

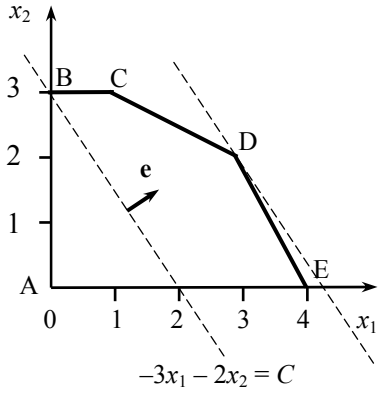


Рис. 7

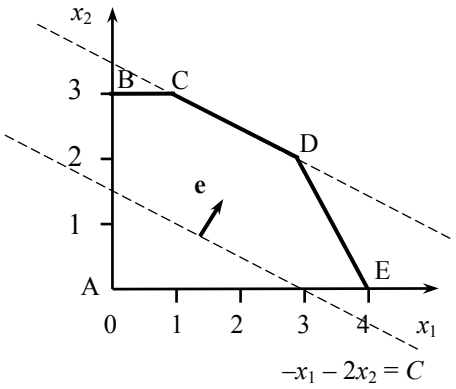


Рис. 8

Изобразим на плоскости (x_1, x_2) допустимое множество U данной задачи (многоугольник $ABCDE$) и одну из линий уровня $-3x_1 - 2x_2 = C$ целевой функции (рис. 7). Направление убывания $f(\mathbf{x})$ указывает вектор $\mathbf{e} = (3,2)$. Совершая параллельный перенос линии уровня вдоль направления **e**, находим ее крайнее положение. В этом положении прямая $-3x_1 - 2x_2 = C$ проходит через вершину D (3, 2) многоугольника $ABCDE$. Поэтому целевая функция $f(\mathbf{x})$ принимает минимальное значение f^* в точке $\mathbf{x}^* = (3, 2)$, причем $f^* = f(3,2) = -13$.

Задача линейного программирования (10)-(12) может иметь и бесконечное множество решений.

Пример 7. Решить задачу линейного программирования с целевой функцией $f(\mathbf{x}) = -x_1 - 2x_2$ и ограничениями на допустимое множество U, взятыми из примера 2.

Множество U построено в примере 6. На рис. 8 изображена линия уровня $-x_1 - 2x_2 = C$ целевой функции $f(\mathbf{x})$. В своем крайнем положении при параллельном переносе вдоль направления $\mathbf{e} = (1,2)$ она содержит сторону CD многоугольника $ABCDE$. Таким образом, все точки отрезка CD являются точками минимума функции $f(\mathbf{x})$ на множестве U. Так как концы C и D этого отрезка имеют координаты (1, 3) и (3, 2) соответственно, то любая точка минимума $f(\mathbf{x})$ представима в виде $\mathbf{x}^* = \alpha(1,3) + (1-\alpha)(3,2) = (3 - 2\alpha, 2 + \alpha)$, где $\alpha \in [0; 1]$. Минимальное значение целевой функции $f^* = f(\mathbf{x}^*) = -7$.

Задание 6. Используя графический метод, найти решение следующей задачи линейного программирования (в соответствии с вариантом).

№ вар.	$f(\mathbf{x})$	Граничные условия	$\mathbf{x}^*$	f^*
1	2	3	4	5
1	$f(\mathbf{x}) = x_1 - 2x_2 \to \min$	$-x_1 + x_2 \le 0$, $2x_1 + x_2 \le 3$, $x_1 - x_2 \le 1$, $x_1 \ge 0$, $x_2 \ge 0$		
2	$f(\mathbf{x}) = -x_1 - 3x_2 \to \min$	$2x_1 + x_2 \le 2$, $x_1 - x_2 \ge 0$, $x_1 - x_2 \le 1$, $x_1 \ge 0$, $x_2 \ge 0$		
3	$f(\mathbf{x}) = -x_1 - 4x_2 \to \min$	$x_1 + 2x_2 \ge 2$, $x_1 \le 2$, $x_2 \le 2$, $x_1 + x_2 \le 3$, $x_1 \ge 0$, $x_2 \ge 0$		
4	$f(\mathbf{x}) = -x_1 - x_2 \to \min$	$x_1 + x_2 \ge 1$, $x_1 - x_2 \ge -1$, $x_1 - x_2 \le 1$, $x_1 \le 2$, $x_2 \le 2$, $x_1 \ge 0$, $x_2 \ge 0$		
5	$f(\mathbf{x}) = 3x_1 + 2x_2 \to \max$	$2x_1 + 3x_2 \le 8$, $2x_1 + x_2 \le 4$, $-2x_1 + x_2 \le 1$, $x_1 \ge 0$, $x_2 \ge 0$		
6	$f(\mathbf{x}) = 3x_1 - x_2 \to \max$	$2x_1 - x_2 \le 4$, $x_1 - 2x_2 \le 2$, $x_1 + x_2 \le 5$, $x_1 \ge 0$, $x_2 \ge 0$		
7	$f(\mathbf{x}) = x_1 + 2x_2 \to \max$	$3x_1 - 2x_2 \le 6$, $-x_1 + 2x_2 \le 4$, $3x_1 + 2x_2 \le 12$, $x_1 \ge 0$		
8	$f(\mathbf{x}) = 2x_1 + x_2 \to \max$	$-x_1 + x_2 \le 2$, $x_1 + 2x_2 \le 7$, $4x_1 - 3x_2 \le 6$, $x_1 \ge 0$, $x_2 \ge 0$		
9	$f(\mathbf{x}) = 7x_1 + 5x_2 \to \min$	$x_1 + x_2 \ge 3$, $x_1 + 5x_2 \ge 5$, $2x_1 + x_2 \ge 4$		

Окончание таблицы

1	2	3	4	5
10	$f(\mathbf{x}) = -x_1 + 2x_2 \to \min$	$2x_1 - 3x_2 \geq 0,\quad x_1 - x_2 \leq 3,$ $2x_1 - x_2 = 4$		
11	$f(\mathbf{x}) = x_1 + x_2 \to \max$	$x_1 + x_2 \leq 1,\ \ x_1 \geq 0,\ \ x_2 \geq 0$		
12	$f(\mathbf{x}) = x_1 + 2x_2 \to \max$	$x_1 + x_2 \leq 1,\ x_1 - x_2 \leq 1,$ $x_1 \geq 0,\ \ x_2 \geq 0$		
13	$f(\mathbf{x}) = x_1 - 2x_2 \to \max$	$-1 \leq x_1 + x_2 \leq 1$ $-1 \leq -x_1 + x_2 \leq 1$		
14	$f(\mathbf{x}) = 3x_1 + 8x_2 \to \max$	$-2 \leq x_1 + x_2 \leq 2$ $-2 \leq -x_1 + x_2 \leq 2,\ -1 \leq x_1 \leq 1$		
15	$f(\mathbf{x}) = 4x_1 + x_2 \to \max$	$-x_1 + x_2 \leq 1,\ \ x_2 \leq 1,$ $-1 \leq x_1 \leq 10$		
16	$f(\mathbf{x}) = x_1 - 2x_2 \to \max$	$x_1 + x_2 \leq 1,\ \ -x_2 \leq 3$		
17	$f(\mathbf{x}) = x_1 - x_2 \to \max$	$2x_1 + x_2 \leq 1,\ x_1 + 2x_2 \leq 1,$ $2x_1 - x_2 = 1,\ \ x_1 \geq 0,\ \ x_2 \geq 0$		
18	$f(\mathbf{x}) = 4x_1 + 2x_2 \to \max$	$2x_1 + 3x_2 \leq 18,\ -x_1 + 3x_2 \leq 9,$ $2x_1 - x_2 \leq 10,\ \ x_1 \geq 0,\ \ x_2 \geq 0$		
19	$f(\mathbf{x}) = 2x_1 + 3x_2 \to \min$	$3x_1 + 2x_2 \geq 6,\ x_1 + 4x_2 \geq 4,$ $x_1 \geq 0,\ \ x_2 \geq 0$		
20	$f(\mathbf{x}) = 2x_1 + 3x_2 \to \max$	$3x_1 + 2x_2 \leq 6,\ x_1 + x_2 \geq 1,$ $x_1 \geq 0,\ \ x_2 \geq 0$		

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Вуколов, Э. А. Сборник задач по математике для втузов. Ч.4. Методы оптимизации. Уравнения в частных производных. Интегральные уравнения: учебное пособие / Э. А. Вуколов, А. В. Ефимов, В. Н. Земсков и др.; под ред. А. В. Ефимова. – 2-е изд., перераб. – Москва : Физматлит, 1990. – 304 с.

2. Пантелеев, А. В. Методы оптимизации в примерах и задачах: учебное пособие / А. В. Пантелеев, Т. А. Летова. – Москва : Высшая школа, 2002. – 544 с.

3. Аттетков, А. В. Введение в методы оптимизации: учебное пособие / А. В. Аттетков, В. С. Зарубин, А. Н. Канатников. – Москва : Инфра-М; Финансы и статистика, 2008. – 272 с.

4. Ашманов, С. А. Теория оптимизации в задачах и упражнениях / С. А. Ашманов, А. В. Тимохов. – Москва : Наука, 1991. – 448 с.

5. Заславский, Ю. Л. Сборник задач по линейному программированию: учебное пособие для вузов / Ю. Л. Заславский; под ред. Д. Б. Юдина. – Москва : Наука, 1969. – 256 с.

6. Калихман, И. Л. Сборник задач по математическому программированию: учебное пособие для вузов / И. Л. Калихман. – 2-е изд., перераб. и доп. – Москва : Высшая школа, 1975. – 270 с.

Приложение 1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Радиотехнический
Кафедра Телекоммуникации
Дисциплина Методы моделирования и оптимизации

КУРСОВАЯ РАБОТА

Тема Поиск экстремумов функции численными методами
(вариант задания №)

Выполнил студент _____ /_____/
подпись инициалы, фамилия

Курс 1 Группа Ткмд-11

Направление/ специальность 11.04.02 «Инфокоммуникационные технологии и системы связи»

Руководитель доцент, к.т.н., Служивый М.Н.
должность, ученая степень, ученое звание

_____
фамилия, имя, отчество

Дата сдачи:
«___» января 20___ г.

Дата защиты:
«__» января 20___ г.

Оценка: _____

Ульяновск
20__ г.

Приложение 2

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Радиотехнический
Кафедра Телекоммуникации
Дисциплина Методы моделирования и оптимизации

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

студенту ТКмд-11 _____
группа фамилия, инициалы

Тема работы Поиск экстремумов функции численными методами (вариант задания №__)

Срок сдачи законченной работы «10» января 20____г.

Исходные данные к работе базовое предприятие – Ульяновский государственный технический университет; функции одной и двух переменных; область определения функции; анализируемый интервал значений независимой переменной; допустимая величина погрешности вычислений

Рекомендуемая литература:

Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – Изд. 2-е. – Москва : – СПб. : Физматлит, 2001. – 630 с.

Васильев Ф.П. Численные методы решения экстремальных задач: учебное пособие для вузов. – Изд. 2-е, перераб. и доп. – Москва : Наука, 1988. – 552 с.

Плохотников К.Э. Вычислительные методы. Теория и практика в среде MATLAB: курс лекций: Учебное пособие для вузов. Изд. 2-е, испр. – Москва : Горячая линия – Телеком, 2013. – 496 с.

Поршнев С.В. Вычислительная математика (курс лекций): учебное пособие для вузов. – СПб. : БХВ-Петербург, 2004. – 320 с.

Семушин И.В. Вычислительные методы алгебры и оценивания: учебное пособие. – Ульяновск : УлГТУ, 2011. – 366 с.

Уайлд Д.Дж. Методы поиска экстремума. / пер. с англ. под ред. А.А. Фельдбаума. – М.: Наука, 1967. – 267 с.

Пантелеев, А. В. Методы оптимизации в примерах и задачах: учебное пособие / А. В. Пантелеев, Т. А. Летова. – М. : Высшая школа, 2002. – 544 с.

Содержание пояснительной записки (перечень подлежащих разработке вопросов)

1. Теоретическая часть, содержащая алгоритмы поиска экстремумов.
2. Формулировка заданий в соответствии с вариантом.
3. Результаты вычислений, представленные в виде формул, таблиц и графиков.
4. Выводы по выполненной работе.

Перечень графического материала (с точным указанием обязательных чертежей)
1. Структурные схемы алгоритмов поиска экстремумов.
2. Графики анализируемых функций.

Руководитель _доцент_____ _____ /__М.Н. Служивый____/
должность подпись инициалы, фамилия

«_30__»__сентября_____20___г

Студент _____ /_____/
подпись инициалы, фамилия

«_30__»__сентября_____20___г

Учебное электронное издание

МЕТОДЫ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

Учебно-методическое пособие

Составитель *Служивый* Максим Николаевич

Редактор Н.А. Евдокимова

Дата подписания к использованию 09.04.2019.
ЭИ № 1259. Объем данных 0,3 Мб. Заказ № 413.

Ульяновский государственный технический университет
432027, г. Ульяновск, ул. Сев. Венец, 32.
ИПК «Венец» УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, 32.

Тел.: (8422) 778-113
E-mail: venec@ulstu.ru
venec.ulstu.ru

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.03 Проектирование ИС

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Декан факультета информационных систем и технологий
_____ Святов К.В.
«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Дисциплина (модуль) Проектирование интеллектуальных систем
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Методические рекомендации составлены

Практическая работа №1-2

Решение задач логического вывода.

1. Изучить основные правила логического вывода MODUS PONENs, правило резолюций. Проанализировать актуальные известные действующие проекты в области искусственного интеллекта (ИИ).

2. Письменно ответить на вопросы, используя только правило вывода PODUS PONENS:

Восстание декабристов 14 декабря 1825 года потерпело поражение

Был ли виноват князь Сергей Трубецкой в поражении восстания?

Почему поводом начала восстания послужила смерть императора Александра Первого

3. Подготовить отчет по практической работе.

Также необходимо быть готовым устно ответить на контрольные вопросы:

Контрольные вопросы

1. Понятие логической модели вывода.
2. Дедукция и индукция.
3. Прямой и обратный вывод.
4. Правило резолюций

Основные понятия логического вывода:

Логические модели

Среди всех моделей знаний логическая модель отличается математической строгостью. В основе логической модели знаний лежит понятие формальной теории и отношения, которые существуют между единицами знаний можно описывать только с помощью синтаксических правил, допустимых в рамках этой теории.

Формальная теория задается всегда четверкой символов

$$S=<B, F, A, R> \quad (1)$$

где B – конечное множество базовых символов, иначе – алфавит теории S; F – подмножество выражений теории S, называемых формулами теории. Обычно имеется эффективная процедура, которая представляет собой совокупность правил, позволяющих из элементов множества B строить синтаксически правильные выражения; A – выделенное множество правил, называемых аксиомами теории, т.е. множество априорно истинных формул; R – конечное множество отношений $\{ r_1, r_2, \dots , r_n \}$ между формулами, называемыми правилами вывода. Для любого ri существует целое положительное число j, такое, что для каждого множества, состоящего из j формул, и для каждой формулы F эффективно решается вопрос о том, находятся ли эти j-формулы в отношении ri с формулой F. Если ri выполняется, то F называют непосредственным следствием F-формул по правилу ri.

Следствием (выводом) формулы в теории S называется такая последовательность правил, что для любого из них представленная формула является либо аксиомой теории S, либо непосредственным следствием.

Правила вывода, которые разрабатываются проектировщиками, позволяют расширить множество формул, которые являются аксиомами теории.

Для понимания идеи логического вывода рассмотрим автоматическое доказательство теорем, основанное на классическом принципе резолюции. Составление программ, доказывающих теоремы на основе математической логики, позволяет изучить дедукцию в чистом виде. Понимание дедукции очень важно для логического вывода. Профессор Дж. Маккарти в 1959 году [2] назвал способность программы делать дедуктивные заключения из заданных фактов «здравым смыслом». Этот тип здравого смысла является составной частью человеческого интеллекта.

Дедукция (лат. *deductio* – выведение) – метод мышления, при котором новое положение выводится чисто логическим путем из предшествующих, вывод по правилам логики; цепь умозаключений (рассуждение), звенья которой (высказывания) связаны отношением логического следования. Началом (посылками) дедукции являются аксиомы, постулаты или просто гипотезы, имеющие характер общих утверждений («общее»), а концом – следствия из посылок, теоремы («частное»). Если посылки дедукции истинны, то истинны и ее следствия. Дедукция противоположна *индукции.* Знаменитый детектив Шерлок Холмс, герой английского писателя Артура Конан-Дойла, в своих умозаключениях использовал дедуктивный метод, позволяющий ему эффективно расследовать преступления.

Основы логического вывода

При автоматическом доказательстве теорем существуют два направления: поиск доказательства и поиск следствия. Программа, выполняющая поиск доказательства, пытается найти доказательство какой-либо заданной теоремы. Программе, выполняющей поиск следствия, задается множество аксиом, после чего она пытается построить дедуктивные следствия из этих аксиом и выбрать следствия, представляющие интерес. Обе программы используют принцип резолюции – естественное и мощное правило вывода. Программа, доказывающая теоремы, обладает свойством «здравого смыла», по определению Маккарти [2], то есть способностью делать дедуктивные заключения из заданных фактов. Этот тип здравого смысла является важной частью человеческого интеллекта.

Математическая логика прекрасно подходит для вычислительных машин, т.к. в течение многих лет логики работали над тем, чтобы правила вывода стали «механическими» [3]. Классическими в этом направлении считаются работы П. Гилмора [4], Х. Ванга [5], М. Дейвиса и Г. Путнама [6]. Основной идеей в этих программах является замена ряда переменными константными термами и проверка того, не получено ли доказательство теоремы. Если нет, то добавляются новые константные термы, делается очередная проверка и т.д. В дальнейшем Дж. Робинсон разработал правило вывода, которое назвал *принципом резолюции* [7].

Рассмотрим подробнее принцип резолюции.

Итак, принцип резолюции выводит из двух заданных утверждений (посылок) наиболее общее возможное заключение и обе посылки обычно содержат переменные. Прежде всего, это **правило вывода**. Для иллюстрации приведем пример [3]:

Теорема 1.

Пусть известно, что

P1: Если x есть часть v и если v есть часть y, то x есть часть y.

P2: Палец есть часть кисти руки.

P3: Кисть руки есть часть руки.

P4: Рука есть часть человека.

Активизируя все 4 посылки P1-P4, мы должны заключить, что палец есть часть человека.

Докажем теорему.

Сначала формируется отрицание доказываемого утверждения, а затем попытаемся вывести противоречие. В нашем примере отрицанием доказываемой теоремы будет следующее утверждение:

P5: Палец не есть часть человека

Резольвентой (следствием) P1 и P2 будет:

P6: Если кисть руки есть часть y , то палец есть часть y.

То есть P6 получается путем отождествления предложения P2 и первой части предложения P1: замены x на «палец» и замены v на «кисть руки». Эта подстановка в P1 даст промежуточный результат, являющийся логическим следствием из P1:

P1': Если палец есть часть кисти руки и кисть руки есть часть y, то палец есть часть y.

Это выражение действительно является логическим следствием из P1, т.к. P1 истинно для всех x и всех v (и для всех y), а поэтому истинно и в частном случае, когда x есть «палец», а v – «кисть руки». Предложение P6 является непосредственным следствием из P1' и P2. Обычно резольвенту P6 получают непосредственно, минуя промежуточный вариант P1'.

Каждая из трех частей предложения P1 называется *атомарной формулой*. Сопоставляем предложение P3 и первую атомарную формулу в предложении P6, заменяя в P6 y на слово «рука». Получаем следующий промежуточный результат:

P6': Если кисть руки есть часть руки, то палец есть часть руки.

Это предложение вместе с P3 дает резольвенту P7.

P7: Палец есть часть руки.

Аналогично получаем резольвенту Р8 путем отождествления Р7 и первой атомарной формулы предложения Р1:

Р8: Если рука есть часть у, то палец есть часть у.

Из Р4 и первой атомарной формулы предложения Р8 получаем резольвенту Р9.

Р9: Палец есть часть человека.

Отождествляющая подстановка для полученного предложения Р9 и предложения Р5 приводит к противоречию, что и требовалось доказать.

Приведем точное определение *принципа резолюции*. Он сочетает две основные идеи:

1. Принцип силлогизма в пропозициональном исчислении.
   Он устанавливает, что из $a \vee b$ и $\neg a \vee c$ можно сделать вывод, что $b \vee c$.
2. Принцип отыскания частных случаев в исчислении предикатов.

Принцип устанавливает, что от формулы $F(v_1, v_2, \ldots, v_n)$, справедливой для всех значений v_i можно перейти к формуле $F(t_1, t_2, \ldots, t_n)$ путем подстановки термов t_i соответственно, вместо переменных v_i. Введем определение терма.

Терм это:
1) индивидуальная константа, например, «палец»;
2) индивидуальная переменная, например, x;
3) функция других термов, например, $g(x,y)$ и $h(s, m(g(s)))$.

Рассмотрим теперь понятие общей резолюции. Дизъюнктивное предложение состоит из дизъюнкции литер. Литера – это атомарная формула или ее отрицание. Резольвента литеры одного предложения и литеры другого предложения является следствием двух этих предложений, рассматриваемых совместно. Резольвента получается следующим образом:

- переменные переименовываются таким образом, чтобы все переменные в одном предложении отличались от переменных в другом предложении;
- находится минимальная подстановка, если она существует, которая делает литеры одинаковыми, но противоположными по знаку;
- подстановка выполняется повсеместно в обоих предложениях;
- если после подстановки одна и та же литера входит в предложение более, чем один раз, в этом предложении вычеркиваются все одинаковые литеры, кроме одной;
- вычеркиваются две литеры, которые стали одинаковыми, но имеют противоположные знаки;
- резольвента ─ это дизъюнкция литер, оставшихся в первом предложении, и литер, оставшихся во втором предложении.

Рассмотрим общее разложение (часть принципа резолюции). Множитель, если он существует, от двух или более литер предложения, есть следствие этого предложения. Множитель получаем следующим образом:

A. Находится минимальная подстановка, если она существует, делающая литеры одинаковыми и с теми же знаками.
B. Выполняется постановка повсеместно в предложении.
C. Вычеркиваются все, кроме одной литеры, которые оказались одинаковыми в результате подстановки.
D. Множитель это дизъюнкция оставшихся литер.

Рассмотрим классический пример применения принципа резолюции.

{(U∨V), (W ∨¬V)}⇒(U ∨W) или {V, ¬ V ∨W)} ⇒W

Известно, что всякий, кто находится в здравом уме, может понимать математику. Ни один из сыновей Гегеля не может понимать математику. Сумасшедшие не допускаются к голосованию.

В результате применения принципа резолюции получаем, что **никто из сыновей Гегеля не допускается к голосованию.**

Дж. Робинсон в 1965 году [7] впервые показал, что принцип резолюции является эффективной и полной процедурой поиска доказательства. Р. Ли впервые доказал, что правило резолюции является полным в применении к отысканию следствий. Эффективность принципа резолюций означает, что при его использовании можно написать программу, которая за конечное число шагов найдет множители любого предложения и резольвенты любых двух предложений.

Принцип резолюции является полным как для нахождения доказательства, так и для отыскания следствий. В 60-х годах XX века среди авторов, разработавших программы, способные находить доказательства, отметим П. Гилмора [4].

Стратегией в доказательстве теорем с использованием принципа резолюции является эвристика, аналогичная порождающей процедуре для деревьев игр.

Метод доказательства теорем является одним из ключевых в решении задачи по искусственному интеллекту, однако решение далеко не всех задач возможно данным методом.

К достоинствам логической модели относят наличие стандартной типовой процедуры логического вывода (доказательства теорем). Однако такое единообразие влечет за собой основной недостаток модели – сложность использования в процессе логического вывода эвристик, отражающих специфику предметной области.

К другим недостаткам логической модели относят:

- монотонность;
- комбинаторный взрыв;
- слабость структурированности описаний.

Важно помнить, выбирая модель логического вывода для формирования базы знаний интеллектуальных систем, что модель логического вывода не может содержать никаких вероятностей или неопределенностей. Если предметная область не может быть описана только с использованием известных фактов или аксиом и правил логического вывода, подразумевающих 100 % уверенность в полученном результате, то нужно выбирать другую форму представления знаний.

Практическая работа № 2

Проектирование семантических сетей.

1. Изучить основные понятия семантических сетей. Классифицирующие сети как разновидность семантических сетей. Типы связей в классифицирующих сетях.

2. Построить классифицирующую сеть для известной системы классификации, например, классификацию растений, животных и т.п. Выбор предметной области согласовать с преподавателем.

3. Построить семантическую сеть для анализа предложения на естественном языке. Варианты предложений получить у преподавателя.

Форма отчета:

1. Граф классифицирующей сети
2. Граф, описывающей предложения естественного языка

Сетевые модели представления знаний

Сетевые модели (СМ) возникли в 60-х годах XX в. Они являются методом представления знаний посредством узлов, соответствующих объектам, концепциям или событиям, связанных дугами, которые описывают отношения между узлами [7,8]. Дуги могут быть определены разными методами в зависимости от вида представляемых знаний, например, дуги, используемые для представления иерархии, включают дуги типа *isa* (является) и *has-part* (имеет часть). Элементы более низкого уровня в сети могут наследовать свойства элементов более высокого уровня, что позволяет экономить память.

В общем виде отметим, что в основе СМ лежит семантическая сеть вида

$$< I, C_1, C_2,..., C_n, \Gamma >,$$

где I - множество информационных единиц;

$C_1, C_2, ..., C_n$ - множество типов связей между информационными единицами;

Γ - отображение, которое задает связи между информационными единицами из I из заданного набора типов связей.

Отношения между вершинами сети несут смысловую нагрузку. Если вершины сети не имеют собственной внутренней структуры, то такие сети называются простыми сетями. Если вершины обладают некоторой структурой, то такие сети называют иерархическими. Сегодня в основном используются иерархические сети. Одно из основных отличий иерархических семантических сетей от простых семантических сетей состоит в возможности разделить сеть на подсети (пространства) и устанавливать отношения не только между вершинами, но и между подсетями. Все вершины и дуги являются элементами, по крайней мере, одного пространства.

Использование СМ дает наилучший эффект для представления знаний о предметной области с хорошо установленной таксономией с целью упрощения поиска решения задачи.

Первые задачи, которые были решены с помощью сетевых моделей, относятся к задачам классификации, проблемам наследования в психологии. Однако лучше всего они зарекомендовали себя для анализа и синтеза грамматических конструкций естественного языка, в т.ч. для машинного перевода.

Типовая грамматическая структура имеет центром действие (ГЛАГОЛ)! (в индо-европейских языках). Рассмотрим следующее предложение.

СОБАКА ЖАДНО ЕСТ КОСТЬ :

ЕСТЬ:

СОБАКА (агент)

КОСТЬ (объект)

ЖАДНОСТЬ

Утверждения, ссылающиеся на ситуации, в т.ч. с помощью точек зрения, будут называться *референциальными (ЗНАЕТ, СКАЗАЛ, МОЖЕТ, НРАВИТСЯ…)*

По типам связей сети бывают:

- Классифицирующие (отношения структуризации)
- Функциональные (функциональные отношения – описывающие вычисление одних отношений через другие)
- Сценарии (каузальные отношения)

Классифицирующие сети

Классифицирующие сети используют таксономическую иерархию – иерархию абстрактных понятий, имеющих структуру дерева, корень является наиболее общим понятием (например, биологическая классификация Карла Линнея 1735 г.)

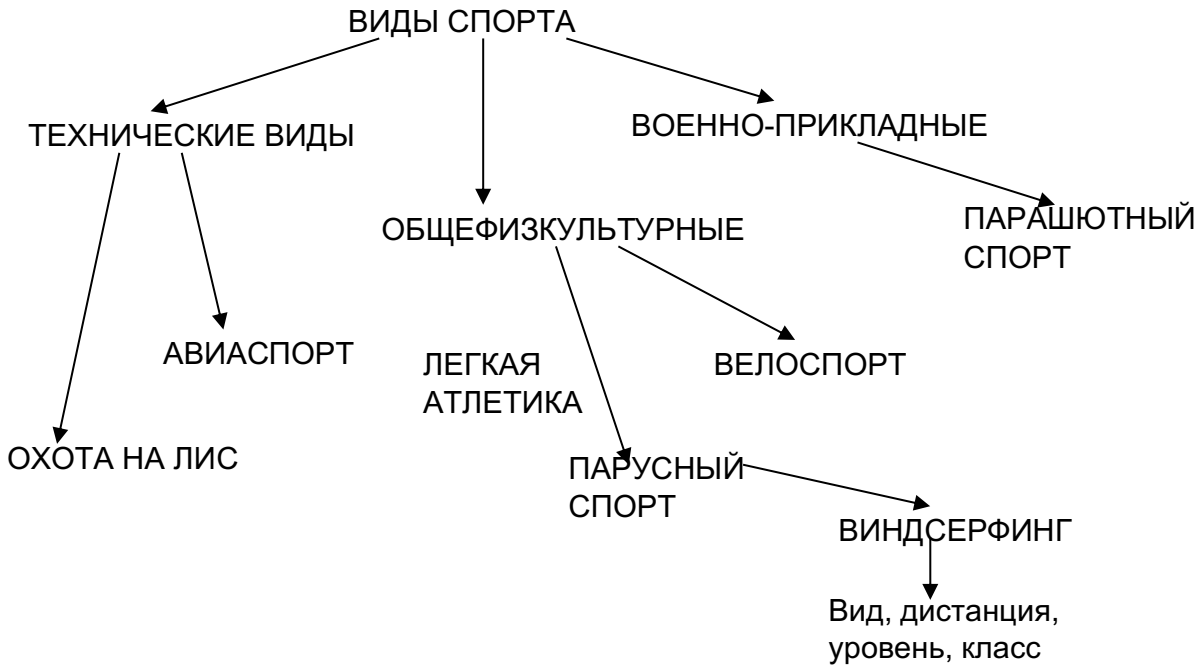


Рис.1 Пример классифицирующей сети

Рассмотрим примеры отношений в классифицирующей сети.

ISA (is a) – отношение классификации **(Member of),** например, *ВСЕ ЗАЙЦЫ СУТЬ МЛЕКОПИТАЮЩИЕ. МУРКА is a КОШКА.* Свойства объекта наследуются от множества.

AKO (a kind of) - разновидность – отношение между множеством и подмножеством. Элемент подмножества – *гипоним,* отношение называется отношением *гипонимии.*

Отношение определяет, что каждый элемент первого множества входит и во второе, а также логическую связь между самими подмножествами: что первое не больше второго и свойства первого множества наследуются вторым

HasPart - отношение части объекта (отношение *меронимии). Мероним* – объект, являющийся частью другого. *Холоним* - объект, включающий в себя другое. Например, КОМПЬЮТЕР – *холоним* для монитора.

Задание для самостоятельного выполнения:

Составить классифицирующую сеть по вариантам задания для определения принадлежности индивида к определенному классу или виду по вариантам задания.

1. Классификация грибов
2. Классификация членистоногих
3. Классификация грызунов
4. Классификация сумчатых
5. Классификация пресмыкающихся
6. Классификация цветковых растений
7. Классификация протистов (простейшие микроорганизмы)
8. Классификация отряда хищников

Если в графе допускаются различные отношения, то это семантическая сеть

Построим семантическую сеть для следующего предложения:

Волга впадает в Каспийское море, но Алексей считает, что это не так. он считает, что Волга впадает в Черное море, что неверно.

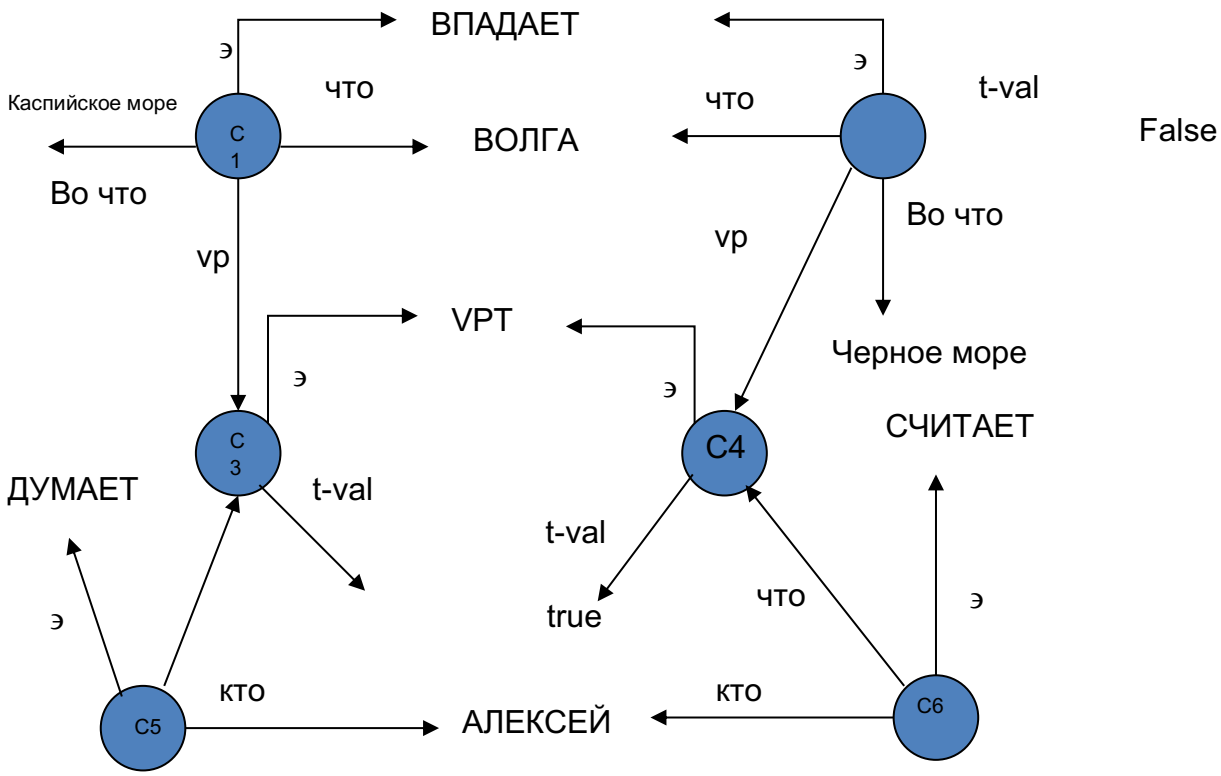


Рис.1 Семантическая сеть предложения

Утверждения точек зрения имеют тип *vpt*

t-val – абсолютно истинный статус (точка зрения природы на состояние модели).

Для того, чтобы семантическая сеть была способна анализировать и производить новые знания, ее необходимо дополнить новыми смысловыми единицами, понятными любому грамотному человеку, но не ясными для семантической сети, например, дать понятия «Волга», «Черное море», «Каспийское море».

Задание для самостоятельного выполнения:

Составить семантическую сеть для высказываний, приведенной в вариантах задания.

1. определить объекты (вершины сети)
2. свойства объектов
3. связи (типы связей, значения)
4. построить графическую модель семантической сети

Варианты заданий:

1. «Олег явился домой около часа ночи. Он терпеть не может музыку, отчаянно скучает на любом спектакле, а в кино, пользуясь темнотой, засыпает. Он трудоголик, целый день и ночь проводящий на службе… Мне *(Виоле)* хотелось романтики, совместных ужинов, кофе в постель, милых подарочков, походов, ладно, не в театр, так хоть в супермаркет за продуктами или в магазин за тряпками»

Д. Донцова «Главбух и полцарства в придачу»

2. Утреннюю гимнастику я не делаю, честно говоря, обожаю поспать часиков до 11, если бы не Лиза и Кирюша, которым нужно попасть в школу к первому уроку, то дрыхла бы до обеда. Ледяная вода вызывает у меня дрожь. Я ни за что не полезу в море при температуре воды меньше 25 градусов по Цельсию. Я отвратительно безграмотна во всем, что связано с математикой, физикой, химией, географией. Бег с сумками наперевес по лестнице тоже не является моим хобби.

Д.Донцова «Обед у людоеда»

3. Меня зовут Иван. К простому мужицкому имени прилагается звучная фамилия Подушкин. Род мой известен издавна. Бояре Подушкины были одними из тех, кто возводил на трон Михаила Романова. Поколения Подушкиных верно служили царю и отечеству, больших чинов не имели, но пользовались уважением и были стабильно богаты. В 1917 году весь род сгинул в пучине революции. Чудом выжил только мой отец, которому еще не исполнилось и 3 года.

Д.Донцова «Букет прекрасных дам»

4. Живем мы в собственном двухэтажном доме возле кольцевой дороги. Вместе с нами обитают 5 собак: питбуль Банди, ротвейлер Снап, английский мопс Хуч, йоркширский терьер Жюли и пудель Черри. Начиналась коллекция псов с пита и ротвейлера. Их купили для охраны. Но, увы, ничего не вышло. И Банди, и Снап - самозабвенные обжоры. Пасти страшных охранников вечно заняты какой-нибудь вкуснятиной. Поэтому любой, предлагающий лакомство, автоматически становится другом, а едят они все: сыр, творог, яйца, печенье, суп, оладьи, чипсы, соленые огурцы и орехи.

Д.Донцова «Эта горькая сладкая месть»

5. Надя говорила, что Богдан раз в месяц обязательно ездит в командировки, закупает медицинскую аппаратуру, повышает свой профессиональный уровень на семинарах, часто зарубежных. Ему приходило много приглашений. Надя никогда с ним не летала, панически боялась самолетов, сесть в железную птицу для нее было равно огромному стрессу, вот милейший Богдан и мотался один.

Д.Донцова «Хождение по мухой»

6. Когда графиня бывала в настроении и выходила к столу, кормили изысканно – блюдами, доставленными из французского ресторана «Эртель». Если же Адди с утра хандрила у себя в будуаре или отправлялась развеяться по галантерейным и парфюмерным магазинам, власть в столовой захватывал Маса, и тут выходил совсем иной коленкор. Из японско-китайской лавки, что на Петровских линиях, фандоринский камердинер приносил пресного

рису, маринованной редьки, хрустящих, похожих на бумагу, водорослей и сладкой жареной рыбы. Надворный советник поедал всю эту отраву с видимым удовольствием, Анисию же Маса выдавал свежий бублик и колбасу.

Б.Акунин «Пиковый валет»

7. Андрей Семенович Лебезятников равлекал прогрессивным разговором своего временного жильца, того самого Петра Петровича Лужина. Приехав в Петербург и пока еще не обставив собственного гнездышка, Петр Петрович из видов экономии поселился у своего младшего товарища и подопечного, при котором в не столь давние времена состоял опекуном и поэтому чувствовал себя вправе обременить.

Лебезятников (худосочный и золотушный человечек малого роста, где-то служивший и до странности белокурый, с бакенбардами, которыми он очень гордился) с азартом пересказывал Петру Петровичу одну из самых новых теорий общественного устройства…

Б.Акунин «ФМ».

8. Я, нижеподписавшийся, подпоручик Федор Михайлович Достоевский, заключил настоящее Условие с купцом Федором Тимофеевичем Стелловским в том, что Достоевский собственное свое сочинение под заглавием «Теорийка» продал ему, Стелловскому, за аванс в 175 рикcталлеров с последующей по издании уплатою в 7000 рублей серебром на вечное время с переходом прав издания и всех прочих прав к наследникам Стелловского.

Б.Акунин «ФМ».

9. Во главе стола сидела Лика, на ней было белое платье и белая фата, прикрепленная к голове с помощью венка из искусственных цветов. Новым супругом был худощавый дядька с красным лицом и шеей. То ли у молодого были проблемы с давлением, то ли ему просто душно в костюме и рубашке с галстуком.

Д. Донцова «Уха из золотой рыбки».

Практическая работа №5-7.

Разработка систем продукций ИС

В выбранной предметной области разработать модель знаний в виде системы продукций.

Форма отчета: модель знаний в виде системы продукций. Возможно выполнение работы в группе студентов.

Продукционные модели

Продукционные модели (ПМ) представляют собой комбинацию элементов логических и сетевых моделей. Из логических моделей взята идея правил вывода, которые здесь называют продукциями, а из сетевых моделей – описание знаний в виде семантической сети. В результате применения правил вывода к фрагментам сетевого описания происходит трансформация семантической сети за счет замены ее фрагментов, наращивания сети и исключения из нее ненужных фрагментов. В продукциях отсутствуют жесткие ограничения, характерные для логических исчислений, что дает возможность изменять интерпретацию элементов продукции.

Продукционные системы базируются на понятии «формальная продукционная система» и берут свое начало с работ Е. Поста, который в 1943 году ввел термины «продукция и каноническая (продукционная) система». Пост доказал, что продукционная система является логической системой, эквивалентной машине Тьюринга. Продукционные системы универсальны, то есть любая формальная система, оперирующая символами, может быть реализована в виде одной из продукционных систем Поста.

Психологические исследования процессов принятия решений человеком показали, что при рассуждении человек использует правила, аналогичные продукциям – правила «условие -> действие», то есть система продукций по своей сути аналогична человеческому мышлению.

Введем следующие определения. Продукционная база знаний (ПБЗ) определяется совокупностью:

$$P=(F, R, G, C, I), \qquad (2)$$

где F – множество фактов о решаемой проблеме (считаем, что число фактов конечно). Каждый факт может быть установленным или неустановленным, совокупность установленных фактов задает некоторую ситуацию в предметной области.

R – множество продукций или правил, включающее правила вида

$$r_m: ЕСЛИ\ f_i\ И\ f_j\ ...\ И\ f_n\ \ ТО\ f_k, \qquad (3)$$

где r_m – имя правила, $r_m \in R$; $f_i, f_j ... f_n$ – условия выполнения правила; f_k –следствие правила, $f_i, f_j, ..., f_n, f_k \in F$; G – множество целей или терминальных фактов ЭС; I – интерпретатор правил, реализующий процесс вывода.

Функционирование интерпретатора I состоит в проверке истинности некоторых целей в заданной для проблемной области ситуации. При этом логический вывод может осуществляться интерпретатором в прямом или обратном направлении.

Пример продукции:

НАЛИЧИЕ ДЕНЕГ; ЕСЛИ ХОЧЕШЬ КУПИТЬ ВЕЩЬ X, ТО ЗАПЛАТИ В КАССУ ЕЕ СТОИМОСТЬ

Постусловие: в БД наличия товаров уменьши на 1 число товаров данного наименования

Ядро продукции составляет $ЕСЛИ\ f_i\ И\ f_j\ ...\ И\ f_n\ \ ТО\ \ f_k$

Ядра бывают:

- детерминированные, когда мы абсолютно уверены, что из А следует В (ЕСЛИ А ТО В);
- недетерминированные, когда этой уверенности не существует, например, ЕСЛИ А ТО ВОЗМОЖНО В. Недетерминированность может присутствовать как в условии продукции, так и в следствии.

Недетерминированность может быть:

- *статистической,* когда можно применить методы математической статистики для оценки неопределенности, например, ЕСЛИ А ТО С ВЕРОЯТНОСТЬЮ p РЕАЛИЗОВАТЬ В. Отметим, что это возможно достаточно сложно реализовать, так как системы, основанные на знаниях, разрабатываются тогда, когда формальные методы трудно применимы. Если в предметной области можно применить математическую статистику, то, скорее всего, это даст более точный результат, чем интеллектуальные методы. Обычно в базах знаний недетерминированность задается при помощи экспертной оценки;
- *лингвистической,* когда для оценки неопределенности используются лингвистические переменные, например, *ЕСЛИ А ТО ЧАЩЕ ВСЕГО НУЖНО РЕАЛИЗОВАТЬ В1 РЕЖЕ В2.*

Большинство человеческих рассуждений могут быть сведены к системе продукций. Люди в своих рассуждениях применяют аналогичные продукциям правила вида «посылка –> заключение», например, ситуация –> действие, причина –> следствие, состояние –> принимаемое решение и пр.

Например, рассмотрим правила некоторых известных медицинских экспертных систем, основанных на продукциях:

1. ЭС VM, управляющая аппаратом искусственного дыхания и интепретирующая данные о пациенте, поступающие от системы физиологического мониторинга. ЭС следит за состоянием больного, которому необходимо применение аппарата искусственного дыхания в послеоперационный период.

Пример правил ЭС VM:

ЕСЛИ текущий контекст системы – «Оказание помощи», и частота дыхания носила устойчивый характер в течение 20 минут, и отношение I/EW было устойчивым в течение 20 минут, ТО больной находится на принудительной контролируемой вентиляции легких (ПКВЛ).

ЕСЛИ больной перешел от «Оказание помощи» к «Применение воздуховода» ИЛИ больной перешел от режима ПКВЛ к «Применение воздуховода», ТО следует ожидать следующих показателей: систолическое давление 110, диастолическое давление 60, частота пульса 60.

2. ЭС DENDRAL, интерпретирует данные и строит полную структуру молекулы на основании измерений, относящихся к частям молекулы. Однако, ввиду сложности

использования ЭС для учебных целей, рассмотрим примеры продукций более простой экспертной системы GA1.

ЕСЛИ генерируются кольцевые структуры молекулы, ТО в качестве начального элемента в списке исходных сегментов следует использовать только наименьший сегмент.

3. ЭС R1 предназначена для анализа конфигурации вычислительной системы VAX известной в свое время фирмы DEC. На вход системы поступает заказ покупателя, а на выходе выдается комплект чертежей, отражающих пространственные отношения между отдельными компонентами, входящими в заказ.

Пример продукции:

ЕСЛИ последним активным контекстом предписывается использование источника энергии и адаптер шины UNIBUS помещен в стойку, и его положение в стойке известно, и в стойке имеется место для источника питания, и отсутствует свободный регулятор H7101, ТО внести регулятор H7101 в заказ покупателя.

Системы продукций (СП) очень широко распространены для построения БЗ ЭС. Это определяется рядом преимуществ, которые продукционные экспертные системы (ПЭС) имеют перед другими способами представления знаний:

1) подавляющая часть человеческих знаний может быть представлена в виде систем продукций;

2) продукции описывают разнообразные знания простыми структурами с высокой степенью стандартизации;

3) продукционные системы в высокой степени удовлетворяют принципу модульности. Любая продукция при программной реализации может рассматриваться как независимый модуль, модификация, добавление которого в продукционную систему и изъятие его из нее происходят достаточно просто и не влияет на остальное содержимое базы знаний;

4) наличие в продукциях указателей на сферу применения продукций позволяет эффективно организовать память, сократить время поиска в ней информации;

5) продукционные системы позволяют легко организовать параллельные процессы, в которых все продукции, входящие во фронт готовых, могут выполняться независимо друг от друга, что позволяет реализовывать системы искусственного интеллекта с базой знаний, состоящей из сотен тысяч продукций, за счет распределенных, например, grid-вычислений.

В процедуры управления легко включаются необходимые дополнительные условия на выбор альтернативных продукций из фронта готовых для выполнения, что не затрагивает основных процедур параллельного выполнения продукций. Асинхронность реализации продукций делает СП удобной моделью вычислений для ЭВМ новой архитектуры, где идея параллельности и асинхронности является центральной.

Достаточно часто в настоящее время также встречаются ЭС, база знаний которых основана на комбинации нескольких типов предоставления знаний, например, фреймов и продукций, семантической сети и продукций. Часто продукции используются для управления процессом вывода.

Продукции не лишены и недостатков:

- при большом числе продукций становится сложной проверка на непротиворечивость;
- при числе продукций более 1000 сложности с корректностью системы;
- продукционным системам не хватает строгой теории, много эвристик, не всегда можно быть уверенным в полноте и непротиворечивости;
- современные тенденции часто требуют создания динамических продукций (меняющихся в процессе принятия решения), следовательно, нужны эффективные стратегии управления.

Рассмотрим основные стратегии управления системой продукций:

- *принцип «стопки книг».* Вспомните, какие книги лежат у вас на столе. Обычно это самые читаемые книги. Аналогично будем считать, что чаще используемые продукции являются самыми полезными. Самая «верхняя в стопке» продукция – та, которая использовалась чаще всех. При активизации фронта продукций выбирается та, у которой частота использования максимальна;
- *принцип наиболее длинного условия выполнения ядра*. Этот принцип основан на том, что частные правила, относящиеся к узкому классу ситуаций, важнее общих правил. Трудность использования данной стратегии заключается в том, что нужно заранее упорядочить условия по вхождению друг в друга по отношению «частное-общее»;
- *принцип метапродукций*. Основан на вводе в базу знаний специальных метапродукций, задачей которых является организация управления в системе продукций при возможности неоднозначного выбора из фронта готовых продукций. При вводе слишком большого числа метапродукций база знаний может превратиться в традиционную программу с большим числом условных операторов и операторов перехода на метку.

Рассмотрим пример метапродукций классической ЭС MYCIN [18]: ЕСЛИ инфекция есть pelvic-abcess и имеются продукции, входящие в состав фронта продукций, в условии которых упоминается grampos-rods ТО продукции, у которых в А имеется enterobacteriaceae, следует активизировать раньше, чем продукции, содержащие в А gramposrods;

- *принцип «классной доски».* Основан на идее спусковых функций [17]. При использовании данного принципа в интеллектуальной системе выделяется рабочая память для обмена информацией между продукциями, аналог классной доски. Здесь параллельно выполняющиеся процессы находят информацию, инициирующую запуск продукций, здесь же хранится информация о их работе, которая может быть полезной для других процессов;
- *принцип приоритетного выбора.* Основан на установке приоритетов (степени важности) для различных продукций;
- *управление по именам.* Реализуется определением имен продукций, некоторой формальной грамматики или другой процедуры, обеспечивающей сужение фронта готовых продукций и выбор из него очередной продукции для выполнения.

Практическая работа №8.

Разработка гибридных моделей знаний ИС

В выбранной предметной области разработать модель знаний, сочетающую фреймовую (табличную) форму и систему продукций.

Форма отчета: гибридная модель знаний, сочетающая фреймовую (табличную) форму и систему продукций.

Практическая работа №9-12

Построение базы знаний интеллектуальной системы

В выбранной предметной области разработать базу знаний, сочетающую фреймовую (табличную) форму, графовую модель, и систему продукций.

Форма отчета: гибридная модель знаний, сочетающая фреймовую (табличную) форму, графовую модель и систему продукций.

Практическая работа №13-14.

Программирование на языке Пролог. Основные принципы

Анализ предметной области. Выявление закономерностей. Составление схемы взаимодействия объектов. Составление простейшей программы на Прологе. Структура программы. Прямая и обратная цепочка рассуждений. Включение цели в структуру программы.

Разработать простейшую программу на языке Пролог. Для реализации использовать любую бесплатную версию языка Пролог, например, TURBO PROLOG, SWI PROLOG.

В задании должны быть рассмотрены два варианта блока goal: вне текста программы и внутри текста программы.

Форма отчета: текст разработанной программы. Готовность объяснить работу программы.

Варианты заданий: Пример:

1. Известны следующие факты:

Том любит Лиз

Джон любит Глорию

Памела любит Майкла

Лиз любит Тома

Лиз любит Джека

Роберт любит Глорию

Майкл любит Памелу

Глория любит Джона

брак считается возможным, если любовь взаимная и только к этому человеку.

Составить программу, отвечающую на вопросы:

1) Кого любит Майкл?
2) Подобрать все возможные сочетания брачных пар.

Текст программы:

```
predicates
        love (symbol, symbol)
        para (symbol, symbol)
        nepara(symbol,symbol)
clauses
        love(tom,liz).
        love(john,glory).
        love(pamela,michael).
        love(liz,tom).
        love(liz,jack).
        love(robert,glory).
        love(michael,pamela).
        love(glory,john).
        nepara(X,Y):-love(X,A),A<>Y;love(Y,B),B<>X.
        para(X,Y):-love(X,Y),love(Y,X),not(nepara(X,Y)).
goal
        love(michael,Z), write("michael loves ",Z), nl, nl,
        para(X,Y), write(X, "  loves  ",Y), nl, fail.
```

Варианты заданий:

1. Известна система родственных связей с отношением РОДИТЕЛЬ. Имена в ней не повторяются. Разработать программу логического вывода, позволяющую определить, кто кому приходится отцом, матерью, братом, сестрой, дядей, тетей, дедушкой, бабушкой.
2. Известны следующие факты:

Преподаватель и учитель имеют высшее образование. Преподаватель ВУЗа имеет высшее образование и аспирантуру. Также известно, что

Джон работает дворником.

Том работает учителем.

Глория – продавец.

Майкл – преподаватель ВУЗа.

Билл – инженер.

Мери – учитель.

Лиз – инженер.

Составить программу, отвечающую на следующие вопросы:

Имеет ли Майкл высшее образование?

Закончил ли Том аспирантуру?

Кто имеет высшее образование?

Кем работает Лиз?

Может ли инженер быть преподавателем ВУЗа?

3. На основании известной таксономической системы (а) составить программу, отвечающую на вопросы:

Кто кормит детей молоком?

Является ли собака позвоночным животным?

Плотоядная ли кошка?

Какие семейства относятся к млекопитающим?

Кормят ли молоком животные, имеющие острые зубы?

Относятся ли копытные к позвоночным?

Все ли кто лает, имеют острые зубы?

Имеет ли позвоночник корова?

Таксономическая система (а):

Каждое млекопитающее – позвоночное

Каждое плотоядное – млекопитающее

Корова мычит

Каждое копытное – млекопитающее

Позвоночное имеет позвоночник

Семейство псовых и кошачьих относятся к плотоядным

Кошка мурлычет

Собака лает

Копытное имеет плоские зубы

Кошка относится к семейству кошачьих

Собака относится к семейству псовых

Млекопитающее кормит детей молоком

Плотоядное имеет острые зубы

Корова относится к копытным

4. Составить программу, осуществляющую подбор партнера и партнерши в бальных танцах на основании следующих правил:

Рост мальчика >роста девочки, но не более, чем на 10 см

Танцевальный класс мальчика выше или равен классу девочки

Известно, есть партнеры:

Боб (рост 175, класс «С»)

Энтони (рост 180, класс «В»)

Поль (рост 160, класс «Д»)

Ник (рост 160, класс «Е»)

Фред (рост 176, класс «А»)

Партнерши:

Мэри (рост 172, класс «В»)

Энн (рост 170, класс «С»)

Луиза (рост 165, класс «Д»)

Натали (рост 168, класс «С»)

Клаудиа (рост 158, класс «А»)

Практическая работа №15-16.

Программирование на языке Пролог. Рекурсия.

Письменный отчет по работе кроме текста задания и исходного текста программы содержит пошаговое описание реализуемой рекурсии для 5 шагов.

Число часов на выполнение работы 6

Варианты заданий:

1. $y = a^n + a^{n-1} + \ldots \quad a^2 + a + 1$, *a* и *n вводятся с клавиатуры*

2. $y = \sqrt{10 + \sqrt{10 + \sqrt{10 + \ldots \quad \sqrt{10}}}}$, *n* –раз

3. $y = na^{n-1} + (n-1)a^{n-2} + \ldots \quad 2a + 1$

4. $y = x^2 + x^4 + x^6 + \ldots$

5. $y=sin(x)+sin(sin(x))+ \ldots + sin(sin(\ldots(sin(x)))$

6. $\frac{\sin(1)}{\cos(1)} + \frac{\sin(1)\sin(2)}{\cos(1)\cos(2)} + \ldots + \frac{\sin(1)\sin(2)\cdot\sin(n)}{\cos(1)\cos(2)\cdot\cos(n)}$

7. $\cos(x) + \cos(x^2) + \ldots \quad \cos(x^n)$

Практическая работа №17

Программирование на языке Пролог. Решение задач на списки

В работе нужно написать программу на языке ПРОЛОГ на тему СПИСКИ. Список представляется в виде головы и хвоста [H|T], где H- голова (обычно первый элемент), а T – хвост (всегда список).

Варианты заданий:

Дан список из 10 элементов. Упорядочить его в порядке возрастания или убывания.

1. Дан список из 10 элементов. Получить список, циклически сдвинутый на один элемент влево, например, исходный список -[1,2,3,4,5], результат- [2,3,4,5,1].
2. Перевести список чисел от 1 до 12 в соответствующие названия месяцев.
3. Дан список из 10 элементов. Найти максимальный элемент списка.
4. Дан список из 10 элементов. Выделить все одинаковые элементы списка в отдельный список.
5. Дан список из произвольного числа элементов. Установить является ли количество элементов четным или нет.
6. Дан список из 5 элементов. Установить, является ли список палиндромом: т.е. читается ли он одинаково, как справа налево, так и слева направо, например, [a,b,c,b,a].
7. Дан список из 10 элементов. Подсчитать количество положительных и отрицательных элементов в списке.
8. Дан список из 10 элементов. Найти сумму элементов в списке.
9. Дан список из 10 элементов. Найти количество элементов в списке, принадлежащих заданному диапазону.

Форма отчета: текст разработанной программы. Готовность объяснить работу программы.

Практическая работа №18-19

Программирование на языке Пролог. Реализация баз данных и баз знаний. Типовые задачи ИИ

Изучаются типовые задачи логического программирования: Ханойская башня, задача о 8 ферзях.

В работе нужно написать программу, реализующую базы данных и базы знаний на языке ПРОЛОГ.

1. Варианты заданий на создание БД:

- Создать БД произведений всех чисел от 1 до 9 в виде число1, число2, произведение. Вывести на печать содержимое БД
- Создать БД, содержащую сведения о сотрудниках предприятия:

Фамилия, Имя, Оклад, кол-во детей, подоходный налог,

Где подоходный налог= 12%(Оклад-1%Оклад-(кол-во детей+2)*необл.минимум)

Необл.минимум (вещественное число) вводится с клавиатуры. Распечатать всех, у кого подоходный налог > определенной величины

- Создать БД, содержащую сведения о книгах в библиотеке:

Раздел, Автор, Название, Год выпуска, Цена

Распечатать все книги, относящиеся к определенному разделу, автору

- Создать БД, содержащую ключевые слова, текстовые варианты вопросов и текстовые варианты ответов на предложения, содержащие данные ключевые слова. Составить программу, осуществляющую диалог с пользователем.
- Создать БД, содержащую сведения о студентах и их успеваемости:

Фамилия, Имя, Группа, Оценка1, Оценка2, Оценка3, Оценка4.

Распечатать всех фамилии и имена студентов, имеющих средний балл, превышающий заданный.

Реализовать предикат Search(X,P,L), порождающий список L всех объектов X, удовлетворяющих цели P. При решении использовать предикаты assert, retract и call(P) (вызов цели).

2. Разработка баз знаний

Варианты заданий на создание баз знаний взять из практической работы №2, в которой разрабатывались модели знаний

Во всех экспертных системах существует зависимость между входным потоком данных и данными в базе знаний. Во время консультации входные данные сопоставляются с данными в базе знаний. Результатом сопоставления является отрицательный или утвердительный ответ.

В системе, базирующейся на правилах, утвердительный результат является действием одного из продукционных правил. Эти продукционные правила определяются входными данными.

Экспертная система, основанная на правилах, содержит множество правил. В реализации на Прологе предикаты в левой части правил определяют один из возможных вариантов решения задачи, предикаты в правой части всегда специфицируются другими правилами, помимо тех случаев, когда предикат просто проверяет, находится ли определенная информация в базе данных. Информация, помещаемая в базу данных, извлекается из ответов пользователя на задаваемые вопросы. Все ответы сохраняются, так как они могут понадобиться позднее.

Экспертная система также содержит интерпретатор в механизме вывода. Работу этого интерпретатора можно описать последовательностью трех шагов: интерпретатор сопоставляет образец правила с элементами данных в базе знаний; если можно вызвать более одного правила, то интерпретатор использует механизм разрешения конфликта для выбора правила; интерпретатор применяет выбранное правило, чтобы найти ответ на

вопрос. Этот процесс интерпретации является циклическим и называется циклом "распознавание-действие".

Рассмотрим в качестве примера экспертной системы систему для идентификации пород собак. Она помогает потенциальному хозяину выбрать породу собаки в соответствии с определенными критериями.

Предположим, что пользователь сообщил множество характеристик собаки в ответ на вопросы экспертной системы. Интерпретатор работает в цикле распознавание-действие. Если характеристики, заданные пользователем, сопоставимы с характеристиками породы собаки, составляющими часть базы знаний, тогда вызывается соответствующее продукционное правило и в результате идентифицируется порода. Затем результат сообщается пользователю. Если порода не идентифицирована, это тоже сообщается пользователю.

Рассмотрим две породы собак, информация о которых содержится в базе знаний. Гончая имеет короткую шерсть, высоту в холке меньше 57 см длинные уши и хороший характер. Датский дог имеет короткую шерсть, низко посаженный хвост, длинные уши, хороший характер и вес более 45 кг.

Из этого описания видно, что обе породы имеют короткую шерсть, длинные уши и хороший характер. Рост гончей меньше 57 см в то время, как ничего не сказано о росте дога. Дог имеет низко посаженный хвост и вес более 45 кг – характеристики отсутствующие для гончей. Описание двух собак в терминах указанных характеристик достаточно, чтобы различить эти две породы, и даже отличить их от любой другой породы в базе знаний.

Введем предикат *positive,* которые описывает свойства, имеющиеся у собаки.

Следующие продукционные правила могут быть составлены по указанным характеристикам:

dog_is("Гончая"):– it_is("короткоерстная собака"),

positive("ее","высота в холке не более 57 см"),

positive("у нее","длинные уши"),

positive("у нее","дружелюбный характер"), !.

dog_is("Great Dane"):– it_is("длинношерстная собака"),

positive("у нее","низко посаженный хвост"),

positive("у нее","длинные уши"),

positive("у нее","дружелюбный характер"),

positive("ее","вес более 45 кг"), !.

Заметим, что в правилах длина шерсти может быть представлена с помощью предиката *positive* в виде:

positive("у нее","короткая шерсть").

Но использование предиката it_is позволяет ограничить "пространство поиска" (количество данных, проверяемых при поиске решения) одним поддеревом древовидной структуры, содержащей информацию о разных породах собак.

Экспертная система, базирующаяся на правилах, позволяет проектировщику строить правила, которые естественным образом объединяют в группы связанные фрагменты знаний. Каждое продукционное правило может быть независимым от других. Эта независимость делает базу продукционных правил семантически модульной, т.е. группы информации не влияют друг на друга. Более того, модульность базы правил позволяет развивать базу знаний, увеличивая ее.

Текст программы

```
/* Программа: эксперт по породам собак                    */

/* Это продукционная система, базирующаяся на правилах*/

database
  xpositive(symbol,symbol)
  xnegative(symbol,symbol)
predicates
  do_expert_job
  do_consulting
  ask(symbol,symbol)
  dog_is(symbol)
  it_is(symbol)
  positive(symbol,symbol)
  negative(symbol,symbol)
  remember(symbol,symbol,symbol)
  clear_facts
goal
  do_expert_job .
clauses
                /* Систесма пользовательского интерфейса */
  do_expert_job :– makewindow(1, 7, 7, "ЭКСПЕРТ ПО  ПОРОДАМ СОБАК, 1, 16, 22, 58),
       nl,write(" * * * * * * * * * * * * * * * * * * * * *  "),
       nl,write("          ДОБРО ПОЖАЛОВАТЬ!          "),nl,nl,
       nl,write(" Проводится идентификация породы "),nl,nl,
       nl,write(" Отвечайте, пожалуйста, 'да' или 'нет' "),
       nl,write(" а вопросы о собаке, породу которой "),
       nl,write(" Вы хотите определить "),
```

```
    nl,write(" * * * * * * * * * * * * * * * * * * * * * * * * * "),
    nl,nl,do_consulting,write("Нажмите любую клавишу"),
    nl, readchar(_), removewindow.
do_consulting :– dog_is(X), !, nl, write("Вероятно Ваша собака – ",X,"."),
                nl,clear_facts.
do_consulting :– nl, write("Извините, я не смогу помочь Вам!"),
                clear_facts.
ask(X,Y) :– write(" ?:– ",X, "  ",Y, " ?  "), readln(Reply),
            remember(X,Y,Reply).
```

/* Механизм вывода */

```
positive(X,Y) :– xpositive(X,Y),!.
positive(X,Y) :– not(negative(X,Y)),!,ask(X,Y).
negative(X,Y) :– xnegative(X,Y),!.
remember(X,Y,yes) :– asserta(xpositive(X,Y)).
remember(X,Y,no) :– asserta(xnegative(X,Y)),fail.

clear_facts :– retract(xpositive(_,_)), fail.
clear_facts :– retract(xnegative(_,_)), fail.
```

/* Продукционные правила */

```
dog_is("Английский Бульдог") :– it_is("короткошерстная собака"),
                positive("ее","высота в холке не более  57 см"),
                positive("у нее","низко посаженный хвост"),
                positive("у нее","дружелюбный характер"),!.
dog_is("Гончая") :– it_is("короткошерстная собака"),
                positive("ее","высота в холке не более  57 см"),
                positive("у нее","длинные уши"),
                positive("у нее","дружелюбный характер"),!.
dog_is("Немецкий Дог") :– it_is("короткошерстная собака"),
                positive("у нее","низко посаженный хвост"),
                positive("у нее","дружелюбный характер"),
                positive("ее","вес более 45 кг"),!.
dog_is("Американский Фоксхаунд") :– it_is("короткошерстная собака"),
```

```
                positive("ее","высота в холке не более  77 см"),
                positive("у нее","длинные уши"),
                positive("у нее","дружелюбный характер"),!.
dog_is("Кокер Спаниель") :– it_is("длинношерстная собака"),
                positive("ее","высота в холке не более  57 см"),
                positive("у нее","низко посаженный хвост"),
                positive("у нее","длинные уши"),
                positive("у нее","дружелюбный характер"),!.
dog_is("Ирландский  Сеттер") :– it_is("длинношерстная собака"),
                positive("ее","высота в холке не более  77 см"),
                positive("у нее","длинные уши"),!.
dog_is("Колли") :– it_is("длинношерстная собака"),
                positive("ее","высота в холке не более  77 см"),
                positive("у нее","низко посаженный хвост"),
                positive("у нее","дружелюбный характер"),!.
dog_is("Сенбернар") :– it_is("длинношерстная собака"),
                positive("у нее","низко посаженный хвост"),
                positive("у нее","дружелюбный характер"),
                positive("ее","вес более 45 кг"),!.
it_is("короткошерстная собака ") :–
                positive("это","короткошерстная собака "),!.
it_is("длинношерстная собака ") :–
                positive("это","длинношерстная собака "),!.
```

Форма отчета: текст разработанной программы. Готовность объяснить работу программы.

Практическая работа №20-22

Построение нечетких и неточных моделей знаний

В данной работе изучаются методы неточного и нечеткого вывода на знаниях

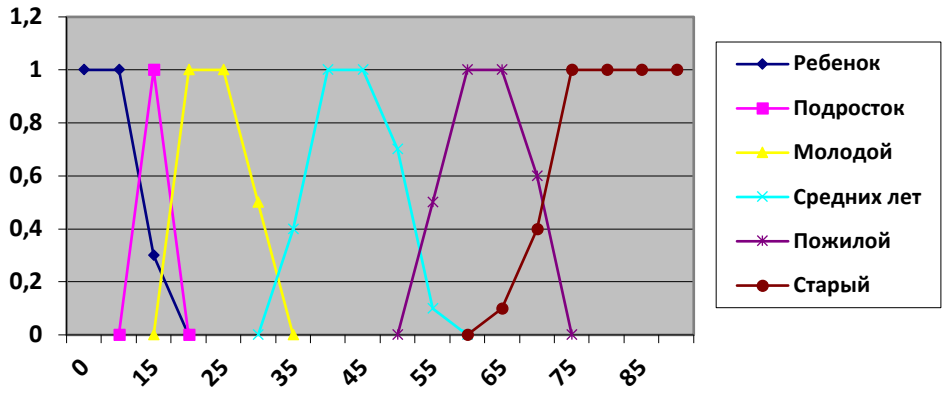
Задание 1:

Разработать типовую систему принятия решения, основанную на нечетком логическом выводе.

Отчет о работе должен содержать

1. Описание входных (не менее 3) и выходных (не менее 2) лингвистических переменных, включающее
   a. Названия переменных
   b. Лингвистические значения переменных
   c. Функции принадлежности нечетких переменных, соответствующих лингвистическим значениям (в графическом и аналитическом виде)

   Пример описания переменной:
   v_1: Возраст человека
   Лингвистические значения: «Ребенок», «Подросток», «Молодой», «Средних лет», «Пожилой», «Старый»
   Нечеткие переменные:



2. Базу нечетких правил (не менее 10 правил)
   Пример правила:
   Если *«Возраст человека»* есть *«Пожилой»* и *«Артериальное давление»* есть *«Высокое»*, то *«Риск инфаркта миокарда»* есть *«Высокий»*.
3. Пример процесса вывода по методу Мамдани и Ларсена.

Примеры тем для экспертной системы:

- Экспертная система составления рейтингов студентов на основе посещаемости, успеваемости, участия в научной и общественной жизни и иных факторов
- Экспертная система определения рейтингов преподавателя на основе аудиторной и внеаудиторной нагрузки, количества дипломных и конкурсных работ, количества изданных публикаций и иных факторов
- Экспертная система определения риска возникновения ДТП на основе скорости и массы транспортного средства, водительского опыта и иных факторов
- Экспертная система выбора автомобиля на основе стоимости автомобиля, уровня доходов и водительского опыта покупателя, расхода топлива и иных факторов
- Экспертная система выбора мобильного телефона на основе его стоимости, уровня доходов покупателя, производительности и срока существования модели на рынке

Метод Мамдани

В базе знаний используются правила вида

r_j: ЕСЛИ $i_{j,1}$ есть $t_{j,1}$ И … И $i_{j,n-1}$ есть $t_{j,n-1}$ ТО g_j есть t_j, (1)

где r_j – уникальное имя правила,

$i_{j,1},\ldots i_{j,n-1}$ – входные лингвистические переменные,

$t_{j,1},\ldots t_{j,n-1}$ – значения входных лингвистических переменных,

g_j – выходная лингвистическая переменная,

t_j – значение выходной лингвистической переменной.

В ходе нечеткого вывода выполняются следующие шаги:

1. Для каждого четкого значения $i^*_{j,n}$, подаваемого на вход системы, вычисляется значение функции принадлежности $\mu_{t_{j,n}}(i^*_{j,n})$.
2. Для каждого правила r_j вычисляется уровень отсечения

$$\alpha_j = \min_n \mu_{t_{j,n}}(i^*_{j,n}) \quad (2)$$

И находятся усеченные функции принадлежности нечетких множеств, соответствующих лингвистическим значениям выходных переменных

$$\acute{\mu}_{t_j}(x) = \min\left(\alpha_j, \mu_{t_j}(x)\right) \quad (3)$$

3. Для каждой выходной переменной g_j формируется результирующая функция принадлежности

$$\mu_{g_j}(x) = \max_j \acute{\mu}_{t_j}(x) \quad (4)$$

4. Для каждой выходной переменной g_j вычисляется четкое значение

$$g^*_j = \frac{\int x \cdot \mu_{g_j}(x)}{\int \mu_{g_j}(x)} \quad (5)$$

Метод Ларсена

Метод Ларсена использует правила вида (1). В ходе нечеткого логического вывода выполняются следующие шаги:

1. Для каждого четкого значения $i^*_{j,n}$, подаваемого на вход системы, вычисляется значение функции принадлежности $\mu_{t_{j,n}}(i^*_{j,n})$.
2. Для каждого правила r_j вычисляется уровень отсечения

$$\alpha_j = \min_n \mu_{t_{j,n}}(i^*_{j,n}). \quad (6)$$

И находятся усеченные функции принадлежности нечетких множеств, соответствующих лингвистическим значениям выходных переменных

$$\acute{\mu}_{t_j}(x) = \alpha_j \cdot \mu_{t_j}(x) \quad (7)$$

3. Для каждой выходной переменной g_j формируется результирующая функция принадлежности

$$\mu_{g_j}(x) = \max_j \acute{\mu}_{t_j}(x) \quad (8)$$

4. Для каждой выходной переменной g_j вычисляется четкое значение

$$g_j^* = \frac{\int x \cdot \mu_{g_j}(x)}{\int \mu_{g_j}(x)} \quad (9)$$

Задание 2.

В реальных экспертных системах чаще всего реализуется неточный вывод, когда нет полной уверенности в принимаемом решении. Существуют различные способы определения результирующего значения определенности вывода: вероятностные, экспертные и пр. Рассмотрим методику расчета определенности, впервые примененную в экспертной системе MYCIN.

Воспользуемся представлением базы знаний в виде И/ИЛИ графа.

Пусть БЗ представлена следующими правилами:

ЕСЛИ A то B

ЕСЛИ A то C

ЕСЛИ A то D

ЕСЛИ B то E

ЕСЛИ B и C и D то F

Соответствующий И/ИЛИ граф будет выглядеть следующим образом:

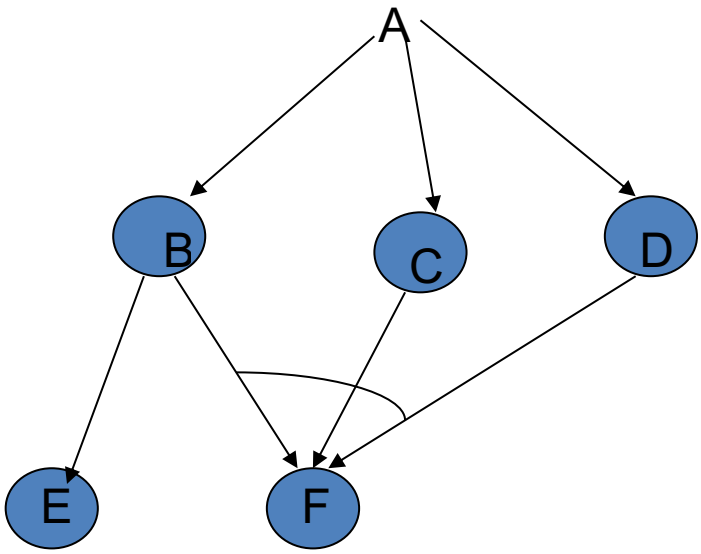


Рис. 1. И/ИЛИ граф базы знаний

Для расчета определенности воспользуемся следующими заключениями: Если соединение вершин графа по **«И»** то степень достоверности заключения не может быть больше самой слабой из составляющих посылки, т.е.

$$K = min\ (K1, K2, \ldots, Kn) \qquad (1)$$

Коэффициент достоверности получаемого заключения сравнивается с некоторым пороговым значением (в MYCIN оно равно 0,2), которое определяется эмпирически.

Для дизъюнктивной вершины считаем, что посылки подкрепляют друг друга при получении заключения.

Если имеются 2 продукции:

P1: A1->D P2: A2->D, то

$$KD = KA1 + KA2\ (1 - KA1) = KA1 + KA2 - KA1KA2 \qquad (2)$$

Если имеется 3 продукции, то коэффициент достоверности KD рассчитывается по формуле:

$$KD = KA1 + KA2 + KA3 - KA1KA2 - KA1KA3 - KA2KA3 + KA1KA2KA3 \qquad (3)$$

Учитывая коэффициенты достоверности продукций, получим

P1 и P2-KD=KA1KP1+KA2KP2-KA1KA2KP1KP2

Будем считать, что информация посылки или заключения с коэффициентом достоверности < 0,4 далее при выводе не используется.

Рассмотрим пример.

В некоторой технической системе произошла авария. Для принятия решений по ее ликвидации необходимо выяснить причину аварии, каждая причина требует своих действий. Диспетчер на основе опыта предполагает, что причиной является R. Нужно определить коэффициент достоверности *Kr* гипотезы *R*. Для подтверждения своего предположения диспетчер обращается к ЭС и вводит данные *(Fi)* с коэффициентами достоверности. И/ИЛИ граф БЗ представлен на рис.2.

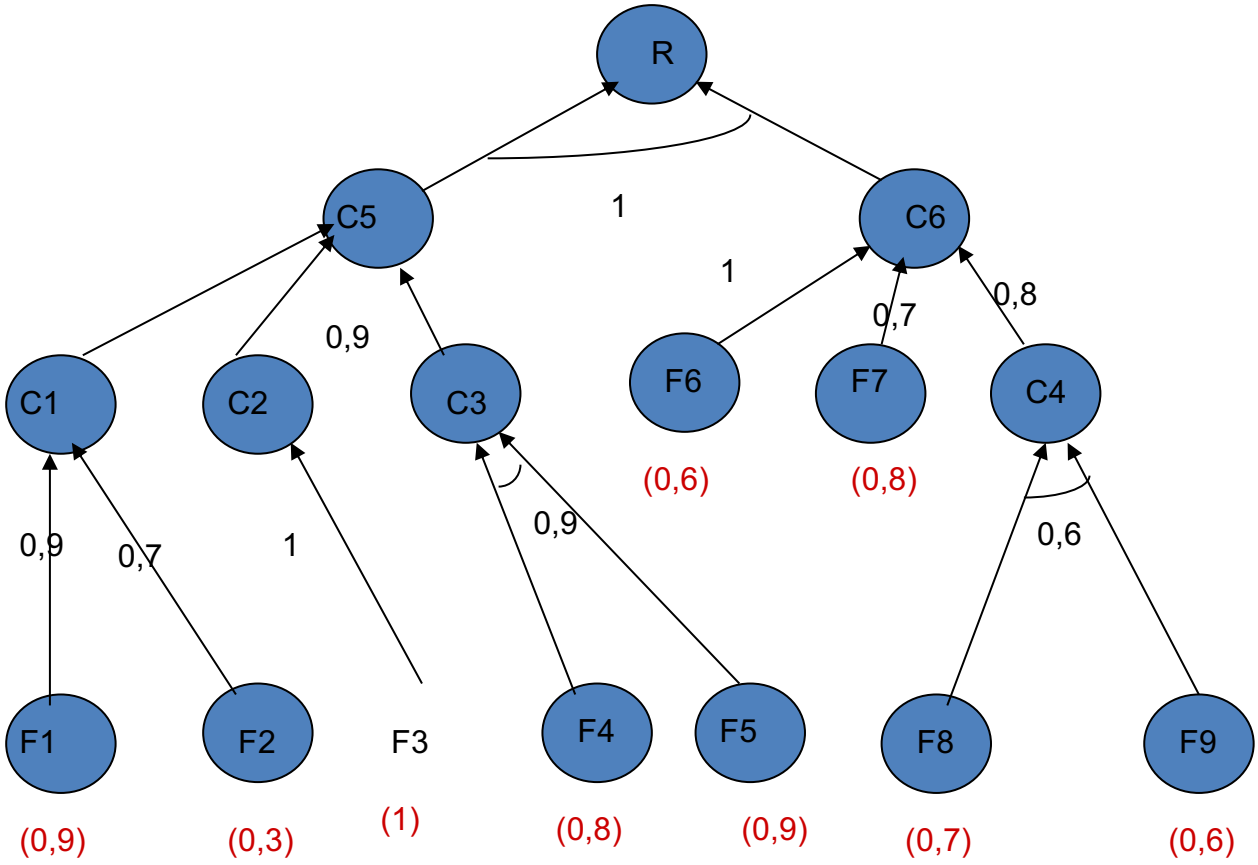


Рис.2 И/ИЛИ граф базы знаний

KC1=KF1 • KP1=0,9•0,9=0,81

KC2=KF3 •KP3=1

KC3=min(KF4,KF5)KP4= 0,72

KC4=min(KF8, KF9)KP7=0,36 гипотеза С4 отбрасывается

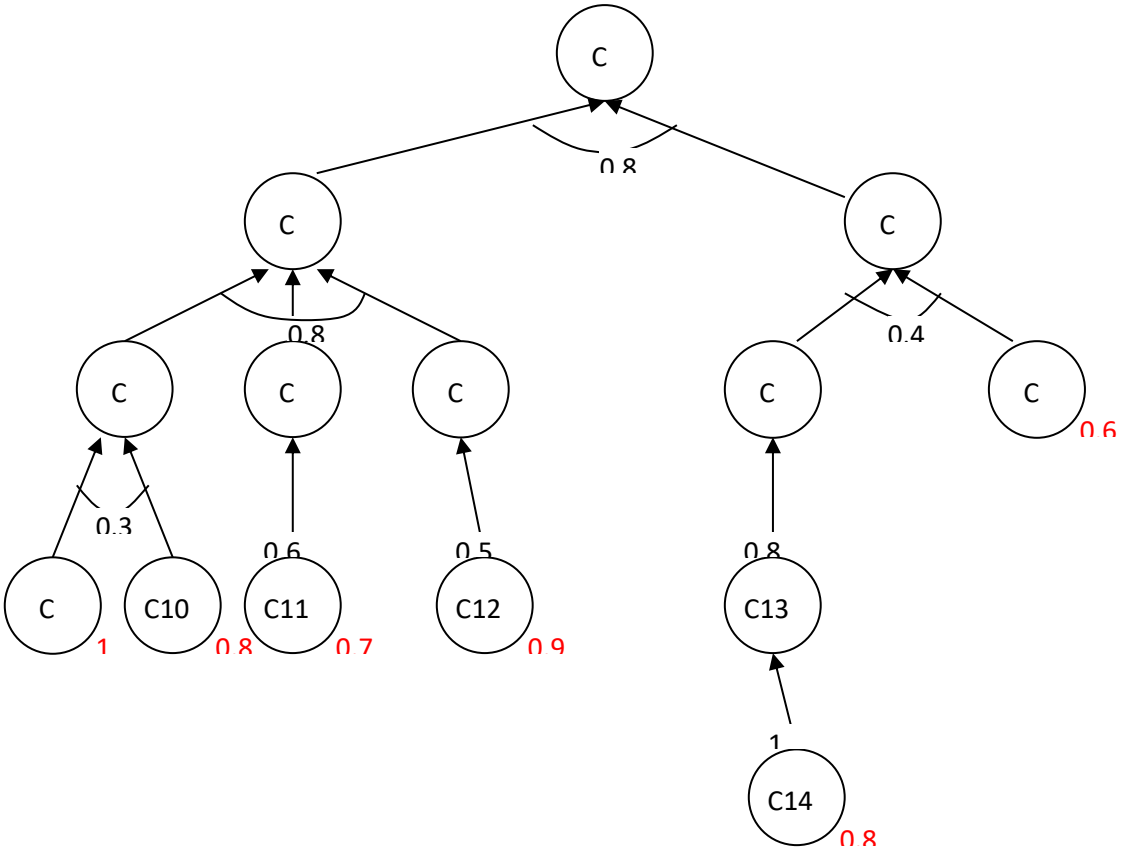
KC5=min(KC1,KC2,KC3)KP8=min(0,81;1;0,72) •0,9=0,648

KC6=KF6KP5+KF7KP6-KF6KF7KP6KP7=0,824

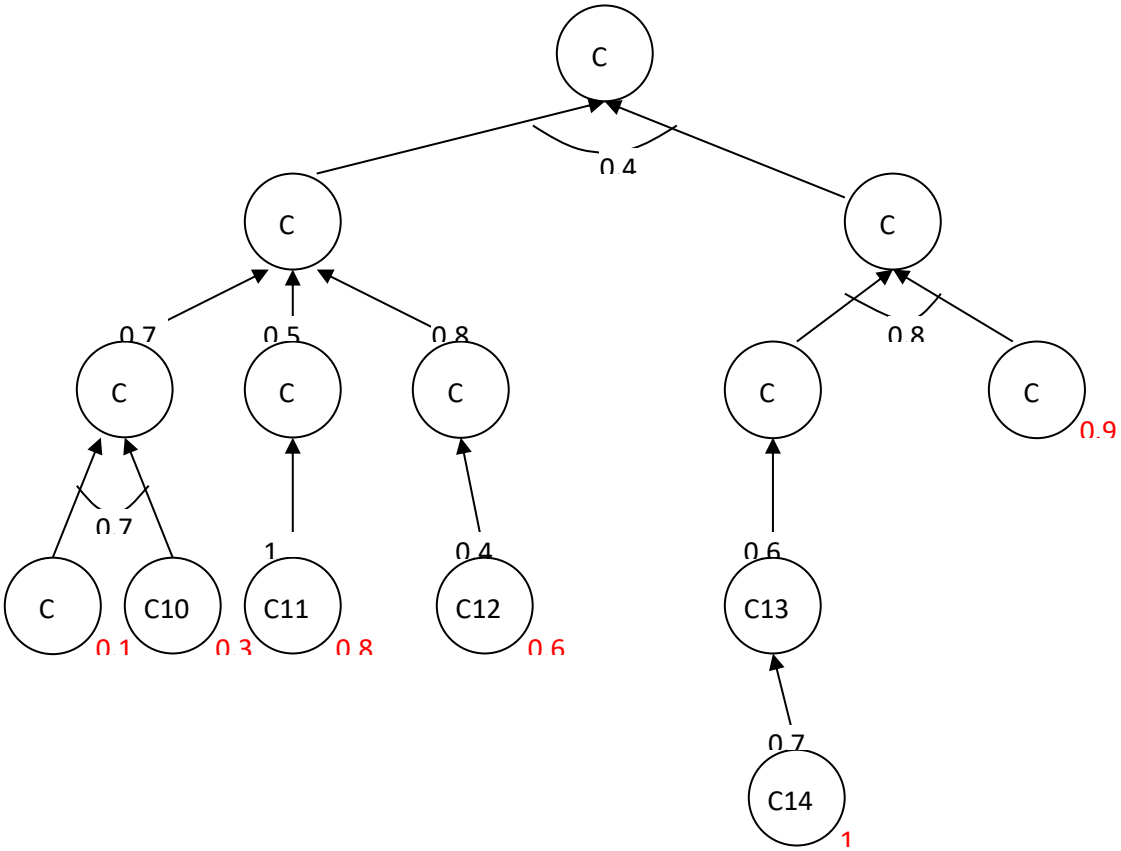
KR=min(KC5,KC6) KP10=0,648 гипотеза подтверждена

Задания для самостоятельного выполнения:

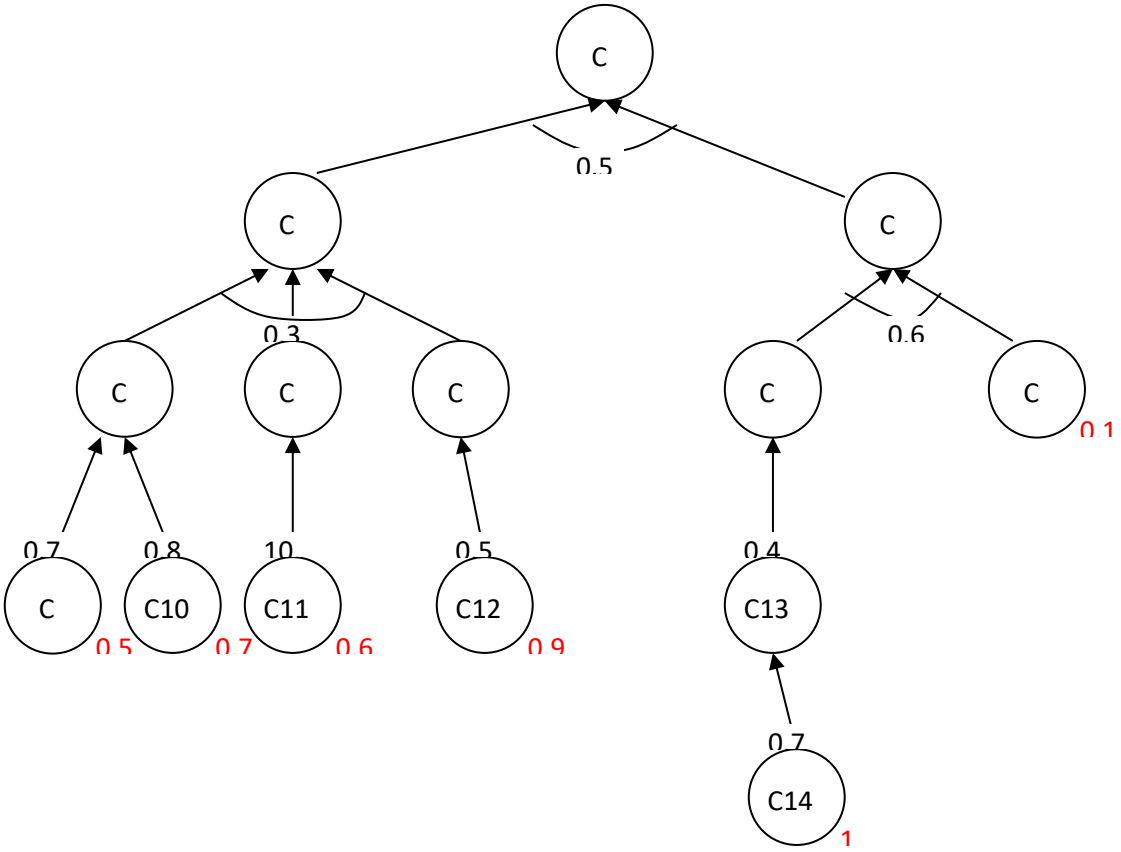
Вариант 1



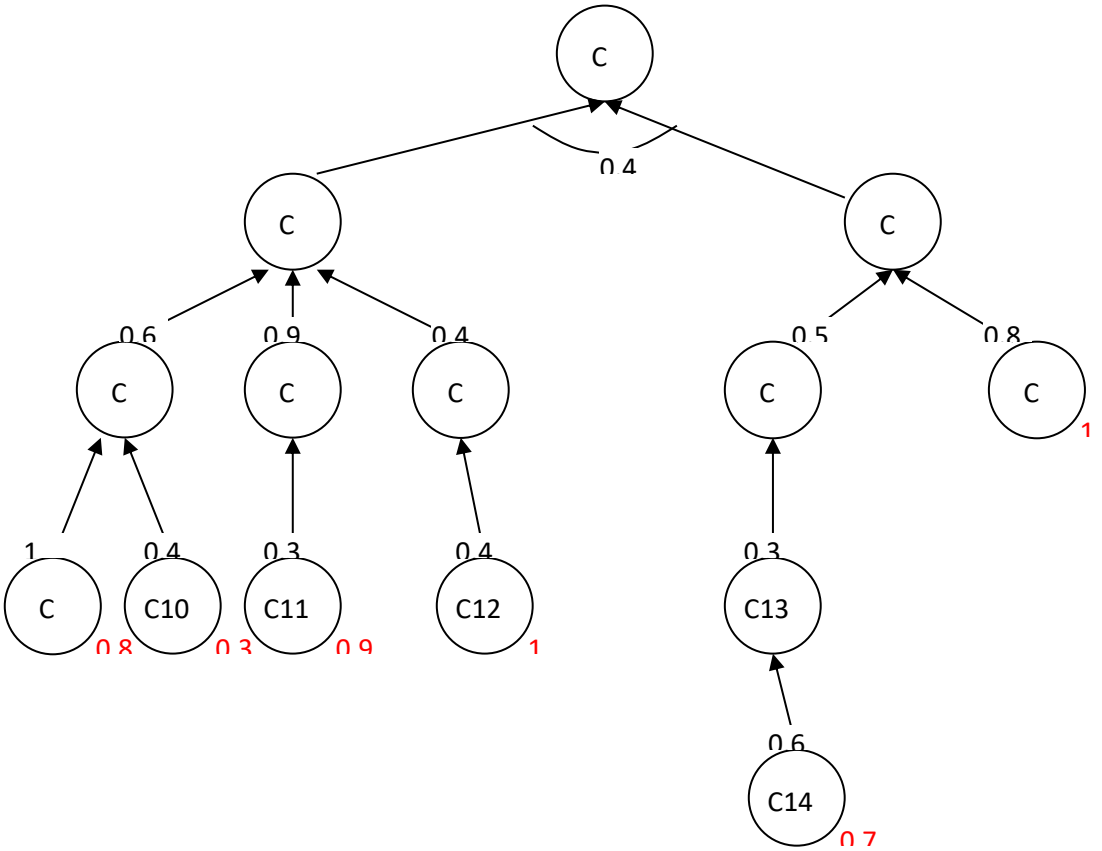
Вариант 2


C
0,4
C
C
0,7
0,5
0,8
0,8
C
C
C
C
C
0,9
0,7
1
0,4
0,6
C
C10
C11
C12
C13
0,1
0,3
0,8
0,6
0,7
C14
1

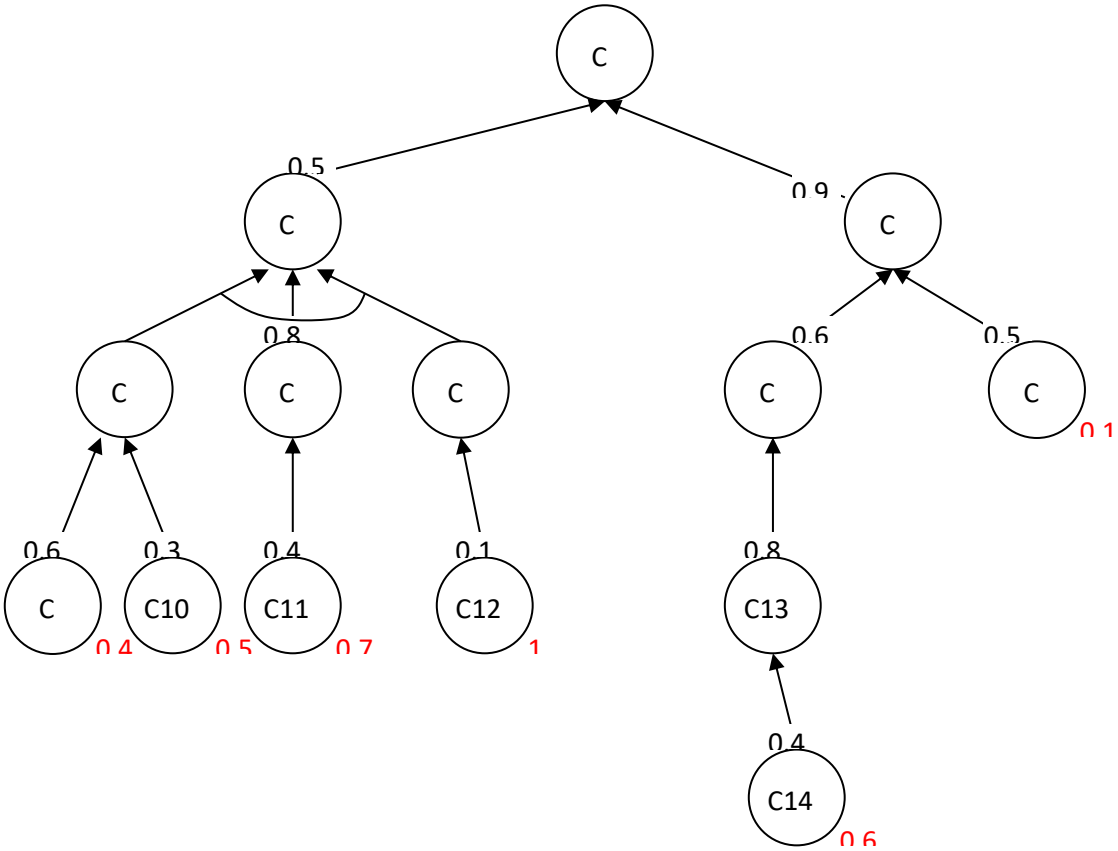
Вариант 3


C
0,5
C
C
0,3
0,6
C
C
C
C
C
0,1
0,7
0,8
10
0,5
0,4
C
C10
C11
C12
C13
0,5
0,7
0,6
0,9
0,7
C14
1

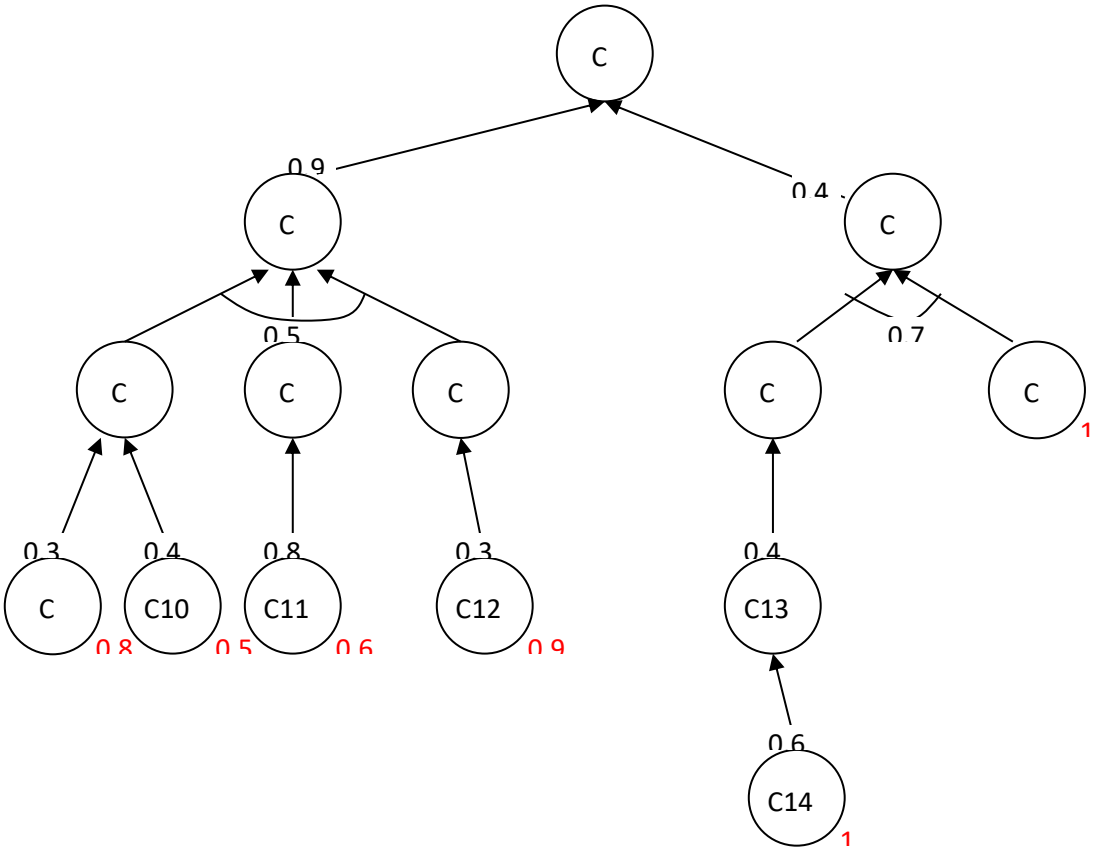
Вариант 4



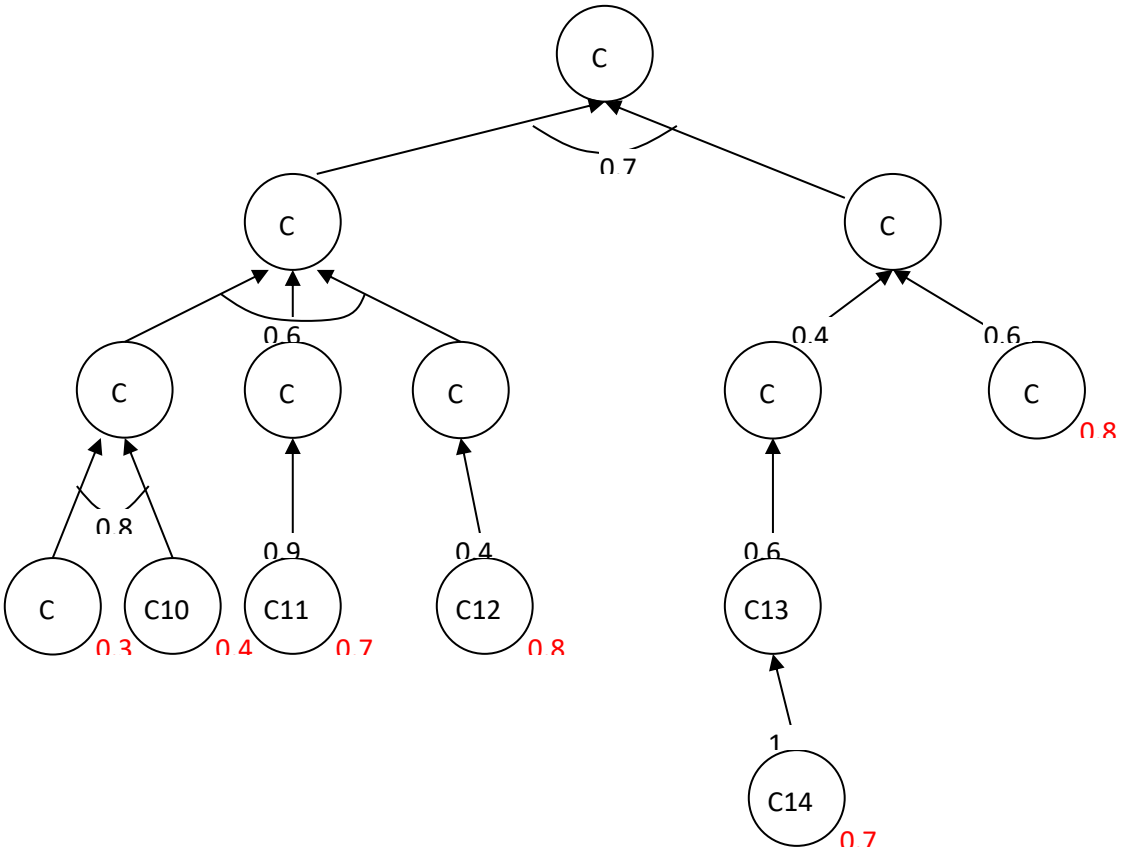
Вариант 5



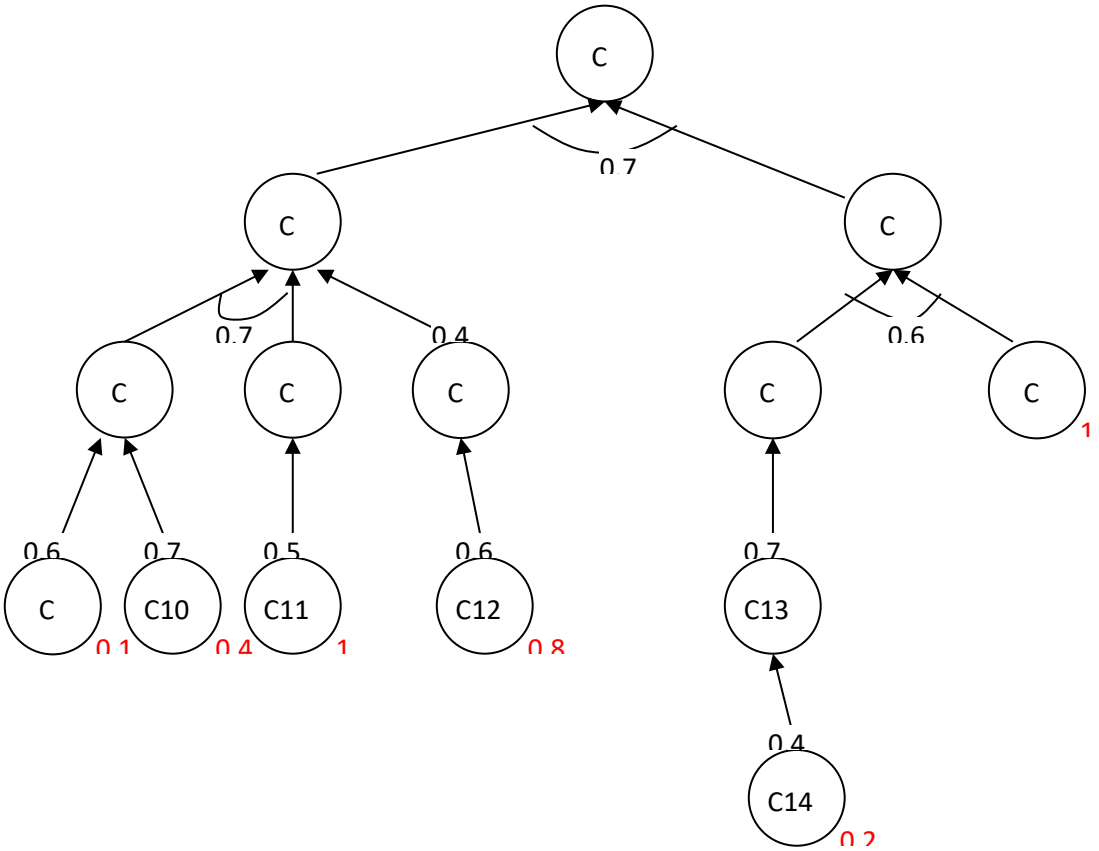
Вариант 6



Вариант 7



Вариант 8



Практическая работа №22-24.

Разработка базы знаний диалоговой системы на основе ELIZA

В качестве модели системы естественно-языкового общения возьмем за основу программу ELIZA, созданную Джозефом Вейценбаумом в лаборатории *искусственного интеллекта* массачусетского технологического института в 1966 году (названную в честь Элизы из "Пигмалиона"). Она была написана на языке Лисп и состояла всего из нескольких десятков строк программного кода. Эта программа моделировала методику известного психотерапевта Карла Роджерса. В этом подходе психотерапевт играет роль "вербального зеркала" пациента. Он переспрашивает пациента, повторяет его слова, позволяя ему самому найти выход из сложившейся ситуации, прийти в состояние душевного равновесия.

Вейценбаум создал эту программу в качестве шутки. Многие пациенты, пообщавшись с детищем Вейценбаума, утверждали, что "Элиза" им помогла, и отказывались верить, что их собеседником был не психотерапевт, а компьютерная программа. Всю оставшуюся жизнь Вейценбаум пытался охладить восторженных поклонников его программы и убедить общественность, что машина не может мыслить.

Несмотря на простоту программы ELIZA опытным разработчикам удается создать иллюзию человеческого общения. Сегодня проводятся мировые чемпионаты по ЭЛИЗЕ, база данных которой наполняется все новыми и новыми темами.

Принцип построения программы ELIZA

Программа пытается сопоставить вводимые пользователем ответы с имеющимися у нее шаблонами и, если ей это удается, шаблонно же отвечает.

Наша программа будет действовать по следующему алгоритму.

1. Попросить человека описать имеющуюся у него проблему.
2. Прочитать строку ответа с клавиатуры.
3. Подобрать шаблон, которому соответствует введенная человеком строка.
4. Если удалось — выдать соответствующий этому шаблону ответ пользователю.
5. Если подобрать шаблон не удалось — попросить продолжать рассказ (ввести строку с клавиатуры).
6. Возвратиться к пункту 2.

Для решения этой задачи нам понадобится предикат, преобразующий строку, вводимую пользователем в список слов. Поэтому введем новый вспомогательный предикат, который будет считывать символ за символом до тех пор, пока не встретит символ-разделитель (пробел, запятая, точка и другой знак препинания). Так как проверять совпадение очередного символа с символом-разделителем нам придется не раз, заведем список символов-разделителей. Поместим его в раздел описания констант и назовем separators (символы-разделители). После этого все символы до символа-разделителя будут помещены в первое слово строки, а все символы, идущие после символа-разделителя, обработаны подобным образом.

Для облегчения распознавания строк переведем все введенные символы в нижний регистр. В этом случае не придется предусматривать всевозможные варианты написания пользователем слова (например, "Да", "да", "ДА"), мы будем уверены, что все символы слова — строчные ("да").

При реализации этого предиката нам понадобятся три вспомогательных предиката.

Первый предикат будет преобразовывать прописные русские буквы в строчные, а все остальные символы оставлять неизменными. У него будет два аргумента: первый (входной) — исходный символ, второй (выходной) — символ, полученный преобразованием первого аргумента.

При написании данного предиката стоит учесть, что строчные русские буквы расположены в таблице символов двумя группами. Первая группа (буквы от 'а' до 'п') имеют, соответственно, коды от 160 до 175. Вторая группа (буквы от 'р' до 'я') — коды от 224 до 239.

С учетом вышеизложенного предикат можно записать, например, так:

```
lower_rus(C,C1):–
    'А'<=C,C<='П',!, /*         символ С лежит между
          буквами 'А' и 'П' */
    char_int(C,I),  /* I — код символа С */
    I1=I+(160–128), /* 160 — код буквы 'а',
```

```
        128 — код буквы 'А'*/
    char_int(C1,I1).
       /* C1 — символ с кодом I1 */
lower_rus(C,C1):–
    'Р'<=C,C<='Я',!, /*          символ С лежит между
         буквами 'Р' и 'Я' */
    char_int(C,I),  /* I — код символа С */
    I1=I+(224–144), /* 224 — код буквы 'р',
         144 — код буквы 'Р'*/
    char_int(C1,I1).
       /* C1 — символ с кодом I1 */
lower_rus(C,C). /* символ С отличен от прописной русской
      буквы и, значит, мы не должны его
      изменять */
```

Второй предикат first_word будет иметь три аргумента. Первый (входной) — исходная строка, второй и третий (выходные) — соответственно, первое слово строки (не содержащее прописных русских букв) и остаток строки, полученный удалением из него первого слова.

Выглядеть его реализация будет следующим образом:

```
first_word("","",""):–!. /* из пустой строки можно
             выделить только пустые
             подстроки */
first_word(S,W,R):– /* W — первое слово строки S, R —
            остальные символы исходной
            строки S */
     frontchar(S,C,R1),
      /* C — первый символ строки S, R1 —
         остальные символы */
     not(member(C,separators)),!,
      /* символ С не является
         символом-разделителем */
     first_word(R1,S1,R),
```

```
        /* S1 — первое слово строки R1,
           R — оставшиеся символы
           строки R1 */
      lower_rus(C,C1),
        /* если C — прописная русская
           буква , то C1 — соответствующая
           ей строчная буква, иначе
           символ C1 не отличается
           от символа C */
      frontchar(W,C1,S1).
        /* W — результат "приклеивания"
           символа C1 в начало строки S1 */
first_word(S,"",R):– /* в случае, если первый символ
            оказался символом-разделителем, */
frontchar(S,_,R). /* его нужно выбросить, */
```

Третий предикат del_sep будет предназначен для удаления из начала строки символов-разделителей. У него будет два аргумента. Первый (входной) — исходная строка, второй (выходной) — строка, полученная из первого аргумента удалением символов-разделителей, расположенных в начале строки, если таковые имеются.

```
del_sep("","").–!.
del_sep(S,S1):–
            frontchar(S,C,R),
              /* C — первый символ строки,
       R — остальные символы */
            member(C,separators),!,
              /* если C является
       символом-разделителем, */
            del_sep(R,S1).
              /* то переходим к рассмотрению
       остатка строки */
del_sep(S,S) . /* если первый символ строки не является
      символом-разделителем, то удалять
```

```
    нечего */
```

И, наконец, предикат, преобразующий строку в список слов.

```
str_w_list("",[]):–!. /* пустой строке соответствует
        пустой список слов, входящих
        в нее */
str_w_list(S,[H     T]):–
    first_word(S,H,R),!,
        /* H — первое слово строки S,
        R — оставшиеся символы
        строки S */
    str_w_list(R,T).
        /* T — список, состоящий из слов,
          входящих в строку R */
```

Основную работу в программе будет осуществлять предикат recognize, задачей которого будет распознавать шаблон, которому можно сопоставить введенную строку. Этот предикат на входе будет получать список слов строки, а на выходе будет выдавать номер шаблона. По этому номеру другой предикат должен будет выдать на экран соответствующую реакцию (вопрос, реплику, уточнение).

Наша учебная программа будет распознавать одиннадцать шаблонов:

1. Человек хочет закончить работу с программой. Об этой ситуации свидетельствует наличие в списке таких слов, как "пока", "свидания" (часть словосочетания "до свидания"). В ответ программа также прощается и выражает надежду, что она смогла чем-нибудь помочь.
2. Человек испытывает какое-то чувство (наличие в списке слова "испытываю"). Программа реагирует вопросом о том, как давно человек испытывает это чувство.
3. Если во вводимой строке встретились слова "любовь" или "чувства", то программа поинтересуется, не боится ли человек эмоций.
4. При обнаружении слова "секс" во входном списке слов будет выдано сообщение о важности сообщения.
5. В случае наличия слов "бешенство", "гнев" или "ярость", программа уточнит, что человек испытывает в данный момент времени.
6. В ответ на краткий ответ ("да" или "нет") будет выдана просьба рассказать подробнее.
7. Если в списке слов найдутся слова "комплекс" или "фиксация", программа отреагирует замечанием о том, что человек слишком много "играет".
8. Появление слова "всегда" в строке, введенной человеком, приводит к ответной реакции — вопросу о том, может ли человек привести какой-нибудь пример.
9. В случае, если человек упомянул кого-то из своих родных ("папа", "мама", "жена", "муж", "брат", "сестра", "сын", "дочь" и т.д.), программа попросит рассказать

поподробнее о его семье. При этом упомянутый родственник будет помещен в базу данных, чтобы потом продолжить этот разговор.

10. Если в процессе разговора была сделана запись во внутреннюю базу данных и в данный момент спросить больше не о чем, программа "вспомнит" об упомянутом родственнике и выдаст фразу: "ранее Вы упоминали ..."
11. И, наконец, если введенная строка не подходит ни под один шаблон, программа просит продолжить рассказ.

А теперь запишем всю программу целиком.

```
CONSTANTS /* раздел описания констант */
separators=[' ', ',', '.', ';']
      /* символы-разделители (пробел,
        запятая, точка, точка с запятой
        и т.д.) */
DOMAINS /* раздел описания доменов */
i=integer
s=string
ls=s* /* список слов */
lc=char* /* список символов */
DATABASE /* раздел описания предикатов базы данных */
Important(s)
PREDICATES /* раздел описания предикатов */
member(s,ls) /* проверяет принадлежность строки списку
    строк */
member(char,lc) /* проверяет принадлежность символа списку
      символов */
lower_rus(char,char) /* преобразует прописную русскую
          букву в строчную букву */
del_sep(s,s) /* удаляет из начала строки
    символы-разделители */
first_word(s,s,s) /* делит строку на первое слово
        и остаток строки */
str_w_list(s,ls)  /* преобразует строку в список слов */
read_words(ls)      /* читает строку с клавиатуры, возвращает
      список слов, входящих в строку*/
```

```
recognize(ls,i) /* сопоставляет списку слов число,
        кодирующее шаблон */
answ(ls) /* выводит ответ человеку */
eliz /* основной предикат */
repeat
CLAUSES /* раздел описания предложений */
eliz:–
    repeat,
    read_words(L), /*          читаем строку с клавиатуры,
          преобразуем ее в список слов L */
    recognize(L,I), /*          сопоставляем списку слов L номер
          шаблона I */
    answ(I),nl, /* выводим ответ, соответствующий номеру
        шаблона I */
    I=0 /* номер шаблона I, равный нулю, означает,
          что человек попрощался */.
read_words(L):–
            readln(S), /* читаем строку */
            str_w_list(S,L). /* преобразуем строку
          в список слов */
recognize(L,0):–
   member("пока",L),!;
   member("свидания",L),!.
recognize(L,1):–
   member("испытываю",L),!.
recognize(L,2):–
   member("любовь",L),!;
   member("чувства",L),!.
recognize(L,3):–
   member("секс",L),!.
recognize(L,4):–
   member("бешенство",L),!;
```

```
    member("гнев",L),!;
    member("ярость",L),!.
recognize(L,5):–
    L=["да"],!;
    L=["нет"],!.
recognize(L,6):–
    member("комплекс",L),!;
    member("фиксация",L),!.
recognize(L,7):–
    member("всегда",L),!.
recognize(L,8):–
    member("мать",L),assert(important("своей матери")),!;
    member("мама",L),assert(important("своей маме")),!;
    member("отец",L),assert(important("своем отце")),!;
    member("папа",L),assert(important("своем папе")),!;
    member("муж",L),assert(important("своем муже")),!;
    member("жена",L),assert(important("своей жене")),!;
    member("брат",L),assert(important("своем брате")),!;
    member("сестра",L),assert(important("своей сестре")),!;
    member("дочь",L),assert(important("своей дочери")),!;
    member("сын",L),assert(important("своем сыне")),!.
recognize(_,9):–
    important(_),!.
recognize(_,10).
answ(0):–
        write("До свидания"),nl,
        write("Надеюсь наше общение помогло Вам").
answ(1):–
        write("Как давно Вы это испытываете?").
answ(2):–
        write("Вас пугают эмоции?").
answ(3):–
```

```
        write("Это представляется важным").
answ(4):–
        write("А что Вы испытываете сейчас?").
answ(5):–
        write("Расскажите об этом подробнее").
answ(6):–
        write("Слишком много игр").
answ(7):–
        write("Вы можете привести какой–нибудь пример?").
answ(8):–
        write("Расскажите мне подробнее о своей семье").
answ(9):–
        important(X),!,
        write("Ранее Вы упомянули о ",X),
        retract(X).
answ(10):–
         write("Продолжайте, пожалуйста").
repeat.
repeat:–
       repeat.
member(X,[X|_]):–!.
member(X,[_|S]):–member(X,S).
lower_rus(C,C1):–
    'А'<=C,C<='П',!, /* символ С лежит между
                буквами 'А' и 'П' */
    char_int(C,I), /* I — код символа С */
    I1=I+(160–128), /* 160 — код буквы 'а',
          128 — код буквы 'А'*/
    char_int(C1,I1). /* С1 — символ с кодом I1 */
lower_rus(C,C1):–
    'Р'<=C,C<='Я',!, /* символ С лежит между
                буквами 'Р' и 'Я' */
```

```
    char_int(C,I), /* I — код символа C */
    I1=I+(224–144), /* 224 — код буквы 'р',
          144 — код буквы 'Р'*/
    char_int(C1,I1). /* C1 — символ с кодом I1 */
lower_rus(C,C). /* символ C отличен от прописной русской
      буквы и, значит, мы не должны его
      изменять */
del_sep("","").–!.
del_sep(S,S1):–
          frontchar(S,C,R),
           /* C — первый символ строки,
      R — остальные символы */
          member(C,separators),!,
           /* если C является
      символом-разделителем, */
          del_sep(R,S1). /* то переходим
        к рассмотрению остатка
        строки */
del_sep(S,S) . /* если первый символ строки не является
      символом-разделителем, то удалять
      нечего */
str_w_list("",[]):–!.
   /* пустой строке соответствует пустой список
      слов, входящих в нее */
str_w_list(S,[H|T]):–
       first_word(S,H,R),!,
          /*          H — первое слово строки S, R —
            оставшиеся символы строки S */
       str_w_list(R,T).
          /*          T — список, состоящий из слов,
            входящих в строку R */
first_word("","","").–!. /* из пустой строки можно
```

```
                выделить только пустые
                подстроки */
first_word(S,W,R):– /* W — первое слово строки S, R —
          остальные символы исходной строки S */
      frontchar(S,C,R1),
            /* C — первый символ строки S,
               R1 — остальные символы */
      not(member(C,separators)),!,
            /* символ C не является
               символом-разделителем */
      first_word(R1,S1,R),
            /* S1 — первое слово строки R1,
               R — оставшиеся символы
               строки R1 */
      lower_rus(C,C1),
            /* если C — прописная русская
               буква , то C1 — соответствующая
               ей строчная буква, иначе символ
               C1 не отличается от символа C */
      frontchar(W,C1,S1).
            /* W — результат "приклеивания"
               символа C1 в начало
               строки S1 */
first_word(S,"",R):– /* в случае, если первый символ
           оказался символом-разделителем, */
       frontchar(S,_,R). /*       его нужно
               выбросить, */
GOAL /* раздел описания цели */
write("Расскажите, в чем заключается Ваша проблема"),nl,
eliz,
readchar(_).
```

Задания для самостоятельного выполнения:

Используя в качестве основы текст программы ELIZA, разработайте программу, ведущую диалог на определенные темы. Темы выбираются и согласовываются с преподавателем. Темы не должны повторяться.

1. Чемпионат по футболу 2010 года.
2. Любимый кинофильм (сюжет, герои, обсуждение)
3. Любимая книга (сюжет, герои, обсуждение)
4. Погода (летняя жара 201 года, пожары, задымление и пр.)
5. Историческое событие (участники, причины, следствие, обсуждение)
6. Постановка диагноза в медицине
7. Выбор места для отпуска, каникул

Форма отчета: Разработанное программное обеспечение на языке ПРОЛОГ

ПЕРЕЧЕНЬ УЧЕБНОЙ ЛИТЕРАТУРЫ:

1. Интеллектуальные системы принятия решений и управления : учеб. пособие / Ю. И. Еременко. - 2-е изд., стер. - Старый Оскол : ТНТ, 2018. - 404 с. ; 21 см. - Библиогр.: с. 395-401 (64 назв.). - Гриф: рек. УМО РАЕ по класс. унив. и техн. образованию в качестве учеб. пособия для студ. вузов, обучающихся по направлению "Информационные системы и технологии". - ISBN 978-5-94178-464-6 - 9 экз.
2. Интернет вещей: будущее уже здесь : пер. с англ. / С. Грингард. - М. : ИГ "Точка" : Альпина Паблишер, 2017. - 224 с. ; 17 см. - (Завтра это будут знать все). - **ISBN** 978-5-9614-6118-3. - **ISBN** 978-5-9908700-0-0 : - 7 экз.
3. Машинное обучение: новый искусственный интеллект : пер. с англ. / Э. Алпайдин. - М. : ИГ "Точка" : Альпина Паблишер, 2017. - 208 с. ; 17 см. - (Завтра это будут знать все). - Библиогр.: с. 185-189 . - **ISBN** 978-5-9614-6114-5. - **ISBN** 978-5-9908700-8-6 - 7 экз.
4. Интеллектуальные системы и технологии : учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. - М. : Юрайт, 2017. - 397 с. : ил. ; 24 см. - (Бакалавр и магистр. Академический курс). - Библиогр. в конце разд. - Гриф: рек. Умо высш. образования в качестве учебника и практикума для студ. вузов, обуч. по инженерно-техн. напр. - **ISBN** 978-5-534-02126-4 – 15 экз.
5. Ясницкий, Л.Н. Интеллектуальные системы [Электронный ресурс] : учебник / Л.Н. Ясницкий. — Электрон. дан. — Москва : Издательство "Лаборатория знаний", 2016. — 224 с. — Режим доступа: https://e.lanbook.com/book/90254. — Загл. с экрана.
6. Кораблев, Ю.А. Интеллектуальные технологии в системах управления и диагностики [Электронный ресурс] : учебное пособие / Ю.А. Кораблев, М.Ю. Шестопалов, М.И. Халиков. — Электрон. дан. — Санкт-Петербург : СПбГЛТУ, 2012. — 112 с. — Режим доступа: https://e.lanbook.com/book/45248. — Загл. с экрана.
7. Афонин, В.Л. Интеллектуальные робототехнические системы [Электронный ресурс] : учебное пособие / В.Л. Афонин, В.А. Макушкин. — Электрон. дан. — Москва : , 2016. — 222 с. — Режим доступа: https://e.lanbook.com/book/100607. — Загл. с экрана
8. Добров, Б.В. Онтологии и тезаурусы: модели, инструменты, приложения [Электронный ресурс] : учебное пособие / Б.В. Добров, В.В. Иванов, Н.В. Лукашевич, В.Д. Соловьев. — Электрон. дан. — Москва : , 2016. — 207 с. — Режим доступа: https://e.lanbook.com/book/100277. — Загл. с экрана.

ПРОЕКТИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Методические указания по выполнению

практических работ по дисциплине

«Проектирование интеллектуальных систем»

Автор

Долинина Ольга Николаевна

УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Декан факультета информационных систем и технологий
_____ Святов К.В.
«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Дисциплина (модуль) Проектирование интеллектуальных систем
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Методические рекомендации составлены

В течение семестра студент должен выполнить теоретическое исследование и создать базу знаний для интеллектуальной системы, реализующей принятие решения на основе составления сценариев.

Под сценарием понимаем совокупность типовых сцен с возможными исходами, последовательность сцен на принятия решения.

Структура сцены:

Наименование сцены:

Участники:

Цель:

Последовательность действий.

Цель курсовой работы: Разработать базу знаний в предметной области и описать способ принятия решения в предметной области, где возможно использование типовых ситуаций

Варианты предметных областей:

- Сценарии народных сказок: цель работы - составить базу знаний и метод принятия решения для генерации произвольных сценариев на основе типовых ситуаций.
- Сценарии правил дорожного движения: цель работы - составить базу знаний и метод принятия решения для генерации возможности безварийного проезда типовых перекрестков при произвольно выбранных участниках дорожного движения.

Разработанная база знаний на примере сказок представляется в следующем виде:

1. Структурированное описание всех активных участников:

Наименование
Перечень характеристик, значения характеристик
Например,
1.1 Персонаж
Человек/животное, волшебный/обычный, умный/глупый, …
1.2 Артефакт
Назначение, условия срабатывания…

2. Цель сказки
3. Число этапов сказки, число сцен в каждом этапе
4. Описание сцен
5. Описание взаимодействия персонажей, артефактов.
6. Описание условий взаимодействия персонажей

Задача: разработать базу знаний, которая позволит автоматически формировать сказку для выбранных случайным образом персонажей, исходя из описанных сцен, правил взаимодействия, артефактов.

Разработанная база знаний для правил дорожного движения представляется в следующем виде:

1. Структурированное описание всех активных участников движения:

Наименование
Перечень характеристик, значения характеристик
Например,
1.1 легковой автомобиль
1.2 трамвай
1.3 автобус, …

2. Описание типовых сцен безаварийного движения
3. Описание правил взаимодействия участников движения.
4. Описание приоритетов взаимодействия участников движения

Задача: разработать базу знаний, которая позволит автоматически формировать ситуаций с правильным поведением выбранных случайным образом участников движения.

ПРОЕКТИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Методические указания по выполнению

курсовой работы по дисциплине

«Проектирование интеллектуальных систем»

Автор

Долинина Ольга Николаевна

УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.04 Международная профессиональная коммуникация

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ

Декан факультета информационных систем и технологий

_____

«____»_____2021 г.

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Международная профессиональная коммуникация

Профиль подготовки

09.04.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Квалификация выпускника

Магистр

Формы обучения

очная

г. Ульяновск, 2021

Раздел 1. Профессиональная коммуникация. Module 1. Professional Communication

ПР01. Тема. Устройство на работу.

Основные виды работы, их краткая характеристика на английском языке; описание обязанностей, связанных с выполнением того или иного вида работы.

ПР01. Application for a job.

Jobs and their brief description in English; a description of job responsibilities.

Exercise 1. Choose the correct answer.

1. A bank teller_____ in a bank.
   A manages advertising B manages the credit department C receives and pays out money D tells banking stories
2. A _____ marks errors in the first printed copy of the text.
   A caretaker B controller C printer D proofreader
3. He is looking for a _____ in electrical engineering.
   A job B labor C occupation D work
4. What do you do in your spare time? What is your favorite _____? – I like to read books about traveling.
   A job B labor C occupation D work
5. He is a member of staff and works from nine to five in the main office. He is a _____ worker.
   A freelance B full-time C part-time D temporary
6. A _____ usually has quite a few vacancies for skilled and unskilled workers.
   A art gallery B large construction company C local school D small travel agency

Exercise 2. Read the job ads and match the descriptions to the position.

A Digital Marketing Manager B Sales Manager C Electronics & Systems Engineer

Advertisement 1

Looking for your next big break? Join a progressive, forward thinking company with an exciting range of products and a fantastic reputation.

Role

You will be responsible for driving sales for Plant and Tool hire services to the construction industry. You will help to grow and develop business and maintain many existing accounts with small to large companies.

Company

A market leading hire and sales company with a fantastic reputation!

To be successful, you may have:

•Previous experience in a similar industry

•A proven track record of winning new business within the plant and tool hire markets or related.

•A high level of motivation, determination and passion.

•The ability to deal with any questions or issues to meet customer's expectations.

•A high level of communicative skills both verbally and written

Benefits

A basic salary of £43,000 - £47,000

Bonus

Company Car, Fuel Card

Advertisement 2

The Opportunity

- An exciting opportunity to join our Navigation Sensors Group.
- The Team are responsible for the specification and assessment of inertial and satellite navigation equipment.
- The successful candidate will work within the inertial sensors team and apply proven understanding of electronics for the specification of inertial systems

The Role

You would get detailed understanding of rate sensors and accelerometers to allow requirements to be created.
Monitoring supplier and technology roadmaps and identifying cutting edge and future technologies for investment.
Supporting sub assembly as part of an international team.

- Undertake research and design activities as part of an international team.
- Monitoring supplier and technology roadmaps and identifying cutting edge and future technologies for investment.
- Collaborating with suppliers and researchers of Navigation Sensors and related technologies.
- Designing and conducting laboratory tests.

What's in it for you?

This role offers an opportunity to be part of a successful team working to develop new and innovative solutions to address complex customer requirements.

- You will develop expertise in navigation sensors; both conventional and cutting-edge technologies.
- Involvement in the strategic growth of a rapidly evolving team.
- You will take a key role in several multi-national research programmes.
- The work is technically challenging, innovative and rewarding.

Advertisement 3

Mobile Fun- Who are we?

We are Europe's leading eCommerce retailer offering the latest mobile devices accessories from our websites, we were established in 2000. We are based in Birmingham and currently employ around 60 staff with a great reputation and having the highest possible standards. This is a fantastic place to work with a great team.

And what would the main responsibilities be?

Accountability and responsibility for the management and ongoing development of Mobile Fun's marketing team, ensuring effective coordination of marketing campaigns and activities across all Mobile Fun channels and the strategic development of all marketing activities and reach across all Mobile Fun's website portfolios.

Self Development

You will have the opportunity to access personalised learning and development opportunities that will expand your existing knowledge and challenge you and equip you with new skills.

Now, here's what we need from you:

- *Confidence and articulation in relation to the analysis of numerical data and statistics*
- *Hands-on with marketing activities*
- *Proven experience in Online marketing*
- *Proven team management experience*
- *Degree in a relevant subject*

As a part of a growing Mobile Fun team: You will receive access to a variety of our excellent benefits which could include;

- up to 25 days holiday,
- pension scheme matched up to 6%,

- Staff discounts (up to 90%)

Please apply by submitting a cover letter and a full CV!
Job Type: Full-time
Salary: £32,500.00 to £40,000.00 /year

Exercise 3. Read the job ads in 2 once again and answer the questions.

- Which job (jobs) requires previous experience in a similar industry?
- Which job (jobs) requires excellent communication skills?
- Which job (jobs) requires conducting laboratory tests?
- Which job (jobs) offers learning and development opportunities?
- Which job (jobs) requires working in a team?
- Which job (jobs) involves dealing with advanced technologies?

ПР02. Тема. Устройство на работу

Современные требования к кандидату при поступлении на работу. Основные документы при принятии на работу.

ПР02. Application for a job

Job requirements. Employment documents.

Exercise 4. Complete the sentences using the following words: *advertisement, applicant, to advertise, to apply for, requirement, position, to require, experience, to provide, curriculum vitae (CV) / resume, application, to assist, reference, referee*

1. This company is looking for a person for the ….....of a Sales Manager who will …... a Managing Director.
2. They placed an ….... in the local newspaper two days ago.
3. Their main ….... are experience and communicative skills. They also ….... that an applicant should be self-disciplined.
4. ….. should have a 3 years' ….... .
5. You must have two ….. from your previous work and give names of your …..... .
6. Candidates should …..... their …..... and send …..... to the address given in the newspaper where the company …..... its products.

Exercise 5. Read Josh Reed's covering letter and complete it with the words and phrases from the box**:** *abilities experience am keen position knowledge skills*

Josh Reed
75 Berry St
Summerville QLD 4536
T: (07) 8222 1111
E: j.reed@email.com

Rachel Forrester
HR Manager
Brighton Mining
Green Plains NSW 2008
February 3, 2016

Dear Ms Forrester

Re: Mechanical Engineer Position

I am writing to apply for the _____ of Mechanical Engineer as recently advertised on SEEK.com.au.

I am a highly motivated Mechanical Engineer with a Bachelor of Engineering (Mechanical Major) and three years of practical on-site mining _____. I am very interested in joining the

engineering team at Brighton Mining given your reputation for world-leading innovation in open cut and underground mining.

In my current position as a mechanical engineer at Newcrest Mining I have developed key project management _____ and the ability to improve communication with the broader project team.

I have a thorough _____ of the processes of open cut and underground mining. I possess excellent interpersonal and communication skills and my multitasking _____ are advanced.

Given my on-site experience I am accustomed to operating in a FIFO environment and working hard to keep projects running on time and within budget. I _____ to employ my skills and enthusiasm as an integral part of your team and I look forward to being able to discuss this position with you further.

Yours sincerely,
Josh Reed

Exercise 6. Read Carmen Frazier's CV and answer the questions.

1. What are the main parts of this CV?

_____

2. What have you learned about Frazier's education?

_____

3. What are her professional abilities?

_____

4. Does Frazier speak any foreign languages?

_____

5. Do you think she is a good candidate for the company Case Consultants? Why/ why not?

_____

PERSONAL INFORMATION
Carmen Frazier
Date of Birth: May 6, 1979
Citizenship: American

PROFILE
• Extensive knowledge in Building Engineering
• Strong technical skills in AutoCad applications
• MS Windows, Mac OS, Archicad, CorelDraw, Sage
• Experienced working in large scale industries

EDUCATION
M.S. in Structural Engineering, 2006, Drexel University, Philadelphia, PA
B.S. in Civil Engineering, 2002, Drexel University, Philadelphia, PA

EMPLOYMENT HISTORY
Chief, Civil Engineer, 2007 - Present

RESPONSIBILITIES:
• Provided cost-effective solutions to recurring construction problems.
• Monitored the status of government projects and ensured compliance with civil engineering standards.

SKILLS
Adept with engineering tools and techniques
Extensive experience in residential, commercial and industrial projects
Certificate in AutoCAD

Exercise 7 Fill the gaps below with the correct present tense (Present Simple or Present Continuous)

Work

a. What _____ (you work on) at the moment?
b. What company/ division/ department/ section/ team _____ (you work) for?
c. What _____ (you do)?
d. What _____ (your company/ division/ department/ section/ team do)?
e. What kind of company _____ (you work) for?
f. Who _____ (you work) for?

English at work

a. _____ (you in the middle of write) anything in English now?
b. _____ (you ever make) English phone calls?
c. _____ (you find) it difficult to write in English?
d. _____ (you have problems) with teleconferences in English?
e. _____ (you normally use) automatic translation?
f. _____(you often write) in English?
g. How many English emails _____ (probably wait) for you right now?
h. What _____ (you find difficult) difficult about English language meetings?
i. As we speak, _____ (you think) in English?

English outside work

a. _____ (you usually watch) movies with English subtitles, with Japanese subtitles or with no subtitles?
b. How often _____ (you socialise) with English speakers?

Language learning

a. _____ (you take) any other English classes now?
b. _____ (you study) on the train on the way home in the evening?
c. _____ (you listen) to anything in English?
d. _____ (you listen) to NHK English radio in the morning?
e. How _____ (you learn) vocabulary?
f. How much English study _____ (you typically do)?
g. _____ (your English improve)?
h. What new English language apps _____ (you use)?

Exercise 8 Fill the gaps in the questions about your company and job (Present Simple and Continuous)

Your job

"What's your job?"
"I _____ [1](work) in the marketing department of a large insurance company."
"How's work?"
"Pretty busy. I _____ [2](get) ready to move to our New York office."
"What _____ [3](you do)?"
"I'm an accountant."
"What _____ [4](you do) here?"
"I'm giving a presentation on our new product range."
"What exactly _____ [5](you do)?"
"I negotiate deals with new suppliers."
"What _____ [6](you work on)?"
"I'm hiring staff for a new branch in Kobe.
"How's your project going?"
"Not very well. We _____ [7](still work) on the initial plan."
"Do you often go abroad on business?"
"No, very rarely, unfortunately. I mainly _____ [8](travel) around Japan."

ПР03. Тема. Компании

Структура компании, названия отделов**.**

ПР03. Companies

Company structure, department names

Exercise 1. Study the information about the people and fill in the gaps below.

'I'm Robert. ***I am responsible for*** *the day-to-day running of the business. I represent the company in the business world'*
Hi! I'm Marina. ***My job is*** *to make sure that the company is producing what people want to buy.'*
'My name is Peter. I am responsible for the entire company when Robert is away on business. '
'Hi! I'm Cecily. ***I deal with*** *personnel matters and recruitment.* ***I also do with issues*** *of staff welfare.'*
' I'm John. ***My area of responsibility is*** *financial issues and money planning'*
'Hi! My name is Liza. ***I do*** *the bookkeeping and the payroll.'*
'Hi! I'm Jake. ***I lead*** *the team which makes our products.'*
' My name is Sonya. I deal with developing and testing our new products'
' I'm Ben. ***I'm in charge of*** *people who sell our products.'*

(1)…… Bradford: Managing Director

(2) ……. Thomson: Assistant Managing Director

(3) …… Gates: Sales Director

(4) …… Johnson: Finance Director

(5) ……. White: Marketing Director

(6) …….. Brown: R&D Manager

(7) …….. Tales: HR Director

(8) …….. Smith: Production Manager

(9) …….. Bay: Accountant

Exercise 2. Look at the list of departments in a company (a-h) and read people's situations (1-6) below. Decide which department each person should ask to speak to when phoning the company. There are more departments than you need.

Names of departments:

Human Resources

Sales

Production

Accounts

Quality Control

Research and Development (R&D)

Technical Support

1 Mr. Mitchell is a marketing executive who has received several complaints from customers about faulty goods.

2 Mr. Davies is a consultant who thinks he has not been paid for an invoice.

3 Mr. Finer has just received the results of the laboratory tests on a possible new product

4 Ms. Smith is a sales executive who is interested in working for the company.

5 Ms. Evans works in the company as a secretary and she has a problem with her computer.

6 Mr. Martins is a retailer who is interested in stocking the company's products.

Exercise 3. Match the words (1-10) to their definitions (A-J)**.**
1 A.G.M.[UK]
2 executive officer[US]
3 board of directors
4 chairman[UK]
5 reception
6 organisation chart
7 shareholder
8 vice president[US]
9 headquarters
10 manager
11 managing director[UK]

A any of several executive officers, each responsible for a separate division
B the place where visitors and clients report on arrival at a company
C person who heads a Board of Directors; head of a company; chairperson
D Annual General Meeting of a company's shareholders
E person who holds or owns shares in or a part of a company or corporation
F group of people chosen to establish policy for and control a company
G person managing the affairs of a corporation - chief executive officer
H a company's principal or main office or centre of control
I person responsible for day-to-day running of a dept.; executive officer[US]
J senior director after the chairman responsible for day-to-day direction
K a table or plan showing a company's structure graphically

Exercise 4. Read the dialogues and fill in the missing phrases.

1) *annual turnover, employ, technical people, workforce*
A: How many people does your company [1]_____?
B: We have sixty employees. We have about forty factory workers and [2]_____ and the rest are admin and sales staff. We started off with only ten people so our [3] _____ has grown a lot. What's your [4]_____?
A: It was over 2 million euro last year.

2) *do manufacture help*
A: What does your company [5]_____?
B: We [6]_____and sell fire prevention and fire control equipment
A: What do your products do?
B: They [7] _____prevent fires and help suppress fires once they have started.

3) provide services guards
A: What does your company do?
B: We [8]_____security services to large businesses and hotels.
A: What kind of[9] _____do you provide?
B: We provide security [10]_____, CCTV and 24-hour monitoring.

ПР04. Тема. Компании.

Характеристика обязанностей работников отделов, описание работы компании.

ПР04. Companies

Description of the staff responsibilities and company activities.

Exercise 5. What do these companies do? Make sentences about the companies' activities, using the words below.

Example: Microsoft designs and sells IT software.

SONY
AUCHAN
CITYBANK
ADIDAS
APPLE
MICROSOFT
TOYOTA
COCA-COLA

Verbs: create, design, develop, manufacture, sell, market, offer, provide

Word combinations: banking services, cars, clothing, electronic goods, food and drinks, Internet services, IT software

Exercise 6. These sentences describe two companies, Autotech and Green Fingers. Choose pairs of sentences, which describe similar things and match them with the correct company. Underline the verbs which mean the same things.

GREEN FINGERS

A small garden-products company

1 George and Tames Hawkins began Green Fingers in the 1920s.
2
3
4
5
6

AUTOTECH

A large car-parts company

1 John Smith started Autotech in 1960.
2
3
4
5
6

John Smith started Autotech in 1960.
It has a workforce of 2,500.
Autotech exports to over 12 countries.
It manufactures car parts.
It introduces one or two new components each year.
It employs about 35 people.
Green Fingers sells some of its products abroad.
It makes garden products.
George and James Hawkins began Green Fingers in the 1920s.
Green Fingers supplies the gardening industry.
It launches 12 new products a year.
Autotech provides components for the car industry.

Exercise 7. Complete the text with the correct form of verbs: *launch have begin manufacture provide export*

Sonara_____ in 1972 near Turin. Today, it _____ mainly aircraft engines, but in the 1970s it also _____ the car industry with components. It _____ a workforce of 2,000. Sonara _____75 % of its engines to other European countries. Last month, it _____a new type of engine which burns 15 % less fuel than other models.

Exercise 8. Match the sentence halves.
1. Panetti employs over 3,500 people,
2. It introduced four new products last year
3. It makes bread and,
4. Panetti only supplies its own shops;
5. It doesn't sell any of its products abroad,

a) but it plans to expand into France.
b) including 1,400 in its own retail outlets.
c) including sandwiches and pies.
d) many other bakery products.
e) it does not make products for anyone else.

Exercise 9. Match these words and phrases from exercise 7 with a word or phrase from exercise 8 that has a similar meaning**.**
1 manufactures makes
2 provided
3 has a workforce of
4 export
5 launched

Exercise 10. Read the text and name the most important inventions of the company.

WHAT IS SO GOOD ABOUT SONY CORPORATION**?**

Since 1946 Sony has been committed to bringing the world the best in technology. It is a leader in consumer electronics. This innovation comes with Sony not just simply recreating products, but actually inventing new technology.
In May 1960 - Sony was the first company to create the world's first direct-view portable television. The first model was developed based on Sony's extensive experience in radio technology. This device opened the door to personal television use.

Sony made further advancements and in 1962 produced the world's lightest and smallest all transistor television (the TV5-303). The advancements however did not stop there. In 1963 the world's first compact transistor VTR, the PV-100, launched. And in 1965 the world's first home-use open-reel VTR, the CV-2000, launched, paving the way to allow people to record and playback over 1 hour of video.
There are so many new offerings that Sony has brought us, which we did not have before. The Walkman is probably one of the greatest inventions from Sony. Sony also brought us the world's first CD player also!

Sony is an electronics manufacturer that has brought us many great inventions, which have benefited humanity. Sony continues to innovate and be a world class leader in reliable electronics, with that elegant and ergonomic Sony style.

Make sentences about the most important events in the history of the company using the Sony timeline:

1946 - _____
1960 - _____
1962 - _____
1963 - _____
1965 - _____

Exercise 11. Put the verbs into Past Simple

1. I (meet) Managing Director at the airport at 7.00 in the morning.
2. We (take) our visitors to the plant.
3. On Wednesday I (fly) to Moscow for a conference.
4. My presentation (not go) very well.
5. Yesterday evening I (write) a proposal for an American company.

Exercise 12. Make questions to the answers.

1. What _____?
They arrived at 10 o'clock.
2. Why _____the meeting?
I left the meeting because I had an urgent phone call.
3. When _____ the company?
She joined the company in 2015.
4. Who _____ at the conference?
We saw our colleagues from Moscow.
5. How long _____ with the visitors?
I spent 2 days with them.

ПР05. Тема. Инновации в производственной сфере.

Описание товаров, их особенностей.

ПР05. Innovation in industry.

Description of goods and their characteristics.

Exercise 1. Read the text and decide whether the statements are true or false.

1. Product is anything which is produced by people or machines and can be sold.
2. Products are always physical. You can touch or smell them.
3. Products can be delivered in the form of services or ideas.
4. Products cannot be virtual.

1. What is a product?

In general, a product is defined as a "thing produced by labor or effort" or the "result of an act or a process".

In marketing, a product is anything that can be offered to a market that might satisfy a want or need. In retail, products are called merchandise. In manufacturing, products are purchased as raw materials and sold as finished goods. Commodities are usually raw materials such as metals and agricultural products, but the term can also refer to anything widely available in the open market.

2. Goods, services, or ideas

Goods are a physical product capable of being delivered to a purchaser and involve the transfer of ownership from seller to customer.

A service is a non-material action resulting in a measurable change of state for the purchaser caused by the provider.

Ideas (intellectual property) are any creation of the intellect that has commercial value, but is sold or traded only as an idea, and not as a resulting service or good. This includes copyrighted property such as literary or artistic works, and ideational property, such as patents, appellations of origin, business methods, and industrial processes.

3. Product classification: tangible or intangible

A product can be classified as tangible or intangible.

A tangible product is a physical object that can be perceived by touch such as a building, vehicle, or gadget. Most goods are tangible products. For example, a soccer ball is a tangible product.

An intangible product is a product that can only be perceived indirectly such as an insurance policy. Intangible data products can further be classified into virtual digital goods ("VDG"), which are virtually located on a computer OS and accessible to users as conventional file types, such as JPG and MP3 files, and real digital goods (“RDG”), such as as 3-D objects or presentational items.

Exercise 2. Fill in the missing words in the sentences: *practical, economical, functional, user-friendly, well-designed*

1. The new air conditioning system in our office is much cheaper than the old one. It is more ….

2. It took us quite a long time designing the new office furniture. Now it is very ….

3. The operating system on my office computer is easy to use. It is very ….

4. My new car is much easier to park. It is very … for driving in the city center.

5. The new office equipment is exactly what we needed. It is very ….

Exercise 3. Read the tips on writing a good product description and match the heading to the paragraphs.

Make it Easy to Scan
Focus on the Product Benefits
Know Who Your Target Audience is
Use Power Words That Sell
Tell the Full Story

1. _____

The first step to writing product descriptions is to define your target audience.
You want to be able to define which features would be of most interest to your potential buyers.
As you are writing your product description, keep these questions in mind:

- How did this person arrive to your page?
- What are his or her interests, generally?
- Why would this person be interested in your Shopify store, specifically?
- How would this person describe the product to a friend?
- What features or benefits would interest this person the most?

By keeping these questions in mind as you write your product copy, you will be better able to write a product description that sells.

2. **_____**

As a business owner, you are understandably excited to share all of the qualities of your products. You want to show that your product has the best features and most unique specs.
The buyer, however, is not necessarily interested in the mundane features of the product. Instead, they want to know how it can benefit them.
A product feature is a factual statement about the product that provides technical information. A product benefit, on the other hand, tells **how the product can improve the buyer’s life**.

3. **_____**

A good product description should give all relevant details, convince the buyer of its benefits, and pack an emotional punch.
Emotions influence buyer behavior, so your product description is the perfect place to elicit emotions.
How do you do this?

By filling in any gaps that potential buyers may have about the product.

4 **_____**

There are certain words and phrases that naturally elicit an emotional response in humans. Luckily for Shopify store owners, this also increases sales.

By being mindful of these words and phrases, you can more easily convince your customers to take the leap and make the purchase.

5 **_____**

People have short attention spans and read only about 16% of what's on the page. So you have to make your descriptions super scannable.

As in, the buyer is able to find exactly the information he or she wants without wasting time looking through other pieces of information.

Make your product descriptions easy to scan by including bullet points, short paragraphs made up of just a few sentences each, lots of white space, and different size fonts.

Exercise 4. Describe the company product (service) and activities using the information below.

1 Company: Translations R Us
Product: Electronic Translation Dictionary
What it does: Translates 45 languages into English, Chinese, Japanese and Spanish

2 Company: Colorado Hot Air, Inc.
Product: Hot Air Balloons
What it does: Takes people on flights up to 10,000 feet (about 3100 meters).

3 Company: Yoga Sensation
Service: Yoga Instruction
What is provided: Yoga classes at all levels worldwide for corporations and resorts.

4 Company: The Pool Doctor
Service: Swimming Pool Maintenance
What is provided: Cleaning, servicing and repair of swimming pools. Everything from small private pools to huge resort and country club pools.

Exercise 5. Match the adjectives with the opposites above.

1. unreliable
2. large
3. heavy
4. unpopular
5. short

a cheap
b unattractive
c fast
d boring
e bad

Fill in the gaps in the sentences.

1. The consumers like *Margin*, it is very _____
2. But they think they have to pay a lot, because the product is _____.
3. Most people say the bottle is nice, it looks _____.
4. Some people think the packages is not _____, it is heavy.
5. The product meets customers' expectations. It is _____.

Exercise 6. Divide the words into three groups: *metal big square tiny glass rectangular wood huge triangular*

Shape:
Material:
Size:

ПР06. Тема. Инновации в производственной сфере.

Анализ рыночной продукции и конкурентоспособности товаров.

ПР06. Innovation in industry.

Product analysis and the competitiveness of goods.

Exercise 7. Read the text about product development.

The development stages of a new product

Before a product can embark on its journey through the four *product life cycle stages*, it has *to be developed*. New product development is typically a huge part of any manufacturing process. Most organizations realize that all products have a limited *lifespan*, and so new products need to be developed to replace them and keep the company in business. Just as the product *life cycle* has various stages, new product development is also broken down into a number of specific phases. Developing a new product involves a number of stages which typically center on the following key areas:

The original idea: Every product has to start with an original idea. In some cases, this might be fairly simple, basing the new product on something similar that already exists. In other cases, it may be something revolutionary and unique, which may mean *the idea generation* part of the process is much more involved.

Market Research: An organization may have plenty of ideas for a new product, but once it has selected the best of them, the next step is to start re-searching the market. This enables them to see if there's likely to be *a demand* for this type of product, and also what specific features need to be developed *in order to* best *meet the needs* of prospective customers.

Design and Development of the Product: The next stage is *the design* and *development* of the product. Prototypes may be modified through various design and manufacturing stages in order to come up with a finished product that consumers will want to buy.

Product Trials (Testing): Before most products *are launched* and the manufacturer spends a large amount of money on *production* and *promotion*, most companies will test their new product with a small group of actual consumers. This helps to make sure that they have *a viable product* that will be *profitable*, and that there are no changes that need to be made before it's launched.

Analysis: Looking at the feedback from consumer testing enables the manufacturer to make any necessary changes to the product, and also decide how they are going to launch it to the market. With information from real consumers, they will be able to make a number of *strategic decisions* that will be crucial to the product's success, including what price to sell at and how the product will be marketed.

Launch of the Product (Introduction): Finally, when a product has made it all the way through the new product development stage, the only thing left to do is *launch* it to the market. Once this is done, good product life cycle management will ensure the manufacturer *makes the most of all their effort* and investment.

Thousands of new products go on sale every year, and manufacturers in-vest a lot of time, effort and money in trying to make sure that any new products they launch will be a success. Creating a profitable product isn't just about getting each of the stages of new product development right, it's also about managing the product once it's been launched and then throughout its lifetime.

Exercise 8. Fill in the gaps using the words in italics from the text.

1. Without a proper … it is impossible to … a product successfully.

2. It is essential for any product to be ….

3. Companies spend a huge amount of money on … … of a new product.

4. Prototypes may be modified through various … …stages.

5. Any new product has to start with …

Exercise 9. Answer the questions:

1. What is the most important stage in the development of a new product?

2. How many stages are there in the product development process?

3. On which stage the companies can realize that their new product may be of demand among prospective customers?

4. How can companies know that their new product will be profitable?

5. How can you explain the statement "to create a profitable product"?

Exercise 10. Choose any product and prepare to give a brief product review.

Your product review should include:
• a description of the product
• the way the product is used
• the cost of the product
• a comparison with other, similar products
• a recommendation to buy or not buy the product
• a reason for your recommendation
• a rating on a scale of 1-5

PRODUCT REVIEW

PRODUCT:
PURPOSE:
AUDIENCE:
USE:
COST:
COMPETITORS:
RECOMMENDATION:
REASON:
RANKNG

ПР07. Тема. Дизайн и спецификация товара.

Описание дизайна и спецификации товара.

ПР07. Design and product specification.
Description of design and product specifications.

Exercise 1. Read the text and correct wrong statements:

1. Design is something related to fashion and style.
2. A good design begins with a good idea.
3. Design is not connected with people's quality of life.
4. Designers try to realize only sensible ideas.
5. Scientists and manufacturers don't have anything in common with designers.

WHAT IS DESIGN?

Design is everywhere. The word "design" means different things to different people. One definition given by designer Richard Seymour is 'making things better for people'. The design activity is focused first and foremost on human behaviour and quality of life. It transfers any idea into a blueprint for something useful, whether it's a car, a building, a graphic, a service or a process.

Scientists can invent technologies, manufacturers can make products, engineers can make them function and marketers can sell them, but only designers can combine insight into all these things and turn a concept into something that's desirable, viable, commercially successful and adds value to people's lives.

A good design begins with the needs of the user. No design, no matter how beautiful and ingenious, is any good if it doesn't fulfill a user need. Finding out what the customer wants is the first stage of what designers do. The designer then builds on the results of that inquiry with a mixture of creativity and commercial insight. Different designers use different methods-combining market research, user testing, prototyping and trend analysis. These methods lead to innovative products and services. Designers learn that ideas that may seem strange are worth exploring and that the 'common-sense' solution is not always the right one.

Exercise 2. Choose one of the products and write its description using the words and the outline

Words: *elegant functional futuristic handmade innovative retro stylish simple mass-produced traditional up-to-date streamlined ergonomic durable easy to use*

Useful phrases:

Visual appeal :	It looks like (a design from the 2010s)
Material**:**	It has a metal/wooden/golden (top/side/base)
Features:	It has several (qualities / special features)....
	It has a unique feature...
	One of its weak/strong points is...(that it is very difficult/easy to use because...)
Use**:**	It is designed for (opening/keeping)...
	It is used for.....

ПР08. Тема. Дизайн и спецификация товара.

Характеристика и сравнение дизайна различных товаров, представленных на современном рынке.

ПР08. Design and product specification.

Description and comparison of design of various products in the modern market

Exercise 3. Complete this presentation introduction with the words: *talk about look at points of view questions brief finally hear act as go along*

Good afternoon and thank you for making the effort to be here with us today. My name's John Smith and I'm responsible for marketing. What I'd like to do today is 1 _____ our recent product promotion campaign. This 2 _____ talk will hopefully 3 _____ a springboard for discussion. I'm going to 4 _____ the marketing campaign from three 5 _____ : firstly, the customers; secondly, the financial institutions; and 6 _____, the shareholders. If you have any 7 _____, just interrupt me as I 8 _____ . Your point of view may well be different, and we'd like to 9 _____ from you.

Exercise 4. Complete the following presentation common expressions and phrases:

i. Today we will _____ at …
ii. I'd like to _____ by …
iii. This _____ us to the next _____ …
iv. Let me _____ you an example …
v. A case in _____ is …
vi. On _____ whole …
vii. Let me end by _____ …

Exercise 5. Choose one of the products given below and complete the product design specification. Invent any additional information you need**.**

PRODUCT DESIGN SPECIFICATION

Product name:
Product description:
Product performance:
Ergonomics features:
Dimensions:
Design:
Safety:

Product 1:

OUTDOOR HEATER
Function: to heat the air outside a building
- gives a lot of heat for 26 hours
- powered by propane gas
- easy to regulate the heat
- light and easy to move
- easy to clean
- can be used in all weathers

Product 2:

BABY MONITOR
Function: to check the health of a sleeping baby
- works up to a 100-metre range
- powered by mains or battery
- low battery indicator
- adjustable volume
- has a belt clip and also a stand
- easy to use and very light

Exercise 6. Read a report on two mobile phones and answer the questions.

1. What mobile phones are compared?
2. Do these phones have any similar features? What are they?
3. What are their peculiarities which differ these models from each other?
4. How many parts does the report consist of? What are they?

Two models of mobile phones the Nokia 6230i and the Samsung SGH-D500 are compared in this report. We have studied their features and found out that they have some similarities and differences.

First, the Samsung SGH-D500 is a little cheaper than the Nokia 6230i. It is very stylish and one of the most popular models on the market. Its features include a 1.3 Mp camera and music player. However, it does not have a memory card so you can store only a limited amount of music tracks and photos on this phone.

Second, the Nokia 6230i is small and light. It weighs only 99g and is very functional. It has a 1.3 Mp camera like the Samsung phone, but it also has a memory card. The panel of the camera can be replaced easily. In addition, covers are available in five different colours.

In conclusion, test results showed that the Samsung SGH-D500 was not very good in capturing movement. Despite this fact both the Nokia and Samsung models are very good buys.

Exercise 7. Write a report on two hand dryers.
AYT
* uses principle of evaporation
* may take up to 44 seconds to dry hands
* warm air flows out at moderate speed
* most of water removed by slow evaporation
* bacteria are not filtered out of washroom air
* unfiltered air blows around the room
* machine starts when user presses start button
* user usually has to press start button several times
* user often walks out with damp hands, wiping on clothes to dry

Dyson
* uses process of scraping water off hands like windscreen wiper
* dries hands in 10-12 seconds
* cold air forced out at high speed
* long ultra-thin apertures run along two blades
* filter removes over 99.9% of bacteria from air* filtered air stays within the machine, not around room
* uses 80% less energy than conventional dryer
* hands completely dry after use
* turns on automatically when user inserts hands

ПР11. Участие в научной конференции. Описание форм участия в научных конференциях.
ПР11. Participating in a research conference. Forms of participation in scientific conferences.

Exercise 1. Read the following five extracts and then write which form (or forms) of communication (*an academic journal, a popular science magazine, a conference, a popular science book, an online forum or science blog, a newspaper*) each one comes from.

1) ... more people were pain-free when using the handheld device than those who had used an identical dummy device. Although the study by Lipton *et al.* (2010) has reliable results, there are some points to consider when putting these findings into context. Importantly, the results will need to be verified in larger trials that directly compare ...

2) Tea and coffee drinkers have a lower risk of developing type 2 diabetes, a large body of evidence shows. And the protection may not be down to caffeine since decaf coffee has the greatest effect, say researchers in *Archives of Internal Medicine.* They looked at...

3) ... can be rapidly generated by lentivirus-mediated transgenesis. RNAi also holds great promise as a novel therapeutic approach. This report provides an insight into the current gene silencing techniques in mammalian systems.

4) Hi! Has anyone had any experiences with nanoparticles sticking to glassware :-(? If so, does anyone know if there's a suitable silylation protocol to pre-treat the glassware to do something about this annoying non-specific adsorption? Thanks!

5) Animal and *in vitro* studies suggest that aspirin may inhibit breast cancer metastasis. We studied whether aspirin use among women with breast cancer decreased their risk of death from breast cancer. This was a prospective observational study based on ...

ПР12. Участие в научной конференции. Проведение научной конференции.

ПР12. Participating in a research conference. Conducting a scientific conference.

Exercise 2. Look at the online poster advertising a conference and complete the phrases below in accordance with the poster using the given words.

Keynote speakers
• Zoltan Szabo
European Institute of Malaria Research (EIMR)
• Miremba Kabasomi
Makarere University, Kampala, Uganda
Preliminary Programme
A list of other invited speakers and preliminary session topics is currently being developed by the Conference Chair and will be announced in due course. Please check back for updates.
For further information about us see www.eimr.org
Online registration only
www.eimr.org/con7/registration
Registration is on a strictly first- come, first-served basis.
Application deadlines
1 April for abstract or poster presentation submissions
7 May for attendees
Registration fees
Academia – €450, Students – €350, Commercial/Industry – €650

basis course deadline keynote preliminary presentation registration (x2) strictly

1. application _____
2. on a _____ first-come, first-served_____
3. _____ speakers
4. online_____only
5. poster_____
6. _____ programme
7. _____ fees
8. to _____ an abstract

9. in due _____
10. check back for _____

ПР15. Презентация исследовательского проекта. Структура презентации в целом и исследовательского проекта, в частности.
ПР15. Presentation of a research project. The structure of the presentation as a whole and the research project in particular.

Exercise 1. Write 'who', 'why', 'what' or 'how' next to each phrase. Check any vocabulary you don't know.

1. On behalf of Mr Keane, may I welcome you to Jackson Inc. My name's Jo Black and I'm responsible for ...
2. My purpose today is to ...
3. I'm going to develop three main points. First, ... Second, ... Third, ...
4. Let me introduce myself. I am ... I am a ...
5. I'll pass round copies of my slides so you can make notes as I go through the presentation.
6. Before I continue, let me tell you something about myself.
7. Today 1 would like to give you a general overview of...
8. I've divided my presentation into three main points. I would like to begin with ...
9. So, I'll be addressing three main points and the first one is going to be ... The second point will be ... And finally the last point is ...
10. I'm going to outline three proposals. Firstly, I'll ... Then, I'd like to ... and finally ...
11. We can take two or three questions at the end of each point.
12. You don't need to take notes as we'll be handing out presentation booklets.

Exercise 2. Match each pair of phrases (1 – 8) from to their correct function (a – f) below. Note that one of the functions may be expressed with three different pairs of phrases.
1. Good afternoon, everybody. / Welcome, ladies and gentlemen.
2. To start, thank you / I'd like to start by thanking you all for coming to my talk today.
3. I'm Milan Poborski and at present / My name is Milan Poborski and I'm a PhD candidate at Northumbria University.
4. I'm going to talk today / My talk today is about my recent research investigating ...
5. I'll begin by explaining / To start with, I'll explain briefly how T-cell responses…
6. After that, I'll / I'll go on to describe the alternative method I have been investigating ...
7. Finally, I will discuss / I'll conclude by discussing why this method could be useful as a way ...
8. I plan to talk for about 40 minutes, leaving plenty of time for /I will talk for about 40 minutes and then I'll answer any questions at the end of my talk.

a. Give instructions for asking questions
b. Greet the audience.
c. Introduce the topic of the presentation
d. Introduce yourself
e. Outline the structure of the presentation.
f. Thank the audience for coming.

ПР16. Презентация исследовательского проекта. Анализ различных проектов и обсуждение их сильных и слабых сторон. Написание теста по пройденному разделу.
ПР16. Presentation of a research project. Analysis of various projects and discussion of their strengths and weaknesses. Test 2.

Exercise 3. Below are *five sentences* from the main part of a presentation. Match the beginnings (1 – 5) to the endings (a – e).

1. A number of potential vaccine types have been developed and
2. As I have already said,
3. As you can see from this image,
4. Let's begin by looking at the size of the malaria problem.
5. That's all I have to say about the vaccine itself,

a. counting IFN-γ secreting cells has been the preferred method to date.
b. using flow cytometry to detect MIG secretion gives us a more accurate way of measuring immune responses.
c. I will be returning to those shortly.
d. Malaria kills over one million people every year in 109 countries.
e. so now I'd like to move on to looking at judging the response of the immune system to the vaccine.

Exercise 4.The underlined phrases in Exercise 3 help speakers to organize their presentation clearly and guide listeners through the information. Write the correct underlined phrase to complete the advice below.
Use:
a. _____ : to introduce a new part of the talk
b. _____ : to conclude one part of the talk and then begin another
c. _____ : to refer back to an earlier part of the talk
d. _____ : to refer forward to a later part of the talk
e. _____ : to refer to a visual aid

Раздел 3. Деловая коммуникация.
MODULE 3. Business Communication

ПР17. Межличностные и межкультурные отношения.
ПР17. Interpersonal and Intercultural Contacts

When you observe people from a certain culture, some characteristics – such as dress and the way people greet each other are easy to see. Others are not so easy.
Culture is sometimes compared to an iceberg, some of which is visible, but much of which is difficult to see, or invisible.

Exercise 1. Look at the list of components of national culture, and place each one in one of the three categories:
A things which you can recognize quite easily
B things which you recognize only when you are very familiar with a culture

1. Beliefs
2. Family values
3. Language

4. Expectations
5. Food
6. Manners
7. Holidays and festivals
8. Rules of conduct
9. Greetings
10. Attitudes to the environment
11. Physical gestures
12. Work ethic
13. Roles of males and females
14. Art and architecture
15. Punctuality
16. Humour

Add any other elements which you think are important in defining a national culture.

ПР18. Межличностные и межкультурные отношения.
ПР18. Interpersonal and Intercultural Contacts

When you meet people for the first time, greet then politely and warmly. Use a mix of questions during the first conversation – try to discover what things you have in common. Be sensitive to the cultural background of the other person during the meeting.

Exercise 2. Nigel Hastings is a director of an intercultural consultancy. Listen to him talking about managing first meeting in different cultures and answer the questions.

1. Why does Nigel say it's important to manage first meetings well when working internationally?
   a) Because it establishes positive relationship with international partners.
   b) Because it helps to learn a foreign language.
2. What does he describe as the purpose of first meetings in the Arab world?
   a) To start talking about business as soon as possible.
   b) To get to know your partner and build trust.
3. When he goes to China, how does he usually start small talk?
   a) He talks about Shanghai.
   b) He talks about a person's roots.
4. Why is asking 'open questions' important?
   a) It's a way to create an emotional connection with your partner.
   b) It's a way to learn about your partner's background.

Audio transcript
Interviewer: How important is it to handle first meetings well?
Nigel: Very important, and breaking the ice can be more difficult due to language differences, cultural unfamiliarity and so on. But when we have to work with people across the globe we don't see very often, it becomes critical to manage that first moment positively, to get the relationship going.
Interviewer: In terms of cultural differences in first meetings, what have you experienced?
Nigel: Well, being British I do a little small talk about the weather, how I travelled to the place, some of the challenges I faced on the way, and a touch of humour but quite quickly getting down to business because I don't want to take up too much of the time of the individual I'm talking to. In other cultural contexts, I've learned you need to take a different approach. For example, working in the Arab world, I think the concept of 'small'talk is less relevant because those opening social moments are critical in a relationship-orientated culture. And showing respect for the local culture,

saying positive things about what you've seen, what you know, why you appreciate where you are, showing and receiving hospitality. There's a slower transition into business, more exchange on the personal level before getting down to the task. And that builds trust. All this can be seen as slow and time-wasting in a UK context.
Interviewer: What kinds of positive things do you say?
Nigel: I think when I go to China I often find myself asking where someone comes from because I know quite a lot about China and it's an opportunity to show a kind of interest which I have for the country and a sensitivity to that person's roots. So I would say something like 'Oh, you come from Shanghai, I've been there a couple of times and I love it there.' This creates a positive impression. And it's true. I do love it there!
Interviewer: Does asking open questions generally work across all cultures to stimulate conversation?
Nigel: I think it definitely does. The open questions beginning with 'what' and 'how' open people up, give you an opportunity to listen to people speak, understand their interests … so questions are very important as a way to create common ground.
Interviewer: Is this one of the secrets of success – creating common ground?
Nigel: I think so. You can ask an open question, listen to the response and then connect your experiences. You have the potential to build common ground also at an emotional level in terms of going through some common challenges. However, you can also ask closed questions to show interest in something around you.

Exercise 3. It is important to find things in common when responding to what people tell you. Match each comment to a response.

Comments

1. I've been to Italy a few times.
2. I studied mechanical engineering at uni.
3. I grew up just outside Madrid.
4. So these are the new offices.
5. Business is a bit challenging at the moment.

Responses

a) I know the city quite well but not the region around it.
b) Really? My brother did something similar and now works in construction.
c) Indeed, but I read that things should be improving by the year end.
d) Impressive. Looks like a nice place to work. Our offices are very different.
e) Have you? Me too. I love the South.

ПР19. Проведение переговоров
ПР19. Negotiations

Exercise 4. Which of the actions a)-g) correspond to the negotiation stages 1-7?

1. build rapport
2. agree on a procedure
3. make proposals and counter-offers
4. probe with questions
5. enter the bargaining zone
6. resolve any areas of conflict
7. conclude the negotiation

a) make concessions
b) find things in common
c) celebrate the deal!

d) state your opening position
e) decide who will speak first
f) clarify anything you don't understand
g) call for a time-out

Exercise 5. In a negotiation each team member must play a specific role. Complete the team roles 1-6 below using appropriate pairs of words a)-f).

1. Decision-maker: …… overall strategy and has the final …… .
2. Facilitator: …… and provides …… of their team's position.
3. Number-cruncher: …… down key figures and does the …… .
4. Chief negotiator: …… the main negotiations and acts as …… .
5. Observer: …… the other team's behaviour and looks for signs of …… .
6. Ideas-generator: …… deadlocks by coming up with creative …… .

a) breaks + solutions
b) monitors + movement
c) formulates + authority
d) notes + calculations
e) conciliates + clarification
f) conducts + spokesperson

ПР20. Проведение переговоров
ПР20. Negotiations

Exercise 6. Match each of the techniques 1-6 from the previous section to a comment a)-f) demonstrating this technique in action.

1 2 3 4 5 6

a) I changed my mind because I believed what he said – he didn't hide anything from me.
b) I felt they really understood my needs so I was happy to accept their proposal.
c) I had to agree. The facts spoke for themselves.
d) She convinced me to join the project because I enjoy working with her.
e) He offered to help me out on my project so I agreed to support him at the meeting.
f) I supported her idea because I could see she really believed in it 100%.

ПР21. Контракты и соглашения
ПР21. Contracts and Agreements

Exercise 7. Match the types of contract 1-6 to their definitions a)-f).

1. a lease
2. an employment contract
3. a contract of sale
4. an insurance policy
5. a credit agreement
6. a software license

a) a contract between a seller (or vendor) and a buyer (or purchaser)
b) a contract between an employer and employee
c) a contract between an insurance company and a person who pays for the insurance
d) a contract which allows one party (the tenant) to use the land or property of the other party (the landlord) for a specified period of time
e) a contract which allows someone to use a computer program

f) a legal contract in which a bank agrees to loan a customer a certain amount of money for a specified amount of time

ПР22. Контракты и соглашения
ПР22. Contracts and Agreements

Exercise 8. Complete the phrases below with the following verbs:
breach complete draw up extend renew sign terminate
1. sign a contract ⇒ to put your signature on a contract to show that you agree to it
2. _____ a contract ⇒ to prepare/write a contract
3. _____ a contract ⇒ to break one or more of the terms of the contract
4. _____ a contract ⇒ to end a contract before the official end date
5. _____ a contract ⇒ to fulfil all the terms of the contract
6. _____ a contract ⇒ to make a contract continue for a longer time
7. _____ a contract ⇒ to sign a new contract (when the previous contract has finished)

Now complete the sentences below:
a) Our lease expires at the end of the year, but we were able to _____ the contract by another year.
b) We cannot sell the goods to a third party. If we do this, we will _____ the contract.
c) We were very happy with the service so after our contract ended, we decided to _____ it.
d) The construction firm couldn't _____ the contract because of the bad weather.
e) We can _____ the contract if we notify the other party one month in advance.
f) In order to make everything legal, the lawyers decided to _____a contract.

ПР13. Принципы составления и написания научной статьи. Анализ отрывков из научных статей по различным темам. Введение и отработка новой лексики, клише.
ПР13. Writing a research paper. Analysis of extracts from scientific articles on various topics. Introduction of a new vocabulary.

Exercise 1. Tony is doing research into the panspermia hypothesis as part of a Master's degree in astrobiology. He has been investigating whether it is possible for bacteria and microorganisms to survive in an environment as harsh as the surface of Mars. He has been advised to organize the text of his introduction around five key questions. Match the beginnings to the endings of the questions.

1. What was I	a. approach the problem?
2. Why was it	b. expect to know after doing the research?
3. What was already	c. important?
4. What did I	d. investigating?
5. How did I	e. known about the subject of my research?

Exercise 2. Read five extracts from the introduction to Tony's paper. Which question from Exercise 7 is each extract answering? Write the questions above the extracts.

1._____
Such an extreme environment was thought to be uninhabitable, but microbial ecology studies reported the presence of microorganisms (Amaral-Zettler et al., 2002). Could the surface composition of Mars protect life against radiation?
2._____

A number of studies have investigated different extreme Martian surface conditions on terrestrial microorganisms. Nicholson and Schuerger (2005) reported that die spores of Bacillus subtilis were able to survive for 19 days under Mars atmospheric pressure and composition. Saffary et al. (2002), however, found that survival decreased due to ...

3. _____

Potential habitability in the subsurface would increase if the overlaying material did play a protective role.

4._____

For many years now, scientists have speculated about the possibility of life on Mars (Klein et al., 1976; McKay, 1997). The discovery of liquid water on Mars would increase its habitability ...

5._____

We report here on our studies of protection by Rio Tinto Basin iron oxides and hydroxides on two microorganisms, AcidithiobaciUiis ferrooxidans and Deinococcus radiodurans, under simulated Mars surface conditions.

ПР14. Принципы составления и написания научной статьи. Анализ различных частей научной статьи и их особенностей.
ПР14. Writing a research paper. Analysis of various sections of a scientific article.

Exercise 3. Read an extract from the introduction of a paper about the ability of lichens and microbes to survive in deep space. Put the verbs into the correct form.

Recent advances in space technology (1)_____(provide) the possibility of studying the survival of different microorganisms in the harsh environment of space *(Demets* et al., *2005; Baglioni* et al., 2007*).* So far, lichens (2) _____(be) the only organisms able to survive exposure to such extreme conditions *(Sancho* et al., *2007; de los Rios* et al., *2010).*

It is believed that, if sufficiently protected by meteorite-like material, microorganisms may also survive the journey through space. However, Brandstatter *et al.* (2008) (3) _____ _____(report) that microorganisms embedded in 2 cm thick rocks on the outer surface of a re-entry capsule, simulating the entry of a meteorite, (4)_____ (not survive).

The aim of this work (5)_____(be) to obtain further information on the resistance of rock-colonising microbial communities and lichens to outer space conditions, during the Biopan-6 flight of ESA on board a Russian Foton satellite.

Exercise 4. Complete the following summary on variables using the given words.

affects collecting controlled data dependent independent

How much a variable (1)_____a relationship can be discovered by (2)_____experimental (3)_____on changes to the relationship as the variable is changed. In an experiment, there will be: one (4)_____variable – this is the feature you are measuring; one or more (5)_____ variables – these are the variables which you change; one or more (6)_____variables – these are not being tested and so they stay the same.

Exercise 5. Complete the lines below using the extract from the following research paper to help you.

A promising candidate among the different adsorbent materials are activated carbons. Through activation, highly porous materials can be prepared. Due to their high porosity, activated carbon

materials are able to adsorb large amounts of hydrogen. Following adsorption, hydrogen molecules can be found at two possible locations: (1) on the surface of the adsorbent, or (2) as a compressed gas in the void space between adsorbent particles. (adapted from *Konowsky et al.* 2009)

Noun	**Verb**	**Adjective**
1. compression	2. compress	3._____
4._____	5. adsorb	6._____
7._____	8. activate	9._____
10._____		11._____

Exercise 6. **The gapped words below all describe physical or chemical properties of substances. The meaning of each word is given on the right. Complete the words with the correct vowels (a, e, i, o, u)**

1. br_ttl_n_ss	how easily something can be broken
2. c_p_c_t_nc_	how well something holds an electrical charge
3. c_nc_ntr_t n	how much of one substance is found in another
4. c_nd_ct_v_ty	how well something allows heat or electricity to go
5. d_ns_ty	how much mass a given volume of a substance has
6. fl_mm_b_l_ty	how easily something burns
7. l_m_n_nc_	how much light passes through or comes from a substance
8. m_ss	how much matter is in a solid object or in any volume of
9. p_rm__b_l_ty	how easily gases or liquids go through a substance
10. p_r_s_ty	how many small holes are in a substance
11. r__ct_v_ty	how easily a chemical substance reacts
12. s_l_b_l_ty	how easily something can be dissolved to form a solution
13. v_l_c_ty	how quickly an object is travelling
14. v_sc_s_ty	how thick a liquid is
15. v_l_m_	how much space is contained within an object or solid

Exercise 7. Complete the paragraphs from the results section of a paper using the following words and phrases in the box.

as can be seen in considerably contrast to noticeably thicker resulted in a longer while

During the rapid heating, the Ni near the Ni/SiC interface reacted with the SiC, which resulted in carbon atoms moving into the Ni. The carbon atoms then separated onto the surface of the Ni during the cooling procedure, forming graphene layers (1)_____Fig. lb. In (2) _____the graphene generated using single-crystalline SiC, the graphene synthesised by this process is (3)_____easier to remove from the SiC surface.

A slower heating rate (4) _____ process. As shown in Fig. 4, more carbon atoms were released into the Ni in a long process. Higher carbon concentration in the Ni produced a (5)_____carbon nanofilm on the Ni surface, (6)_____a lower carbon concentration reduced the thickness of the carbon nanofilm and formed graphene.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.05 Технологическое предпринимательство

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Полная версия пособия доступна по ссылке

Путилов, А. В. Коммерциализация технологий и промышленные инновации : учебное пособие / А. В. Путилов, Ю. В. Черняховская. — Санкт-Петербург : Лань, 2021. — 324 с. — ISBN 978-5-8114-3371-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/169312 (дата обращения: 28.02.2022). — Режим доступа: для авториз. пользователей.

Логин: ulgtu2019@yandex.ru

Пароль: 778452asd

А. В. ПУТИЛОВ,
Ю. В. ЧЕРНЯХОВСКАЯ

КОММЕРЦИАЛИЗАЦИЯ ТЕХНОЛОГИЙ И ПРОМЫШЛЕННЫЕ ИННОВАЦИИ

Учебное пособие

САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2021

ББК 65.290я73
П 90

Путилов А. В., Черняховская Ю. В.

П 90 Коммерциализация технологий и промышленные инновации: Учебное пособие. — СПб.: Издательство «Лань», 2021. — 324 с.: ил. — (Учебники для вузов. Специальная литература).

ISBN 978-5-8114-3371-1

Пособие посвящено изучению вопросов промышленной коммерциализации технологий — деятельности, направленной на получение дохода от использования результатов научных исследований и разработок. Приводятся примеры предварительного комплексного изучения финансовых и рыночных перспектив научных разработок и доказательства их будущих экономических и технологических преимуществ, прогнозирование развития рынков, сравнение с лучшим опытом и оценка конкурентных преимуществ разработок и технологий. Настоящее учебное пособие предназначено для студентов магистратуры по направлениям подготовки: «Системный анализ и управление», «Экономика», «Менеджмент», «Бизнес-информатика».

ББК 65.290я73

Рецензенты:

А. И. АГЕЕВ — доктор экономических наук, профессор, генеральный директор Института экономических стратегий РАН; *А. В. ФОМИНА* — доктор экономических наук, профессор, генеральный директор ЦНИИ «Электроника».

Обложка
Е. А. ВЛАСОВА

ПРЕДИСЛОВИЕ

В последнее время в высокотехнологичных отечественных отраслях, прежде всего в атомной отрасли, происходят существенные экономические сдвиги, характеризующиеся переходом к долгосрочной инновационной политике и внедрению технологий цифровой экономики, связанных с освоением технологий следующего поколения и развитием новых инновационных подходов к созданию принципиально новой продукции и оказанию высокотехнологичных услуг. Книга «Коммерциализация технологий и промышленные инновации» является универсальным пособием, которое может быть использовано при обучении студентов как технических, так и управленческих специальностей, связанных с высокими технологиями, например с созданием ядерных реакторов, производством ядерного топлива, разработкой систем управления АЭС. Актуализация образовательного контента, ярко проявленная в данном издании, — это технология выявления наиболее одаренных учащихся в режиме реального времени, которая позволяет производить отбор из больших коллективов потенциальных кандидатов на «инновационную образовательную спираль»: последовательное чередование новых курсов дисциплин, ориентирующих обучающихся стремиться к инновационным решениям в дальнейшей профессиональной деятельности.

Развитие цифровой экономики является одним из ключевых направлений государственной политики Российской Федерации, обусловленной модернизацией производственных отраслей и проникновением цифровых технологий в экономические процессы на всех уровнях производства. Перед ведущими госкорпорациями «Росатом» и «Ростех», как одними из участников Центра компетенций программы «Цифровая экономика Российской Федерации», стоит задача по интеграции всех предложений в план мероприятий по цифровой трансформации промышленности и внедрению «сквозных» технологий. Например, в атомной отрасли был разработан новый инструментарий управления знаниями на основе больших данных с использованием нейросетей и машинного обучения. На ряде конференций в последнее время было отмечено, что будущее развитие цифровых платформ и цифровых технологий заключается именно в управлении результатами интеллектуальной деятельности и их коммерциализации. Расширение использования цифровых технологий позволит более активно вовлекать интеллектуальную собственность в хозяйственный оборот. Система управления знаниями выходит на новый уровень: постоянно развиваются и адаптируются продукты и технологии под меняющиеся условия рынка и цифровые особенности производства, что делает их еще более востребованными. Материалы данного издания позволят расширить использование новых технологий на новые области знаний и инженерные системы нового поколения.

Заместитель Президента РАН,
доктор экономических наук,
член-корреспондент РАН *В. В. Иванов*

ВВЕДЕНИЕ В ЦИФРОВУЮ ЭКОНОМИКУ

Переход к цифровой экономике — общемировая тенденция, поэтому существует большое число определений этого нового экономического уклада. По определению Всемирного банка цифровая экономика — система экономических, социальных и культурных отношений, основанных на использовании цифровых информационно-коммуникационных технологий. В отечественной практике чаще всего используется следующее определение, приведенное в указе Президента России от 9 мая 2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы»: «цифровая экономика — хозяйственная деятельность, в которой ключевым фактором производства являются данные в цифровом виде, обработка больших объемов и использование результатов анализа которых по сравнению с традиционными формами хозяйствования позволяют существенно повысить эффективность различных видов производства, технологий, оборудования, хранения, продажи, доставки товаров и услуг». Очевидно, что коммерциализация технологий и реализация промышленных инноваций должны учитывать эти макроэкономические особенности ближайшего будущего.

Наступающую эру можно охарактеризовать как эпоху цифровых платформ, вытесняющих с рынка и из сферы производства неэффективных посредников и заменяющих их эффективными алгоритмами. Существуют разновидности этих платформ, можно различить две большие группы: электронные торговые площадки (например, Uber, Avito, Cainiao и др.) и инструменты автоматической внерыночной координации совместной деятельности (виртуальные офисы, интернет-взаимодействие отдельных производств, а также цифровые инструменты для более крупных хозяйственных единиц), производства товаров, оказания услуг. Существует тезис о том, что подключение к цифровым платформам дает субъектам рынка такие конкурентные преимущества, что по мере захвата этими платформами национальных и мировых рынков в полной мере начнет срабатывать принцип «кто не с нами, тот банкрот». Можно выделить американскую (англосаксонскую) и китайскую (восточноазиатскую) модели построения цифровой экономики и провести сопоставление этих моделей: по способам инвестирования, роли государственного и частного капитала, доле провальных и сверхприбыльных стартапов за последние несколько лет, по динамике роста и суммарному экономическому эффекту от «цифрового» сектора. Сопоставления явно свидетельствуют в пользу китайской модели, а нехитрая экстраполяция показывает, что если ничего не предпринимать в сфере высоких технологий прежде всего, то к середине 30-х гг. нынешнего столетия эти два мировых лидера по цифровым технологическим платформам захватят не менее 90% совокупных рынков планеты. И если не ответить на вызов чем-то «несимметричным», то места для России в дележе этого рыночного пирога может и не остаться.

Можно и нужно продемонстрировать математическую ясность и логику в осмыслении темы цифровых платформ, постоянно искать ответы на несколько принципиальных вопросов «цифровизации» реального сектора экономики. Следует выявлять схематическую модель каждой такой технологической еди-

ницы в «идеальной» полноте необходимых функций и элементов, искать ответ на вопрос об оптимальном размере формируемых цифровых платформ. В некоторых работах считается, что пределом целесообразной масштабируемости является отраслевая платформа, причем отрасль должна рассматриваться в контексте глобального разделения труда. Важен также вопрос о границах охвата цифровыми платформами того или иного сегмента экономики. Уже существующие современные цифровые платформы в состоянии глубоко автоматизировать производства и рынки тех товаров и услуг, характеристики которых поддаются жесткой количественной или качественной параметризации. Неподатливыми для этой платформенной логики пока остаются сектора, продукты которых обладают значимыми «невербализуемыми» свойствами и спрос на которые существенно зависит от персонального вкуса того или иного потребителя. Но по мере развития возможностей искусственного интеллекта новые типы цифровых платформ будут захватывать все новые и новые области «неформализуемой» экономики.

Можно дать высокие оценки наиболее продвинутым современным платформам Alibaba Group (Китай) и Abbyy SmartCAT (Россия): по мнению многих экспертов, они уже сегодня на 70% соответствуют перспективным платформенным моделям. Кроме того, важен вопрос о корпоративных платформах глобальных производителей (например, авиастроительной корпорации Boeing), масштабы которых сильно повышают барьер выхода на рынок платформ собственных отраслевых разработок. Большие данные и результаты их анализа дают основу развития нового экономического уклада, поэтому технологии искусственного интеллекта или создаваемые квантовые компьютеры могут в перспективе дать кардинальные преимущества в этой экономической гонке своим разработчикам.

В настоящее время в нашей стране высказываются идеи относительно путей «платформенной» оптимизации системы госзакупок и государственных информационных систем, предложено несколько конкретных проектных решений. В обсуждениях государственного участия в создании цифровых платформ часто звучит предостережение: рассматривать государство в качестве субъекта интересов, решений и действий — значит самим запутываться в мифах и запутывать других. Главным предметом дискуссий является проблема субъекта «информационного реформирования» государства: как достичь синергии несочетаемых, казалось бы, в одном лице качеств — компетентности, заинтересованности и полномочий.

В ряде работ делаются попытки корректно ввести понятие «платформа», но указывается необходимость более глубокого переопределения этой экономической сущности. Наиболее целесообразно не изобретать велосипед, а «встать на плечи гигантов»: восстановить систему смысловых координат для обсуждения, обратившись к наработкам классиков экономических и общественных наук последних десятилетий. Всем нам нужно осмыслить фундаментальный сдвиг мировой экономической мысли в понимании, что такое хозяйственная деятельность по своей сути. Это позволит разработать понятие технологической платформы в контексте современного понимания. Тогда и в теоретических разработках, и в практических реализациях идеи «цифровых платформ» в нашей стране может появиться реальный шанс возглавить «миро-

вой мейнстрим». Догоняющая стратегия обречена на неудачу, и только понимание глубокой значимости теории дает возможность «ухватить тренд и забежать вперед». Все вышесказанное дает основание утверждать, что мы все находимся в преддверии масштабных экономических изменений и на конкретных примерах следует искать ответы на поставленные жизнью экономические вопросы. Коммерциализация технологий и промышленные инновации — это та сфера, которая наиболее перспективна при рассмотрении ближайшего экономического будущего.

Государственный и корпоративный подход к разработкам информационных систем производственных предприятий в эпоху цифровой экономики

В течение 2017 г. в нашей стране было выпущено несколько документов наивысшего уровня, в которых цифровая экономика была обозначена совершенно конкретно. Это весенние указы Президента Российской Федерации:

- от 9 мая 2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы»;
- от 13 мая 2017 г. № 208 «О Стратегии экономической безопасности Российской Федерации на период до 2030 года».

Наконец, летом 2017 г. программа долгосрочного экономического развития «Цифровая экономика Российской Федерации» была утверждена распоряжением Правительства России от 28 июля 2017 г. № 1632-р.

Основные цели этой программы:

- создание экосистемы цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности;
- создание необходимых и достаточных условий институционального и инфраструктурного характера для создания и (или) развития высокотехнологичных бизнесов;
- повышение конкурентоспособности на глобальном рынке как отдельных отраслей экономики Российской Федерации, так и экономики в целом.

При формировании данной программы подразумевались следующие основные уровни цифровой экономики:

- среда (регуляторика, инфраструктура, кадры, информационная безопасность);
- платформы и технологии (где формируются компетенции для сфер деятельности);
- рынки и отрасли экономики (сферы деятельности), где и осуществляется взаимодействие конкретных экономических субъектов.

Отмеченная в программе «Цифровая экономика Российской Федерации» роль государства — это создание фундамента развития цифровой экономики путем развития компонентов платформ и среды:

- институтов цифровой экономики (нормативное регулирование, кадры и образование, исследовательские компетенции и технологические заделы);
- инфраструктуры цифровой экономики (информационная инфраструктура, информационная безопасность).

По мнению разработчиков программы, которое поддержано на правительственном уровне, рынки и отрасли экономики самоорганизуются в новой среде и вокруг создаваемых и развивающихся цифровых платформ.

Базовые направления развития программы «Цифровая экономика Российской Федерации» (табл. 1) обеспечивают комплексный подход к разработке принципиальных основ цифровой экономики. При реализации программы предусматривается, что каждое из этих направлений развития цифровой среды и ключевых институтов цифровой экономики учитывает как поддержку развития уже существующих условий для возникновения прорывных и перспективных сквозных цифровых платформ и технологий, так и создание условий для возникновения новых платформ и технологий. Введено понятие «сквозные» технологии: основными сквозными цифровыми технологиями, которые входят в рамки реализуемой программы, являются:

1) новые производственные технологии;
2) большие данные;
3) нейротехнологии и искусственный интеллект;
4) системы распределенного реестра;
5) квантовые технологии;
6) промышленный Интернет;
7) компоненты робототехники и сенсорика;
8) технологии беспроводной связи;
9) технологии виртуальной и дополненной реальностей.

Таблица 1

Программа «Цифровая экономика Российской Федерации»: базовые направления

№	Направления	Задачи развития направлений
1	Нормативное регулирование	Формирование регуляторной среды, обеспечивающей благоприятный правовой режим для возникновения, развития и использования цифровых технологий
2	Кадры и образование	Подготовка кадров, модернизация системы образования, создание рынка труда для цифровой экономики
3	Формирование исследовательских компетенций и технических заделов	Создание системы поддержки поисковых, прикладных исследований в области цифровой экономики, обеспечивающей технологическую независимость, национальную безопасность и конкурентоспособность на мировых рынках
4	Информационная инфраструктура	Создание отечественных сетей связи, центров обработки данных (ЦОД), обеспечение доступа к создаваемым цифровым данным и внедрение цифровых платформ
5	Информационная безопасность	Обеспечение безопасности данных цифровой экономики, инфраструктуры, институтов, граждан

Эти девять сквозных технологий должны пронизывать все проектные направления, которые в настоящее время активно формируются в рамках программы. В целях управления развитием цифровой экономики определяются цели и задачи в рамках пяти центров компетенций, целью которых является соз-

дание благоприятной экосистемы развития основных сквозных технологий. Формирование исследовательских компетенций и технических заделов закреплено на правительственном уровне (Постановление Правительства России от 28 августа 2017 г. № 1030 «О системе управления реализацией программы „Цифровая экономика Российской Федерации“») за АО «РВК», которое опирается на госкорпорации «Росатом» и «Ростех». Зоны ответственности этих госкорпораций по развитию «сквозных» технологий распределяются примерно поровну. Госкорпорация «Росатом»: новые производственные технологии, большие данные, квантовые технологии, технологии виртуальной и дополненной реальности. Госкорпорация «Ростех»: промышленный Интернет, компоненты робототехники и сенсорика, нейротехнологии и искусственный интеллект, системы распределенного реестра, технологии беспроводной связи.

Государственные корпорации — база для разработки «сквозных» технологий цифровой экономики

Современная стратегия, направленная на модернизацию отечественной экономики, тесно связана с развитием крупных вертикально-интегрированных государственных компаний и корпораций, которые нацелены на решение как коммерческих, так и специальных государственных задач. Заявленные цели инновационной модернизации требуют интеграции широкого спектра современных инструментов развития, в том числе принципов технологических платформ (ТП), ресурсов «институтов развития» (ИР) на федеральном и региональном уровнях, а также механизмов государственно-частного партнерства (ГЧП). Рациональное комбинирование методик и подходов ГЧП и ТП с вовлечением ведущих федеральных и региональных ИР позволяет достигать синергетического эффекта и реального стимулирования процессов инновационной модернизации высокотехнологической промышленности. Атомная энергетика и развитие ядерных технологий в России в настоящее время являются абсолютной монополией государства. Отечественная атомная энергетика вертикально интегрирована и централизована в Государственной корпорации по атомной энергии «Росатом» (Госкорпорации «Росатом»), которая была преобразована из соответствующего министерства, затем федерального агентства и, по сути, сохранила все признаки государственной структуры, поскольку является государственной компанией и занимает монопольное положение на рынке атомной энергетики в стране. Однако по многим параметрам эта ситуация не отвечает требованиям времени и подлежит определенному реформированию. Необходимо проведение модернизации атомного энергетического комплекса как на институциональном, так и на технологическом уровне. Оба вектора модернизации тесно взаимосвязаны друг с другом, что не позволяет осуществлять реализацию одного вектора без другого. Для дальнейшего развития, сохранения и укрепления конкурентоспособности отечественная атомная энергетика нуждается в масштабной инновационной модернизации. Очевидно, отрасль необходимо «открыть» для частных инвестиций как капитального, так и интеллектуального характера. Отрасли необходима реформа, которая повлечет за собой повышение инвестиционной привлекательности в долгосрочной перспективе, а следовательно, инновационности. Это должно быть одной из ключевых целей в стратегии развития атомной энергетики, стратегической установкой Госкорпорации

«Росатом» как ключевого игрока атомной отрасли. Стоит отметить, что качество стратегического управления в отрасли существенно выросло за последние десять лет с момента начала реорганизации управления атомной отрасли в 2007 г. В настоящее время поддерживается постоянный процесс повышения эффективности и непрерывного качественного развития. Однако недостаточное внимание уделяется внедрению в практическое использование современного инструментария стратегического управления, нет последовательности в реализации принятой стратегии. Очевидна высокая вероятность того, что принятая стратегия так и не будет реализована, «останется на бумаге». Необходимо сфокусировать внимание на повышении инновационности и инвестиционной привлекательности отрасли, в поиске путей запуска межотраслевого мультипликативного эффекта для целей стратегического развития и инновационной модернизации высокотехнологичных отраслевых комплексов. Руководством Госкорпорации «Росатом» разработана и утверждена Программа инновационного развития, что подтверждает ориентацию на формирование технологической компании мирового уровня. Однако зафиксированные в этой программе неэнергетические направления развития (ядерная медицина, облучение материалов, системы безопасности и пр.) пока не вносят достойного вклада в экономические результаты деятельности корпорации, сопоставимого с атомной энергетикой. Коммерциализация неэнергетических ядерных технологий (медицинских, материаловедческих и пр.) позволит внести новый импульс в реализацию промышленных инноваций.

Государственные подходы к развитию цифровой экономики

Правительством Российской Федерации разработан и утвержден (или пока еще находится в процессе утверждения) ряд стратегических документов, определяющих общий долгосрочный вектор модернизации и инновационного развития промышленности России. На правительственном уровне утвержден План мероприятий по реализации Стратегии научно-технологического развития Российской Федерации на 2017–2019 гг. (распоряжение Правительства Российской Федерации от 24 июня 2017 г. № 1325-р), в соответствии с которым будет принят целый комплекс важных документов. Достаточно еще раз упомянуть программу «Цифровая экономика Российской Федерации», утвержденную в июле 2017 г. Принимаемые документы должны обозначить вертикаль стратегического планирования в экономике страны. Ключевую роль в единой системе планирования играют государственные корпорации. В принятых документах роль государственных корпораций обозначена формально и нуждается в конкретизации, увязке с верхними уровнями стратегического планирования, отраслевыми стратегиями, региональными и муниципальными стратегиями, а также стратегиями отдельных корпораций (на паритетных началах и принципах консенсуса в случае частных корпораций). Однако пока российские госкорпорации (доля государства, превышающая 50% уставного капитала, характерна более чем для пятидесяти отечественных корпораций) недостаточно эффективны и прозрачны в реализации целенаправленной стратегии инновационной модернизации. Хотя Госкорпорация «Росатом» и демонстрирует существенные успехи в повышении качества корпоративного управления, эффективности стратегического управления и движения по пути инновационной модернизации, существуют при этом и направления для дальнейшего совершенствования в области

стратегического управления отраслью. Главное в этом направлении развития — систематизации регулярного управления за счет повышения инвестиционной привлекательности отечественной атомной энергетики, цифровая трансформация атомной отрасли в целом. Ключевыми инструментами экономического подхода к совершенствованию управления госкорпорацией являются:

- государственно-частное партнерство — выстраивание механизмов кооперации государства с частными предприятиями, интенсификация инновационного развития и модернизации. В развитии российских высокотехнологических отраслевых комплексов применение инструментов и механизмов ГЧП решает не только проблему привлечения дополнительных инвестиций, но и способствует обновлению управленческих ресурсов, внедрению современных моделей стратегического управления отраслевыми комплексами, проведению совместных научно-исследовательских разработок;
- развитие совместных проектов между государством (в том числе в лице госкорпраций) и частными бизнес-структурами. Эти связи пока затруднены из-за отсутствия нормативно-правовой базы ГЧП на федеральном уровне и фрагментарности правового поля в регионах. Существующие нормативные документы либо носят особый региональный характер (например, для Санкт-Петербурга), либо обладают выраженной отраслевой направленностью (закон о концессионных соглашениях, разработанный Минтрансом России). Необходимо скорейшее принятие федерального закона о ГЧП, регулирующего максимально полный спектр взаимоотношений государства и частного бизнеса в реализации общественно значимых проектов в различных отраслях экономики, в том числе в высокотехнологичных разработках и долгосрочных проектах, например в атомной энергетике.

Для перехода к инновационной модернизации экономики крайне важно четко обозначить права и обязанности сторон в рамках ГЧП. Необходимо минимизировать риски вмешательства государства в развитие нормальных рыночных отношений, снижение конкурентоспособности через создание неравных условий для различных участников рынка. Правила участия в общественно значимых государственных проектах с использованием ГЧП должны быть едины, понятны и прозрачны для всех участников рынка. Очень важно обеспечить неизменность государственной политики в рамках реализации проектов ГЧП в долгосрочной перспективе (15–20 лет). Здесь важную роль играет создание эффективных механизмов оценки, одобрения и контроля осуществления соглашений — органов, уполномоченных на выполнение указанных функций на межведомственном уровне. Среди них можно выделить следующие:

- технологические платформы создаются для более эффективного функционирования госкорпораций. В Госкорпорации «Росатом» наиболее известны технологическая платформа реакторов типа ВВЭР и разрабатываемая в настоящее время технологическая платформа замыкания ядерного топливного цикла с реакторами на быстрых нейтронах (проект «ПРОРЫВ»). Концепция технологических платформ не только позволяет обеспечить выбор стратегических научных направлений и всесторонний анализ рыночного потенциала технологий, но и предполагает учет точек зрения всех заинтересованных сторон: государства, промышленности, научного сообщества, контролирующих орга-

нов, пользователей и потребителей. В конечном итоге, использование инструментов и методов технологических платформ ведет к дополнительной мобилизации общественных и частных источников финансирования;

- цифровые платформы являются системой объединения ресурсов различных участников процесса в рамках решения научной, технической или технологической задачи. В частности, цифровые технологические платформы в атомной энергетике должны решить задачу объединения ресурсов коллективов инженеров и ученых в стране и за рубежом в целях разработки новых прорывных технологий энергогенерации и энергосбережения.

Институты развития как драйвер роста цифровой экономики в реализации промышленных инноваций

В России сложилась достаточно обширная система институтов развития в сфере инноваций, обеспечивающая грантовое и заемное финансирование инновационной сферы. К таким институтам можно отнести Российский фонд фундаментальных исследований (РФФИ, 1992 г.), Фонд содействия инновациям (1994), Российскую венчурную компанию (2006), ОАО «Росинфокоминвест» (2006), РОСНАНО (2007) и ФИОП РОСНАНО (2010), Фонд «Сколково» (2010), Фонд «ВЭБ Инновации» (2011), Фонд развития интернет-инициатив (ФРИИ, 2013 г.), Российский научный фонд (РНФ, 2013 г.), Фонд развития промышленности (с 2014 г., ранее — Российский фонд технологического развития), а также Российский экспортный центр (2015) и ГК Внешэкономбанк (2007). Кроме того, выделяется ряд особых институтов, к числу которых можно, в частности, отнести АНО «Агентство стратегических инициатив по продвижению новых проектов (АСИ)». К нефинансовым инновационным институтам развития также должны быть отнесены: ОАО «Особые экономические зоны», НКО «Фонд развития моногородов», инновационная инфраструктура (индустриальные парки, технопарки, инновационные кластеры и др.), Корпорация развития Дальнего Востока, Фонд развития Дальнего Востока, ОАО «Корпорация развития Северного Кавказа», региональные корпорации развития.

Важным звеном в стратегии цифровой трансформации атомной отрасли и повышения инвестиционной привлекательности этой отрасли, в конечном счете — интенсификации инновационной модернизации атомной энергетики, должны быть институты развития — федеральные и региональные. Необходимо вовлекать в развитие атомной энергетики федеральные и региональные институты развития — Российскую венчурную корпорацию (РВК), «Роснано», Инновационный центр «Сколково», Инвестиционный фонд РФ, Внешэкономбанк (и его дочерние компании — Российский банк развития и Экспортное страховое агентство России), Евразийский банк развития и пр.

Стратегические задачи развития, технологические и научные ресурсы Госкорпорации «Росатом» позволяют ей играть ведущую роль в развитии страны — принять активное участие в формировании системы территориальных научных и производственных центров (инновационных кластеров и индустриальных парков), территорий опережающего развития (ТОР). В этой связи требуется решение следующих задач: законодательные ограничения на привлечение иностранных инвестиций и организацию предприятий на территориях для отраслевых кластеров (большая часть предприятий расположена на закрытых

территориях — ЗАТО), низкое качество транспортной, энергетической и социальной инфраструктуры, недостаточная квалификация специалистов региональных и муниципальных органов власти в вопросах проектного управления и инновационного развития и пр.

Развитие атомной энергетики в России всегда осуществлялось и финансировалось государством как по причине стратегической важности отрасли (она появилась «в дополнение» к военному ядерному комплексу), так и в связи с крайне высокой капиталоемкостью и длительным сроком возврата инвестиций. Тем не менее принятая стратегия развития атомной отрасли (в том числе экономическая экспансия за рубеж) и потребность в скорейшей модернизации существующих генерирующих мощностей ставят перед Госкорпорацией «Росатом» задачу создания условий для привлечения российского и иностранного частного капитала для развития атомной энергетики страны. Сегодня существует ряд ограничений для приватизации «гражданской» части отечественного атомного комплекса. Основное препятствие — интегрированность научно-технической базы отрасли с оборонным ядерным комплексом, международные требования МАГАТЭ и пр. В ведущих ядерных странах мира подобное сращивание гражданской и военной частей атомного комплекса не является типичным, что, соответственно, наносит определенный ущерб глобальной конкурентоспособности российской атомной энергетики и энергомашиностроения. Необходимо проводить целенаправленные управленческие и инженерно-технические действия в целях разделения атомного и оборонного направлений Госкорпорации «Росатом» с целью частичной приватизации атомного энергетического комплекса России в перспективе.

В ближайшем будущем даже частичная приватизация Госкорпорации «Росатом» не может быть эффективно реализована как в силу технико-экономических причин (внутренних — неготовность компании к приватизации из-за значительной интеграции ресурсной базы и системы НИОКР, и внешних — негативный макроэкономический фон в мире), так и в силу специфики функционирования данной отрасли в России. Даже с учетом негативных качеств, присущих организационно-правовой форме государственной корпорации (также во многом и публично-правовой компании), экономический анализ показал, что именно эта форма (при организации должного надзора) является наиболее адекватной профилю и стратегии компании, ее отраслевой специфике.

Стратегические направления развития цифровой экономики и государственно-частное партнерство

Стратегическое управление отраслью атомной энергетики реализуется на сравнительно высоком и качественном уровне. Это подтверждается высокими результатами, достигнутыми российской атомной энергетикой в последние годы, — принят ряд стратегических документов, приняты цели по научным исследованиям, сделана фокусировка на ключевых (прорывных) проектах по НИОКР. Также отрасль проводит активную модернизацию существующих мощностей, строительство новых объектов, реализуется масштабная экспансия на мировом рынке строительства и эксплуатации АЭС, совершен ряд сделок по приобретению активов за рубежом, в первую очередь в целях дальнейшего развития ресурсной базы и минимизации рисков топливного дефицита на среднесрочную и долгосрочную перспективу. Тем не менее важно обратить внимание

на то, что функционал стратегического управления в Госкорпорации «Росатом» (и соответственно в отрасли в целом) серьезно размыт, отсутствует необходимая для эффективной деятельности централизация ответственности и соответствующих полномочий, что могло бы позволить реализовывать политику перманентного повышения инвестиционной привлекательности отрасли. Следует продолжать работу в направлении дальнейшего совершенствования стратегического управления в российской атомной отрасли для формирования новых цифровых платформ, что неизбежно потребует развития ГЧП.

Привлечение инвестиционных ресурсов в развитие атомной энергетики как высокотехнологичной стратегической отрасли зависит от комплекса факторов. Прямолинейное увязывание доступности инвестиционных ресурсов в экономике с инвестиционной привлекательностью и соответственно наличием или отсутствием инвестиций в отрасли оказывается слишком упрощенным, поверхностным и не позволяет эффективно управлять инвестиционным потенциалом в долгосрочной перспективе. В ряде исследований показано, что последовательное стратегическое повышение инвестиционной привлекательности отраслевого комплекса возможно только при создании и управлении системой, включающей управление следующими инфраструктурными блоками:

- нормативно-правовой блок;
- ресурсный блок;
- технологический блок, включая цифровые технологии.

Причем эффективность данной работы по повышению инвестиционной привлекательности непосредственно зависит не только от координации этапов работ по каждому из направлений, но и от конструктивного сотрудничества между этими блоками и отдельными проектными группами, специалистами и экспертами на основе матричного принципа организации процесса. В управлении атомной отраслью стратегически обоснованным будет сосредоточение всей полноты ответственности и полномочий по разработке инвестиционной политики и осуществления деятельности по повышению инвестиционной привлекательности в рамках одного структурного подразделения или блока.

Инвестиции совершенно необходимы в атомной отрасли для реализации перспективных цифровых технологий. В частности, при помощи когнитивных технологий машины поиска информации в Интернете перестанут выдавать абсурдные миллионы ссылок. Они станут сами обрабатывать собранные ссылки, соревнуясь в полноте, достоверности и доступности для восприятия человеком создаваемых ими рефератов. Потребитель самостоятельно найдет производителя, а, учитывая возможность автоматического документооборота, тот сможет напрямую взаимодействовать со всеми своими контрагентами. Так появятся бизнес-модели M2C (manufacturer to customer, производитель — потребителю) и обратный C2M, при котором возможна реализация персонализированного производства, предполагающего производство товара, обладающего необходимыми (или желательными) для данного потребителя оригинальными свойствами. Облачные технологии или облачные вычисления (Cloud Computing) — информационно-технологическая концепция, подразумевающая обеспечение повсеместного и удобного сетевого доступа по требованию к общему объему конфигурируемых вычислительных ресурсов, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами

или обращениями к провайдеру. Примерами ресурсов могут являться сети передачи данных, серверы, устройства хранения данных, приложения и сервисы — как вместе, так и по отдельности. Иначе говоря, облачные технологии — это технологии обработки данных, в которых компьютерные ресурсы предоставляются интернет-пользователю по запросу (on demand) как онлайн-сервис. Необходимо подчеркнуть, что облачные технологии внесли колоссальный вклад в фундамент зарождающейся цифровой экономики. Этот вклад не ограничивается лишь технологической составляющей, но включает еще экономическую и идеологическую компоненты. Развитие облачных технологий, например, привело к появлению таких понятий, как производство по требованию (production on demand), программное обеспечение как услуга (software as a service) и многих других, которые станут лейтмотивом большинства бизнес-моделей будущего и принципом большинства экономических взаимодействий.

Перспективные интернет-технологии и информационная среда цифровой экономики

Промышленный Интернет или Интернет вещей — это концепция, объединяющая множество технологий, подразумевающая оснащенность датчиками и подключение к Интернету всех приборов (и вообще вещей), что позволяет реализовать удаленный мониторинг, контроль и управление процессами в реальном времени (в том числе в автоматическом режиме). Сегодня сформированы два крупных направления: Интернет вещей (IoT — Internet of Things) и промышленный Интернет вещей (IIoT — Industrial Internet of Things). Инструментально данные технологии очень похожи, ключевая разница в предназначении: если основная задача Интернета вещей — это сбор всевозможных данных (которые будут приоритетно использоваться для построения моделей и прогнозов), то предназначение промышленного Интернета вещей состоит в автоматизации производства (за счет удаленного управления ресурсами и мощностями по показаниям датчиков).

Технологии цифровой трансформации экономики прежде всего подразумевают массовое использование интернет-технологий. Использование интернет-систем в этом случае должно базироваться на стандартах, которые подразумевают пятый (следующий за действующим сейчас четвертым уровнем) уровень стандартизации или стандарты 5G. Целевые характеристики 5G предусматривают резкий рост производительности и скорости реализации интернет-протокола:

- скорость передачи данных должна обеспечить рост производительности интернет-связи в 10–100 раз в расчете на абонента — до 10 Гбит/с (DL) и до 5 Гбит/с (UL);
- потребляемый трафик абонентов должен обеспечить рост в 1000 раз — до 500 Гб на пользователя в месяц;
- предусматривается увеличение количества подключаемых абонентских устройств в интернет-соте в 10–100 раз (до 300 000 на узел). Рост M2M-устройств с 50 до 500 млрд;
- для промышленного интернет-протокола потребуется увеличение в 10 раз времени автономной работы абонентских устройств с небольшим энергопотреблением, таких как сенсоры M2M;

- потребуется сокращение времени задержки в цепочке E2E с 5 до 1 мс и менее;
- новые технологии предусматривают снижение стоимости эксплуатации и энергопотребления сетей 5G до 10% от текущего потребления сетей 4G.

По своим производственным возможностям технология 5G должна составить к 2025 г. альтернативу наземным сетям цифрового телевидения DVB-T. Все эти изменения и многие другие технологические нововведения дадут возможность в массовом масштабе создавать цифровые платформы.

Платформа цифровой экономики — это цифровая среда (программно-аппаратный комплекс) с набором функций и сервисов, обеспечивающая потребности потребителей и производителей, а также реализующая возможности прямого взаимодействия между ними. Ценность цифровой платформы — в предоставлении самой возможности прямой коммуникации и облегчении процедуры взаимодействия между участниками бизнес-процессов. Платформы снижают издержки и предоставляют дополнительный функционал как для поставщиков, так и для потребителей. Также они предполагают обмен информацией между действующими лицами, что должно существенно улучшать сотрудничество и способствовать созданию инновационных продуктов и решений. Цифровая платформа как бизнес-модель существует давно. Простым примером может служить классический рынок, на котором продавцы и покупатели (производители и потребители) находят друг друга. В современном мире можно привести много активно растущих компаний, в основе которых функционируют принципы «платформенной бизнес-модели», и самые яркие — это Uber и Airbnb. Для сферы образования интернет-технологии (рис. 1) дают возможность не только дистанционного обучения, но и создания новых образовательных платформ.

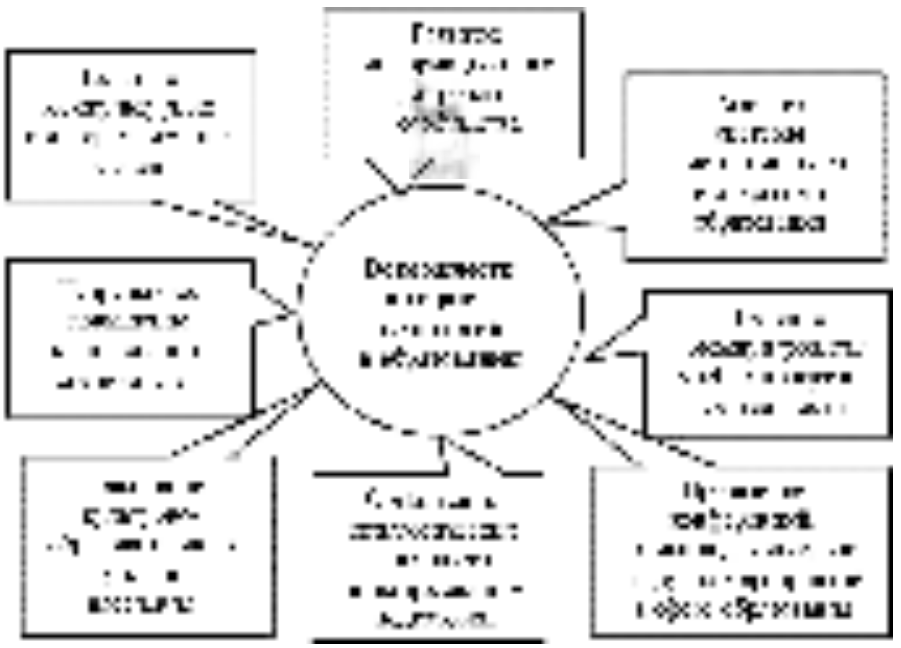


Рис. 1

Основные направления использования интернет-технологий в образовании

Организация педагогических сообществ в интернет-среде может позволить быстро и качественно формировать новое содержание образовательных программ, которые пока сильно отстают от потребностей реальной жизни. А ведение международной учебно-проектной деятельности станет новым стратегическим направлением, позволяющим объединить консалтинг, обучение, повышение квалификации и формирование «интеркультурных» творческих коллективов для решения крупных международных проблем.

Указ Президента России от 09.05.2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы» декларирует, что развитие цифровой экономики является стратегически важным вопросом для России в целом, определяющим ее конкурентоспособность на мировой арене. Необходимо признать, что в России сегодня нет условий для стихийного формирования зрелой цифровой экономики за приемлемый период времени — в первую очередь из-за технологического отставания и отсутствия критической массы экономических субъектов. Это значит, что государству необходимо стимулировать и направлять развитие цифровой экономики. Важной отличительной особенностью российской экономики является тот факт, что львиная доля ВВП создается государственными корпорациями (или компаниями со значительной долей государственного участия). Во многих отраслях производства игроки с государственным участием могут составлять до 80% рынка. В таких условиях наиболее рациональным шагом представляется создание ряда индустриальных цифровых платформ под руководством профильных министерств или госкорпораций. Такие платформы создадут необходимый инфраструктурный базис для максимально быстрого развития цифровой экономики и распространения сопутствующих технологий. При построении платформ цифровой экономики необходимо фокусировать усилия на ключевых направлениях: транспорт, телекоммуникации, энергетика, обработка данных. Развитие именно этих областей позволит создать инфраструктурный и технологический базис, тиражируя который на другие области Россия сможет максимально быстро развить зрелую цифровую экономику.

Теоретические подходы к обоснованию методов развития цифровой экономики

Фундаментальная экономическая теория пока отстает от практики. На сегодняшний день не существует теоретической базы не только для будущей цифровой экономики, но даже для современной экономики услуг, основанной во многом на цифровых платформах (системы такси, интернет-торговли и пр.). После первой четверти XX в. базовая экономическая теория практически не имела развития — некоторые успехи были достигнуты лишь в рассмотрении отдельных вопросов. Начиная со второй половины XX в. официальные социально-экономические науки беззастенчиво обслуживали интересы финансово-олигархических «элит». Важнейшие вопросы системных свойств капитализма, на которые обращали внимание классики, старательно игнорировались. Все основные экономические законы и метрики (в том числе ВВП) были введены и сформулированы еще в XIX — первой половине XX в. и хорошо описывают реальный сектор (производящую промышленные товары экономику). Со второй половины XX в. сектор услуг и нематериального производства по-

лучил значительное развитие и со временем стал основным сектором экономики в наступившем XXI в. Свойства производства и потребления в нематериальной сфере значительно отличаются, но человечество пока не создало соответствующей теоретической базы для корректного описания новой экономики. Вместо этого создавались и постоянно пересматривались методики, позволявшие «привести» нематериальную сферу к уже имевшимся метрикам и показателям, чтобы иметь возможность включить ее в уже привычные формы описания экономики. До определенного момента эти попытки давали приемлемые результаты, но только до тех пор, пока нематериальный сектор производства услуг не стал превышать реальный сектор экономики по производству товаров. Еще одним отягчающим обстоятельством является политизированность и предвзятость современной экономической науки, что приводит к спекулятивным и преднамеренным искажениям общей картины (например, из-за практики постоянного пересмотра методик подсчета ВВП). Одним из ключевых вопросов при формировании новой теории является выбор адекватных интегральных параметров и формирование новых метрик. В нашем мире уже есть несколько подходящих устойчивых тенденций, учет которых может помочь в формировании необходимого базиса обновленной экономической теории:

- информация постепенно становится товаром и происходит конвергенция и взаимообогащение программной продукции и аппаратных средств;
- благосостояние общества коррелирует с удельным потреблением энергии;
- социальный статус постепенно вытесняется социальным авторитетом.

Возможно, валютой будущего может стать субстанция, которую можно охарактеризовать как «количество произведенной вами полезной информации на количество потребленной энергии», где «полезность» измеряется в сетевых «лайках». Сегодня такое предсказание кажется чрезмерно футуристическим, но прогресс экономического развития постоянно ускоряется и подобная перспектива может ожидать нас совсем не за горами, а уже через 15–20 лет. Достаточно вспомнить, что такому важному технологическому изобретению для человечества, как ткацкий станок, понадобилось 120 лет, чтобы покинуть пределы Европы, а Интернету понадобилось всего 10 лет, чтобы охватить всю планету.

Цифровая экономика предлагает широкие возможности для развития системы государственного управления. Современные технологии позволяют в ближайшем будущем создать среду высокотехнологичной цифровой платформы государственного управления, которая обеспечит минимизацию человеческого фактора, сопутствующей ему коррупции и ошибок, автоматизирует сбор статистической, налоговой и иной отчетности, обеспечит принятие решений на основе анализа реальной ситуации. Оказание государственных услуг будет строиться на базе единой цифровой облачной платформы, имеющей открытые интерфейсы межмашинного взаимодействия и позволяющей расширять возможности взаимодействия граждан с государством путем создания ими собственных приложений, работающих на базе этой платформы (с обязательной сертификацией по безопасности и соблюдению законодательных норм). В атомной отрасли информационные системы и комплексы должны учитывать все описанные выше ас-

пекты, давая возможность на производственных предприятиях получать экономические выгоды от создания и использования цифровых платформ.

Цифровая трансформация традиционных общественных отношений

Современный период развития реального сектора экономики зачастую отождествляют с технологической революцией, оказывающей все большее влияние на общество в целом. Следствием широкого распространения цифровых решений и прорывных технологий являются изменения не только в производстве, но и на рынках труда. Во всех сферах задаются жесткие требования к скорости и качеству информации, являющейся основой государственных и бизнес-решений. По многим традиционно «неторгуемым» (т. е. непереносимым в географическом пространстве) товарам и услугам кратно снижаются издержки перемещения. Новые форматы товаров и услуг из неторгуемых становятся торгуемыми. В первую очередь это касается сфер образования, здравоохранения, безопасности, а также других услуг, традиционно предоставляемых государством:

- практически весь спектр государственных услуг для граждан становится электронным;
- контроль юридически значимых действий и финансовых операций с помощью интеллектуальных роботизированных систем начинает осуществляться по цифровому следу;
- беспрепятственный доступ в Интернет увеличивает число безналичных финансовых расчетов;
- получают широкое распространение цифровые сервисы в социальной сфере, связанные прежде всего с контролем в режиме реального времени самых разнообразных объектов (от использования ресурсов в «умном» доме до мониторинга состояния здоровья с помощью подключенных цифровых устройств);
- развиваются компьютерные форматы образования, в том числе дистанционного.

Количественные характеристики новых образовательных форматов можно проиллюстрировать тем, что в конце 2016 г. цифровая образовательная платформа Coursera объявила, что число ее студентов достигло 18 млн человек, число пройденных курсов превысило 3,6 млн, а совокупный трафик видеопросмотров курсов составил 17 тыс. лет. Количество учащихся, зарегистрированных на цифровой образовательной платформе edX, достигло 7 млн человек. В то же время испанская MOOC-платформа Miriada X достигла показателя в 2 млн студентов. Всеиндийский совет по техническому образованию (AICTE) принял решение о том, что 10% учебного плана в 10,8 тыс. технических институтов страны должно базироваться на платформе MOOC. Под давлением технологической революции быстро перестраивается качество услуг в социальной сфере в целом:

- профессиональное образование во всем мире сталкивается с растущей конкуренцией со стороны неакадемической сферы. Квалификации выпускников устаревают быстрее, чем успевает среагировать традиционная система образования. Происходит переключение с проектно-ориентированного образования на экспериментально-ориентированное, а передача практических навыков обеспечивается не только за счет стажировок, но и за счет распространения специально созданных учебных заводов и учебных лабораторий. Использование в образовании цифровых технологий, в том числе больших данных, искусственного

интеллекта, разного рода нейротехнологий, претендует на изменение природы познания и предоставление образовательных услуг в целом;

- на базе цифровых и биологических технологий продолжается развитие медицины «4П»: превентивной/предупредительной, прогностической, пациент-ориентированной, персонифицированной. В последние годы закладываются основы «5П-медицины», которая дополняется решениями на базе цифровых платформ, основанных на использовании математических моделей состояния здоровья или связанных с данными моделями комплексов методов его контроля и лечения;
- в науке и научной деятельности происходят преобразования, связанные прежде всего с использованием больших данных, искусственного интеллекта и цифровизации исследований: аналоговые системы фиксации экспериментальных данных вытесняются цифровыми, происходит замена натуральных экспериментов цифровым моделированием, растет использование цифровых средств сбора, обработки и хранения информации.

Цифровая экономика и цифровая трансформация производственных комплексов вызовут значительные изменения в ближайшее время на рынке труда, особенно в традиционных отраслях и профессиях:

- изменение структуры рынка труда в сторону роботизации производств будет иметь последствия для занятости в сегменте рабочих профессий: по прогнозам, к 2035 г. в развитых странах роботы заместят работу, выполняемую людьми, в 25–30% видах профессиональной деятельности;
- изменение структуры занятости в сторону использования искусственного интеллекта будет иметь последствия для «белых воротничков» — менеджеров, аналитиков и пр., а в некоторых случаях и для высших управленческих кадров. Ожидается, что к 2025 г. до 30% корпоративных аудиторских проверок будет осуществляться с использованием технологий искусственного интеллекта. Кроме того, полная роботизация возможна в отдельных сегментах банковской деятельности, юридических услугах, бухгалтерском учете, сложной аналитике;
- повсеместный отказ от систем пожизненного найма и быстрая смена квалификационных требований к работникам приводят к изменению ими своего отношения к профессиональной карьере и выбору занятий: работники все больше должны стремиться сами создавать себе работу и заботиться о ее рентабельности;
- одновременное формирование на рынке труда огромной потребности в новых занятиях и профессиях, связанных с использованием передовых производственных технологий, интеллектуализацией, роботизацией производства и т. п., формирует новые требования к персоналу.

Новые требования к работникам будут постоянно изменяться, трансформируясь совместно с изменениями цифровых платформ в различных сферах. При этом движущей силой этих изменений будут новые подходы к формированию потоков в цифровой экономике: информационных, энергетических, материальных, финансовых и пр. Изменения внешней среды будут воздействовать на структуру производственных комплексов, а внутри этих комплексов новые потоковые конфигурации последовательно определят новые необходимые компетенции работающих в каждой конкретной сфере (рис. 2).

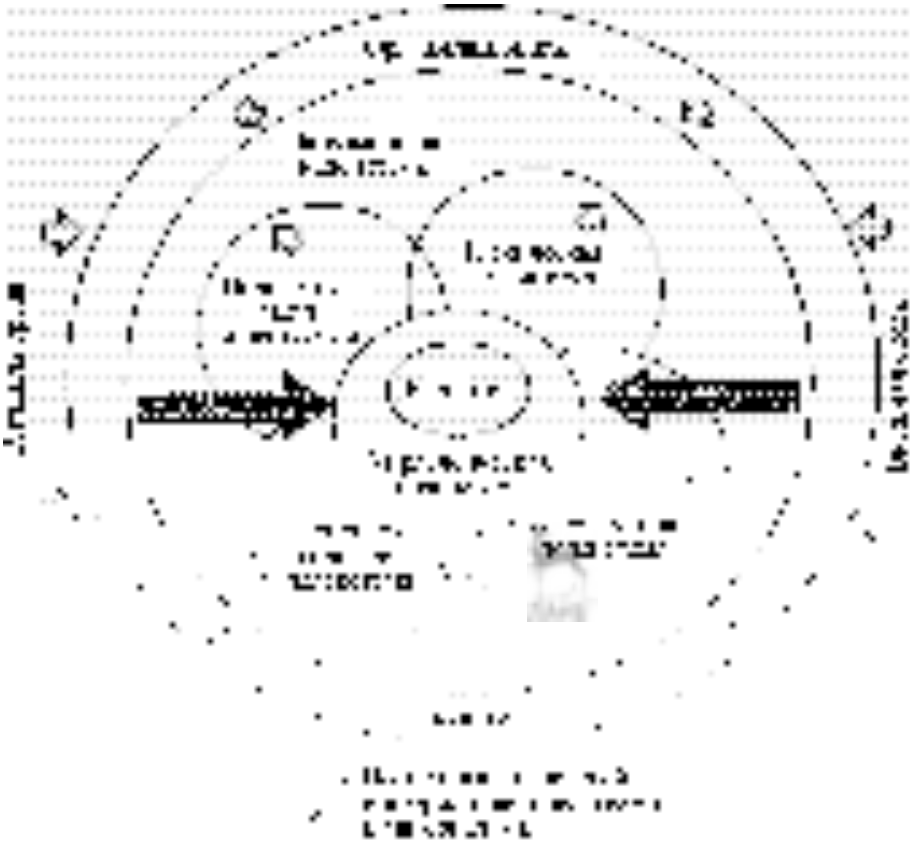


Рис. 2

Подсистемы производственных комплексов, в центре развития которых формируются новые требования к персоналу

Анализ и сопоставление зарубежного опыта цифровой трансформации

В результате проходящей в настоящее время «цифровой революции» государственная политика во всем мире претерпела в последнее время резкие изменения. Сформировать ответ на вызов сохранения конкурентоспособности и достижения высоких темпов производительности в настоящее время призвана проводимая государствами научно-технологическая и инновационная политика. Ее цель — стимулировать разработку и внедрение передовых технологий, которые отличаются высокой производительностью и могут обеспечить наибольший вклад в технологический и экономический рост. Индустриально развитые страны (США, Германия, Великобритания, Япония, Китай, Южная Корея и др.) приняли решение о развертывании новой технологической революции в виде государственной политики. Кроме того, эти государства хотят сосредоточить у себя ключевые универсальные (цифровые) платформы, агрегирующие так называемые стратегические данные и алгоритмы их обработки. Страны-лидеры уже сегодня реализуют целый пакет больших государственных программ в сфере передовых технологий в промышленности и непромышленных секторах экономики, рассчитанных на запуск новой технологической революю-

ции и радикальное укрепление конкурентных позиций на глобальных рынках. Так, в Германии в 2012 г. была инициирована промышленная стратегия «Индустрия 4.0» (Industrie 4.0) как один из десяти «проектов будущего» в рамках «Плана действий по реализации обновленной федеральной Стратегии в области высоких технологий». В США приняты «Стратегия инновационного развития», «Национальный стратегический план развития передовых промышленных технологий США», а также реализуется ряд профильных межведомственных инициатив, таких как «Инициатива генома материалов», национальные инициативы в сфере робототехники и т. п. Великобритания реализует собственный план развития передовых производств, а также программу развития «Восемь великих технологий». В 2013 г. Франция запустила программу «Новая промышленная Франция», в рамках которой реализуются проекты по 10 перспективным технологическим направлениям развития индустрий и технологий будущего. В Японии запущен уже 5-й пятилетний план развития науки, технологий и инноваций (2016–2020). Китай с 2015 г. реализует программы «Сделано в Китае — 2025» и «Интернет+». Кроме того, в июле 2017 г. в КНР был утвержден «Национальный план стимулирования технологических разработок в сфере искусственного интеллекта».

Практически все развитые страны с конца нулевых годов нового века, помимо традиционной активизации промышленной и технологической политики, увеличивают инвестиции в научные исследования — источник прорывных технологий. Особенностью данного инвестиционного цикла является то, что при сокращении государственных бюджетов на НИОКР растут частные инвестиции в исследования и разработки. Это связано с тем, что критическим условием успешности в разворачивающейся технологической гонке является ранний доступ к прорывным технологиям на том этапе, когда они еще не доведены до прототипа, а являются научной гипотезой и ранней технологической идеей. Это заставляет всех, кто включился в данную гонку, переоценить риски, связанные с созданием продукта, переписать ключевые составляющие формулы времени продвижения на рынке (time-to-market). Инвестиции в науку и исследования, увеличение пространства для экспериментов, различные способы ускорения исследований и доведения научных гипотез до полезных продуктов на рынке (R&D fast track programs), коллаборативные механизмы объединения усилий и разделения ответственности (исследовательские и технологические консорциумы) — это широко применяемые в настоящее время инструменты хеджирования рисков.

Учет российских реалий при вступлении в эпоху цифрового бизнеса

Анализируя тематику разного рода конференций и публикаций о трендах развития информационных технологий (ИТ) в 2017 г., нетрудно заметить нарастающую волну направления «цифрового предприятия» (Digital Enterprise). Особенно четко это проявилось в попытке преобразования проекта DOCFLOW с двадцатилетней историей из выставки-конференции по проблемам развития ИТ в мероприятие DigEn, посвященное Digital Enterprise. Данная конференция помимо воли ее организаторов наглядно отразила специфику текущего момента в плане трансформации бизнеса от его «доцифровой» стадии развития к «цифровой». По сути мероприятие состояло из двух мало связанных между собой

частей, в первой из которых шел сугубо теоретический разговор о «цифровом будущем», а во второй — традиционная демонстрация ИТ-решений для нашего «сегодня». Тем не менее, хотя эпоха «цифровой экономики» еще не наступила, уже понятно, что она не за горами и к ней нужно готовиться. И на DigEn были не только представлены «розовые» перспективы цифрового будущего, но и показаны имеющиеся трудности на пути к нему. Сегодня есть общее понимание того, что речь идет о трансформации не просто ИТ-рынка, но всей мировой экономики под влиянием новых возможностей ИТ. Разумеется, степень такой цифровой трансформации зависит от уровня экономического развития и специфики разных стран, а также от вертикальных отраслей. Однако технологии — условие необходимое, но недостаточное. Для реализации Digital Enterprise нужны новые модели организации бизнеса, позволяющие создавать такие предприятия, вся деятельность которых будет базироваться на обработке и анализе данных. И необходим также третий обязательный компонент — стратегия развития, подразумевающая возможность не только постоянного оперативного изменения предприятия в меняющихся условиях окружающего мира, но и заблаговременную подготовку к будущим изменениям на основе точных прогнозов, обеспечиваемых как раз новыми технологиями.

Конечно, цифровая трансформация компаний, производств, отраслей — очень непростое дело: существует классический пример компании «Кодак», которая, став одним из инициаторов цифровой революции в фотографии, как раз в результате начатого преобразования отрасли потеряла свое лидерство. Разумеется, темпы цифровой трансформации различаются в отраслевом разрезе. Лидерами являются ритейл, банковский сектор, телекоммуникационная отрасль, т. е. те, кто в наибольшей степени работает в массовом потребительском секторе, кому нужно быстро реагировать на изменения в спросе и на внедрение новейших технологий. В физическом производстве темпы будут медленнее, и там объектом анализа станет улучшение внутренних производственных процессов.

Одним из активных пропагандистов идей «цифрового предприятия» в России выступает Сбербанк, который не только расширяет использование ИТ в своей внутренней деятельности, но все чаще выступает и как ИТ-провайдер. Например, руководство компании «Сбербанк-Технологии» (дочерняя структура Сбербанка) совсем недавно предложило такое определение цифрового предприятия: «Это организация, способная быстро адаптироваться к быстро меняющимся окружающим условиям, в том числе на основе проактивного прогноза развития ситуации в будущем. Жизнь — это движение, таков её лозунг». Одна из главных характеристик современного мира — высокая скорость его изменения, которая, по мнению экспертов, характеризуется несколькими трендами:

- активы (любые — финансовые, материальные, интеллектуальные) в цифровую эпоху становится легче получать;
- связность общества и его технологического обеспечения повышается: «люди и вещи» постоянно на связи;
- возникают новые производственные системы: появляются возможности использования умных материалов, аддитивных технологий и пр.;
- сложившиеся жесткие организационные иерархии постепенно сменяются гибкими сетями;

- массовые решения и услуги (в здравоохранении, ЖКХ и пр.) становятся персонализированными;
- относительно несовершенные человеческие отношения постепенно заменяются машинными алгоритмами, которые все больше управляют бизнесом и обществом.

В цифровом бизнесе исчезает былое деление специалистов по роду деятельности: «информационщики» должны быть глубоко погружены в деловые бизнес-процессы, а управленцы — отлично знать возможности ИТ и тренды их развития. Например, все ведущие менеджеры Сбербанка не только читают книги по бизнес-аналитике, большим данным и нейронным сетям, но и сдают зачеты по этим дисциплинами. Хотя, конечно, организационная трансформация требует и иных базовых технологий. В частности, Сбербанк сейчас ведет работы по созданию новой технологической ИТ-платформы, которая использует механизмы in-memory, ориентируется на применение аппаратных средств стандартной low-end-архитектуры и идей Open Source. На одну важную проблему следует также обратить внимание: трудность трансформации заключается также во внешних условиях для бизнеса, а именно в нормативно-законодательной базе, которая по-прежнему нацелена на поддержку старой жесткой иерархической системы управления. Существуют примеры трудностей работы внешних поставщиков информационных решений с тем же Сбербанком, который, как государственная компания, должен работать по 223-ФЗ, фактически не позволяющему вносить какие-то изменения в исходное техзадание. А современные командные и итеративные технологии создания ИТ-решений вступают с этими обстоятельствами в противоречие: при этом нормативном подходе невозможно применять Agile-методы. Именно поэтому Сбербанк развивает внутреннее ИТ-направление (в том числе в виде выделенных «дочек»), поскольку оно позволяет взаимодействовать с командами разработчиков в рамках гибких моделей управления.

Следует признать, что сейчас тема «цифрового предприятия» находится пока в стадии Hype («много шума, маркетинга, рекламы») и задачей сегодняшних дискуссий во многом является отделение зерен от плевел, обсуждение реальной содержательной сущности каждого тренда и перспектив его развития. Но уже сейчас ясно, что речь идет не о мыльном пузыре, а о реальных изменениях в экономике, связанных с ИТ: можно сравнить сегодняшний этап зрелости данной темы с разговорами об облаках и облачных сервисах в 2009–2010 гг. А это означает, что цифровое предприятие, основанное на информационных технологиях, должно отвечать модели рисков, причем весьма значительных. Нужно быть готовым к тому, что придётся провести десять пилотных проектов, в то время как реального результата достигнет лишь пара из ваших начинаний. Главное достижение и одновременно главная проблема современных ИТ-систем — это их сложность. Дальнейшее повышение их производительности и адаптивности даже при неизменном составе решаемых бизнес-задач невозможно без радикального пересмотра бизнес-архитектуры и организационного обеспечения. Это можно сделать на базе концепции иерархической эмерджентной стратификации систем при организации разработки и внедрения ИТ, реализуя отдельные рабочие места в виде своеобразных «инфороботов», которые смогут на основе имеющихся данных выполнять часть процессов в автономном режиме. Многие руководители

ИТ-компаний сегодня постулируют: если Сбербанк и другие традиционные компании становятся ИТ-поставщиками, то и собственно ИТ-бизнес должен заниматься другими видами деятельности, используя свои преимущества, в том числе в плане умения эффективно управлять данными. ИТ-специалисты должны погружаться в бизнес-схемы не только на уровне отношений внутри отдельного предприятия, но и на уровне самостоятельного бизнеса. Собственно, известный пример компании Uber как раз об этом и говорит.

Тренды цифровой трансформации, связи с общественностью и рекламный рынок

Рассматривая проблему перехода к цифровой экономике, невозможно не замечать ту сторону больших данных, которые относятся к СМИ, связям с общественностью и просто рекламному бизнесу. Международные компании в своих информационных кампаниях все чаще пользуются цифровыми носителями. Например, недавний отчет Digital PR and Communications Report, авторитетной британской ассоциации консультантов по связям с общественностью PRCA, отмечает, что средний процент от маркетингового бюджета компаний, потраченного на цифровые социальные медиа, к осени 2016 г. составил 25% по сравнению с 16% в предыдущем. При этом участники исследования ожидают роста digital-затрат в ближайшие год-полтора до 62%. Соответственно, чтобы не просто удержаться на плаву, а достичь преимущества, компаниям имеет смысл уже сейчас активнее интегрировать новые технологические инструменты коммуникации в продвижение своих продуктов и услуг.

Структура групп, на которые планируются воздействия при формировании связей с общественностью (рис. 3), определяет и выбор цифровых технологий, которые могут быть использованы в перспективном развитии этого направления деятельности компаний. Следует рассмотреть, какие ключевые тренды в перспективной области digital-продвижения сейчас наиболее актуальны.

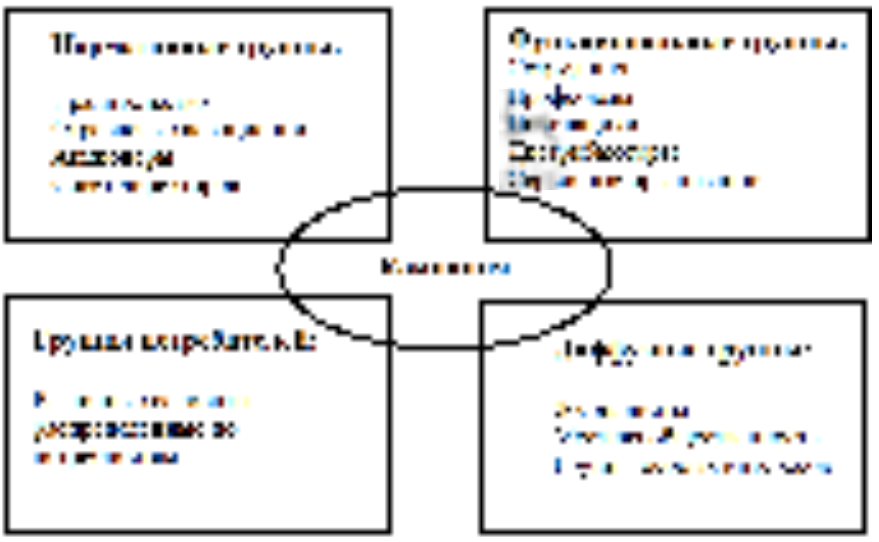


Рис. 3

Структура групп, на которые планируются воздействия при формировании связей с общественностью в любой компании

Тренд № 1: рост использования Интернета вещей (*IoT*)

Об Интернете вещей говорят уже давно и много, но по-прежнему далеко не все понимают, как правильно использовать его возможности в коммуникациях с клиентами. Сейчас в России IoT применяют в основном в промышленном секторе, на производствах, а также в инфраструктурных объектах в рамках концепции «Умный город» (камеры видеонаблюдения, датчики энергопотребления, расхода тепла и т. д.). В то же время на Западе большую популярность приобрело применение интеллектуальных устройств на персональных продуктах, прежде всего потребительской электроники. Особенно актуально это для сферы розничной торговли, где уже предлагаются соответствующие решения. К примеру, компания Intel предлагает платформу Intel Retail Sensor, которая использует специальную аналитическую систему на базе RFID-меток (радиочастотная идентификация), размещенных на продаваемых в магазине товарах, в комплексе с камерами видеонаблюдения и интерактивными экранами Digital Signage (видеореклама на мониторах). Система отслеживает действия покупателя, определяет его социально-демографический портрет и на основе полученных данных делает персональное предложение. Например, покупатель берет рубашку в руки, а на экране рядом приятный женский голос предлагает приобрести к ней галстук со скидкой. Частично эта технология Intel уже применяются в магазинах Levi Strauss & Co., Brooks Brothers и Nordstrom. Из отечественных ритейлеров в начале 2017 г. сеть Media Markt внедрила в своих магазинах электронные ценники, которые в режиме реального времени отображают цены и информацию по ассортименту. Они позволяют не только вывести на экран всю необходимую информацию о продукте, но и использовать устройства как медианосители. В скором времени с помощью NFC-технологий и анализа покупательского поведения клиентам будут предлагаться индивидуальные условия покупки. Это позволит привлечь дополнительное внимание покупателей и, как следствие, увеличить продажи.

Тренд № 2: использование виртуальной реальности (*VR*) *и дополненной реальности* (*AR*)

Технологии виртуальной и дополненной реальности стремительно из области развлечений переходят в средство коммуникации с клиентами для продвижения продуктов компаний. Доступность контента, очков и смартфонов делает VR-технологии удобным каналом коммуникаций с рядовыми потребителями. Первые шаги к массовому вовлечению пользователей с помощью VR-технологий уже сделал Facebook, анонсировав в 2017 г. бета-версию продукта Spaces. Пользователи сервиса будут подключаться через свои аккаунты в Facebook и погружаться в виртуальное пространство, где общение осуществляется с помощью голоса и движения тела. Там можно общаться, рисовать, смотреть сферические ролики, совершать видеозвонки через Facebook Messenger и создавать VR-селфи. В таком пространстве можно будет предлагать пользователям как виртуальные, так и физические товары, рекламировать услуги, создавать сообщества по интересам, обсуждать важные темы. Что касается традиционного маркетинга, там уже сейчас набирает популярность демонстрация клиентам продуктов и услуг с помощью VR- и AR-технологий, из актуальных направлений — VR-туры по недвижимости. Отечественные девелоперы элитной недвижимости предлагает своим потенциальным клиентам с помощью технологий виртуальной реальности (специально разработанной программы, уста-

навливаемой в виде приложения на смартфон, и VR-очков), не вставая с дивана, познакомится с жилым комплексом, изучить планировку квартир, интерьер, испытать эмоции, сравнимые с реальным присутствием на объекте. Необычное применение AR-технологий можно наблюдать на примере «умной фермы» от компании DeLaval. Здесь клиенту, чтобы увидеть работу молочной фермы, не обязательно ехать в коровник, достаточно воспользоваться специальным голографическим столом и очками, которые позволят в деталях рассмотреть, например, процесс дойки коров. О финансовых эффектах подобных инструментов судить пока сложно: это выставочный инструмент, его задача продемонстрировать масштабное оборудование на обычном столе.

Тренд № 3: массовое использование мессенджеров (*Telegram-каналы*)

Только в 2016 г. за счет стремительно растущего числа каналов аудитория активных пользователей мессенджера Павла Дурова выросла в 3 раза до 6 млн человек, зацепив самую продвинутую аудиторию российского Интернета, так называемых «трендсеттеров»: журналистов, блогеров, пиарщиков, а актуализировалась только сегодня. Как по теоретической «диффузной» модели коммуникации Э. Роджерса — от «новаторов» к «ранним последователям». Поэтому именно сейчас самое время заводить канал для распространения информации, если нацеливаться на завтрашнее «раннее большинство». Особенно это актуально для компаний, чей бизнес связан с Интернетом. На сегодняшний день свои успешные каналы уже имеют мобильные операторы, банки, продавцы онлайн-услуг. Уже вышло немало статей на тему роста количества «полезных» телеграм-каналов. При этом необходимо осознавать, что в свете последних событий есть вероятность «антитеррористической» блокировки таких систем. Некоторые каналы уже предложили пользователям подписаться на аналогичные страницы в Facebook. Несмотря на существующую угрозу, скорее всего компромисс в том или ином виде будет найден противоборствующими сторонами, что будет играть на руку специалистам по PR.

Тренд № 4: широкое использование возможностей YouTube

Система YouTube в последнее время обретает новое дыхание, не раз уже хоронили эту сеть, но с каждым годом она наполняется все новыми смыслами. Новый виток развития связан с «цифровым поколением Z» (возраст 13–24 лет), потребляющим информацию исключительно визуально. Это подтверждает недавнее исследование Google, в котором утверждается, в частности, что четверть этих пользователей используют YouTube для поиска релевантных ответов на многие жизненные вопросы. То есть видеоконтент становятся для них главным источником не только развлечений, но и быстрых ответов на возникающие вопросы. Фактически происходит замещение традиционных поисковиков типа Google и Yandex. При этом, несмотря на то что молодые пользователи часто не обладают финансовыми возможностями для совершения покупки, их мнение влияет на выбор старшего поколения, особенно в вопросах гаджетов и трендовых продуктов. Помимо этого, сеть с конца 2016 г. начала внедрять в интерфейс каналов новый инструмент общения «Сообщество». Теперь создатели каналов смогут размещать свои сообщения, не обязательно привязывая их к видео, они могут содержать только текст, картинку или просто ссылку. Инструмент уже доступен для некоторых западных блогеров и в ближайшее время появится и у отечественных.

Тренд № 5: развитие PR-Digital

Если раньше ключевая функция PR-специалиста сводилась к общению с традиционными или гибридными СМИ (Media relations), а для работы в социальных сетях привлекался SMM-специалист (существовало разделение на офлайн и онлайн), то в условиях digital-трансформации две эти компетенции окончательно слились воедино. В настоящее время ключевой компетенцией PR-специалиста стало управление всеми каналами, при этом модель односторонней коммуникации изменилась на двустороннюю, где во главе угла — вовлечение аудитории как более эффективный способ транслирования информации.

Подводя итог обзору трендов и тенденций, следует отметить, что цифровая трансформация — это реалии сегодняшнего дня, которые необходимо учитывать. Если компания намерена оставаться конкурентоспособной или повышать эффективность, она должна меняться в соответствии с требованиями современности.

Этапы перехода к использованию цифровых технологий в жизненном цикле сложных систем

Реализацию конкретных мер по формированию цифрового экономического пространства и внедрению «сквозных» технологий программы «Цифровая экономика Российской Федерации» по приоритетным направлениям на период до 2024 г. можно условно разбить на два этапа.

На первом этапе (2021–2020) основные действия должны быть направлены на запуск организационных изменений и пилотных проектов: этот этап может объединить все мероприятия, связанные с нормативным регулированием приоритетных сфер научно-технологической политики, организационными изменениями, выработкой новых стандартов и механизмов финансирования предлагаемых изменений. Второй этап (2021–2024) должен быть связан с переходом к действию в регулярном режиме в масштабе всей экономики: вступление в силу и применение правовых норм и организационных изменений, совершенных на предшествующем этапе, масштабирование содержательных мер, предусмотренных к реализации по каждому из направлений. Целесообразно отметить, что для обеспечения дальнейших изменений по итогам двух этапов реализации программы может быть подготовлен новый стратегический пакет действий на перспективу примерно до 2035 г., который будет отталкиваться от уже достигнутых результатов включения России в новую глобальную технологическую реальность, актуальных мировых тенденций технологического развития и позиции России на глобальных высокотехнологичных рынках. В этот период на базе передовых производственных технологий может быть совершена смена моделей развития ряда ключевых секторов российской экономики:

- в сфере энергетики — переход к цифровой и интеллектуальной энергетике, энергетическим мультиагентным системам, развитие технологий так называемой «постуглеродной энергетики», включая масштабное развитие атомной энергетики и ВИЭ;
- в сфере природных ресурсов — переход к предельно рациональному использованию природных ресурсов России, а в отдаленной перспективе — к использованию природоподобных технологий;
- в сфере здравоохранения — переход к «4П»-медицине (предсказательной, превентивной, персонализированной, партисипативной, т. е. предполагающей активное управление здоровьем самим пациентом), а затем и к «5П»-

медицине (так называемой платформенной медицине, т. е. базирующейся на общих программно-информационных и продуктовых платформах);

- в сфере финансовых технологий и управления — переход к мобильным расчетам и электронным финансам, в перспективе — к расчетам на базе технологий распределенных реестров (блокчейн) и управлению на основе алгоритмического регулирования и так называемых «умных» контрактов (автоматических и роботизированных сделок, большая часть которых осуществляется программными агентами);
- в сфере агропромышленного производства — создание новой отрасли «продовольственной системы», включающей в себя не только часть сельского хозяйства (высокоточное земледелие, «Интернет вещей» в АПК, широчайшее распространение биотехнологий, включая генную инженерию и пр.), но и требования к логистическим и торговым системам (короткие цепочки поставок, обеспечение качества продовольствия, рациональное использование последнего и сокращение отходов), к индивидуальным диетам, функциональным продуктам.

Цифровые оценки промышленной коммерциализации технологий

Промышленная коммерциализация технологий — деятельность, направленная на получение дохода от использования результатов научных исследований. Это самое простое определение понятия, учебное пособие по которому вы держите в руках.

Рыночные реформы в России изменили лицо российской науки, как следствие — произошло изменение в подходах в формировании отечественных производственных технологий, цифровая трансформация затронула целый ряд отраслей. Эти изменения проявились прежде всего в децентрализации управления и изменении схем финансирования научных исследований. Однако российская наука по-прежнему остается по сути государственной: подавляющее большинство научных учреждений, экспериментальное оборудование и опытные производства принадлежат государству, а ученые получают заработную плату в основном из бюджетных средств. В этом заключается основное отличие нашей науки от западной, где существенная доля исследовательских работ выполняется в частных лабораториях и научных центрах. В то же время на протяжении последних лет наша наука финансируется на уровне нескольких десятков процентов от требуемого объема средств.

Многие экономисты и политики понимают необходимость серьезной реорганизации системы финансирования науки, что означает прежде всего более широкое привлечение частных инвестиций. Однако частные инвесторы вкладывают средства в научные изыскания только в расчете на будущую прибыль. Это осуществляется путем закрепления прав инвесторов на результаты научных исследований и использования их в дальнейшем для производства новых товаров или для последующей перепродажи. Фактически, частные инвестиции в науку означают приобретение опциона на специфический товар — интеллектуальную собственность, — способный в дальнейшем приносить прибыль. Для того чтобы стать таким товаром, научные исследования должны быть облечены в соответствующую «упаковку», что означает как минимум проведение предварительного комплексного изучения финансовых и рыночных перспектив научных разработок и доказательства их будущих экономических и технологических преимуществ.

Описанный выше процесс инвестирования в перспективные научные разработки носит название *коммерциализации технологий.* Коммерциализация предполагает прогнозирование рынков, поиск, экспертизу и отбор разработок для финансирования, привлечение инвестиций, распределение и юридическое закрепление прав на будущую интеллектуальную собственность между всеми участвующими в процессе сторонами, управление научным проектом, внедрение результатов в производство, дальнейшую модификацию и сопровождение интеллектуального продукта. Желательно, чтобы все эти мероприятия выполнялись в рамках цифровой платформы, где практически автоматически можно проводить расчеты вариантов.

К сожалению, в силу оторванности науки от бизнеса развитие технологий в России происходит по своим собственным законам, без учета конкретных потребностей промышленного производства, и в большинстве случаев — без использования цифровых технологий. Чаще всего процесс коммерциализации начинается в результате счастливой встречи ученого или инженера — носителя некоторой передовой идеи — и предпринимателя, способного эту идею оценить и поддержать в финансовом плане. То, что такие встречи происходят редко, и мы имеем на сегодняшний день не так много примеров успешной коммерциализации технологий, объясняется прежде всего тем, что у российских участников этого процесса отсутствует понимание механизмов функционирования современного рынка интеллектуальной собственности и правил игры на нем. Особенно эта тенденция проявляется у ученых, которые воспринимают коммерциализацию своих идей прежде всего как проблему поиска финансовых средств для продолжения исследований. Многие ученые достаточно поверхностно относятся к вопросам оценки окупаемости своих разработок и необходимости учета интересов инвесторов. Все эти обстоятельства, безусловно, осложняют работу по поиску перспективных для коммерческого использования технологий в России.

Важно отметить также отличие коммерциализации технологий и понятия «трансфер технологий». Понятие «трансфер технологий» появилось в русскоязычной научной литературе сравнительно недавно и напрямую связано с переориентацией на рыночные отношения в большинстве сфер человеческой деятельности. Часто его употребляют в связке с другим понятием — «коммерциализация технологий», хотя смысловое содержание этих понятий неодинаково. Англоязычное слово «трансфер» успешно заместило насильственный термин «внедрение», которым административно-командная система наградила процесс претворения в жизнь инновационного предложения. Однако это не простое замещение, а существенное преобразование смысла процесса. Вместо насильственного «внедрения» (предполагающего активное или пассивное сопротивление среды, в которую производится это «внедрение» чего-то инородного) «трансфер» предполагает не только передачу информации о новшестве, но и ее освоение при активном позитивном участии и источника этой информации (например, автора изобретения), и реципиента, приемника и реализатора на практике информации о новой технологии, и конечного пользователя продукта, производимого с помощью этой технологии. Поэтому, кстати, основной акцент при трансфере технологии делается не столько на технологии как таковой, сколько на субъектах — участниках этого процесса.

Понятие «коммерциализация технологии» предполагает обязательное коммерческое использование информации о технологии, т. е. использование с обязательным извлечением выгоды. Чаще всего эта выгода непосредственно измеряется

в конкретных денежных единицах, гораздо реже — в тех же единицах, но опосредованно, через, например, увеличение эффективности другой технологии. Но деньги в этих расчетах присутствуют всегда и являются определяющим критерием успешности процесса. В то же время вопрос о том, кто, какой субъект осуществляет непосредственное использование технологии, при коммерциализации не является первостепенным, и в частности коммерциализацией нередко пытается заняться сам автор, первоисточник новой технологии (физическое лицо или организация).

Исторический пример масштабной коммерциализации промышленных технологий

Успешным и в своем роде уникальным опытом такого рода в области распространения передовых промышленных технологий можно назвать в эпоху деятельности административно-командной системы программу Совета экономической взаимопомощи (СЭВ) в области атомной энергетики в 1970-х гг. Уникальность и значимость Программы заключалась в том, что за достаточно короткий отрезок времени в странах — членах СЭВ был создан мощный производственно-промышленный потенциал для изготовления специализированного оборудования для АЭС, сооружаемых в этих странах (локализация), что позволило обеспечить непрерывное и поточное строительство АЭС в Чехословакии, а также сооружение АЭС «Пакш-2» в Венгрии и Ровенской АЭС в СССР (ныне — Украина) и др. Были определены предприятия — участники системы кооперации, на которые комплексно и последовательно был осуществлен трансфер технологий изготовления оборудования для АЭС с ядерными реакторами типа ВВЭР. Это создавало экономико-технический базис сотрудничества и дало значительный рывок в развитии атомного энергомашиностроения стран-участниц.

На рисунке 4 представлена схема из документа Международного хозяйственного объединения (МХО) «Интератомэнерго», показывающая распределение обязанностей между социалистическими странами при производстве специализированного оборудования АЭС с серийными блоками ВВЭР-1000. Этот пример свидетельствует о том, что в любой общественно-экономической формации находятся инструменты и механизмы, которые способствуют распространению передовых технологий за рубежи отдельных государств, фактически реализуя подходы, которые в современную эпоху характеризуются термином «глобализация». Таким образом, экономические принципы и закономерности являются объективной реальностью, приспособление которых к определенной политической ситуации является скорее искусством, чем наукой. В то же время международное разделение труда, параметры которого в современную эпоху постоянно изменяются и модифицируются, является устойчивым трендом, и все инструменты промышленной коммерциализации технологий и реализации крупных промышленных инноваций, описанные в настоящем пособии, — это живая экономическая реальность. Российская промышленность, базовые параметры которой закладывались еще в бывшем СССР, должна развиваться и совершенствоваться, перенимая в зарубежном опыте самое передовое, но не растрачивая и положительного потенциала, накопленного десятилетиями упорного труда наших предшественников. В последующих разделах пособия будут приведены отдельные примеры и зарубежного опыта (Южная Корея, КНР) трансфера ядерных технологий, что свидетельствует об объективном характере процессов глобализации развития наукоемких технологий и локализации производственных возможностей их осуществления.

Рис. 4

Описание специализации стран — членов СЭВ и СФРЮ в производстве и поставках оборудования для АЭС с реакторами ВВЭР-1000 в эпоху деятельности СССР и стран социалистического лагеря

Западный бизнесмен прекрасно понимает, что одним-единственным удачным изобретением, доведенным до массового производства, он может обеспечить безбедное будущее себе и своим потомкам, поэтому он активно ищет такое изобретение и вкладывает деньги в изобретение и в изобретателя (разумеется, за соответствующую долю в будущем предприятии). Однако он понимает и то, что это вложение средств — долгосрочное, это не челночный торговый бизнес с длительностью «производственного цикла» в недели, а многостадийный и тщательно планируемый процесс, сочетающий прогнозирование потребностей и научное исследование технологии, и ее масштабирование и оптимизацию, и маркетинг, и заботы о сбытовой сети, и многое-многое другое.

Менталитет же отечественных бизнесменов в большинстве случаев пока еще далек от такого скрупулезного подхода и поиска революционных технологий или продуктов, да и с финальной частью проблемы (маркетинг и сбыт) у нас пока отношения отнюдь не блестящие. Поэтому мы имеем достаточно сложную картину: по широким российским научным и технологическим полям рыщут акулы развитого капитализма и скупают на корню самые экономически перспективные инновационные разработки. Причем они бы и рады организовать производство по этим разработкам здесь же (хотя бы из-за дешевизны рабочей силы), но, с одной стороны, легионы «доброжелателей», воспитанные в известном духе, ставят столько препятствий, что никакого терпения все это понять и пережить не хватает, а с другой стороны, наличная производственная база и ее владельцы настолько озабочены перманентным форс-мажором, что найти с ними общий язык преуспевающему западному бизнесмену непросто.

Еще одна особенность словоупотребления в этом курсе состоит в несколько необычном употреблении слова «технология». В действительности «видимым» предметом трансфера может быть и конкретный объект предметного типа (например, материал, система, микропроцессор нового типа и пр.), который сам по себе технологией не является. Однако и в этом случае с инновационным предметом обязательно связано то или иное умение, искусство, та или иная технология (либо технология производства, либо технология применения или осуществления), поэтому такие словосочетания, как «коммерциализация технологий», так и «трансфер технологии», всегда несут вполне определенный реальный смысл. Этот смысл становится еще более понятным, если учесть, что наиболее надежным способом защиты любого объекта и любой технологии как интеллектуальной собственности является не патент, а так называемое «нераскрываемое ноу-хау».

Ведь почему наши ведомства, занимающиеся экспортом технологий, так настойчиво выясняют, есть ли в предлагаемой на экспорт технологии это самое «нераскрываемое ноу-хау»: именно потому, что обойти практически любой патент для современных мастеров этого дела не составляет никакого труда, если патентный документ содержит информацию, достаточную для запуска технологии. А доказывать нарушение патентных прав — это такая дорогостоящая и сомнительная (в смысле успешности) операция, которая не только нам, но и более богатым странам представляется крайне нежелательной.

Так что в настоящем пособии смысл понятия «технология» ближе всего не к устоявшемуся производственному смыслу, а к его прямому переводу (tehne —

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.06 Аналитическое моделирование в проектировании автоматизированных систем

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Декан факультета информационных систем и технологий
_____ Святов К.В.
«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Дисциплина (модуль) Аналитическое моделирование в проектировании автоматизированных систем
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Аналитическое моделирование в проектировании автоматизированных систем

А.А.Куликова, В.Н.Негода

Аннотация. Методические материалы предназначены для студентов магистратуры кафедры ВТ УлГТУ, обучающихся по направлению «Информатика и вычислительная техника».

Излагается методика построения онтологических моделей (ОМ) проекта, использования графовых СУБД для хранения ОМ и использования их спецификаций для генерации программного кода. Методика способствует снижению рисков нарушения концептуальной целостности, непротиворечивости и полноты всего набора проектных решений, а также риски нарушения сроков и бюджета проекта автоматизированной системы (АС). В ходе разработки АС используется широкий спектр инструментальных средств, которые обеспечивают автоматизацию многих проектных работ через множество существенно различающихся информационных процессов. Интеграция всех этих информационных процессов и их результатов достаточно эффективно автоматизируется средствами онтологического моделирования.

Методические материалы включают в себя изложение общих понятий, связанных с онтологическим моделированием, методологии применения концепции проектного мышления (design thinking), основы представления онтологических спецификаций средствами графовой СУБД NEO4j и язык запросов Cypher, описание информационных процессов автоматизированного проектирования АС, описание примера проектирования системы логического управления роботом-плиткоукладчиком, охватывающего все этапы, вплоть до генерации кода на основе онтологических спецификаций.

Ключевые слова: онтологические модели, автоматизированные системы, проектное мышление, автоматизация проектирования

Введение

Потребность в онтологическом моделировании возникает при проектировании автоматизированных систем (АС) в связи со многими обстоятельствами, основные из которых представлены ниже.

Во-первых, в разработке достаточно масштабных АС принимают участие десятки и даже сотни самых различных специалистов: системных аналитиков, проектировщиков, программистов, тестировщиков, технических писателей и прочих. Более того, даже в рамках одной и той же роли существуют различные специализации и ролевые ответственности. Например, разработка подсистемы сопряжения с техническим объектом управления и разработка интерфейса с человеком-оператором заметно различаются, что требует участия в разработке разных специалистов. Одни должны хорошо разбираться в физике сенсоров, теории и технике измерений, особенностях реализации требований режима жесткого реального времени. Другим нужно знать эргономику человеко-машинных интерфейсов, возможности и ограничения операторских панелей и средств программирования диалога, а также хорошо понимать особенности поведения объекта и человека-оператора в той части функционирования пары «АС – объект управления», где возможно вмешательство оператора. Все это требует вовлекать в разработку знания из различных предметных областей, каждая из которых имеет свой профессиональный язык, изобилующий инженерными жаргонизмами. Для обеспечения единообразного понимания проектных спецификаций всеми разработчиками в таких условиях целесообразно снабжать документацию автоматизированными тезаурусами, включающими в себя словари задействованных предметных областей и отношения между понятиями, а также поддерживающими возможности вовлекать объекты словарей в интерпретацию спецификаций проектных решений средствами автоматизации проектирования (САПР).

Во-вторых, процесс разработки достаточно сложной АС является длительным, причем, многие работы выполняются параллельно. В этих условиях обеспечение концептуального единства и непротиворечивости проектных решений очень непросто. Для оперативного согласования всех проектных решений приходится слишком много времени тратить на поддержание активных отношений между разработчиками. Затраты времени на эту работу и обеспечение непротиворечивости можно существенно сократить, если в ходе документирования явным образом специфицировать связи и взаимозависимости, относящиеся к разным проектным решениям, обеспечив при этом интерпретируемость сформированных спецификаций средствами САПР.

В-третьих, довольно частой причиной неуспешности проектов АС является неодинаковое понимание требований заказчиками и разработчиками.

Изначальная причина этого - неполнота и неопределенность в спецификациях требований технического задания. На этапе разработки и реализации проектных решений неполнота и неопределенность воспринимается как свобода выбора, позволяющая выгодно вовлекать в проект типовые и даже уже реализованные проектные решения. На фоне значительной похожести многих автоматизируемых процессов в разных АС кажется естественным доопределять детали технического задания тем, что уже несколько раз фигурировало в других проектах. В результате упускаются специфические детали и вполне конкретный автоматизируемый процесс заменяется на некоторый типовой. Более того, детали часто упускаются при сочетании сжатых сроков разработки с большим объемом проектных спецификаций и высокой степенью разделения труда проектировщиков. Несоответствие проектных спецификаций одних этапов проектирования спецификациям других в проектировании АС часто называют информационным разрывом. Важным способом уменьшения информационных разрывов является обеспечение доминирования результатов анализа требований в проектных спецификациях всех последующих этапов разработки.

В-четвертых, при проектировании достаточно сложной АС качество проектных решений часто обеспечивается через верификацию проектных решений всех этапов, т.е. проверку их соответствия изначальным требованиям заказчика. Большое количество проектных спецификаций существенно усложняет эту работу. Для ее автоматизации важно обеспечить интерпретируемость спецификаций требований и других проектных спецификаций. На различных этапах проектирования зачастую используются различающиеся инструменты и форматы данных, что существенно усложняет задачу автоматизации верификации. В этой связи целесообразно расширять использование универсальных и достаточно легко интерпретируемых представлений проектных спецификаций на всех этапах проектирования.

В-пятых, в ходе достаточно продолжительного процесса проектирования АС часто возникают изменения требований заказчика. Это происходит в связи с объективными изменениями в автоматизируемых процессах, а также в связи с развитием представлений заказчиков о возможностях автоматизации. Рост аппетита заказчика к расширению или изменению функциональных возможностей АС больше всего стимулируется позитивными эффектами, обнаруживаемыми представителями заказчика или иными потенциальными пользователями в ходе знакомства с прототипами создаваемых компонентов. Изменения требований довольно коварны, поскольку внесение одного изменения в спецификации требований может потребовать большое число изменений во многие проектные решения, специфицированные в самых разных документах, исходных кодах, тестах. В отработку одного изменения могут быть вовлечены десятки исполнителей. Чтобы обеспечить полноту изменений и их

согласованность, целесообразно иметь такие представления связей и взаимозависимостей проектных спецификаций, которые за счет их интерпретируемости облегчают автоматическое формирование трасс изменений.

Все приведенные выше обстоятельства порождают риски нарушения концептуальной целостности, непротиворечивости и полноты всего набора проектных решений, а также риски нарушения сроков и бюджета проекта АС. В ходе разработки АС используется широкий спектр инструментальных средств, которые обеспечивают автоматизацию многих проектных работ. Интеграция всех этих инструментальных средств предполагает рассмотрение процесса автоматизированного проектирования АС как информационного процесса, на каждом шаге которого фигурируют входные спецификации, проектная процедура и выходные спецификации, являющиеся результатом выполнения этой процедуры. Часто выходные спецификации являются некоторым приращением к множеству всех проектных спецификаций. Именно это объясняет активное использование терминов «инкрементальное конструирование», «инкрементальное проектирование».

Сама проектная процедура предполагает исполнение нескольких проектных операций, значительная часть из которых выполняется автоматически средствами автоматизации проектирования, однако имеются операции, которые должен выполнять проектировщик. Наиболее трудно автоматизируются операции, требующие интерпретации плохо формализованных спецификаций. Онтологическое моделирование является технологией, которая позволяет расширить пространство автоматически интерпретируемых спецификаций.

1 Онтологии в информатике и компьютерные технологии работы со знаниями

1.1 Понятие онтологии

Онтология в информатике есть ***способ формализации некоторой области знаний с помощью концептуальной схемы***. Данный термин является производным от философского понятия онтологии, которое определяется как раздел философии, представляющий собой учение о сущем (или бытии) – его сущностях, категориях, структуре и закономерностях.

Наиболее известное определение онтологии в информатике принадлежит американскому ученому Томасу Груберу: «***Онтология – это точная спецификация концептуализации***». Более поздняя модификация данного определения сформулирована следующим образом: «онтология есть формальная спецификация согласованной концептуализации». Данная формулировка подчеркивает наличие формальной структуры в онтологии, а также согласованность концептуализации, которая заключается в том, что описываемое знание является общим для некоей группы людей, использующих онтологию.

В специализированной литературе можно встретить и другие, более детализированные определения онтологии. Однако, в общем случае онтология характеризуется наличием структуры, состоящей из следующих компонентов:

- ***Понятия (концепты)*** – термины, характерные для области знаний (предметной области, проекта, системы), описываемой с помощью онтологии. В некоторых источниках под понятиями скрываются *классы* объектов, которые образуют иерархию: в этом случае в структуре онтологии также предусмотрены *экземпляры* таких классов (при этом, наличие экземпляров в онтологической модели не является обязательным).

  Пример: «Модуль управления светофором на перекрестке ул. Гагарина и ул. Орлова» является экземпляром класса «Модули управления светофорами», который связан с классом «Средства управления дорожным движением» иерархическим отношением «быть подклассом».

- ***Атрибуты*** – свойства понятий. Служат для хранения информации, характерной для того или иного понятия.

Пример: понятие «Модуль управления светофорами» может иметь такие атрибуты, как «Тип», «Время цикла управления», «Поддержка слежения за движением», «Число режимов управления» и т.п.

- ***Отношения*** – зависимости между понятиями, несущие семантическую нагрузку. В зависимости от конфигурации онтологии отношения могут являться как отдельными сущностями, так и частными случаями *атрибутов*, значениями которых выступают другие понятия.

  Пример: понятие «Индикатор времени» связано с понятием «Модуль управления светофорами» отношением «быть частью»; это отношение в спецификации онтологии может быть представлено либо атрибутом «быть частью» со значением «Модуль управления светофорами», либо элементом множества отношений, содержащим код типа отношения «быть частью» и две ссылки: на понятие «Индикатор времени» и «Модуль управления светофорами».

К дополнительным структурообразующим элементами онтологической модели можно отнести ***аксиомы***, т.е. утверждения относительно понятий, атрибутов и отношений онтологии, которые всегда считаются истинными. Аксиомы служат для различных целей: например, проверки модели на непротиворечивость, задания некоторых ограничений модели или вывода новых знаний. В проектировании автоматизированных систем аксиомы являются хорошей основой для представления элементов палитры компонентов, которые используются в готовом виде для встраивания в спецификации проектных решений через механизмы композиции.

Конструктивное использование онтологических моделей в автоматизированном проектировании АС предполагает, что инфологические представления онтологических моделей становятся частью проектных спецификаций. Именно это обстоятельство облегчает обеспечение интерпретируемости представлений проектных решений – интерпретатор языка специфицирования представления онтологической модели

Логико-алгебраическая Интересам поддержки автоматизированного проектирования в большей степени удовлетворяет понятие «компьютерная онтология предметной области», которое определяется как тройка:

$O = <X, R, F>$,

где

$X = \{X_1, X_2, \ldots, X_n\}$ − конечное множество концептов (понятий-объектов) заданной АС; каждый концепт в формальной модели представляется кортежем, в полях которого фигурируют атрибуты.

$R = \{R_1, R_2, \ldots, R_m\}$, $R \subseteq X_1 \times X_2 \times \ldots \times X_n$, − конечное множество семантически значимых отношений между понятиями-объектами ПО; спецификатор отношения также является кортежем, представляющим атрибуты (свойства) отношения;

F: $X \times R$ − конечное множество функций интерпретации, заданных на понятиях-объектах и/или отношениях; эти функции поддерживают анализ атрибутов совокупностей концептов и отношений и порождение на выходе информационных объектов, которые могут быть логическими выводами в формате предикатов, подмножеством онтологических спецификаций, продуктом трансформации онтологических спецификаций.

Частным случаем задания множества функций интерпретации F является глоссарий, составленный для множества понятий X

Частным случаем аксиом являются ***функции интерпретации***, как способ содержательного толкования онтологической модели, с помощью которых одним сущностям онтологической модели ставятся в соответствие другие.

Пример аксиомы: Если роботу поступает сигнал START, то он должен начать движение.

В данном утверждении задействованы понятия «Робот», «Сигнал START», «Начало движения». При этом, если понятие «Сигнал START» связано с понятием «Робот» отношением «поступать», то понятие «Робот» должно быть связано с понятием «Начало движения» отношением «осуществлять». На языке логике предикатов это выглядит следующим образом:

```
поступать(Сигнал START, Робот) → осуществлять(Робот, Начало движения)
```

1.2 Компьютерные технологии для работы с онтологиями

1.2.1 RDF Triple Store

Существует несколько формальных языков для описания онтологических моделей. Наиболее распространенным на сегодняшний день является язык ***OWL (Web Ontology Language)***, который является стандартом Консорциума Всемирной паутины W3C – язык для семантических утверждений, разработанный как расширение стандартов *RDF (Resource Description Framework)* и *RDFS (RDF Schema)*. Однако, специалисты в области онтологического моделирования обычно используют на практике смесь из выражений всех трех стандартов.

Все три стандарта базируются на формализмах дескрипционной логики с использованием синтаксиса XML. В основе данных стандартов лежит особая

синтаксическая структура, именуемая ***триплетом***. Триплет есть набор из трех сущностей, который формальным образом описывает некое знание и в переводе на термины синтаксиса естественного языка соответствует трем основным членам предложения: подлежащее (объект), сказуемое (предикат) и дополнение (субъект).

Базовые классы и свойства онтологических моделей заданы конструкциями стандартов. Так, например, предикат «быть классом» задается с помощью субъекта `owl:Class` (для OWL) или `rdfs:Class` (для RDF), а для отнесения объекта к типу используется свойство `rdf:type`.

Пример: для объявления класса «Человек» могут быть задействованы следующие триплеты:

- `my_ontology:Human – rdf:type – owl:Class` (что дословно означает: «В онтологии my_ontology есть класс Human»; префиксу `my_ontology` соответствует пользовательское название онтологии);
- `my_ontology:Human – rdfs:comment – "Человек"` (что дословно означает: «Классу Human соответствует описание “Человек”»; обычно с помощью свойства `rdfs:comment` задаются человекочитаемые названия объектов).

В качестве предикатов могут быть использованы не только стандартные конструкции, но и пользовательские – таким образом могут быть заданы новые свойства понятий и отношения между ними.

Помимо языков описания онтологий, «существуют программные продукты класса Triple store – хранилища триплетов, функционирующие аналогично базам данных. Наиболее известные решения такого класса – Apache Jena, Virtuoso, Sesame, GraphDB и множество других. В том числе, создать triple store можно на основе Oracle 11g. Хранилища триплетов имеют программные интерфейсы, позволяющие обращаться к ним из различных средств программирования, использовать машины логического вывода. Существует стандарт ***SPARQL***, описывающий программный интерфейс и синтаксис запросов к онтологическим моделям; уже из его названия очевидна аналогия с языком SQL. Программная реализация SPARQL-интерфейса к содержимому хранилища триплетов называется точкой доступа SPARQL (SPARQL endpoint)» [1].

Наиболее распространенным ПО, использующим внутри себя стандарты W3C, является открытый редактор онтологий и фреймворк для построения баз знаний ***Protégé***, разработанный Стэнфордским университетом [3]. Более подробно ознакомиться с нюансами построения онтологий с использованием стандартов W3C и технологии Semantic Web можно в книге С. Горшкова «Введение в онтологическое моделирование» [1].

1.2.2 Labeled Property Graph

Альтернативным способом представления знаний в форме онтологии, активно развивающимся на сегодняшний день, является схема ***LPG (Labeled Property Graph)*** – схема представления данных в виде графа, помеченного набором свойств. Данная технология была разработана группой шведских инженеров в ходе работы над системой управления корпоративным контентом (Enterprise Content Management, ECM), данные в которой они решили хранить в виде графа.

Основное отличие LPG от RDF заключается в том, что, в LPG как узлы, так и связи графа имею внутреннюю структуру: т.е. могут быть помечены любым количеством свойств, выраженных в виде объектов любого типа.

Такая модель имеет ряд преимуществ перед RDF: так, конструкции LPG-схемы имеют более лаконичную структуру, более удобный и человекочитаемый вид, а также в ряде случаев обращение к таким базам знаний обладает лучшим быстродействие. По этим причинам в рамках данных методических материалов будем рассматривать именно эту технологию. Подробнее об отличиях двух технологий можно прочитать в статье “RDF Triple Stores vs. Labeled Property Graphs: What’s the Difference?” [4].

1.3 Рекомендуемая литература

1. Горшков, С. Введение в онтологическое моделирование: учебное пособие [Электронный ресурс] / С. Горшков; ООО «ТриниДата». Екатеринбург, 2016. – 165 с. — Режим доступа: http://trinidata.ru/files/SemanticIntro.pdf.
2. Добров Б. В., Иванов В.В., Лукашевич Н.В., Соловьев В.Д. Онтологии и тезаурусы: модели, инструменты, приложения. — М.: Бином. Лаборатория знаний, 2009. — 173 с.
3. Protégé. A free, open-source ontology editor and framework for building intelligent systems [Электронный ресурс] — Режим доступа: https://protege.stanford.edu/.
4. Barrasa J. RDF Triple Stores vs. Labeled Property Graphs: What’s the Difference?, Neo4j Blog. [Электронный ресурс] — Режим доступа: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/.

2 Построение онтологической модели по схеме LPG

Одним из наиболее известных инструментов, базирующихся на схеме LPG, является графовая система управления базами данных Neo4j [1].

2.1 Структура данных

Рассмотрим подробнее структуру данных, управляемой данной СУБД на примере следующего простого графа:

Рисунок 1. Структура данных

Вершины графа в терминологии Neo4j называются ***узлами (nodes)*** и представляют собой сущности.

Любые узлы в графе могут иметь ***метки (labels)***. Метки используются, чтобы агрегировать узлы в подмножества. При этом, любой узел может иметь неограниченное количество таких меток (или не иметь никаких меток вовсе). В представленном примере оба узла имеют метку `State`, которая обозначает, что данный узел представляет собой состояние плиткоукладчика.

Два узла связаны ***отношением (relation)***. Любое отношение обязательно имеет определенное направление, а также единственный тип (в нашем примере это тип `TRANSIT`).

Как узлы, так и отношения могут иметь любое количество ***свойств (properties)***. Свойства представляют собой пары вида «ключ – значение» и используются для описания характеристик сущностей. Так, в нашем примере узлы имеют свойства `name` (название сущности), `description` (описание сущности на естественном языке), `codename` (имя сущности, материализованное в исходном коде); отношение задано свойствами аналогичными name и codename, а также свойством condition, служащим для описания условия перехода.

Более подробно со структурой данных Neo4j можно ознакомиться в документации к проекту [1].

Переходя на термины онтологического моделирования, ***понятия можно задавать с помощью узлов, классы или агрегаты – с помощью меток, свойства и отношения – с помощью одноименных сущностей***. Если в онтологической модели необходимо использовать иерархию классов, то можно задать классы с помощью отдельных узлов, добавив к ним метки `Class` и связав их между собой иерархическими отношениями `IS_SUBCLASS_OF` (быть подклассом). С узлами-экземплярами их можно связать отношениями типа `IS_INSTANCE_OF` («являться экземпляром»); кроме того, можно зафиксировать принадлежность к классу с помощью меток. Отметим, что это лишь варианты возможной конфигурации онтологической модели, которые можно и нужно модифицировать в зависимости от поставленных задач.

Ограничения, обеспечивающие целостность онтологической модели, можно задать на уровне конфигурации структуры БД с помощью специальной сущности ***constraints***.

Аксиоматику онтологической модели можно формулировать на уровне запросов к базе данных. Кроме того, существуют инструментальные средства поддержки автоматической проверки на непротиворечивость [3] и логического вывода новых знаний [4] (плагин Neosemantics для Neo4j).

2.2 Язык запросов Cypher

Cypher – это декларативный язык запросов к графовой СУБД. Часть его конструкций, такие как, например `WHERE` или `ORDER BY`, заимствованы из SQL. Для реализации конструкции сопоставления различным шаблонам (`MATCH`) используется подход, характерный для SPARQL. Еще часть конструкций наследуется из языков программирования Haskell и Python.

Как и SQL-запрос, запрос на языке Cypher представляет собой набор последовательно связанных клаузул.

2.2.1 Запрос на запись

Запрос на запись обычно осуществляется с помощью клаузулы `CREATE`. Рассмотрим пример создания простого графа (см. Рисунок 2) с помощью запроса (см. Листинг 1):

Листинг 1. Пример Cypher-запроса на запись

```
   // создание узлов
   CREATE (machine1:Tiler {name: "Плиткоукладчик 1", codename:
"Machine1"})
   CREATE (stop:State {name: "Не запущено", description: "Состояние
плиткоукладчика перед запуском или после завершения работы.", codename:
"STOP", startTime: 1616746906, endTime: 1616746913})
```

```
   CREATE (init:State {name: "Инициализация", description: "Состояние
плиткоукладчика после поступления сигнала для начала работы.", codename:
"INIT", startTime: 1616746913, endTime: 1616746918})
   CREATE (run:Operation {name: "Запуск плиткоукладчика", codename: "start
= ext:: machine.run()"})

   // создание связей
   CREATE (machine1)-[:HAS_STATE {name: "иметь состояние"}]->(stop)
   CREATE (machine1)-[:HAS_STATE {name: "иметь состояние"}]->(init)
   CREATE (stop)-[:TRANSIT {name: "переходить в", condition: "Получен
сигнал для начала работы"}]->(init)
   CREATE (run)-[:PRECEDE {name: "предшествовать"}]->(stop)
```

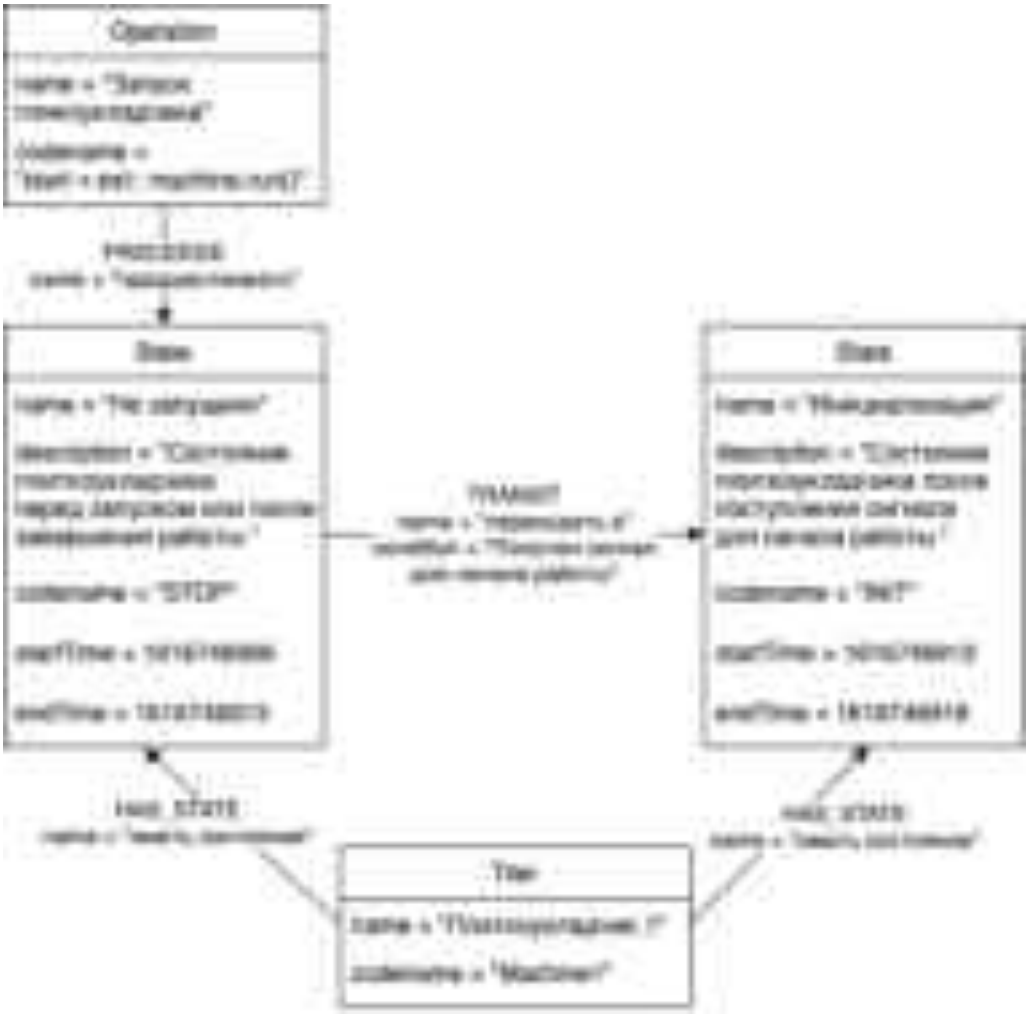


Рисунок 2. Запрос на создание графа

2.2.2 Запрос на чтение

Запрос на чтение обычно осуществляется с помощью следующих клаузул:

- MATCH: шаблон графа, с которым нужно сопоставить результат выполнения запроса;

- `WHERE`: используется как часть клаузул `MATCH`, `OPTIONAL MATCH` или `WITH`; накладывает ограничения на шаблон или фильтрует промежуточный результат;
- `RETURN`: служит для описания того, какие сущности необходимо вернуть.

Допустим, мы хотим узнать, какое состояние плиткоукладчика №1 следует за операцией «Запуск плиткоукладчика». Тогда выполним следующий запрос:

Листинг 2. Пример Cypher-запроса на чтение

```
   MATCH (tiler:Tiler {name: "Плиткоукладчик 1"})-[:HAS_STATE]->(state)<-
[:PRECEDE]-(:Operation {name: "Запуск плиткоукладчика"})
   RETURN tiler.name, state.name
```

Результат выполнения запроса будет выглядеть следующим образом:

```
+-----------------------------------+
| tiler.name          | state.name    |
+-----------------------------------+
| "Плиткоукладчик 1" | "Не запущено" |
+-----------------------------------+
1 row
```

Можно получить ответ в формате JSON:

```
[
  {
    "tiler.name": "Плиткоукладчик 1",
    "state.name": "Не запущено"
  }
]
```

Подробнее с синтаксисом и возможностями языка Cypher можно ознакомиться в документации к нему.

2.3 Рекомендуемая литература

1. The Neo4j Getting Started Guide v4.2. [Электронный ресурс] — Режим доступа: https://neo4j.com/docs/getting-started/current/.
2. The Neo4j Cypher Manual v4.2. [Электронный ресурс] — Режим доступа: https://neo4j.com/docs/cypher-manual/current/.
3. Neosemantics(n10s) User Guide. Validating Neo4j graphs against SHACL. [Электронный ресурс] — Режим доступа: https://neo4j.com/labs/neosemantics/4.0/validation/.
4. Neosemantics(n10s) User Guide. Inferencing/Reasoning. [Электронный ресурс] — Режим доступа: https://neo4j.com/labs/neosemantics/4.0/inference/.

3 Методика формирования онтологической модели и возможности ее использования в ходе разработки АС

3.1 Содержание проектного процесса на основе онтологического моделирования

Проектный процесс с применением технологии онтологического моделирования основан на применении ГОСТ 34.601, в котором определены стадии создания АС [1], а также парадигму Design Thinking [2]. Излагаемая ниже методика предполагает формирование спецификаций онтологических моделей на каждой стадии проектного процесса. При этом, проектные процедуры каждой стадии выполняются с использованием средств поддержки формирования и интерпретации онтологических спецификаций, описанных в предыдущем разделе.

Этот процесс обеспечивает реализацию ***алгоритма формирования*** онтологической модели, который в обобщенном виде можно свести к следующим шагам:

1. Вычленение сущностей, участвующих в онтологической модели. Формирование списка понятий ***C***.
2. Формирование пространства агрегатов ***Agg***. В рамках данных методических материалов целесообразным представляется выделение агрегатов, соответствующих фазам жизненного цикла проекта:
    - понятия, относящиеся к уровню требований к проекту;
    - понятия, относящиеся к уровню проектных решений и содержанию процесса их формирования; включение сущностей, относящихся к процессу формирования проектных решений, обеспечивает развитие средств автоматизации проектирования за счет формализации не только спецификаций самих проектных решений, но и процессов их построения;
    - понятия, относящиеся к уровню реализации, которые охватывают не только поддержку создания целевого кода АС, но и формирование таких шаблонов проектирования и прототипов, которые становятся базой опыта для следующих проектов.

    Кроме того, возможно расширение пространства агрегатов за счет тематических групп – например, можно выделить группу понятий, относящихся к той или иной проектной задаче, функциям некоторого объекта проектирования, его состояниям и т.д..

3. Агрегация или классификация понятий. На этом этапе необходимо определить агрегаты для каждого понятия (при этом, одно понятие может относиться сразу к нескольким агрегатам). Если принято решение использовать классификацию, то необходимо также построить иерархию классов.
4. Описание свойств понятий и их значений $\boldsymbol{Pr \rightarrow C}$.
5. Описание отношений между понятиями $\boldsymbol{R}$.
6. Описание аксиом (правил логического вывода) над элементами онтологии $\boldsymbol{A}$.

Таким образом, онтологическая модель $\boldsymbol{O}$ в самом общем виде может быть представлена множеством спецификаций, описанных выше:

$$\boldsymbol{O = \{C, Pr, Agg, R, A\}.}$$

Каждая из спецификаций представляется в формате информационного объекта, который может быть погружен в базу данных проекта, а вся совокупность информационных объектов может интерпретироваться средствами формирования запросов, описанными выше.

Отметим, что, согласно предлагаемой методике, данный процесс является *инкрементальным*: т.е. по мере развития проекта онтологическая модель должна пополняться новыми сущностями.

В обобщенном виде проектный процесс с применением технологии онтологического моделирования представлен на рисунке 3.

Проектный процесс начинается с этапа ***формирования требований R*** к проекту, после чего каждое требование из множества $\boldsymbol{R}$ подвергается анализу (можно, например, использовать технологию вопросно-ответного или QA-анализа [3] для детализации требований). В результате такого анализа формируется ***онтологическая модель требований***. Сущности онтологической модели требований необходимо ***связать с материализациями требований в проекте***. Сделать это можно разными способами: например, добавить ссылки на требования или непосредственно их формулировки в качестве свойств узлов онтологии и/или сформировать подпространство агрегатов в соответствии с требованиями и добавить узлам соответствующие метки.

Далее следует этап ***постановки задачи Z***, в результате чего должна быть сформулирована исходная постановка задачи с опорой на требования $\boldsymbol{R}$. При этом учитывается факт, что в практике проектирования АС зачастую используются так называемые продукты лоскутной автоматизации – тексты, модели, UML-диаграммы, куски кода и т.п., относящиеся к автоматизации отдельных частей объекта автоматизации. Например, до создания комплексной АС производства были во многом независимо автоматизированы склады цехов, внутрицеховое

оперативно-календарное управление, выдача и закрытие нарядов на работы и т.п. Продукты лоскутной автоматизации будем называть ***исходными прототипами IPt***.

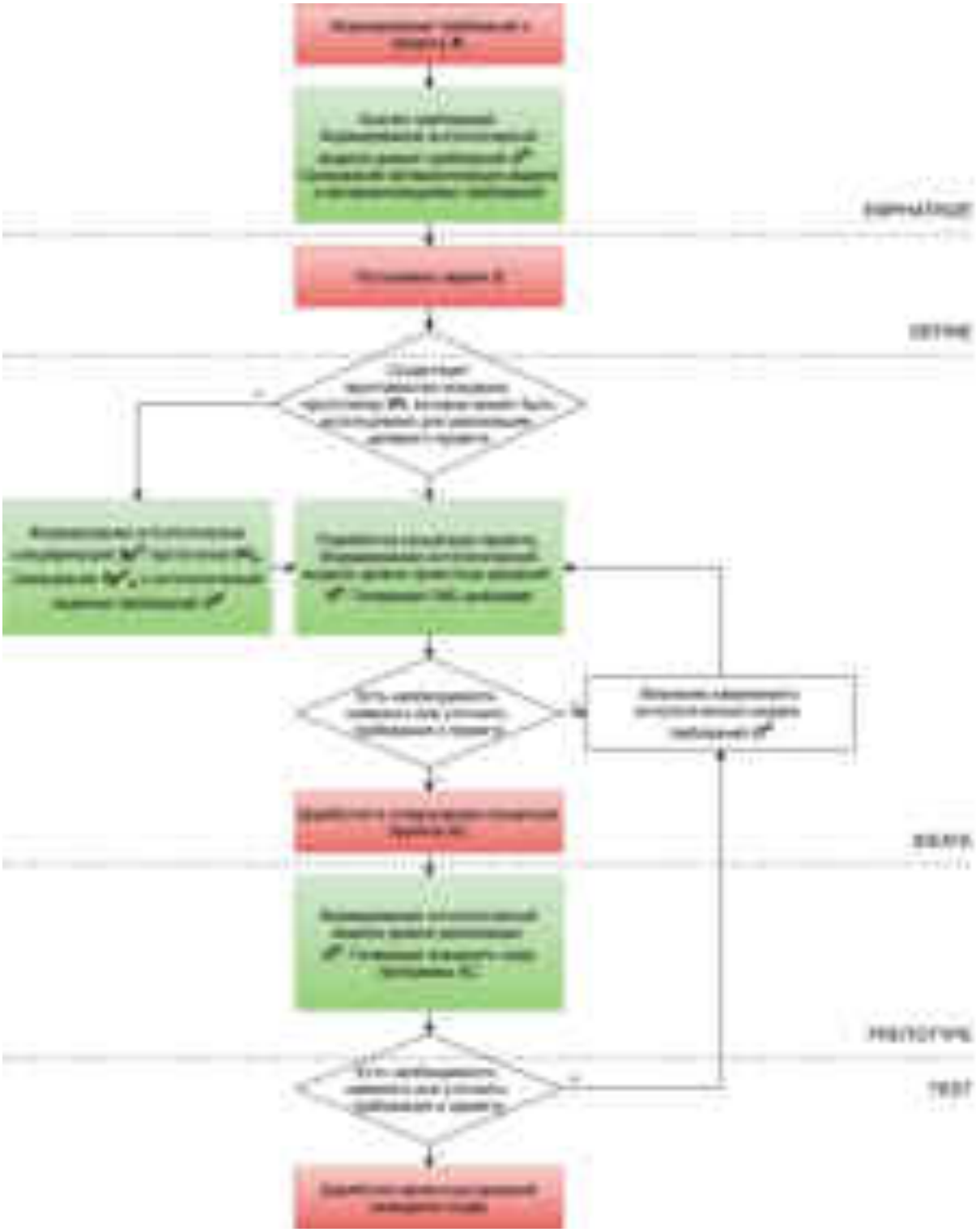


Рисунок 3. Проектный процесс с применением технологии онтологического моделирования в парадигме Design Thinking

3.2 Анализ требований

Итерационный процесс анализа требований представлен на рисунке 4. В ходе этого процесса осуществляется циклический обход всех потенциально возможных требований и их онтологическое специфицирование.

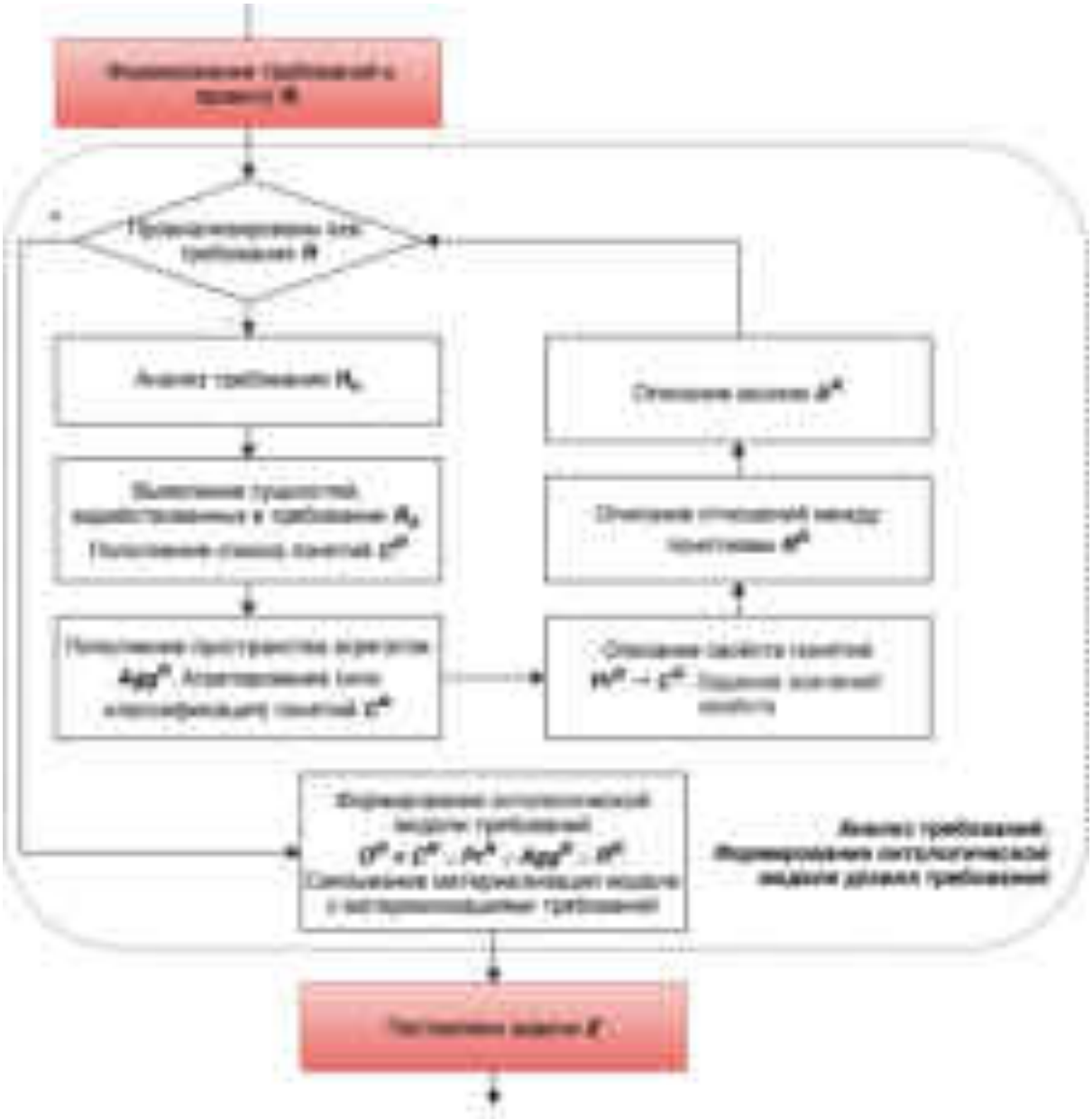


Рисунок 4. Анализ требований и формирование онтологической модели соответствующего уровня

Традиционные технологии формирования технического задания (ТЗ) часто предполагают использование шаблонов ТЗ, содержащих множество прототипов

требований, содержащих готовые фразы. В ходе анализа требований на основе таких шаблонов системный аналитик в ходе диалога с представителями заказчика применительно к прототипам отдельных требований выполняет такие информационные операции, как заполнение пропусков во фразах, выбор вариантов из списков, зачеркивание ненужного или подчеркивание нужного. Благодаря шаблонам ТЗ обеспечивается достаточная полнота требований не только в части того, что должна обеспечивать АС, но даже и в части того, что она не обязана обеспечивать.

В более современных процессах анализа требований используются технологии вопросно-ответного моделирования на базе вопросно-ответных программ [3], являющимися сценариями деятельности системного аналитика, проводящего анализ требований. Методика онтологического моделирования позволяет формировать такие сценарии и даже части ТЗ через запросы к базе знаний онтологических спецификаций, например, с помощью инструментальных средств, описанных в предыдущем разделе.

3.3 Специфицирование исходных прототипов

Если проектировщик располагает исходными прототипами, релевантными задаче ***Z***, то ему необходимо проанализировать, может ли он использовать какие-либо из них для удовлетворения требований к проекту. Если такие прототипы существуют, то необходимо подвергнуть их онтологическому специфицированию согласно методике, описанной в начале данного параграфа, и связать полученные спецификации с онтологической моделью требований (см. рисунок 5).

Онтологическое специфицирование исходных прототипов является важной работой не только для обеспечения возможностей применять методологию онтологического моделирования в проекте создания конкретной АС. В том процессе изначально выявляется потенциал повторного использования какого-то исходного прототипа в текущем проекте, однако этот потенциал с высокой вероятностью может быть потреблен в другом проекте. Для поддержки этой возможности достаточно снабдить объекты онтологической спецификации признаками наличия такого потенциала.

Связывание с онтологической моделью требований может быть осуществлено, как и в случае с материализациями требований, на уровне специальных меток. В дальнейшем – при повторном использовании таких прототипов в рамках других проектов – онтологическое специфицирование требоваться уже не будет: необходимо будет лишь связать существующие онтологические спецификации с ОМ требований конкретного проекта.

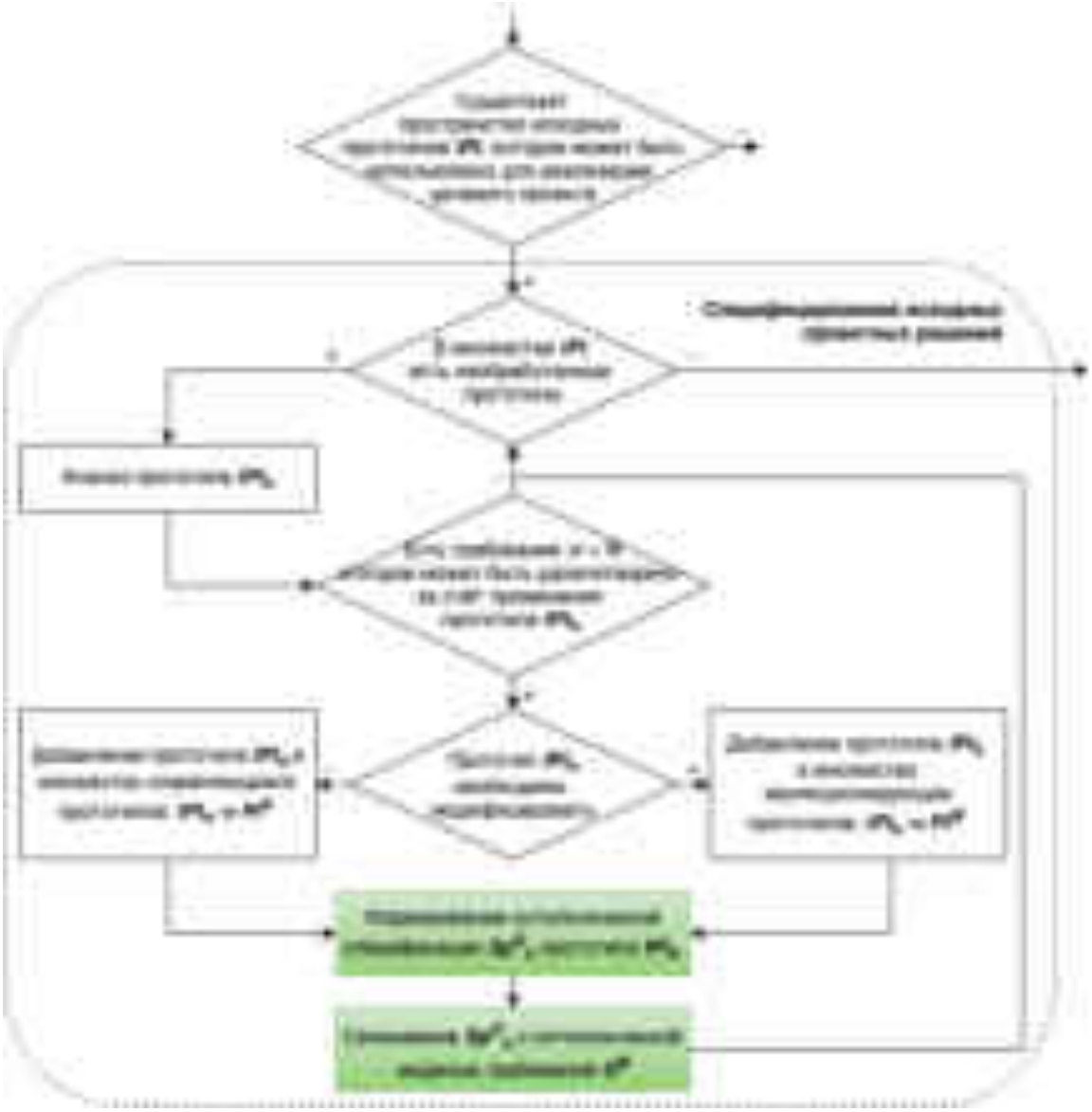


Рисунок 5. Специфицирование исходных проектных решений

3.4 Формирование концептуальной модели

После этого необходимо проанализировать задачу и определить, каким образом можно ее решить и что для этого требуется, чтобы в конечном итоге сформировать концепцию проекта. На этом этапе также целесообразно применять технологию вопросно-ответного анализа, который позволяет осуществлять пошаговую детализацию задачи – вплоть до конкретных проектных решений. В ходе такого анализа формируется ***онтологическая модель уровня проектных решений***, которая по своей сути является формой выражения концептуальной модели проекта. Поскольку такая модель содержит в себе максимально полную информацию о проекте на данном этапе, она может быть использована для автоматического или автоматизированного генерирования

UML-диаграмм, которые на сегодняшний день являются общепринятой и наиболее распространенной формой представления концептуальных моделей. Более подробно процесс формирования онтологической модели уровня проектных решений показан на рисунке 6.

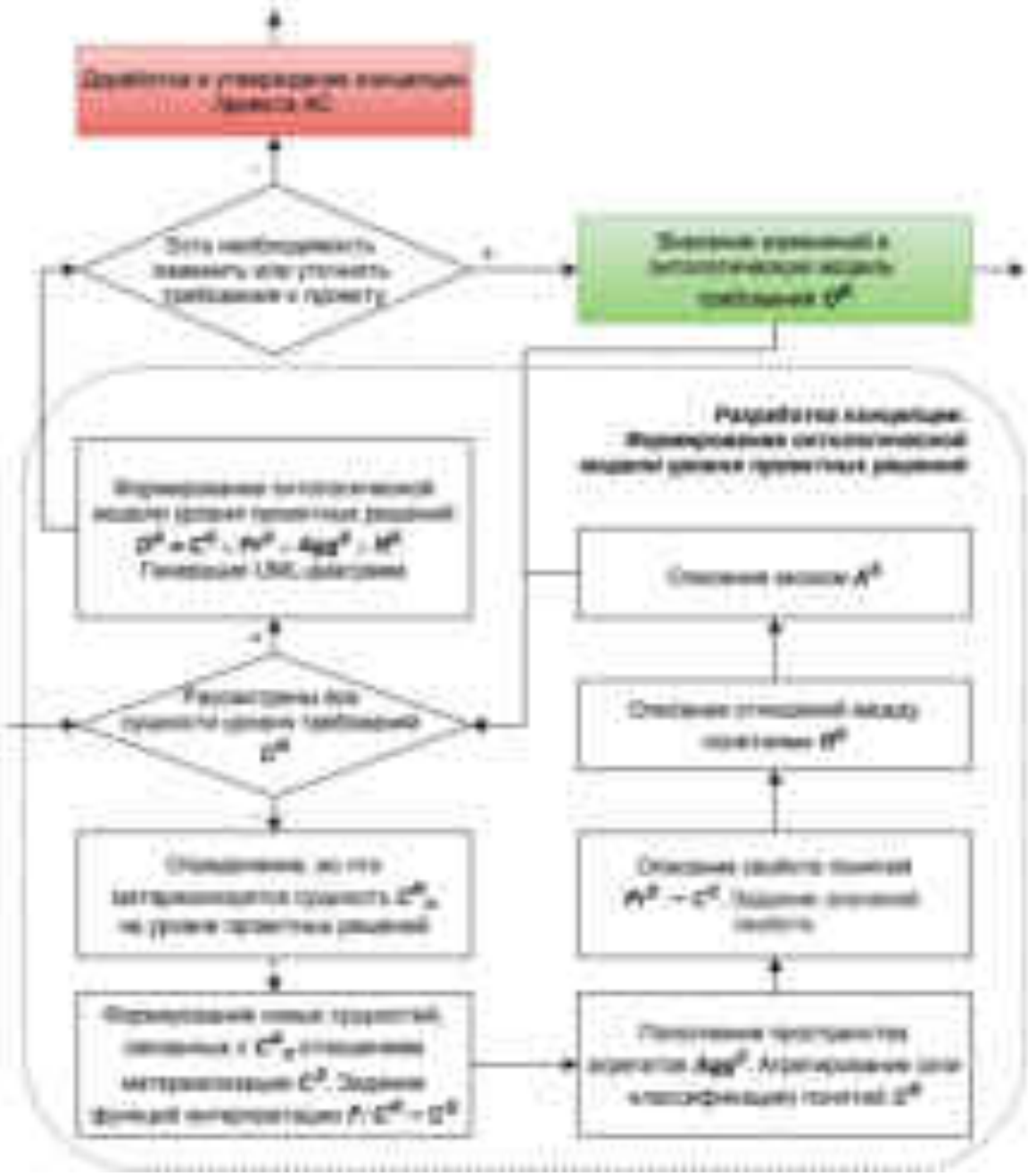


Рисунок 6. Формирование концептуальной модели проекта в форме онтологии

3.5 Формирование онтологии реализации

Дальнейшая детализация задачи ***Z*** предполагает формирование ***онтологической модели уровня реализации*** проекта, которое строится на основе сценария деятельности, представленного на рисунке 7.

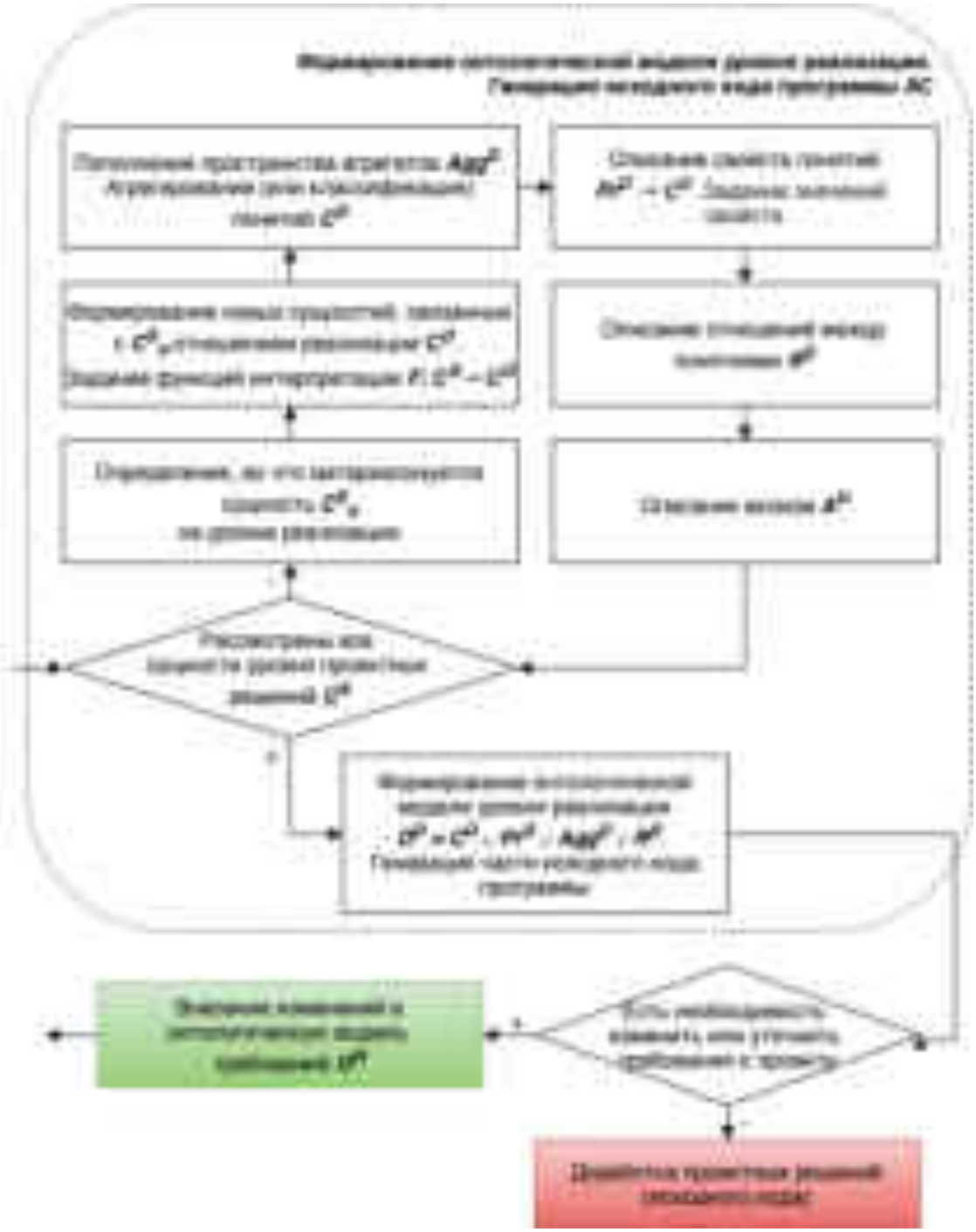


Рисунок 7. Формирование онтологической модели уровня реализации

На этом этапе задается пространство сущностей, представленных в исходном коде программы; данным сущностям присваиваются соответствующие имена, а

между ними и сущностями уровня проектных решений устанавливаются отношения типа «быть реализованным». Такая конфигурация ОМ позволяет автоматически ***генерировать часть исходного кода программы***.

Детали этого процесса представляются в следующем разделе в ходе описания конкретного примера онтологического моделирования.

3.6 Позитивные эффекты проектного процесса на основе онтологического моделирования

В рассмотренном выше процессе понятия уровня требований сопоставляются с понятиями уровня проектных решений с помощью соответствующих функций интерпретации (как и понятия уровня проектных решений с понятиями уровня реализации), что обеспечивает верифицируемость проектных решений. Для выполнения функций верификации вполне пригодны механизмы запросов к базе знаний онтологических спецификаций, рассмотренные в разделе 2. Через запросы легко выделить степень покрытия спецификаций требований спецификациями проектных решений. Запросы, обрабатывающие значения атрибутов сущностей, Информационные процессы построения онтологических спецификаций уровня реализации вполне пригодны для построения онтологических спецификаций поддержки тестирования и генерации исходных текстов Unit-тестов.

Еще одним важным позитивным свойством такой конфигурации онтологии проекта является возможность ***автоматизированного реинжиниринга*** при возникновении необходимости внесения изменений или уточнений в набор требований. Поскольку сущности ОМ уровня требований связаны с сущностями онтологии уровня проектных решений, а сущности онтологии уровня проектных решений – с сущностями онтологии реализации, то при внесении изменений в ОМ требований, соответствующие изменения также должны быть внесены в ОМ проектных решений и в ОМ реализации (вручную или автоматически – в зависимости от заданных функций интерпретации). Следовательно, данные изменения легко проникают через онтологическую модель непосредственно в артефакты проектирования (например, в UML-диаграммы и в исходный код программы), что способствует повышению степени автоматизации управления изменениями. В качестве метаданных, поддерживающих распространение изменений в этом случае используются отношения между сущностями требований и сущностями проектных решений.

Дополнительным полезным эффектом является материализация многих пространств имен в онтологической модели проекта: в частности, пространства имен требований, исходных прототипов, проектных решений и реализации. В рамках каждого из этих пространств одни и те же понятия могут быть сформулированы по-разному, но благодаря их связыванию в ОМ формируется

собственный ***язык проекта***, за счет которого достигается концептуальная его целостность и значительно снижается уровень необходимых когнитивных усилий на понимание проектных задач и проектных ситуаций в рамках множества различных пространств имен.

Простота манипулирования онтологическими спецификациями с помощью языка запросов дает возможность автоматически выполнять селекцию проектных спецификаций для поддержки аспектно-ориентированного и объектно-ориентированного управления разработкой. Рабочие совещания по аспектам «состояние документации», «состояние оттестированности», «состояние отработки изменений» и т.п. легко снабжаются результатами выборки из базы онтологических спецификаций проекта. Аналогично формируются выборки тех частей спецификаций, которые относятся к каким-то подсистемам или компонентам, являющимися предметами рабочих совещаний.

3.7 Рекомендуемая литература

1. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания
2. Коломенский А. Design Thinking — гайд по применению методологии дизайн–мышления. [Электронный ресурс] — Режим доступа: https://leadstartup.ru/db/design-thinking.
3. Соснин П.И. Вопросно-ответное программирование человеко-компьютерной деятельности / П. И. Соснин. – Ульяновск : УлГТУ, 2010. – 240 с.
4. Горшков С. Введение в онтологическое моделирование.

4 Формирование онтологической модели на примере системы логического управления роботами-плиткоукладчиками

Рассматриваемый пример представляется в упрощенном виде, достаточном для иллюстрации предлагаемого подхода.

Логическое управление охватывает группу роботов-плиткоукладчиков, укладывающих плитки на негоризонтальной плоскости, что требует различать движение вверх, вниз, влево и вправо, поскольку распределение нагрузок на ходовую часть при каждом виде движения и смене направлений различается. Каждый робот оснащен механизмом укладки, закрепленном на мобильной платформе с правой стороны. Это предопределяет укладку по траектории раскручивающейся спирали с вращением по часовой стрелке. Все плитки квадратные и в логическом управлении используется дискретное координатное пространство с центром {x: 0, y: 0}, где располагается первая укладываемая плитка. Увеличению координаты y на 1 соответствует перемещение на одно плиткоместо вверх, а уменьшению – на одно плиткоместо вниз. Аналогично увеличению координаты x на 1 соответствует перемещение на одно плиткоместо вправо, а уменьшение – на одно плиткоместо влево. Одно из требований предписывает формирование для каждого шага процесса протокольной записи следующего формата:

- момент времени начала шага с точностью до миллисекунд;
- номер плиткоукладчика; предполагается общий протокол для всей группы плиткоукладчиков;
- порядковый номер укладываемой плитки; нумерация начинается с 0;
- номер витка спирали; нумерация начинается с 1;
- значение координаты x текущей плитки;
- значение координаты y текущей плитки;
- состояние процесса на данном шаге (направление движения, либо подготовительные состояния);
- состояние автомата, в которое переходит автомат.

4.1 Структура онтологической модели

Общая онтологическая модель проекта включает в себя онтологию требований, онтологию проектирования и онтологию реализации, структура которой задана в СУБД Neo4j в формате метамодели.

Формальные спецификации композиции онтологий, ориентированных на автоматическую генерацию программ логического управления, целесообразно базировать на логико-алгебраических моделях, позволяющих представлять сущности и отношения всех этапов проектирования таким образом, чтобы модели обслуживали не только спецификацию проектных решений, но и процесс автоматизации проектирования. Поскольку любой проектный процесс является процессом трансформации одних проектных спецификаций в другие, важное значение для формальных спецификаций имеют функциональные отношения, определяющие отображение исходных спецификаций в результирующие для каждой проектной операции.

Этим условиям удовлетворяет набор общих формализмов, представляемый ниже.

4.2 Формирование онтологии требований

Обобщенная структура онтологии требований может быть представлена в следующем виде:

$$RO = (Source, Inputs, Outputs, Actions, Events, Messages, Processes, Protocols, Conditions \mid RInputs, ROutputs, RActions, REvents, RMessages, RProcesses, RProtocols, RConditions),$$

- где $Source$ – множество предложений неформального описания объекта управления, протекающих в нем процессов и технических условий, определяющих требования;
- $Inputs$ – множество спецификаций входов СЛУ;
- $Outputs$ – множество спецификаций выходов СЛУ;
- $Actions$ – множество спецификаций операций и действий СЛУ;
- $Events$ – множество спецификаций событий, возникающих в объекте управления;
- $Messages$ – множество спецификаций сообщений о событиях, действиях, значениях входов и выходах;
- $Conditions$ – множество спецификаций элементов технических условий, включающее в себя функциональные, параметрические и иные требования;
- $Processes$ – множество спецификаций процессов, осуществляемых в ходе управления;
- $Protocols$ – множество спецификаций протоколов о ходе управления;

- *RInputs: Source* → Inputs** – отношение «быть входом», обеспечивающее связывание подмножества предложений неформального описания с подмножеством входов СЛУ; здесь и далее конструкция *S** означает множество всех подмножеств *S*;
- *ROutputs: Source* → Outputs** – отношение «быть выходом», обеспечивающее связывание подмножества предложений неформального описания с подмножеством выходов СЛУ;
- *RActions: Source* × Inputs* × Outputs* → Actions** – отношение «быть операцией», обеспечивающее связывание подмножества предложений неформального описания с подмножеством операций и действий СЛУ, а также определяющее зависимости операций от входов и выходов;
- *REvents: Source* × Inputs* × Outputs* × Actions* → Events** – отношение «быть событием», обеспечивающее связывание подмножества предложений неформального описания с подмножеством событий в ходе функционирования СЛУ а также определяющее зависимости событий от входов, выходов и операций;
- *RMessages: Source* × Events* → Messages** – отношение «быть сообщением», обеспечивающее связывание подмножества предложений неформального описания с подмножеством сообщений, возникающих в ходе функционирования СЛУ в контексте конкретных событий или их групп;
- *RProcesses: Source* × Inputs* × Outputs* × Actions* × Events* → Processes** – отношение «быть процессом», обеспечивающее связывание подмножества предложений неформального описания с подмножеством процессов, протекающих в СЛУ и охватывающих изменения входов и выходов, выполнение операций, возникновение событий;
- *RProtocols: Source* × Inputs* × Outputs* × Actions* × Events* × Messages* × Processes* → Protocols* – отношение «быть протокольной записью», обеспечивающее связывание подмножества предложений неформального описания с подмножеством процессов, протекающих в СЛУ и охватывающих последовательности изменения входов и выходов, выполнения операций, возникновение событий и сообщений;
- *RConditions: Source* × Inputs* × Outputs* × Actions* × Events* × Protocols* × Messages* → Conditions** – отношение «быть условием», обеспечивающее связывание подмножества предложений неформального описания с подмножеством условий,

задающих требования к СЛУ в контексте связей этих требований со входами, выходами, событиями, операциями и сообщениями.

Сформируем метамодель онтологии требований в СУБД Neo4j с помощью следующего запроса:

Листинг 3. Cypher-запрос для создания метамодели онтологии требований

```
// добавление узлов
CREATE(source:RO {name: 'Source'})
CREATE(inputs:RO {name: 'Inputs'})
CREATE(outputs:RO {name: 'Outputs'})
CREATE(actions:RO {name: 'Actions'})
CREATE(events:RO {name: 'Events'})
CREATE(messages:RO {name: 'Messages'})
CREATE(processes:RO {name: 'Processes'})
CREATE(protocols:RO {name: 'Protocols'})
CREATE(conditions:RO {name: 'Conditions'})

// добавление связей между узлами
CREATE(source)-[:RInputs]->(inputs)

CREATE(source)-[:ROutputs]->(outputs)

CREATE(source)-[:RActions]->(actions)
CREATE(inputs)-[:RActions]->(actions)
CREATE(outputs)-[:RActions]->(actions)

CREATE(source)-[:REvents]->(events)
CREATE(inputs)-[:REvents]->(events)
CREATE(outputs)-[:REvents]->(events)
CREATE(actions)-[:REvents]->(events)

CREATE(source)-[:RMessages]->(messages)
CREATE(events)-[:RMessages]->(messages)

CREATE(source)-[:RProcesses]->(processes)
CREATE(inputs)-[:RProcesses]->(processes)
CREATE(outputs)-[:RProcesses]->(processes)
CREATE(actions)-[:RProcesses]->(processes)
CREATE(events)-[:RProcesses]->(processes)

CREATE(source)-[:RProtocols]->(protocols)
CREATE(inputs)-[:RProtocols]->(protocols)
CREATE(outputs)-[:RProtocols]->(protocols)
CREATE(actions)-[:RProtocols]->(protocols)
CREATE(events)-[:RProtocols]->(protocols)
CREATE(messages)-[:RProtocols]->(protocols)
CREATE(processes)-[:RProtocols]->(protocols)

CREATE(source)-[:RConditions]->(conditions)
CREATE(inputs)-[:RConditions]->(conditions)
CREATE(outputs)-[:RConditions]->(conditions)
CREATE(actions)-[:RConditions]->(conditions)
CREATE(events)-[:RConditions]->(conditions)
CREATE(messages)-[:RConditions]->(conditions)
CREATE(protocols)-[:RConditions]->(conditions)
```

На рис. 8 показана метамодель в виде графа, на котором присутствуют все узлы, за исключением *Source* (в целях повышения наглядности иллюстрации, поскольку последние связаны со всеми остальными без исключений).

Рисунок 8. Метамодель онтологии требований

Как было отмечено в предыдущем параграфе, процесс построения онтологии требований начинается с анализа требований, представленных в нашем примере множеством неформальных описаний объекта управления *Source*, которые чаще всего являются текстами на естественном языке. Ниже – примеры таких описаний:

Source1: Существует некоторый процесс инициализации, за которым могут скрываться последовательные подпроцессы загрузки контейнера с плитками и подпроцесс движения к начальной точке монтажа. Начало этого процесса задается некоторым внешним воздействием – старт укладки плитки.

Source2: Необходимо установить таймаут – ограничение на время пребывание в состоянии инициализации.

Source3: Необходимо формировать протокольную запись, содержащую информацию о старте и завершении процесса инициализации с точностью до миллисекунд с указанием идентификатора плиткоукладчика.

Последовательный анализ множества *Source* приводит нас к формированию формальных спецификаций онтологии требований. Отметим, что связывание элементов множеств таких спецификаций между собой основано на функциональных отношениях, описанных выше, за которыми скрываются двухместные семантические отношения между концептами онтологии.

Функциональные многоместные отношения, описанные в метамодели (рис. 8) обеспечивают такую последовательность формирования онтологии требований, которая способствует достижению необходимой степени полноты, непротиворечивости и концептуальной целостности. Так, в соответствии с отношением *RInputs* и *ROutputs* на основе множества *Source* сначала формируются входы (*Inputs*) и выходы (*Outputs*), затем элементы множества *Inputs* и *Outputs* вместе с элементами множества *Source* используются для формирования операций (*Actions)* и связывания их с другими элементами онтологии требований – и так далее. Таким образом, опираясь на такую метамодель, проектировщик или системный аналитик получает некую инструкцию, благодаря которой облегчается формирование онтологии требований и снижаются риски, связанные с получением недостаточной полной модели требований.

На рис. 9 показан фрагмент онтологической модели требований, пригодный для реализации в СУБД Neo4j, сформированный в результате анализа элементов множества *Source*, представленных выше. Принадлежность концептов к множествам *Inputs, Outputs, Actions, Events, Messages, Processes, Protocols, Conditions* задано с помощью соответствующих меток (*labels*). Типы отношений указывают на функции, вычисление которых привели к их формированию.

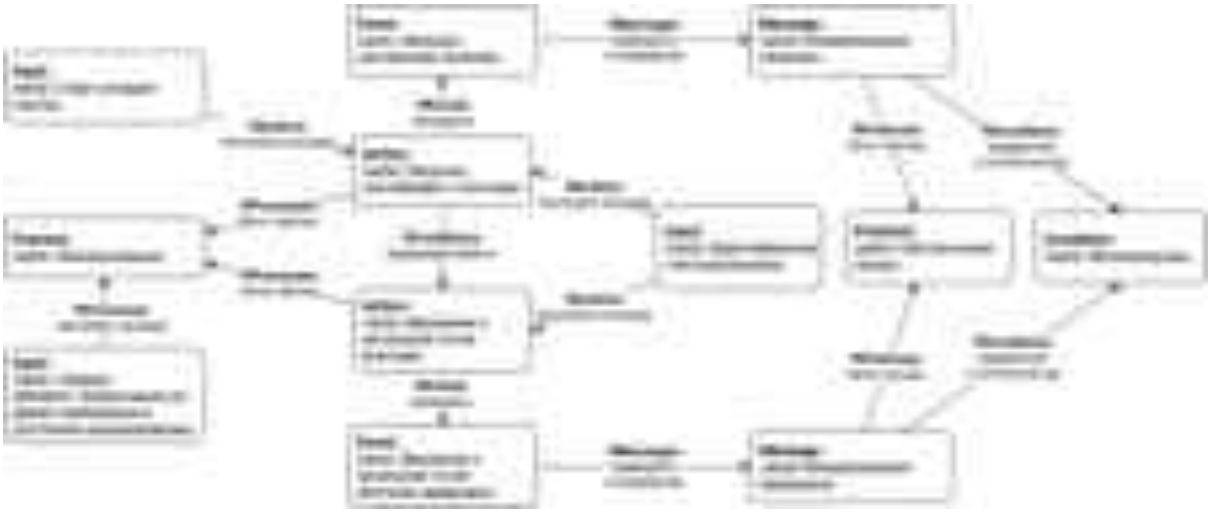


Рис. 9. Фрагмент онтологии требований

4.3 Формирование онтологии проектирования

Онтология проектирования в дополнение к спецификациям требований определяет проектные операции, проектные задачи, проектные решения и проектные процессы, а также некоторые сущности, характерные для автоматной модели.

$$DO = (DActions, DTasks, DSolutions, DProcesses \mid RDActions, RDTasks, RDSolutions, RDProcesses),$$

- где $DTasks$ – множество проектных задач;
- $DActions$ – множество проектных операций;
- $DSolutions$ – множество проектных решений, представляющих собой спецификации методов решения задач управления и соответствующих алгоритмов, способов представления функциональных зависимостей и данных, в том числе схемы баз данных, спецификации интерфейсов СЛУ с объектом управления и человеком-оператором;
- $DProcesses$ – множество проектных процессов по проектированию СЛУ;
- $RDActions: DStates^* \times Inputs^* \times Outputs^* \times Actions^* \times Events^* \times Messages^* \times Conditions^* \times Protocols^* \rightarrow DActions^*$ – отношение «быть проектной операцией», обеспечивающее порождение или модификацию подмножества проектных операций применительно к одному или нескольким состояниям процесса управления и вовлечение в операцию спецификаций из онтологии требований;

- *RDTasks: DActions* × Inputs* × Outputs* × Actions* × Events* × Messages × Conditions* × Protocols* → DSolutions** – отношение «быть проектной задачей», обеспечивающее агрегацию проектных операций и порождение или модификацию подмножества проектных решений из множества *DSolutions* в контексте вовлечения в агрегат операций спецификаций из онтологии требований;
- *RDSolutions: DSolutions* × DTasks* × Inputs* × Outputs* × Actions* × Events* × Messages* × Conditions* × Protocols* → DSolutions** – отношение «быть проектным решением», обеспечивающее порождение или модификацию подмножества проектных решений из множества *DSolutions* в контексте использования наборов проектных операций и спецификаций из онтологии требований;
- *RDProcesses: DTasks* × DSolutions* → DTasks** – отношение «быть проектным процессом», обеспечивающим связывание подмножества проектных решений и проектных задач.

Аналогичным онтологии требований образом может быть построена метамодель, которая на этом данном служит не только инструкцией для проектировщиков по формированию онтологии проектирования, но и базой для частичной автоматизации трансформации онтологии требований в онтологию проектирования, поскольку содержит в себе некие правила порождения новых сущностей (например, такие, что для всех процессов, протекающих в СЛУ, должны быть заданы состояния автомата; состояния автомата должны быть связаны соответствующими переходами, удовлетворяющими техническим условиям и т.д.).

По ходу порождения новых сущностей между узлами описанных типов могут появляться следующие семантические отношения:

- взаимообратные отношения между переходами и состояниями (*from* и *to*); при этом, каждый уникальный предикат всегда связан с двумя состояниями двумя такими отношениями («быть переходом из» и «быть переходом в»), которые, в свою очередь, автоматически порождают отношение «переходить в» между данными состояниями;
- «переходить в» (*transitTo*) – отношение перехода из состояния в состояние (генерируется автоматически на основании наличии предиката перехода между состояниями и служит для создания более упрощённого представления онтологической модели);

- «быть предикатом перехода» (*cause*) – отношение причинно-следственного характера между условием (предикатом перехода) и переходом из одного состояния в другое;
- «вызывать» (*call*) – ещё одно отношение причинно-следственного типа между переходом из одного состояния в другое и действием или процессом, которые он вызывает;
- отношения предшествования (*precede*) – отношения между процессами и состояниями.

Дополнительным средством автоматизации построения онтологии проектирования выступают формальные аксиомы (запишем их в формате SWRL):

- `State(?a), State(?b), Transition(?t), from(?a, ?t), to(?t, ?b) -> transitTo(?a, ?b),`

  что означает: если между состоянием `a` и переходом `t` есть связь «быть переходом из» и между переходом `t` и состоянием `b` есть связь «быть переходом в», то в онтологической модели должна появиться связь перехода из состояние `a` в состояние `b`;
- `State(?a), State(?b), Transition(?t), from(?a, ?t), to(?t, ?b), precede(?a,?x), call(?t, ?y) -> precede(?x, ?t), precede(?x, ?y), precede(?y, ?b),`

  что означает: если между состоянием `a` и переходом `t` есть связь «быть переходом из» и между переходом `t` и состоянием `b` есть связь «быть переходом в», а также состояние `a` предшествует некоему процессу `x` и переход `t` вызывает некое событие `y`, то в онтологической модели должна появиться связь предшествования между событием `x` и переходом `t`, между событиями `x` и `y`, а также между событием `y` и состоянием `b`.

В Neo4j такие аксиомы реализуются на уровне запросов к БД (левая часть правила задается клаузулой `MATCH`, правая – клаузулой `CREATE` с использованием выбранных в правой части сущностей). В качестве примера приведем реализацию последнего правила:

Листинг 4. Пример реализации правила логического вывода с помощью Cypher-запроса

```
   MATCH (state_1:State {name: '%s'}), (state_2:State {name: '%s'}),
(t:Transition {name: '%s'}),
   (t)-[:Solutions {name: 'быть переходом из'}]->(state_1), (t)-
[:Solutions {name: 'быть переходом в'}]->(state_2),
   (state_1)-[:Solutions {name: 'предшествовать'}]->(x), (t)-[:Solutions
{name: 'вызывать'}]->(y)
   CREATE (x)-[:Solutions {name: 'предшествовать'}]->(y)
```

Продемонстрируем описанную структуру онтологической модели на примере модели перехода из состояния «Движение вперед» в состояние «Движение вправо». На рис. 10 изображен прототип части онтологической модели (в формате, пригодной для реализации в среде Neo4j), где пунктиром показаны отношения, генерируемые на основе правил логического вывода, цветами показаны типы концептов и связи, относящиеся к онтологии проектирования, черно-белым – связанные с ними концепты онтологии требований.

Рис. 10. Фрагмент онтологии проектирования

4.4 Формирование онтологии реализации и генерация исходного кода программы

Онтология реализации в дополнение к множествам, описанным ранее, включает в себя следующее:

$$IO = (IStates, IPredicates, IFunctions, IOperations, IParameters \mid RIStates, RIPredicates, RIFunctions, RIOperations, RIParameters),$$

- где $IStates$ – множество состояний системы СЛУ на уровне реализации;
- $IPredicates$ – множество условий перехода из одного состояния в другое;
- $IFunctions$ – множество функций исходного кода программы СЛУ;
- IOperations – множество операций исходного кода программы СЛУ;
- IParameter – множество параметров функций исходного кода программы СЛУ;
- $RIStates: DStates^* \to IStates^*$ – отношение «быть состоянием уровня реализации», обеспечивающее связывания состояний уровня проектирования и уровня реализации;
- $RIPredicates: DPredicates^* \to IPredicates^*$ – отношение «быть условием переходы», обеспечивающее связывания предикатов перехода уровня проектирования и условий перехода уровня реализации;
- $RIFunctions: IStates^* \times IPredicates^* \times IFunctions^* \times IOperations^* \times IParameters^* \times DActions^* \to IFunctions^*$ – отношение «быть функцией», обеспечивающее порождение или модификацию подмножества функций исходного кода программы из множества $DActions$ и вовлечение в них состояний, условий перехода, функций, операций и параметров;
- $RIOperations: DActions^* \to IOperations^*$ – отношение «быть операцией», обеспечивающее связывания операций уровня проектирования и уровня реализации;
- $RIParameters: Predicates^* \times IFunctions^* \to IParameters^*$ – отношение «быть параметром», обеспечивающее порождение параметров, задействованных в функциях исходного кода программы и условиях.

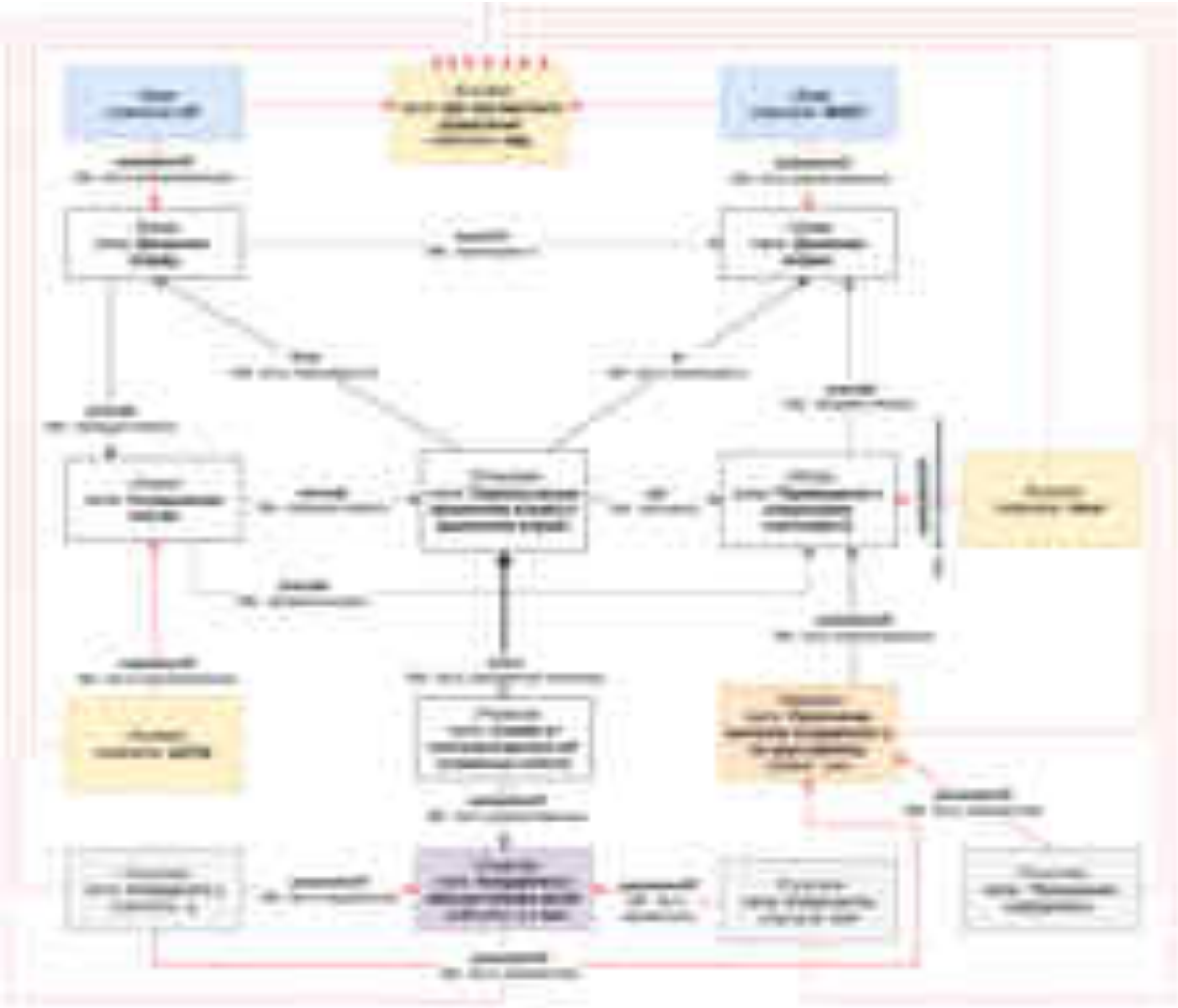


Рис. 11. Фрагмент онтологии реализации

Прототип онтологической модели на этапе реализации на примере перехода из состояния «Движение вперед» в состояние «Движение вправо» показан на рис. 11.

Помимо этого, в онтологии реализации появляются поля, отражающие пространство имён, задействованное в исходном коде (`codename`), а также следующие семантические отношения:

- «быть параметром» (`parameterOf`) – отношения между условиями и параметрами, а также функциями и параметрами;
- «быть реализованным» (`realizationOf`) – отношения между понятиями уровня проектных решений и понятиями уровня реализации.

4.4.1 Генерация исходного кода программы

На основе данной онтологической модели возможна генерация исходного кода функции на языке C, реализующий шаг автоматного управления (`step`). Обобщенная схема работы генератора представлена на рис. 12.

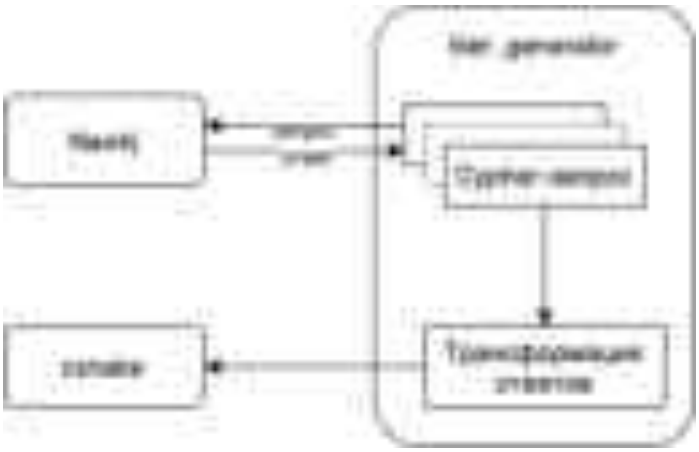


Рис. 12. Структура модуля генерации кода

Для обеспечения генерации кода используется следующий набор Cypher-запросов на чтение данных из онтологии реализации (запросы реализованы в формате отдельных функций на языке Python):

- получение всех состояний уровня реализации (функция `get_states`);
- получение состояний, в которые может быть осуществлён переход из текущего состояния (функция `get_states_to_transit`, которая принимает на вход имя текущего состояния);
- получение списка операций, которые необходимо выполнить до осуществления перехода из текущего состояния (функция `get_operations_before_transition`, которая принимает на вход имя текущего состояния);
- получение условия перехода из состояния А в состояние Б (функция `get_condition`, которая принимает на вход имена состояний);
- получение списка операций, которые необходимо выполнить после осуществления перехода из текущего состояния (функция `get_operations_before_transition`, которая принимает на вход имя предиката перехода).

В таблице представлены листинги данных функций, а также примеры ответов, получаемых при выполнении запросов к базе данных Neo4j. Для удобства обработки ответы трансформируются в словари (тип `dict`) и таблицы (тип `pandas.DataFrame`), которые формируются на выходе функций

Таблица 1. Функции реализации запросов на чтение данных из онтологии, обслуживающих генерацию кода

Листинг функции	Пример ответа

`def get_states():`
`query = """`
`MATCH (code:IState),`
`(code)-[ {name: 'быть реализацией'}]->(solution)`
`RETURN code.codename, solution.name"""`
`res = conn.query(query, db=db_name)`
`return {dict(r)['code.codename']: dict(r)['solution.name'] for r in res}`
	`[<Record code.codename='UP' solution.name='Движение вперёд'>, <Record code.codename='RIGHT' solution.name='Движение вправо'>, <Record code.codename='STOP' solution.name='Не запущено'>, <Record code.codename='INIT' solution.name='Инициализация'>]`

`def get_states_to_transit(state_name):`
`query = """`
`MATCH (state_1:DState {name: '%s'}),`
`(state_1)-[ {name: 'переходить в'}]->(state_2),`
`(state_2_dev)-[ {name: 'быть реализацией'}]->(state_2)`
`RETURN state_1.name, state_2.name, state_2_dev.name""" % state_name`
`res = conn.query(query, db=db_name)`
`return {dict(i)['state_2_dev.name']: dict(i)['state_2.name'] for i in res}`
	`[<Record state_1.name='Движение вперёд' state_2.name='Движение вправо' state_2_dev.name='RIGHT'>]`

`def get_operations_before_transition(state_name):`
`query = """`
`MATCH (state_1:DState {name: '%s'}),`
`(state_1)-[ {name: 'предшествовать'}]->(process_name),`
`(process_code)-[ {name: 'быть реализацией'}]->(process_name)`
`RETURN process_name.name as name, process_code.codename as codename, labels(process_code) as labels`
`""" % state_name`
`res = conn.query(query, db=db_name)`
`return query_res_to_df(res)`
`# return {dict(i)['process_code.codename']: dict(i)['process_name.name'] for i in res}`
	`[<Record name='Укладывание плитки' codename='setTile' labels=['Function', 'Transition', 'Development']>]`

`def get_condition(s_1, s_2):`
`query = """`
`MATCH (state_1:DState {name: '%s'}), (state_2:DState {name: '%s'}),`
`(t)-[{name: 'быть переходом из'}]-
	`[<Record p.name='Справа от плиткоукладчика нет уложенных плиток' p_code.codename='y < turn'>]`

>(state_1), (t)-[{name: 'быть переходом в'}]->(state_2), (p)-[{name: 'быть предикатом перехода'}]->(t), (p_code)-[{name: 'быть реализацией'}]->(p) RETURN p.name, p_code.codename""" % (s_1, s_2) res = conn.query(query, db=db_name) return dict(res[0])['p_code.codename'], dict(res[0])['p.name']	
def get_operations_after_transition(predicate): query = """ MATCH (pr:DPredicate {name: '%s'}), (pr)-[{name: 'быть предикатом перехода'}]->(t), (t)-[{name: 'вызывать'}]->(p), (p_code)-[{name: 'быть реализацией'}]->(p) RETURN p.name as name, p_code.codename as codename, labels(p_code) as labels""" % predicate res = conn.query(query, db=db_name) return query_res_to_df(res)	[<Record name='Перемещение к следующему плиткоместу' codename='move' labels=['Function', 'Process', 'Development']>]

При выполнении запроса необходимо указать название базы данных (db_name). Соединение с базой данных Neo4j осуществляется с помощью класса Neo4jConnection, реализованного с помощью специального модуля neo4j для Python, при объявлении которого необходимо указать параметры доступа к БД (uri, имя пользователя и пароль). Более подробную информацию об использовании Neo4j с помощью Python можно получить в документации [3].

Листинг 5. Реализация класса для осуществления программного доступа к базе Neo4j

```
from neo4j import GraphDatabase


class Neo4jConnection:

    def __init__(self, uri, user, pwd):
        self.__uri = uri
        self.__user = user
        self.__pwd = pwd
        self.__driver = None
        try:
            self.__driver = GraphDatabase.driver(self.__uri, auth=(self.__user, self.__pwd))
        except Exception as e:
            print("Failed to create the driver:", e)
```

```
        def close(self):
            if self.__driver is not None:
                self.__driver.close()

        def query(self, query, db=None):
            assert self.__driver is not None, "Driver not initialized!"
            session = None
            response = None
            try:
                session = self.__driver.session(database=db) if db is not
None else self.__driver.session()
                response = list(session.run(query))
            except Exception as e:
                print("Query failed:", e)
            finally:
                if session is not None:
                    session.close()
            return response
```

Программная реализация генератора кода (см. Листинг 6) осуществляется с использованием библиотеки **csnake** [2]. В начале необходимо создать экземпляр класса `CodeWriter`. Для объявления конструкции `SWITCH` служат несколько методов данного класса:

- `start_switch` (начало конструкции);
- `add_switch_case` (добавление нового условия);
- `add_switch_break` (добавление оператора `break`).

Кроме того, для генерации используются следующие методы класса `CodeWriter`:

- `add_line` (добавление новой строки);
- `indent` (добавление отступа);
- `close_brace` (закрытие скобок);
- `add_enum` (объявление переменной перечисляемого типа);
- `add_function_definition` (объявление функции).

Для объявления переменной перечисляемого типа используется экземпляр класса `Enum`. Добавление новых значений осуществляется с помощью метода `add_values` (в нашем случае в неё записываются состояния).

Для объявления функций используется экземпляр класса `Function` и его метод `add_code` (добавление новых строк кода).

При добавлении новых строк в метод `add_line` можно передать необязательный параметр `comment`, который служит для добавления комментария к коду. Поскольку онтологическая модель СЛУ содержит также и

имена сущностей на естественном языке, то целесообразно использовать их в качестве комментариев.

Более подробно с использованием модуля csnake можно ознакомиться в документации [2].

В результате выполнения функции генерируется код следующего вида (см. Листинг 7).

Листинг 6. Фрагмент генератора кода

```
    from csnake import CodeWriter, Function, Variable, Enum


    def add_process_lines(p_cwr, df):
        for _, row in df.iterrows():
            line = row['codename']
            if 'Function' in row['labels']:
                line += '()'
            p_cwr.add_line(line, comment=row['name'])


    def get_code():
        cwr = CodeWriter()

        # состояния
        states = get_states()
        enum = Enum('STATES')
        enum.add_values(states.keys())
        cwr.add_enum(enum)

        # switch
        cwr_switch = CodeWriter()
        var = Variable('state', 'str')

        cwr_switch.start_switch(var)

        for key_state in states:
            # состояние, из которого осуществляется переход
            state_var = Variable(key_state, 'str')
            cwr_switch.add_switch_case(state_var)

            # действие, которое необходимо выполнить до перехода
            pr_before                                                    =
get_operations_before_transition(states[key_state])
            add_process_lines(cwr_switch, pr_before)

            sts_to_transit = get_states_to_transit(states[key_state])

            for key_state_2 in sts_to_transit:
                # условие перехода
                cond,    cond_comm    =    get_condition(states[key_state],
sts_to_transit[key_state_2])
                cwr_switch.add_line('if (%s) {' % cond, comment=cond_comm)
                cwr_switch.indent()
```

```
                # изменение состояния
                cwr_switch.add_line('state = %s;' % key_state_2)

                # действие, которое необходимо выполнить после перехода
                pr_after = get_operations_after_transition(cond_comm)
                add_process_lines(cwr_switch, pr_after)
                cwr_switch.close_brace()

            cwr_switch.add_switch_break()

        # объявляем функцию step()
        step_fun = Function('step', return_type='void')
        step_fun.add_code(cwr_switch)
        cwr.add_function_definition(step_fun)
```

Листинг 7. Пример сгенерированного кода

```
    enum STATES
    {
        UP,
        RIGHT,
        STOP,
        INIT
    };
    void step(void)
    {
        switch (state)
        {
            case UP:
                setTile() /* Укладывание плитки */
                if (y < turn) { /* Справа от плиткоукладчика нет уложенных
плиток */
                    state = RIGHT;
                    move() /* Перемещение к следующему плиткоместу */
                }
                break;
            case RIGHT:
                break;
            case STOP:
                start = ext:: machine.run() /* Запуск плиткоукладчика */
                if (!start) { /* Нет сигнала на старт укладки плитки */
                    state = STOP;
                }
                if (start) { /* Получен сигнал на старт укладки плитки */
                    state = INIT;
                    init() /* Выполнение инициализации */
                    timeOut = clock() + timeInit /* Установка ограничения
на пребывание в состоянии инициализации */
                }
                break;
            case INIT:
                break;
    }
```

4.5 Трансформации онтологических спецификаций

На вход модулю формирования онтологии поступают формализованные требования, формальная структура которых определяется в том числе типом задействованных, согласно требованиям к проекту, проектных решений. Так, в случае разработки СЛУ плиткоукладчиками формальная структура требований сводится к описанию следующих сущностей (часть этих сущностей наследуется из онтологии требований):

- исходное состояние;
- результирующее состояние;
- условия перехода из исходного состояние в результирующее;
- действия или процессы, которые необходимо осуществить до перехода;
- В действия или процессы, которые необходимо осуществить после перехода.

Пример фрагмента таких формализованных требований представлен в табличном виде на рис. 4.2 (действия или процессы, которые необходимо осуществить до или после перехода из состояния в состояние перечислены через запятую).

Рис. 13. Фрагмент формализованных требований

На основании данной формализации модуль формирования онтологии автоматически генерирует онтологию проектирования и онтологию реализации с занесением соответствующей информации в базу данных Neo4j. Всякий раз, когда в данную структуру вносятся изменения, модуль формирования онтологии может быть запущен повторно, в результате чего в онтологии проекта будут изменены сущности (добавлены новые сущности, удалены неиспользуемые сущности, изменены имена сущностей).

Модуль формирования онтологии генерирует и изменяет онтологию проектирования в соответствии с правилами логического вывода типа:

- если существуют два связанных состояния (состояние A и состояние B), из и в которое необходимо осуществить переход соответственно, то необходимо
    - добавить вспомогательную сущность «Переход из состояния A в состояние B»;
    - связать данную сущность с состоянием A отношением «быть переходом из» и с состоянием B отношением «быть переходом в»;
    - добавить отношение «переходить в» между состоянием A и состоянием B;
- а также если существует условие перехода из состояния A в состояние B, то необходимо
    - добавить отношение «быть предикатом перехода» между условием перехода и новой сущностью «Переход из состояния A в состояние B»;
- а также если существуют действия A (которые необходимо выполнить до перехода), то необходимо
    - добавить отношение «предшествовать» между состоянием A и каждым действием A (которое необходимо выполнить до перехода), а также между каждым действием A и новой сущностью «Переход из состояния A в состояние B»;
- а также если существуют действия B (которые необходимо выполнить после перехода), то необходимо
    - добавить отношение «вызывать» между новой сущностью «Переход из состояния A в состояние B» и каждым действием B (которое необходимо после перехода), а также отношение «предшествовать» между каждым действием B и состоянием B;
- если существуют и действия A, и действия B, то необходимо
    - добавить отношение «предшествовать» между каждым действием A и каждым действием B.

Алгоритм формирования онтологии реализации заключается в том, что для каждой сущности онтологии проектирования (за исключением вспомогательных) необходимо создать соответствующую сущность в онтологии реализации, связанную с первой отношением «быть реализованным» и обладающую следующими свойствами:

- имя на уровне реализации (необязательный параметр; если не задано, то наследуется от онтологии проектирования);
- имя, используемой в исходных кодах программ (обязательный параметр; может быть сгенерировано автоматически с использованием средств автоматического перевода);

- тип (функция, операция, параметр).

Поскольку онтология реализации используется для генерирования исходного кода программы СЛУ и, как было описано выше, ее сущности генерируются на основе онтологии проектирования, формируемой, в свою очередь, на основе формализованных требований; то автоматическая модификация исходного кода также возможна при внесении изменений в требования.

Так, например, в требования может потребоваться внести следующее изменение:

> `SourceN`: перед переходом из состояния «Движение вперед» в состояние «Движение вправо», необходимо переключить передачу на более высокую, поскольку робот осуществляет движение по наклонной плоскости, и движение вперед (вверх) требует пониженной передачи.

В этом случае в онтологию требований и проектирования добавится сущность типа Action «Изменить скорость» с описанием – «Переключить передачу на более высокую», а в онтологию реализации – новая сущность типа Function (функция) changeSpeed, связанная с сущностью «Изменить скорость» отношением «быть реализованным». После чего можно будет запустить скрипт генерации программного кода на основе онтологии проекта и получить модифицированный код.

4.6 Рекомендуемая литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб, 2008. – 167 с.
2. Andrej Radović. Csnake. Developer Interface (API) [Электронный ресурс] — Режим доступа: https://andrejr.gitlab.io/csnake/api.html
3. Using Neo4j from Python. [Электронный ресурс] — Режим доступа: https://neo4j.com/developer/python/

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.07 Организация и автоматизация научных исследований

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Методическое пособие по дисциплине «Б1.О.07 Организация и автоматизация научных исследований» доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/152439

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б1.О.09 Управление проектами в области искусственного интеллекта

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Дисциплина (модуль) Управление проектами в области искусственного интеллекта
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Практическая работа 1

Понятие инновационного проекта. Определение тематики проекта в области ИИ

1. Проанализировать актуальные известные действующие проекты в области искусственного интеллекта (ИИ).

2. Обсудить проекты с преподавателем. Определить инновационную составляющую известных проектов.

3. Сгенерировать собственную тематику возможного проекта в области искусственного интеллекта.

4. В отчет по практической работе включить обзор актуальных известных действующих проектов с описанием инновационной составляющей, предложение по тематике возможного проекта в области искусственного интеллекта с описанием инновационного эффекта.

5. Подготовить отчет по практической работе.

7. Также необходимо быть готовым устно ответить на контрольные вопросы.

Методические указания

Практическая работа может быть выполнена с помощью поиска информации в сети Интернет

Контрольные вопросы

1. Содержание понятие «инновационный проект». Системное представление проекта.
2. Примеры инновационных проектов в области искусственного интеллекта.

Практическая работа № 2-3.
Планирование проекта. Организационная структура проекта

Преподаватель анализирует сформированные студентами предлагаемые темы инновационного проекта в области ИИ. В аудитории проводится голосование / он-лайн голосование по оценке привлекательности предлагаемых тематик проектов. По результатам голосования отбирается несколько проектов, рекомендуемых для разработки в рамках данной дисциплины. Авторы проектных идей, набравшие наибольшее число голосов, получают возможность сформировать проектную команду из числа студентов, изучающих данную дисциплину (до 4 человек в одной команде в зависимости от числа студентов). По определенной тематике разрабатываемого проекта в области искусственного интеллекта разрабатывается организационная структура проекта.

Что такое организационная структура проекта

Организационная структура проекта – это временная организационная структура, созданная для повышения качества управления и взаимодействия в проекте путем определения и визуализации процессов взаимодействия как между внутренними, так и с внешними участниками проекта.

условные типы организационных структур проекта:

Организационная структура управления проектом. Предназначена для определения уровней принятия решений

Организационная структура выполнения проекта предназначена для организации взаимодействия между командами, вовлеченными в проект (архитектура, тестирование, разработка, анализ и проч.).

Организационная структура работы с подрядчиком или подрядчиками в проекте. Согласуется на уровне ответственных за проект от каждой вовлеченной стороны для определения процесса работы и точек принятия решений.

Организационная структура программы проектов. Согласуется на уровне руководителя программы и ее возможного спонсора (в случае наличия спонсоров, готовых профинансировать открытие стартапа) для определения процесса взаимодействия между проектами (и, конечно, руководителями проектов), включенными в программу.

Понять, кто вообще будет вовлечен в проект. Понять, достаточно ли вам будет одной орструктуры или необходимо построить несколько, и для чего вообще вы ее строите. Например, организационная структура управления проектом, которую вы будете согласовывать на уровне управляющего комитета будет отличаться от организационной структуры выполнения проекта для организации взаимодействия между командами или от организационной структуры, которую вы делаете, чтобы четко определить процесс взаимодействия с подрядчиками в этом проекте.

Разработать слайд презентации, например, в MS visio, mindmap или в любом другом инструменте список всех участников.

Определить, какую информацию помимо ролей вам необходимо видеть. Обычно это как минимум должности и подчиненность, а как максимум – уровень принимаемых решений, конкретные имена, регулярность встреч и проч. Пытаться вставить туда все я не

рекомендую – для этого есть план коммуникаций, а картинку с оргструктурой лучше этим не перегружать.

Прорисовать подчиненность/иерархию и направления коммуникации.

Эффектно оформить организационную структуру, избавившись от всей лишней информации, «потерявшихся» людей и стрелок и проч. Организационная структура проекта – один из основополагающих документов и должен выглядеть эффектно, чтобы его воспринимали всерьез.

Показать получившуюся оргструктуру проекта кому-нибудь, не входящему в нее, но понимающему контекст. Этот человек сможет вам подсказать, что в ней непонятно, и, возможно, обратит вниманием на какие-то логические или политические несоответствия, т.к. в процессе разработки взгляд все-таки замыливается.

В случае наличия возможного спонсора согласовать построенную организационную структуру со спонсором проекта или с другими заинтересованными лицами, чем мнение неплохо бы получить до обнародования проекта.

Рис 1. Организационная структура выполнения проекта

Примечание. Это пример реального проекта, для студенческой работы схема будет более общей.

Практическая работа №4-5.

Подбор персонала проекта. Распределение обязанностей и активностей проекта.

На данном этапе детально должны быть прописаны роли всех членов команды, а также возможные роли среди иных специалистов, не присутствующих среди студентов, например, инженеров по знаниям, веб-дизайнеров, программистов, бухгалтеров и пр. Роли прописываются и оформляются в виде таблицы 1.

Таблица 1.

Распределение функционала проектной команды

№пп	Наименование должности в проекте	ФИО	Число ставок
1	Руководитель	Степанов Илья Сергеевич	1
	….		

Практическая работа №6

Процессы и функции управления проектами. Основные и вспомогательные процессы в управлении проектами.

Студенты в командах продолжают работать над проектом.

1. По сформированным проектным идеям необходимо расписать процессы управления проектами.
2. Подготовить презентацию по процессам управления проектом

Управление проектами — это приложение знаний, опыта, методов и средств к работам проекта для удовлетворения требований, предъявляемых к проекту, и ожиданий участников проекта. Чтобы удовлетворить этим требованиям и ожиданиям, необходимо найти оптимальное сочетание между целями, сроками, затратами, качеством и другими характеристиками проекта.

Управление проектами подчиняется четкой логике, которая связывает между собой различные области знаний и процессы управления проектами.

Прежде всего у проекта обязательно имеются одна или несколько целей. Под целями мы будем далее понимать не только конечные результаты проекта, но и выбранные пути достижения этих результатов (например, применяемые в проекте технологии, система управления проектом).

Достижение целей проекта может быть реализовано различными способами. Для сравнения этих способов необходимы критерии успешности достижения поставленных целей. Обычно в число основных критериев оценки различных вариантов исполнения проекта входят сроки и стоимость достижения результатов. При этом запланированные цели и качество обычно служат основными ограничениями при рассмотрении и оценке различных вариантов. Конечно, возможно использование и других критериев и ограничений, в частности ресурсных.

Для управления проектами необходимы рычаги. Влиять на пути достижения результатов проекта, цели, качество, сроки и стоимость исполнения работ можно, выбирая применяемые технологии, состав, характеристики и назначения ресурсов на выполнение тех или иных работ. Таким образом, применяемые технологии и ресурсы проекта можно отнести к основным рычагам управления проектами. Кроме этих основных существуют и вспомогательные средства, предназначенные для управления основными. К таким вспомогательным рычагам управления можно отнести, например, контракты, которые позволяют привлечь нужные ресурсы в нужные сроки. Кроме того, для управления ресурсами необходимо обеспечить эффективную организацию работ. Это касается структуры управления проектом, организации информационного взаимодействия участников проекта, управления персоналом.

Информация, используемая в управлении проектами, обычно не бывает стопроцентно достоверной. Учет неопределенности исходной информации необходим и при планировании проекта, и для грамотного заключения контрактов. Анализу и учету неопределенностей посвящен анализ рисков.

Любой проект в процессе своей реализации проходит различные стадии, называемые в совокупности жизненным циклом проекта. Для реализации различных функций управления проектом необходимы действия, которые в дальнейшем именуются процессами управления проектами.

Процессы управления проектами

Управление проектами — интегрированный процесс. Действия (или их отсутствие) в одном направлении обычно влияют и на остальные направления. Такая взаимосвязь заставляет балансировать между задачами проекта — часто улучшение в одной области может быть достигнуто лишь за счет ухудшения в другой. Для лучшего понимания интегрированной

природы управления проектами опишем его через процессы, из которых оно состоит, и их взаимосвязи.

Проект состоит из процессов. Процесс — это совокупность действий, приносящая результат. Процессы проекта обычно выполняются людьми и распадаются на две основные группы:

- процессы управления проектами — касающиеся организации и описания работ проекта (которые будут подробно описаны далее);
- процессы, ориентированные на продукт — касающиеся спецификации и производства продукта. Эти процессы определяются жизненным циклом проекта и зависят от области приложения.

В проектах процессы управления проектами и процессы, ориентированные на продукт, накладываются и взаимодействуют. Например, цели проекта не могут быть определены при отсутствии понимания того, как создать продукт.

Процессы управления проектами могут быть разбиты на шесть основных групп, реализующих различные функции управления:

- процессы инициации — принятие решения о начале выполнения проекта;
- процессы планирования — определение целей и критериев успеха проекта и разработка рабочих схем их достижения;
- процессы исполнения — координация людей и других ресурсов для выполнения плана;
- процессы анализа — определение соответствия плана и исполнения проекта поставленным целям и критериям успеха и принятие решений о необходимости применения корректирующих воздействий;
- процессы управления — определение необходимых корректирующих воздействий, их согласование, утверждение и применение;
- процессы завершения — формализация выполнения проекта и подведение его к упорядоченному финалу.

Процессы управления проектами накладываются друг на друга и происходят с разной интенсивностью на всех стадиях проекта, как показано на рис. 1.

Кроме того, процессы управления проектами связаны своими результатами — результат выполнения одного становится исходной информацией для другого. Эти взаимосвязи проиллюстрированы на рис. 2.

И наконец, имеются взаимосвязи групп процессов различных фаз проекта. Например, закрытие одной фазы может являться входом для инициации следующей фазы (пример: завершение фазы проектирования требует одобрения заказчиком проектной документации, которая необходима для начала реализации).

В реальном проекте фазы могут не только предшествовать друг другу, но и накладываться.

Повторение инициации на разных фазах проекта помогает контролировать актуальность выполнения проекта. Если необходимость его осуществления отпала, очередная инициация позволяет вовремя это установить и избежать излишних затрат.

Рис. 2. Взаимосвязи групп процессов управления проектами в фазе

Взаимосвязи процессов

Процессы инициации

Инициация включает единственный подпроцесс — авторизацию, то есть решение начать следующую фазу проекта.

Процессы планирования

Планирование имеет большое значение для проекта, поскольку проект содержит то, что ранее не выполнялось. Естественно, что планирование включает сравнительно много процессов. Однако не следует считать, что управление проектами — это в основном

планирование. Усилия, прилагаемые для планирования, следует соизмерять с целями проекта и полезностью полученной информации.

Напомним, что следует различать цели проекта и цели продукта проекта, под которым понимается продукция (или услуги), созданная или произведенная в результате исполнения проекта.

- Цели продукта — это свойства и функции, которыми должна обладать продукция проекта.
- Цели проекта — это работа, которую нужно выполнить для производства продукта с заданными свойствами.

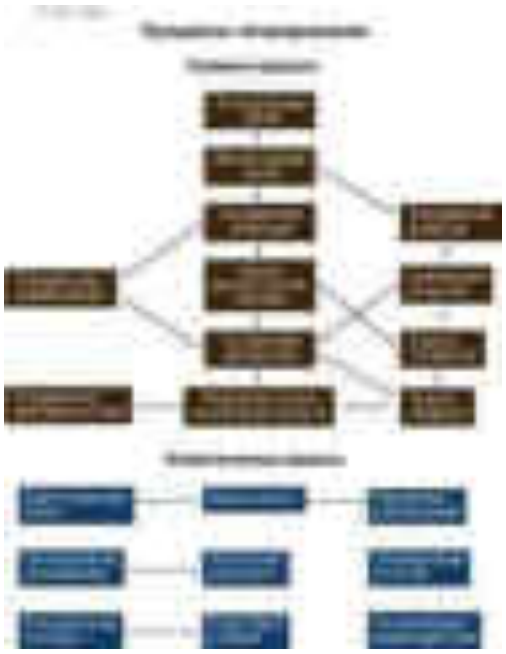


Рис. 3. Взаимосвязи процессов планирования

Взаимосвязи между процессами планирования представлены на рис. 3.

В ходе исполнения проекта эти процессы многократно повторяются. Изменениям могут подвергнуться цели проекта, его бюджет, ресурсы и т. д. Кроме того, планирование проекта — это не точная наука. Различные команды проекта могут разработать различные планы для одного и того же проекта. А пакеты управления проектами могут составить различные расписания выполнения работ при одних и тех же исходных данных.

Некоторые из процессов планирования имеют четкие логические и информационные взаимосвязи и выполняются в одном порядке практически во всех проектах. Так, например, сначала следует определить, из каких работ состоит проект, а уж затем рассчитывать сроки выполнения и стоимость проекта. Эти основные процессы выполняются по нескольку раз на протяжении каждой фазы проекта.

Кроме перечисленных основных процессов планирования имеется ряд вспомогательных процессов, необходимость в использовании которых сильно зависит от природы конкретного проекта:

- планирование качества — определение того, какие стандарты качества использовать в проекте, и того, как этих стандартов достичь;
- планирование организации — определение, документирование и назначение ролей, ответственности и взаимоотношений отчетности в организации;
- назначение персонала — назначение человеческих ресурсов на выполнение работ проекта;
- планирование взаимодействия — определение потоков информации и способов взаимодействия, необходимых для участников проекта;
- идентификация риска — определение и документирование событий риска, которые могут повлиять на проект;
- оценка риска — оценка вероятностей наступления событий риска, их характеристик и влияния на проект;
- разработка реагирования — определение необходимых действий для предупреждения рисков и реакции на угрожающие события;
- планирование поставок — определение того, что, как и когда должно быть поставлено;
- подготовка условий — выработка требований к поставкам и определение потенциальных поставщиков.

Взаимосвязи между вспомогательными подпроцессами, как и само их наличие, в большой мере зависят от природы проекта.

Процессы исполнения и контроля

Под исполнением подразумеваются процессы реализации составленного плана. Исполнение проекта должно регулярно измеряться и анализироваться для того, чтобы выявить отклонения от намеченного плана и оценить их влияние на проект. Регулярное измерение параметров проекта и идентификация возникающих отклонений далее также относится к процессам исполнения и именуется контролем исполнения. Контроль исполнения следует проводить по всем параметрам, входящим в план проекта.

Рис. 4. Взаимосвязи процессов исполнения

Как и в планировании, процессы исполнения (рис. 4) можно подразделить на основные и вспомогательные.

К основным можно отнести сам процесс исполнения плана проекта.

Среди вспомогательных процессов отметим:

- учет исполнения — подготовка и распределение необходимой для участников проекта информации с требуемой периодичностью;
- подтверждение качества — регулярная оценка исполнения проекта с целью подтверждения соответствия принятым стандартам качества;
- подготовка предложений — сбор рекомендаций, отзывов, предложений, заявок и т. д.;
- выбор поставщиков — оценка предложений, выбор поставщиков и подрядчиков и заключение контрактов;
- контроль контрактов — контроль исполнения контрактов поставщиками и подрядчиками;
- развитие команды проекта — повышение квалификации участников команды проекта.

Процессы анализа

Процессы анализа включают как анализ плана, так и анализ исполнения проекта.

Анализ плана означает определение того, удовлетворяет ли составленный план исполнения проекта предъявляемым к проекту требованиям и ожиданиям участников проекта. Он выражается в оценке показателей плана командой и другими участниками проекта. На стадии планирования результатом анализа плана может быть принятие решения о необходимости изменения начальных условий и составления новой версии плана либо принятие разработанной версии в качестве базового плана проекта, который в дальнейшем служит основой для измерения исполнения. В дальнейшем изложении анализ плана не выделяется в качестве отдельной группы процессов, а включается в группу процессов планирования, делая эту группу процессов по своей природе итеративной. Таким образом, под процессами анализа в дальнейшем понимаются процессы анализа исполнения.

Рис. 5. Взаимосвязи процессов анализа

Процессы анализа исполнения предназначены для оценки состояния и прогноза успешности исполнения проекта согласно критериям и ограничениям, определенным на стадии планирования. В силу уникальности проектов эти критерии не являются универсальными, но для большинства проектов в число основных ограничений и критериев успеха входят цели, сроки, качество и стоимость работ проекта. При отрицательном прогнозе принимается решение о необходимости корректирующих воздействий, выбор которых осуществляется в процессах управления изменениями.

Процессы анализа также можно подразделить на основные и вспомогательные.

К основным относятся те процессы анализа, которые непосредственно связаны с целями проекта и показателями, характеризующими успешность исполнения проекта:

- анализ сроков — определение соответствия фактических и прогнозных сроков исполнения операций проекта директивным или запланированным;
- анализ стоимости — определение соответствия фактической и прогнозной стоимости операций и фаз проекта директивным или запланированным;
- анализ качества — мониторинг результатов с целью их проверки на соответствие принятым стандартам качества и определения путей устранения причин нежелательных результатов исполнения качества проекта;
- подтверждение целей — процесс формальной приемки результатов проекта его участниками (инвесторами, потребителями и т. д.).

Вспомогательные процессы анализа связаны с анализом факторов, влияющих на цели и критерии успеха проекта. Эти процессы включают:

- оценку исполнения — анализ результатов работы и распределение проектной информации с целью снабжения участников проекта данными о том, как используются ресурсы для достижения целей проекта;
- анализ ресурсов — определение соответствия фактической и прогнозной загрузки и производительности ресурсов запланированным, а также анализ соответствия фактического расхода материалов плановым значениям.

В число процессов анализа не включены анализ взаимодействия с целью оптимизации процедур обработки информации, анализ исполнения контрактов с целью своевременного внесения изменений и предотвращения споров и ряд других процессов, которые не носят регулярного характера (как анализ взаимодействия) либо составляют часть включенных процессов (как анализ контрактов).

В результате анализа либо принимается решение о продолжении исполнения проекта по намеченному ранее плану, либо определяется необходимость применения корректирующих воздействий.

Процессы управления

Управление исполнением проекта — это определение и применение необходимых управляющих воздействий с целью успешной реализации проекта. Если исполнение проекта происходит в соответствии с намеченным планом, то управление фактически сводится к исполнению — доведению до участников проекта плановых заданий и контролю их реализации. Эти процессы нами включены в процессы исполнения.

Другое дело, если в процессе реализации возникли отклонения, анализ которых показал, что необходимо определение и применение корректирующих воздействий. В этом случае требуется найти оптимальные корректирующие воздействия, скорректировать план оставшихся работ и согласовать намеченные изменения со всеми участниками проекта. Итак, процессы управления предназначаются для определения, согласования и внесения необходимых изменений в план проекта. Такие процессы управления часто называются управлением изменениями и инициируются процессами анализа (рис. 6).

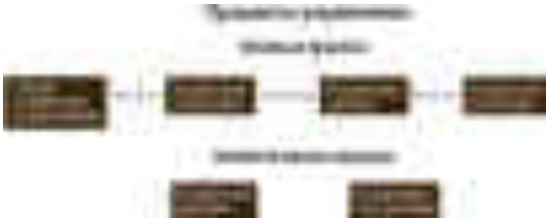


Рис. 6. Взаимосвязи процессов управления

К основным процессам управления, встречающимся практически в каждом проекте, относятся:

- общее управление изменениями — определение, согласование, утверждение и принятие к исполнению корректирующих воздействий и координация изменений по всему проекту;

- управление ресурсами — внесение изменений в состав и назначения ресурсов на работы проекта;
- управление целями — корректировка целей проекта по результатам процессов анализа;
- управление качеством — разработка мероприятий по устранению причин неудовлетворительного исполнения.

Среди вспомогательных процессов управления отметим:

- управление рисками — реагирование на события и изменение рисков в процессе исполнения проекта;
- управление контрактами — координация работы (суб)подрядчиков, корректировка контрактов, разрешение конфликтов.

Рис. 7. Взаимосвязи процессов завершения

Процессы завершения

Завершение проекта сопровождается следующими процессами (рис. 7):

- закрытие контрактов — завершение и закрытие контрактов, включая разрешение всех возникших споров;
- административное завершение — подготовка, сбор и распределение информации, необходимой для формального завершения проекта.

Методы и технологии реализации перечисленных процессов, их интеграция составляют сущность управления проектами. Обратите внимание, что все перечисленные процессы приложимы к проектам любой природы — и к строительным, и к информационным, и к любым другим. Однако имеются и существенные отличия в управлении проектами различных типов. Следует также отметить, что успешное внедрение системы управления проектами связано с определенной организационной перестройкой и с внедрением специализированных программных средств. Перечисленные вопросы, а также специализированные методы решения отдельных задач управления проектами, технология, опыт и проблемы внедрения будут раскрыты в последующих публикациях.

Практическая работа №7.

Понятие инициации, планирования, выполнения, контроля и закрытия проекта.

Студенческие команды определяют основные процессы управления проектом и готовят презентации по следующим 5 стандартным процессам управления проектом: инициации, планирования, выполнения, контроля и закрытия проекта.

Стандартная схема жизненного цикла проекта состоит из пяти фаз:

- Инициация
- Планирование
- Исполнение
- Контроль
- Завершение

Обычно, когда начинается новый проект, эти пять фаз следуют друг за другом по порядку, но так бывает не всегда. Например, в случае каких-то изменений в ходе выполнения проекта вы можете вернуться к фазе планирования, чтобы учесть изменения, но не станете снова повторять весь цикл, начиная с фазы инициации.

Такой уровень гибкости упрощает процессы управления изменениями в ходе жизненного цикла проекта.

В чем важность жизненного цикла проекта? Эта последовательность фаз может показаться вам формальностью, но на самом деле она очнь помогает. Она позволяет применять организованный подход к управлению проектом. Это значит, что у вас будут более четкие роли и обязанности, улучшенное взаимодействие и стабильные результаты (а упущений и забытых задач станет намного меньше!).

Фаза 1. Инициация

Это подготовительный этап, когда вам нужно убедиться, что проект действительно можно осуществить, прежде чем вкладывать силы в планирование и последующие задачи.

В контексте нашей работы этот этап включает в себя описание проекта, создание экономического обоснования, выявление ключевых участников и утверждение проекта соответствующими сторонами.

Не нужно недооценивать важность фазы инициации. На этом этапе закладывается фундамент для успешного выполнения проекта.

Когда вы займетесь задачами и сроками, будет очень легко забыть о том, в чем заключается важность проекта. На самом деле четко представляют себе бизнес-цели проекта только 55% участников, преимущественно руководители команд и менеджеры проектов.

При описании проекта нужно обязательно указать конкретные преимущества для бизнеса, связанные с вашим проектом. Это поможет вам и участникам вашей команды не только нацелиться на достижение целей, но и заранее убедиться, что время не будет потрачено зря.

Составление устава проекта поможет вам предоставить руководителям конкретные факты и получить их одобрение.

Фаза 2. Планирование

На этапе планирования вы составляете смету и назначаете дату, когда проект будет запущен в опытную эксплуатацию.

Трудно переоценить важность тщательного планирования, но очень многие команды и компании пропускают этот этап, чтобы сразу приступить к работе. Таким образом, фаза планирования стоит того, чтобы вложить в нее время и силы. Будьте уверены, это сэкономит вам кучу времени и нервов на последующих этапах.

Фаза планирования требует тщательного обдумывания множества элементов, связанных с вашим проектом. В числе прочего это могут быть следующие элементы:

- Ключевые роли и сферы ответственности
- Показатели успеха
- Возможные риски и препятствия, снижающие эффективность
- Ожидания в отношении внутрикомандного взаимодействия

- Календарный план проекта

Все эти элементы должны быть описаны в подробном плане проекта, который вы представите команде на организационном совещании, а затем станете ссылаться на него на протяжении работы над проектом.

Кстати, об организационном совещании. Оно заслуживает особого внимания, поскольку является одним из важных этапов планирования.

В ходе организационного совещания вы и участники проектной группы должны будете обсудить несколько важнейших элементов плана, в том числе сферы ответственности, рекомендации по организации взаимодействия и критерии успешности проекта. Также можно выяснить, какие программные решения для управления жизненным циклом проекта или другие инструменты вы будете использовать.

Помните, что это совещание не просто дает вам возможность донести до всех свою идею и раздать указания. Выделите время для вопросов и отзывов участников проектной группы, чтобы заранее разрешить любые сомнения и избежать недоразумений. В результате у вас должны сформироваться общие представления о том, как будет выполняться проект.

Фаза 3. Исполнение

Во время этой фазы разработанные планы начинают выполняться. Как ни странно, это именно та фаза, когда все может пойти не так, как планировалось.. Планировать действия гораздо проще, чем эти действия выполнять.

Но помните о том, что ничто не случится само по себе только лишь потому, что у вас есть план. Необходимо убедиться, что каждый участник придерживается этого плана и выполняет свою часть работы.

Сразу же после организационного совещания займитесь распределением дополнительных ресурсов, раздайте указания к действию, ссылки и план проекта, чтобы участники могли сверяться с ним. Воспользуйтесь моментом и не теряйте времени, а сразу приступайте к делу.

Фаза 4. Контроль

Эта фаза обычно совпадает по времени с этапом выполнения проекта. Пока разработчики работают, например, над программным обеспечением, вы тщательно следите за ходом работ и заодно сверяетесь со сметой и графиком, чтобы удостовериться, что все идет по плану.

Учтите, что жизненный цикл управления проектом требует определенной гибкости.

Во время фазы контроля могут возникать ситуации, требующие корректировки первоначального плана. Это нормально — именно для этого фаза контроля и нужна!

Организационные совещания необходимы для запуска проектного механизма в действие, но кроме него периодически следует проводить контрольные совещания для мониторинга хода работ и корректировки курса. Кроме того, такие встречи — отличная возможность для оперативного сбора отзывов, чтобы не запускать проблемные или конфликтные ситуации.

Использование платформы управления проектами обеспечивает прозрачность данных о процессе. Этот инструмент позволит вам получать общее представление о ходе выполнения проекта, а также отслеживать отдельные задачи и рабочую загрузку исполнителей, обходясь без лишних совещаний.

Следя за ходом работ, обязательно уделяйте внимание ресурсам (в число которых всходит абсолютно все — от времени до материалов). Выясняйте, выделяются ли они в нормальном объеме или их не хватает. При управлении ресурсами очень важно действовать на опережение, особенно если учесть, что по результатам исследования, проведенного Институтом управления проектами (PMI) в 2018 году, в 21% случаев проекты срываются из-за недостающих или ограниченных ресурсов.

Фаза 5. Завершение

Вот и все. Проект закончен.

Фаза завершения — это финал проекта, когда проще не подчищать хвосты, а оставить все как есть. Но последнее впечатление не менее важно, чем первое, и для завершения необходимо выполнить еще несколько действий.

Вместо того, чтобы счесть проект «законченным», во время этой последней фазы стоит позаботиться о паре финальных задач:

- Сдача проекта клиенту или команде, которая будет работать с ним дальше
- Размещение всех документов и материалов в централизованном хранилище (чтобы обратиться к ним в будущем, если потребуется)
- «Разбор полетов», чтобы извлечь полезные уроки из успехов и неудач

Практическая работа №7. Часть 2.

Целеполагание. Формулировка целей.

На предыдущих этапах проектной работы цели проекта уже были сформированы. На данном этапе сформированные цели уточняются и окончательно прописываются в разрабатываемой проектной отчетности. Для целеполагания рекомендуется использовать технологию SMART.

Целеполагание — это процесс определения целей в жизни, бизнесе, творчестве и любой другой деятельности. Протекает осознанно или стихийно, бессистемно.

При осознанном целеполагании постановка целей осуществляется поэтапно и основывается на жизненной/деловой миссии, анализе потребностей и ресурсов.

При стихийном подходе цели рождаются спонтанно, без участия самого субъекта, под напором внешних обстоятельств. Например, человек хочет поступить в экономический ВУЗ, потому что настояли родители. Компания начинает выпуск продукции, потому что появилось дешёвое сырьё. Такая схема не приводит к высоким результатам, поскольку цели выбираются хаотично, а в их основе лежат не реальные потребности, а случайные обстоятельства.

Осознанное целеполагание базируется на системе истинных потребностей и ценностей. Оно предполагает:

- Осознание собственных потребностей – того, в чём человек нуждается больше всего: в любви, внимании, развитии, образовании, власти, и т.д.
- Определение ценностей, миссии компании или человека, которые раскрывают смысл существования.
- Выбор приоритетов – основных и второстепенных целей.
- Построение дерева целей – системы целей, распределенных во времени или по структурным подразделениям (в случае компании).

SMART — прекрасный инструмент. Один из пунктов SMART гласит, что любая цель должна иметь количественный показатель, то есть результат по ней должен быть оцифрован. Я убеждена, что каждая цель может быть оцифрована. Любой процесс, любое решение в компании влияет на ее финансовый результат в той или иной степени и соответственно его можно оценить. Это вопрос широты мышления, способности увидеть весь процесс целиком и практики.

Рис.8 Описание технологии SMART

SMART является набором критериев, по которым пишется формулировка цели. Аббревиатура SMART сформирована от 5 английских слов:

S	**Specific** — конкретный.
M	**Measurable** — измеримый.
A	**Achievable** — достижимый.
R	**Relevant** — соответствующий.
T	**Time bounded** — ограниченный во времени.

Эффективность цели также повысится, если она будет про конкретный результат, а не процесс. Не работают цели с формулировками «принимать участие», «формировать», «обеспечивать». Данные глаголы не предусматривают конкретного завершения какого-либо действия и соответственно трудно измеримы. Уверена, что вы понимаете почему. Используйте глаголы совершенного вида прошедшего времени, то есть «выпустить», «создать», «обеспечить». Если вам необходимо поставить цель, касающуюся длительного проекта, но в рамках более короткого, чем предусмотрено проектом, периода, то рекомендую разбить эту цель на несколько промежуточных и сформулировать их по каждому этапу отдельно.

Целеполагание — это процесс определения целей в жизни, бизнесе, творчестве и любой другой деятельности. Протекает осознанно или стихийно, бессистемно.

При осознанном целеполагании постановка целей осуществляется поэтапно и основывается на жизненной/деловой миссии, анализе потребностей и ресурсов.

При стихийном подходе цели рождаются спонтанно, без участия самого субъекта, под напором внешних обстоятельств. Например, человек хочет поступить в экономический ВУЗ, потому что настояли родители. Компания начинает выпуск продукции, потому что появилось дешёвое сырьё. Такая схема не приводит к высоким результатам, поскольку цели

выбираются хаотично, а в их основе лежат не реальные потребности, а случайные обстоятельства.

Осознанное целеполагание базируется на системе истинных потребностей и ценностей. Оно предполагает:

- Осознание собственных потребностей – того, в чём человек нуждается больше всего: в любви, внимании, развитии, образовании, власти, и т.д.
- Определение ценностей, миссии компании или человека, которые раскрывают смысл существования.
- Выбор приоритетов – основных и второстепенных целей.
- Построение дерева целей – системы целей, распределенных во времени или по структурным подразделениям (в случае компании).

Отчетность по практической работе.

Описать все цели проекта по технологии SMART.

Практическая работа №8-9.

Календарное планирование и организация системы контроля проекта. Структурная декомпозиция работ.

Необходимо составить календарный план работ по проекту и сформировать систему контроля проекта. Для уточнения работ нужно провести структурную декомпозицию работ.

Отчетность по практической работе:

Календарный план работ по проекту и структурная декомпозиция работ. Студенческие команды представляют также в виде презентации разработанные материалы.

Структурная декомпозиция работ (СДР или WBS - Work Breakdown Structure) – это представление проекта в виде иерархической структуры работ, полученной путем последовательной декомпозиции. СДР предназначена для детального планирования, оценки стоимости и обеспечения персональной ответственности исполнителей.

Структурная декомпозиция работ (СДР) имеет следующие характеристики:

описывает с необходимой точностью содержание работ по проекту;

определяет весь объем работ по проекту;

формируется в виде иерархической структуры;

имеет измеримый результат, который рассматривается как результат совокупности работ.

СДР является средством для разделения всех работ по проекту на управляемые пакеты работ, позволяющие достичь такого уровня детализации информации, который соответствует потребностям руководства проекта для осуществления контроля.

Разработка СДР имеет две основные цели:

обеспечение планирования всех необходимых работ проекта,

обеспечение отсутствия работ, не связанных с реализацией проекта.

Для руководителя проекта важны обе эти цели. Если в плане отсутствуют все необходимые работы, проект будет задержан, бюджет, скорее всего, будет превышен. Если выполняются работы, не относящиеся к данному проекту – деньги заказчика тратятся нецелевым образом. Если СДР не объединяет обе эти цели, проект может потерпеть неудачу.

Проект разворачивается вокруг СДР. СДР является основным «стержнем» для четырех основных и одного вспомогательного процессов:

определения работ,

планирования ресурсов,

оценки стоимости,

бюджетирования,

определения рисков.

Структурная декомпозиции работ может разрабатываться «с нуля» либо с использованием компонентов уже созданных СДР.

Прежде всего, нужно ответить на целый ряд вопросов:

что нужно сделать (определить продукты проекта);

как это нужно будет делать (определить технологические этапы проекта);

кто это будет делать (определить исполнителей, соисполнителей, субподрядчиков);

кто и в какой форме будет оплачивать работы (определить, какие и с кем будут заключены контракты).

Правила разработки СДР

При разработке СДР необходимо принимать во внимание следующие основные правила:

каждый элемент СДР должен обеспечивать достижение измеримого результата;

каждый элемент СДР должен являться агрегатом всех подчиненных элементов, перечисленных непосредственно под ним;

результаты должны логически декомпозироваться до уровня, на котором можно определить, как они будут достигаться (проектирование, поставки, заключение договоров, производство). Результаты проекта от верхнего до нижнего уровня декомпозиции должны быть логически связаны;

результаты пакетов работ должны быть уникальными и отличаться от результатов других пакетов работ того же уровня;

пакеты работ должны декомпозироваться до уровня детализации, обеспечивающей успешное планирование, координацию и контроль работ, связанных с достижением поставленных целей;

при изменении содержания проекта СДР должна быть откорректирована;

для всех важных событий, связанных с отчетностью (например, ежемесячные отчеты, отчеты о проведении испытаний и т.д.) должны быть определены соответствующие пакеты работ;

все пакеты работ должны быть совместимы с организационной структурой и структурой затрат;

результаты должны быть четко определены так, чтобы исключить дублирование объемов работ внутри элементов СДР, в целом по организации или отдельными ответственными за выполнение работ;

результаты должны иметь размер, достаточный для эффективного управления, но не настолько малый, чтобы сделать затраты на контроль чрезмерными.

Этапы разработки СДР

СДР разрабатывается путем итерационного рассмотрения целей и результатов проекта, критериев планирования объема работ, реализации технических требований. Верхние уровни СДР могут быть разработаны на ранней, концептуальной стадии проекта.

Дальнейшая детализация СДР возможна, как только будет определен проект и подготовлены спецификации.

Очень важно понять, что первоочередная задача составления СДР – разделить проект на подпроекты до той степени детализации, когда появится возможность распределить элементарные работы.

Основной процесс разработки СДР состоит из следующих шагов:

Определение конечных результатов проекта – что должно быть произведено для обеспечения успешного завершения проекта. В качестве руководства рекомендуется проанализировать, рассмотреть документы, описывающие общий объем работ по проекту.

Определение основных пакетов работ, необходимых для получения продукта проекта. Часто такими основными пакетами работ являются результаты, необходимые для создания продукта проекта, но вместе с тем, сами по себе они не являются целями проекта (например, технические требования к разработке ИС).

Приведение в соответствие элементов дополнительных уровней детализации и внутренней системы управления и контроля. Такие элементы обычно связаны с четким и раздельным определением отдельных результатов (продуктов) проекта.

Пересмотр (анализ) и усовершенствование СДР до тех пор, пока все участники проекта не будут согласны, что планирование проекта может быть успешно завершено, и можно будет успешно управлять, контролировать и регулировать получаемые результаты.

Подготовку структурной декомпозиции работ можно считать законченной, когда определены мелкие индивидуальные (элементарные) работы. Ответственность за каждую элементарную работу должна быть поручена одному и только одному члену команды проекта. Если этот человек (или группа) собираются выполнять работу, а не руководить ее выполнением, этот уровень может быть признан самым нижним уровнем СДР.

Календарное планирование в управлении проектами – это ключевой и важный процесс, результатом которого является утвержденный руководством компании календарный план проекта (часто его называют еще **планом-графиком**, календарным графиком, планом управления проектом). Цель календарного планирования – получить точное и полное расписание проекта с учетом работ, их длительностей, необходимых ресурсов, которое служит основой для исполнения проекта.

Календарное планирование включает в себя:

планирование содержания (scope) проекта и построение СДР - структурной декомпозиции работ, или WBS (Work Breakdown Structure);

определение последовательности работ и построение сетевого графика;

планирование сроков, длительностей и логических связей работ и построение диаграммы Ганта;

определение потребности в ресурсах (люди, машины и механизмы, материалы и т.д.) и составление ресурсного плана проекта;

расчет затрат и трудозатрат по проекту.

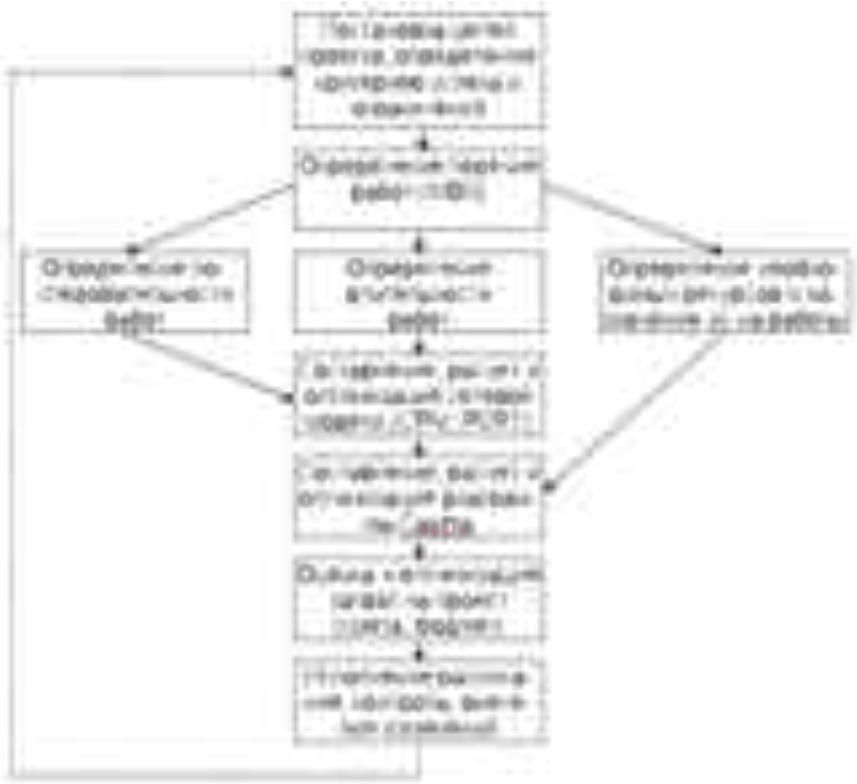


Рис.10. Алгоритм планирования проекта

Составление календарного плана-графика проекта включает в себя несколько аспектов. Мы должны спланировать сроки и длительности работ, определить их последовательность и взаимосвязи, подумать о необходимых ресурсах, учесть стоимость этих работ и ресурсов. В дальнейшем, когда проект перейдет на стадию исполнения, то есть практической реализации запланированных действий, именно по этому плану-графику мы отслеживаем ход выполнения работ. И, если что-то в проекте пойдет не так, можно, сверив с первоначальным планом проекта, внести соответствующие изменения.

Как правило, план-график проекта разрабатывается менеджером проекта с привлечением людей, которые являются экспертами в той или иной области. Например, содержание строительных работ лучше всего знает специалист по строительству; а мероприятия по продвижению продукта, скорее всего, спланирует маркетолог. В результате мы получаем полный перечень работ, структурированный по иерархическому признаку, то есть структурную декомпозицию работ (СДР).

Следующий шаг по созданию календарного плана проекта – это определение длительностей работ и их взаимосвязей. Например, какие-то работы в нашем списке могут выполняться строго последовательно, а какие-то – параллельно друг с другом во времени. Для того чтобы «увязать» сроки работ по проекту, их продолжительность и зависимости, сегодня во всем мире менеджеры проектов используют простой и вместе с тем полезный инструмент календарного планирования – диаграмму Ганта (иногда пишется «диаграмма Гантта»). Диаграмма Ганта – это наглядное представление календарного плана-графика проекта, в котором слева расположен иерархический перечень всех работ проекта (СДР), и справа – календарь с конкретными датами. Работы обозначены полосками, связи между работами - стрелками.

Практическая работа №10.

Управление рисками проекта. Мониторинг и контроль рисков.

В практической работе студенческие команды должны определить риски своего проекта, а также сформулировать мероприятия по мониторингу рисков.

Отчетность по практической работе:

Описание возможных рисков проекта и плана мониторинга рисков, доклад студенческой команды на практическом занятии.

Основными задачами мониторинга и управления рисками являются отслеживание идентифицированных рисков, поиск и получение информации для выявления новых рисков и планирование управления ими, обеспечение своевременной реализации плана мероприятий по управлению рисками, а также оценка их эффективности.

Получаемая в процессе мониторинга информация используется для добавления в реестр рисков не выявленных ранее угроз или возможностей, а также для повторной качественной и количественной оценки рисков и корректировки плана мероприятий по управлению рисками. Пересмотр рисков основывается на сопоставлении получаемой информации о рисках с установленными на этапе планирования и идентификации индикаторами и триггерами риска. По результатам мониторинга принимаются решения о реализации соответствующих мероприятий по управлению.

В рамках мониторинга осуществляется аудит рисков, цель которого — сбор и документирование информации о реализовавшихся рисках, принятых мерах по управлению рисками, а также проводится общая оценка эффективности управления рисками проекта. Результаты аудита рисков используются для совершенствования и повышения эффективности системы управления рисками проекта, а отчетные документы передаются в архив.

Процесс мониторинга и управления рисками осуществляется непрерывно на протяжении всего жизненного цикла проекта и тесно взаимосвязан с процессом коммуникаций и консультаций в области управления рисками. На начальных этапах проекта коммуникации со всеми участниками проекта необходимы для определения внешних и внутренних условий, а также четкой постановки целей и критериев успешности управления рисками проекта. На последующих стадиях процесса управления рисками проекта, от идентификации до планирования мероприятий но управлению рисками, коммуникации необходимы для получения информации о рисках и их анализа, для определения доступных альтернатив и инструментов управления рисками и формирования плана мероприятий по управлению рисками, а также для согласованного определения ответственных за их выполнение. Непосредственно в процессе мониторинга коммуникации направлены не только на получение новой информации, но, возможно, в первую очередь на осведомление основных участников проекта о существенных событиях, произошедших в проекте, о реализовавшихся и потенциальных рисках и их влиянии на достижение целей проекта. Открытые и регулярные коммуникации позволяют не только обеспечить прозрачность управления рисками проекта, но и повысить осведомленность и вовлеченность участников в данный процесс.

Практическая работа №11.

Управление персоналом в проекте. Подбор экспертов для формирования баз знаний.

В данной работе проводится детальный анализ персонала, имеющегося в проекте, в числе студенческой команды, а также функционал привлекаемого персонала. Описываются основные требования к специалисту-эксперту, используемого для формирования базы знаний проекта в сфере ИИ.

Форма отчетности:

Детальное описание функционала персонала проекта, способы руководства персоналом.

Управление персоналом проекта включает организационное планирование, кадровое обеспечение проекта, создание команды проекта, а также реализует функции контроля и мотивации трудовых ресурсов проекта для эффективного выполнения работ и успешного завершения проекта. Целью при этом являются руководство и координация деятельности команды проекта. Для достижения цели используются различные стили руководства, разнообразные административные методы и методы мотивации, повышение квалификации кадров на всех этапах жизненного цикла проекта.

Иногда в качестве синонима к термину «управление персоналом» приравнивают и более узкое, но в рамках управления проектами наиболее важное понятие — «управление командой проекта». Следует оговориться, что управление командой проекта хотя и является основной частью управления персоналом в рамках управления проектами, но далеко не исчерпывает основную его специфику. Управление персоналом проекта помимо управления командой также еще включает и проблемы, свойственные управлению персоналом в рамках общего менеджмента.

Управление персоналом проекта включает процессы, необходимые для того, чтобы сделать использование персонала, вовлеченного в проект, наиболее эффективным.

Организационное планирование (Organizational Planning) — определение, документирование и распределение ролей в проекте, ответственности и отчетности.

Подбор персонала (Staff Acquisition) — подбор персонала для работ над проектом.

Развитие команды (Team Development) — развитие индивидуальных и групповых навыков для улучшения качества работы над проектом.

Основные проблемы, свойственные управлению персоналом любого проекта:

- 1) управление командой проекта (образование команды проекта, ее развитие, проблемы расформирования команды);
- 2) разрешение конфликтов, возникающих в связи с использованием проектно-матричных организационных структур управления проектами;
- 3) проблемы общего управления, связанные со взаимодействием участников проекта с другими членами организаций.

При использовании проектно-матричных организационных структур управления проектами возникает ставшая уже классической проблема управления персоналом, связанная с двойным подчинением сотрудников функциональных подразделений, участвующих в проекте. Так как двойное подчинение изначально закладывается в механизм организационной структуры, то управление конфликтами — одна из главных задач управления человеческими ресурсами проекта.

Одним из основных отличий управления персоналом в рамках управления проектом является использование особого образования — команды.

Команда проекта — одно из главных понятий управления проектами. Это группа сотрудников, непосредственно работающих над осуществлением проекта и подчиненных руководителю проекта; основной элемент структуры проекта, так как именно команда проекта обеспечивает реализацию замысла проекта.

Команду характеризует размытость, нечеткость структуры и формального разделения полномочий. Команда — не простое структурное подразделение, деятельность которого, как правило, формируется по функциональному признаку. Деятельность команды является целевой, она нацелена на конкретный результат (result-driven), а не ориентирована на выполнение некоторой деятельности (activity-oriented), будь то функция или процесс. Это существенным образом повышает эффективность использования человеческих ресурсов, но ввиду нетрадиционное вызывает определенные трудности в формировании команды и управлении ею.

Основные проблемы, возникающие при управлении командой проекта:

• как быстро и эффективно сформировать команду из некогда незнакомых людей с разными характерами, с разными уровнями знаний и опыта, а главное — с разными областями профессиональной деятельности;

• каким образом быстро наладить между ними эффективное взаимодействие и ориентировать их деятельность на достижение общего результата;

• как обеспечить эффективность управления командой и ее взаимодействие с другими участниками проекта;

• как управлять командой в условиях, когда достижение цели проекта означает роспуск команды (какие средства мотивации применять, как управлять судьбой и карьерой членов команды).

3. Вовлеченность

Это минимум два человека, которые занимаются проектом фултайм. Варианты с постоянной занятостью в другой компании и работой над проектом по вечерам не рассматриваются. Бывает, что в Акселератор проходят стартапы с одним человеком на фултайм, но это скорее исключение. Двум членам команды проще распределить задачи, поэтому вовлеченность определяет скорость

4. Мотивация

Для нас идеальный кейс — когда двум-трем участникам, костяку команды принадлежит если не 100%, то хотя бы 70-80%. Это гарантия того, что команда будет заниматься проектом долгосрочно. Как правило, руководитель владеет более

50%. Сомнителен кейс, когда 100% принадлежит руководителю, а команда при этом никаким образом не замотивирована. В этом случае мы спрашиваем, как основатель планирует в ближайшей перспективе мотивировать сотрудников: выделять опционы, перераспределять часть своей доли сотрудникам, делать их соучредителями. Бывают кейсы, когда внутри команды есть договоренность о распределении долей, которая не закреплена юридически. Это понятно членам команды, но для инвестора надежнее, когда у участников есть реальные доли. Опцион — довольно индивидуальная вещь. На самой ранней стадии, когда есть команда, базовый продукт и только начинаются продажи, идеально, если у всех ключевых членов команды есть доли. Если же уже есть объем продаж, проект какое-то время функционирует — могут быть KPI и завязанные на них функционалы. Обычно это все же KPI, который определяет основатель проекта.

•

Практическая работа №13-14.

Мотивация участников проекта. Распределение ролей в команде.

В данной работе происходит дальнейшее развитие способов управления персоналом для достижения результатов проекта и окончательно определяются роли в проектной команде.

Форма отчетности по практической работе:

Отчет по распределению ролей в команде и способов мотивации участников проекта.

1. Опыт лидера проекта

У студенческого руководителя проекта должны быть, как минимум, лидерские качества, желателен опыт работы в индустрии, в которой он разрабатывается проект — экспертиза, а также опыт ведения собственных проектов.

2. Компетенции команды и баланс в распределении ролей

Идеальное распределение ролей — когда лидер проекта параллельно обладает какой-то второй компетенцией (маркетолог, продажи, разработка, веб-продвижение). Второй человек дополняет его компетенцию. В идеале — должны быть закрыты техническая часть и маркетинговая. В совокупности у них должна быть синергия, это позволит им быстрее развивать свой проект и масштабироваться. От технической части в ИТ-проекте команде должен быть тим-лид, разработка же может оставаться на аутсорсе.

Практическая работа №15.

Управление коммуникациями в проекте. Распределение проектной информации, представление отчетности. Разработка плана управления коммуникациями проекта

В данной работе необходимо разработать план управления коммуникациями в проекте.

Форма отчетности по практической работе:

план управления коммуникациями в проекте, представляемый студенческой командой.

Управление информацией и коммуникациями проекта (Project Communications Management)

Обеспечение участников и процессов проекта информацией включает каналы связи, накопление данных, обмен и актуализацию данных, ведение баз данных, распределение информации по потребителям. Управление информацией обеспечивает предоставление, оценку, переработку, мониторинг, анализ информации, информационных потоков в течение жизненного цикла проекта.

Коммуникации и сопутствующая им информация — своего рода фундамент для координации действий участников проекта.

Под информацией понимают собранные, обработанные и распределенные данные. Чтобы быть полезной для принятия решений, информация должна предоставляться своевременно, по назначению и в удобной форме. Это решается использованием современных информационных технологий в рамках системы управления проектом.

Управление коммуникациями проекта (управление взаимодействием, информационными связями) — управленческая функция, направленная на обеспечение своевременного сбора, генерации, распределения и хранения необходимой проектной информации.

Управление коммуникациями проекта включает процессы:

Планирование коммуникаций (Communications Planning) — определяет информационные и коммуникационные нужды команды проекта (кому, когда и какая необходима информация).

Распределение информации (Information Distribution) — своевременное предоставление необходимой информации участникам проекта.

Отчетность об исполнении (Performance Reporting) — сбор и распространение информации о ходе выполнения проекта.

Административное завершение (Administrative Closure) — подготовка, сбор и распространение информации и материалов для официального завершения фазы или проекта.

Основными потребителями информации проекта выступают:

Менеджер проекта — для анализа расхождений фактических показателей выполнения работ от запланированных и принятия решений по проекту.

Заказчик — для осведомленности о ходе выполнения проектных работ.

Поставщики — при возникновении потребности в материалах, оборудовании и т.п., необходимых для выполнения работ.

Проектировщики — при необходимости внесения изменений в проектную документацию.

Непосредственные исполнители работ.

Содержание управления коммуникациями проекта. Управление коммуникациями проекта обеспечивает поддержку системы связи (взаимодействий) между участниками проекта, передачу управленческой и отчетной информации, направленной на обеспечение достижения целей проекта. Каждый участник проекта должен быть подготовлен к взаимодействию в рамках проекта в соответствии с его функциональными обязанностями.

Управление коммуникациями включает следующие процессы.

Планирование системы коммуникаций — определение информационных потребностей участников проекта (определение состава информации, сроков и способов ее доставки).

Сбор и распространение информации — процессы регулярного сбора и своевременной доставки необходимой информации участникам проекта.

Подготовка отчетов о ходе выполнения проекта — обработка фактических результатов состояния работ проекта, сравнение их с плановыми показателями, анализ тенденций, прогнозирование.

Документирование хода работ — сбор, обработка и организация хранения документации по проекту.

Рассмотрим перечисленные процессы подробнее.

Планирование системы коммуникаций. План коммуникаций — составная часть плана проекта. Он включает:

план сбора информации, в котором определяются источники информации и методы ее получения;

план распределения информации, в котором определяются потребители информации и способы ее доставки;

детальное описание каждого документа, который должен быть получен или передан, включая формат, содержание, уровень детальности и используемые определения;

план ввода в действие тех или иных видов коммуникаций;

методы обновления и совершенствования плана коммуникаций.

План коммуникаций формализуется и детализируется в зависимости от потребностей проекта.

Сбор и распространение информации. В рамках проекта существует потребность в осуществлении различных видов коммуникаций:

- • внутренних (внутри команды проекта) и внешних (с руководством компании, заказчиком, внешними организациями и т. д.);
- • формальных (отчеты, запросы, совещания) и неформальных (напоминания, обсуждения);
- • письменных и устных;
- • вертикальных и горизонтальных.

Отчетность о ходе выполнения проекта. Процессы сбора и обработки данных о фактических результатах и отображение информации о состоянии работ в отчетах обеспечивают основу для координации работ, оперативного планирования и управления. Отчетность о ходе выполнения работ включает:

- • информацию о текущем состоянии проекта в целом и в разрезе отдельных показателей;
- • информацию об отклонениях от базовых планов;
- • прогнозирование будущего состояния проекта.

Системы сбора и распределения информации должны обеспечивать потребности различных видов коммуникаций. Для этих целей могут использоваться автоматизированные и неавтоматизированные методы сбора, обработки и передачи информации.

Неавтоматизированные методы включают сбор и передачу данных на бумажных носителях, проведение совещаний.

Автоматизированные методы предусматривают использование компьютерных технологий и современных средств связи для повышения эффективности взаимодействия: электронную почту, системы документооборота и архивирования данных.

Документирование хода работ. Основные промежуточные результаты хода работ должны быть формально задокументированы. Документирование результатов хода работ включает:

- • сбор и верификацию окончательных данных;
- • анализ и выводы о степени достижения результатов проекта и эффективности выполненных работ;
- • архивирование результатов с целью дальнейшего использования.

Компьютерные системы ведения электронных архивов позволяют автоматизировать процессы хранения и индексации текстовых и графических документов, значительно облегчить доступ к архивной информации.

Практическая работа №16.

Информационное обеспечение управления проектами: состав, структура, характеристики. Программные средства для управления проектами. Характеристика состояния рынка программных продуктов по управлению проектами. Планирование проекта с использованием MS Project.

В данной работе используется программное обеспечение для управления проектами MS Project.

Форма отчетности:

План выполнения проекта, созданный в MS Project. Диаграмма Ганта.

Процесс планирования в MS Project Online – это процесс, в раках которого проводится оценка ключевых параметров проекта, формируется перечень задач и оценивается объемы финансирования и необходимых ресурсов. Этот процесс включает в себя предварительную оценку всех элементов проекта. В результате реализации данного этапа формируется план-график проекта. Если данный план-график устраивает основные заинтересованные стороны проекта, сохраняется базовый план проекта и начинается его реализация.

Удобная форма представления плана работ – диаграмма Ганта.

Диагра́мма Га́нта — это тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами. Первый формат диаграммы был разработан Генри Л. Ганттом в 1910 году.

Рис.11 Формирование проектных заданий

Рис.12 Формирование задач и указание предшествующих задач

Рис.13 Определение ресурсов при выполнении проекта

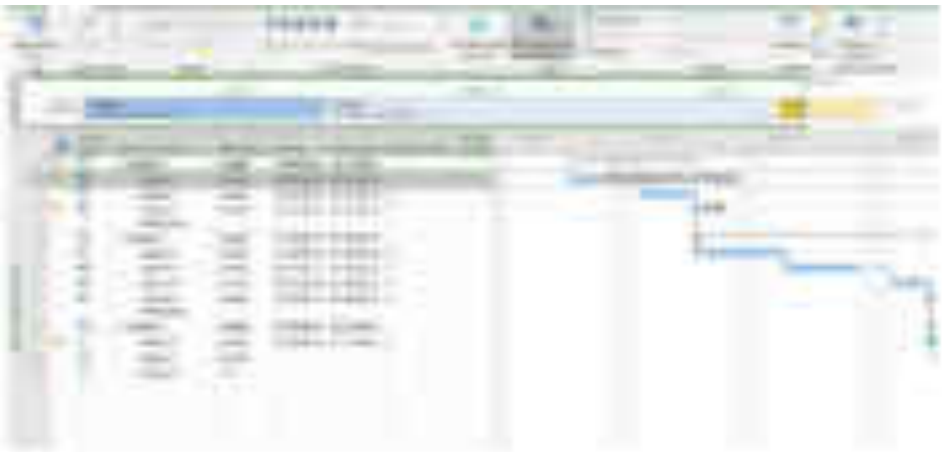


Рис.14 Пример формирования диаграммы Ганта

Список используемой литературы:

Баранчеев, В. П. Управление инновациями в 2 т : учебник для академического бакалавриата [Текст] / В. П. Баранчеев, Н. П. Масленникова, В. М. Мишин. – 3-е изд., перераб. и доп. – М. : Юрайт, 2015. – 782 с.

2. Зуб, А. Т. Управление проектами : учебник и практикум для академического бакалавриата [Текст] / А. Т. Зуб. : МГУ им. М.В. Ломоносова. – М. : Юрайт, 2017. – 422 с.

3. Первушин, В.А. Практика управления инновационными проектами : учебное пособие [Текст] / В. А. Первушин ; РАНХиГС – М. : Дело, 2015. – 208 с.

4. Поляков, Н. А. Управление инновационными проектами : учебник и практикум для академического бакалавриата [Текст] / Н. А. Поляков, О. В. Мотовилов, Н. В. Лукашов. — М. : Юрайт, 2017. – 330 с.

5. Первушин, В.А.. Практика управления инновационными проектами : [учеб. пособие] / В. А. Первушин; – М.: ИД «Дело» РАНХиГС, 2013. – 208 с.

6. Попов, В.Л. Управление инновационными проектами : учебное пособие [Текст] / В. Л. Попов и др. ; под ред. В. Л. Попова. – М.: Инфра-М, 2015. – 336.

7. Туккель, И.Л., Сурина, А.В., Культин, Н.Б. Управление инновационными проектами: учеб. для студентов вузов [Текст] / И.Л. Туккель, А.В. Сурина, Н.Б. Культин; под общ. ред. И. Л. Туккеля – СПб. : БХВ-Петербург, 2011. – 416 с.

Дополнительная литература:

1. Алексеева, М. Б. Анализ инновационной деятельности : учебник и практикум для бакалавриата и магистратуры [Текст] / М. Б. Алексеева, П. П. Ветренко. — М. : Издательство Юрайт, 2017. – 303 с.

2. Гончаренко, Л. П. Инновационный менеджмент : учебник для академического бакалавриата [Текст] / Л. П. Гончаренко, Б. Т. Кузнецов, Т. С. Булышева, В. М. Захарова ; под общ. ред. Л. П. Гончаренко. — 2-е изд., перераб. и доп. – М. : Юрайт, 2016. – 487 с.

3. Друкер, П.Ф. Менеджмент. Вызовы XXI века [Текст] / П.Ф. Друкер ; пер. с англ. Н. Макарова. – М.: Манн, Иванов и Фербер, 2012. – 256 с.

4. Кремер, Н. Ш. Математическая статистика : учебник и практикум для академического бакалавриата [Текст] / Н. Ш. Кремер. – М. : Юрайт, 2017. – 259 с.

5. Тарасенко, Ф.П. Прикладной системный анализ. Учебное пособие [Текст] / Ф.П. Тарасенко. – М.: КноРус, 2010. – 224 с.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б2.О.01(П) НИР

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Методическое пособие доступно в электронно-библиотечной системе Лань.
Ссылка: https://e.lanbook.com/book/159496

Данные для подключения:
Лань:
https://e.lanbook.com
логин: ulgtu2019@yandex.ru
пароль: 778452asd

ЭБС IPR BOOKS:
https://www.iprbookshop.ru/
логин:ulgtu2019@yandex.ru
пароль:8nhJHXDcTg64

Юрайт
https://biblio-online.ru
логин:ulgtu-ulgtu2019@yandex.ru
пароль:e8f9d8

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б2.О.02(У) Ознакомительная практика

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ

Декан факультета информационных систем и технологий

_____ Святов К.В.

«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Дисциплина (модуль) Ознакомительная практика

наименование дисциплины (модуля)

Уровень образования магистратура

(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр

Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Методические указания по дисциплине «Ознакомительная практика» для магистрантов, обучающихся по направлению 09.04.01 «Информатика и вычислительная техника», профиль «Искусственный интеллект в автоматизации проектирования»

По результатам прохождения практики у обучающихся формируется отчёт. Отчёт по практике содержит:

1. титульный лист;
2. задание на практику, включающее рабочий график (план) проведения практики,
3. индивидуальное задание, планируемые результаты практики;
4. отзыв руководителя практики от профильной организации о работе обучающегося в период прохождения практики;
5. дневник практики;
6. аннотированный отчёт;
7. приложения (при необходимости)}.

Аннотированный отчёт о прохождении практики должен включать краткое описание проделанной работы.

В качестве приложений могут быть необходимые для дальнейшего использования в учебном процессе нормативные документы, таблицы обработки измерений, схемы устройств, графики, копии необходимых документов и т.д.

1. Шкала оценивания отчёта с учетом срока сдачи

Критерии оценки качества отчета	Балл
Оценка «отлично» ставится, если выполнены все требования к написанию отчета: содержание разделов соответствует их названию, собрана полноценная, необходимая информация, выдержан объём; умелое использование профессиональной терминологии, соблюдены требования к внешнему оформлению	отлично
Оценка «хорошо» - основные требования к отчету выполнены, но при этом допущены недочёты. В частности, имеется неполнота материала; не выдержан объём отчета; имеются упущения в оформлении.	хорошо
Оценка «удовлетворительно» - имеются существенные отступления от требований к отчету. В частности: разделы отчета освещены лишь частично; допущены ошибки в содержании отчета; отсутствуют выводы.	удовлетворительно
Оценка «неудовлетворительно» - задачи практики не раскрыты в отчете, использованная информация и иные данные отрывисты, много заимствованного, отраженная информация не внушает доверия или отчет не представлен вовсе.	неудовлетворительно

В конце Ознакомительной практики проводится зачёт с оценкой. Процедура проведения зачёта представлена в таблице.

Общее количество вопросов к зачету	8 вопросов
Количество основных задаваемых вопросов	3 вопроса
Формат проведения	Устно

Шкала оценивания с учетом текущего контроля работы обучающегося в семестре

Критерии оценки уровня сформированности компетенций по практике	Балл
Выставляется обучающемуся, если студент показал глубокие знания теоретического материала по поставленному вопросу, грамотно логично и стройно его излагает, а также выполнил в полном объеме практические задания и способен обосновать свои решения	Отлично
выставляется обучающемуся, если студент твердо знает теоретический материал, грамотно его излагает, не допускает существенных неточностей в ответе на вопрос, выполнил практические задания не в полном объеме (не менее ¾) либо в полном объеме, но с несущественными погрешностями и ошибками	Хорошо
выставляется обучающемуся, если студент показывает знания только основных положений по поставленному вопросу, требует в отдельных случаях наводящих вопросов для принятия правильного решения, допускает отдельные неточности; выполнил практические задания не в полном объеме (не менее ½) либо в полном объеме, но с существенными погрешностями и ошибками	Удовлетворительно
выставляется обучающемуся, если студент допускает грубые ошибки в ответе на поставленный вопрос, не справился с выполнением практических заданий	Неудовлетворительно

Полный перечень вопросов для проведения собеседования

1) Каким образом производилась оценка риска в проекте, выполняемом в рамках практики?
2) Как осуществляется управление проектной деятельностью в рамках выполнения работы по практике?
3) Что является объектом управления в разрабатываемой в рамках практики системе?
4) Какие обратные связи присутствуют в разрабатываемой в рамках практики системе?
5) Какие модели были построены в рамках выполненной проектно-исследовательской работы?
6) Каким образом была осуществлена калибровка моделей?
7) Какие программные средства были использованы для моделирования?
8) Каковы результаты проведённого моделирования?

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б2.О.03(П) Технологическая практика

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Декан факультета информационных систем и технологий
_____ Святов К.В.
«____»_____20____ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Дисциплина (модуль) Технологическая практика
наименование дисциплины (модуля)

Уровень образования магистратура
(СПО/бакалавриат/магистратура/специалитет/подготовка кадров высшей квалификации)

Квалификация Магистр
Техник/Бакалавр/Магистр/Инженер/ Исследователь. Преподаватель-исследователь

г. Ульяновск, 2021

Методические указания по дисциплине «Технологическая (проектно-технологическая) практика» для магистрантов, обучающихся по направлению 09.04.01 «Информатика и вычислительная техника», профиль «Искусственный интеллект в автоматизации проектирования»

ЦЕЛИ И ЗАДАЧИ ПРАКТИКИ

Целями практики является углубление и закрепление у студентов знаний, умений и навыков, приобретаемых в ходе освоения дисциплин профессиональной подготовки путем фокусирования на основных направлениях научных исследований в сфере информатики и вычислительной техники, соответствующих образовательной программе 09.04.01 «Информатика и вычислительная техника».

Задачами технологической практики являются:

- Систематизация, закрепление и расширение теоретических знаний и практических навыков проведения исследований;
- Углубление полученных теоретических знаний в области информатики и вычислительной техники и их применение в решении конкретных проектно-исследовательских задач;
- Развитие и стимулирование навыков самостоятельной научно-исследовательской работы;
- Выявление и формулирование актуальных научных проблем в области информатики и вычислительной техники;
- Поиск, обработка, анализ и систематизация информации по теме исследования;
- Разработка программ научных исследований и организация их выполнения;
- Овладение навыками выступлений с докладами и проведения содержательных научных дискуссий, оценок и экспертиз.

ОТЧЁТ ПО ПРАКТИКЕ

Отчет по практике, формируемый обучающимся по итогам прохождения практики, содержит:

1. титульный лист;
2. задание на практику, включающее рабочий график (план) проведения практики,
3. индивидуальное задание, планируемые результаты практики;
4. отзыв руководителя практики от профильной организации о работе обучающегося в период прохождения практики;
5. дневник практики;
6. аннотированный отчет;
7. приложения (при необходимости)}.

Аннотированный отчет о прохождении практики должен включать краткое описание проделанной работы.

В качестве приложений могут быть необходимые для дальнейшего использования в учебном процессе нормативные документы, таблицы обработки измерений, схемы устройств, графики, копии необходимых документов и т.д.

Критерии оценки качества отчета	Балл
Оценка «отлично» ставится, если выполнены все требования к написанию отчета: содержание разделов	отлично

соответствует их названию, собрана полноценная, необходимая информация, выдержан объём; умелое использование профессиональной терминологии, соблюдены требования к внешнему оформлению	
Оценка «хорошо» - основные требования к отчету выполнены, но при этом допущены недочёты. В частности, имеется неполнота материала; не выдержан объём отчета; имеются упущения в оформлении.	хорошо
Оценка «удовлетворительно» - имеются существенные отступления от требований к отчету. В частности: разделы отчета освещены лишь частично; допущены ошибки в содержании отчета; отсутствуют выводы.	удовлетворительно
Оценка «неудовлетворительно» - задачи практики не раскрыты в отчете, использованная информация и иные данные отрывисты, много заимствованного, отражённая информация не внушает доверия или отчет не представлен вовсе.	неудовлетворительно

ЗАЧЁТ С ОЦЕНКОЙ

Процедура проведения зачёта с оценкой представлена в таблице

Общее количество вопросов к зачету	15 вопросов
Количество основных задаваемых вопросов	2 вопроса
Формат проведения	Устно

Шкала оценивания с учетом текущего контроля работы обучающегося в семестре

Критерии оценки уровня сформированности компетенций по практике	Балл
Выставляется обучающемуся, если студент показал глубокие знания теоретического материала по поставленному вопросу, грамотно логично и стройно его излагает, а также выполнил в полном объеме практические задания и способен обосновать свои решения	Отлично
выставляется обучающемуся, если студент твердо знает теоретический материал, грамотно его излагает, не допускает существенных неточностей в ответе на вопрос, выполнил практические задания не в полном объеме (не менее ¾) либо в полном объеме, но с несущественными погрешностями и ошибками	Хорошо
выставляется обучающемуся, если студент показывает знания только основных положений по поставленному вопросу, требует в отдельных случаях наводящих вопросов для принятия правильного решения, допускает отдельные неточности; выполнил практические задания не в полном объеме (не менее ½) либо в полном объеме, но с существенными погрешностями и ошибками	Удовлетворительно
выставляется обучающемуся, если студент допускает грубые ошибки в ответе на поставленный вопрос, не	Неудовлетворительно

справился с выполнением практических заданий	

1. Вопросы к зачету с оценкой

1) Какие научно-практические материалы были подготовлены?

2) На каких семинарах и конференциях были обсуждены полученные результаты?

3) Каковы выводы по проделанной работе можно заключить?

4) Каковы результаты аналитического обзора по научно-технической проблеме?

5) Результаты литературного обзора по рассматриваемой проблеме.

6) Какие практические задачи были решены в процессе практики?

7) Какие навыки приобрели в процессе реализации проекта по теме научного исследования?

8) Какими нормативными документами пользовались для оформления научных результатов?

9) Какие сетевые протоколы были использованы для взаимодействия между компонентами спроектированной системы?

10) Каким образом осуществляется передача данных между компонентами спроектированной системы?

11) Каким образом организовано управление потоками проектно-исследовательской работы в рамках практики?

12) Каким образом производилась оценка риска в проекте, выполняемом в рамках практики?

13) Как осуществляется управление проектной деятельностью в рамках выполнения работы по практике?

14) Каким образом проектируемая в рамках практики система интегрируется в план информатизации предприятия?

15) Какие стадии жизненного цикла продукции предприятия охватывает проектируемая в рамках практики система?

ПРОЦЕДУРА ТЕСТИРОВАНИЯ

Параметры процедуры проведения тестирования представлены в таблице

Количество проводимых тестов в течение всего периода освоения практики	1 тестов
Общее количество тестовых вопросов в банке тестов	36 вопросов
Количество задаваемых тестовых вопросов в одном тесте	10 вопросов
Формат проведения тестирования	Электронный
Сроки / Периодичность проведения тестирования	Последняя неделя практики

Шкала оценивания с учетом срока сдачи представлена в таблице

Количество правильных ответов / Процент правильных ответов	Балл
75-100%	Отлично
60-75 %	Хорошо
40-60 %	Удовлетворительно
менее 40 %	неудовлетворительно

Вопросы для подготовки к тестированию

I. Какие новые управленческие действия появились в рамках функции управления знаниями?

1. **Приобретение знаний, усвоение знаний, передача знаний**.
2. Отбор и обучение персонала, кодификация информации.
3. Передача информации, исследования, развитие инноваций.
4. Повышение квалификации сотрудников, развитие информационной системы организации.

II. Какие новые должности вводятся в компаниях для решения задач по управлению знаниями?

1. **Директор по управлению знаниями, менеджер по интеллектуальным активам, вице-президент по управлению интеллектуальными ресурсами**.
2. Технолог производства, директор по обучению.
3. Директор отдела развития, экспедиторы решений, менеджеры по продажам.
4. Аналитик управления знаниями, директор информационной службы.

III. Какие задачи решает управление знаниями в организации?

1. **Управление знаниями систематизирует процессы информационного обмена в организациях.**
2. Управление знаниями направлено на добавление реальных ценностей к информации с помощью ее фильтрации, синтеза, обобщения и предоставления ее в таком виде, который позволяет сотрудникам приобрести необходимые знания.
3. Управление знаниями направлено на развитие инноваций в организациях.

IV. Основная задача экспертов**:**

1. Выявление недостатков концепции, заложенной в технологию принятия решения;
2. Подготовка альтернативных решений;
3. Выявление недостатков и достоинств представленных вариантов принятия решений;
4. **Оценка последствий выбора того или иного варианта принятия решений.**

V. Специалисты в области принятия решений должны обладать**:**

1. Знаниями о существующих методах поддержки принятия решений;
2. Умениями и навыками работы со средствами поддержки принятия решений;
3. Способностями в области математического моделирования планируемых процессов;
4. **Умениями применять на практике накопленный опыт принятия решений**.

VI. Какими значениями обладает слово «решение»**:**

1. Множество рассматриваемых возможностей, выделенных человеком, делающим выбор;
2. **Процесс поиска наиболее предпочтительного варианта (обдумывание, изучение вопроса или задачи, нахождение правильного ответа);**
3. Полученный ответ в ходе поиска, один или несколько выбранных вариантов, результат анализа проблемы или задачи, нахождение правильного ответа;
4. Указы, постановления, распоряжения, приказы, акты органов законодательной и исполнительной власти, судебные и иные решения.

VII. Выберите правильное определение термина «Принятие решения»**:**

1. Спектр человеческой деятельности, состоящий в оптимальном выборе наилучшего варианта из имеющихся с учетом критериев оптимизации;

2. Процесс поиска наиболее предпочтительного варианта без учета критериев оценки;
3. Поиск вариантов, направленных на решение поставленной проблемы или задачи;
4. **Особый вид человеческой деятельности, состоящий в обоснованном выборе наилучшего в некотором смысле варианта из имеющихся возможных.**

VIII. При принятии решения следует**:**

1. Рассмотреть различные варианты;
2. Оценить возможные варианты;
3. Сопоставить однотипные варианты;
4. Учесть разные точки зрения экспертов, консультантов, аналитиков.

IX. При принятии политических**,** экономических**,** производственных и др**.** решений следует**:**

1. Учитывать интересы заинтересованных сторон;
2. Абстрагироваться от возможных вариантов;
3. Прислушиваться к собственной интуиции и своим предпочтениям;
4. Отыскивать и анализировать разнообразную информацию.

X. Для сравнения различных вариантов необходимо**:**

1. Провести всесторонний анализ проблемной ситуации;
2. Выбрать из предложенных вариантов наиболее привлекательный вариант;
3. Использовать средства вычислительной техники и необходимое программное обеспечение (в том числе, Системы поддержки принятия решений);
4. Разработать специальные (в том числе и математические) модели.

XI. Лицо**,** принимающее решение должно**:**

1. Оперативно принимать решения в любых ситуациях;
2. Выбирать из предложенных вариантов тот, который соответствует его точке зрения;
3. Абстрагироваться от возможной ответственности;
4. Всегда основываться на применении математических моделей.

XII. Современные СППР **(Decision Support System, DSS),** возникашие как естественное развитие автоматизированных систем управления и систем управления базами данных**,** представляют собой**:**

1. системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении неструктурированных задач;
2. системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении слабоструктурированных задач многокритериальных;
3. системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении чисто информационных задач;
4. системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении неструктурированных и слабоструктурированных задач.

XIII. Выберите свойства**,** общепризнанные специалистами для СППР**:**

1. использование и данных, и моделей, а также решение слабоструктурированных и неструктурированных задач;

2. решение задач, связанных с использованием вероятностных методов и теории массового обслуживания;
3. поддерживают, а не заменяют, выработку решений ЛПР;
4. СППР целенаправленны на повышение эффективности (оперативность и обоснованность и др.) решений, обеспечивающих потенциальные возможности объекта управления.

XIV. Выделите среди предложенных правильную архитектурно **-** технологическую схему информационно**-**аналитической поддержки принятия решений**:**

1. Метаданные -> Хранилище данных -> анализ данных -> интеллектуальный анализ;
2. Оперативные данные -> Хранилище данных -> анализ данных -> интеллектуальный анализ;
3. Модели данных -> СУМД -> анализ данных -> интеллектуальный анализ;
4. Данные -> СУБД -> Извлечение данных -> анализ данных.

XV. Современные СППР **(Decision Support System, DSS)** могут содержать такие блоки**,** как**:**

1. База данных и/или База знаний:
2. СУБД и/или систему управления базой знаний;
3. системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении чисто информационных задач;
4. **системы, приспособленные к решению задач управленческой деятельности, являются инструментом, призванным оказать помощь ЛПР в решении неструктурированных и слабоструктурированных задач.**

XVI. Выделите правильную последовательность процедур технологии генерации решения с помощью СППР **(**интеллектуальной**):**

1. Анализ полученного варианта решения (варианты) и в случае надобности изменение условий их получения.
2. Выполнение постановки задачи и выбор модели базы знаний;
3. Наполнение системы знаниями и данными;
4. Формирование проблемы, цели или гипотезы, а также выбор критерия оценки принятого решения;

XVII. Раздел информатики, изучающий возможность обеспечения разумных рассуждений и действий с помощью вычислительных систем и иных искусственных устройств, называется

- **искусственным интеллектом** (+)
- кибернетикой
- теорией нечетких множеств
- алгеброй логики

XVIII. Глобальное информационное пространство, основанное на физической инфраструктуре Интернета и протоколе передачи данных http, называется

- ISO/OSI
- **World Wide Web** (+)
- Local Area Network
- TCP/IP

XIX. Создание программ, которые создают другие программы как результат своей работы

- проектный менеджмент
- метапроектирование

- **метапрограммирование** (+)
- макропрограммирование

XX. Основанные на суждениях специалистов количественные оценки процессов или явлений, не поддающихся непосредственному измерению, представляют собой оценки
- нечеткие
- экспертные (+)
- абстрактные
- оптимальные

XXI. Назовите характерный признак системы, основанной на знаниях:
- выделение метазнания, описывающего структуру знаний и отражающего модель предметной области
- **выделение операционного знания в базу знаний**
- разделение фактуального и операционного знаний
- неотделимость операционного и фактуального знаний

XXII. В качестве единиц знаний используются:
- правила
- факты
- **правила и факты**
- нет правильного ответа

XXIII. Экспертная система:
- это интеллектуальная система, обрабатывающая знания
- это **интеллектуальная система, позволяющая решать сложные задачи на основе накапливаемого экспертного знания**
- это интеллектуальная система, осуществляющая поиск релевантной для принятия решений информации

XXIV. Экспертная система состоит из:
- интеллектуального интерфейса
- базы знаний
- механизма вывода заключений
- **интеллектуального интерфейса, базы знаний и механизма вывода заключений**

XXV. Процедура, выполняющая интерпретацию запроса пользователя к БЗ и формирующая ответ в удобной для него форме:
- это механизм объяснения
- это **интеллектуальный интерфейс**
- это механизм приобретения знаний
- это механизм вывода

XXVI. Концептуализация знаний:
- это получение инженером по знаниям наиболее полного из возможных представлений о предметной области и способах принятия решения в ней создание прототипа ЭС
- это **разработка неформального описания структуры знаний о предметной области в виде графа, таблицы, диаграммы или текста**
- это разработка БЗ на языке представления знаний

XXVII. Инженер по знаниям:
- это **специалист, занимающийся извлечением знаний и их формализацией в БЗ**
- это специалист, знания которого помещаются в БЗ
- это специалист, интеллектуальные способности которого расширяются благодаря использованию ЭС

XXVIII. Правило построения дерева целей:

- все вершины нижнего уровня подчиняются всем вершинам вышестоящего уровня иерархии
- вершины нижнего уровня подчиняются одной вершине вышестоящего уровня иерархии
- **вершина нижнего уровня подчиняются только одной вершине вышестоящего уровня иерархии**

XXIX. Фрейм – это:
- это модель, позволяющая представить знание в виде предложений типа «ЕСЛИ (условие), ТО (действие)»
- это ориентированный граф, вершины которого – понятия, а дуги – отношения между ними
- это **структура данных, предназначенная для представления некоторой стандартной ситуации**
- это совокупность классов и объектов предметной среды

XXX. Объектно-ориентированный подход – представление системы в виде:
- модели, позволяющей представить знание в виде предложений типа «ЕСЛИ (условие), ТО (действие)»
- ориентированного графа, вершины которого – понятия, а дуги – отношения между ними
- **совокупности классов объектов, отвечающих требованиям инкапсуляции, полиморфизма, наследования**
- структуры данных, предназначенной для представления некоторой стандартной ситуации

XXXI. Модель, реализующая и объекты, и правила с помощью предикатов первого порядка, являющаяся строго формализованной моделью с универсальным дедуктивным и монотонным методом логического вывода «от цели к данным»:
- это продукционная модель
- это фреймовая модель
- это семантическая сеть
- это **логическая модель**
- это объектно-ориентированная модель

XXXII. Модель, позволяющая представить знания в виде ориентированного графа, вершины которого – понятия, а дуги – отношения между ними, – это:
- это продукционная модель
- это **семантическая сеть**
- это фрейм
- это объектно-ориентированная модель

XXXIII. Структура данных, предназначенная для представления некоторой стандартной ситуации:
- это продукционная модель
- это семантическая сеть
- это **фрейм**
- это объектно-ориентированная модель

XXXIV. Для чего используется детектор Даугмана
- Идентификация людей по отпечатку пальцев
- Идентификация Y-связных углов
- Идентификация людей по картинке их радужной оболочки (+)
- Идентификация L-связных углов

XXXV. Каким из способов на практике можно бороться с проблемой Out of Vocabulary (отсутствие слова в словаре) ?

- генерация всех возможных слов словаря перед обучением
- добавление признака на входном слове и до-обучение модели (+)
- использование буквенных триграмм дополнительно к словарю (+)
- приведение неизвестных слов к наиболее близким

XXXVI. Какой подход называется обучение с учителем?

- Когда нерегулярно происходит «наказание» модели за неправильный результат
- Когда нерегулярно происходит «вознаграждение» модели за правильный результат
- Когда модели предоставляются входные данные и правильный ответ (+)
- Ничего из вышеперечисленного

XXXVII. Для чего используется регрессия в машинном обучении?

- Формализация знаний экспертов и их перенос в компьютер в виде базы знаний
- Поиск набора признаков их значений, которые особенно часто встречаются в признаковых описаниях объектов
- Обнаружение в обучающей выборке небольшого числа нетипичных объектов
- Предсказание каких-либо значений по набору признаков

XXXVIII. Поисковый индекс построен как инвертированный индекс над коллекцией текстов. Результаты поиска на запрос «тот который не этот» ...

- Вероятно будут избыточны, так как все слова высокочастотные и попадут в словарь «стоп-слов»
- Вероятно будут неполны, так как все слова высокочастотные и попадут в словарь «стоп-слов» (+)
- Вероятно будут избыточны, если не используется словарь «стоп-слов», так как все слова высокочастотные (+)
- Вероятно будут неполны, если не используется словарь «стоп-слов», так как все слова высокочастотные

XXXIX. Модель continuousbag of words из word2vec представляет собой нейронную сеть с 1 скрытым слоем. Где в модели можно найти скрытые векторные представления заданной (малой) размерности:

- Это значения на выходе нейронной сети
- Это значения на скрытом слое
- Это значения, подаваемые на входной слой

XXXX. Какие бывают метрики машинного обучения?

- Чувствительность (+)
- Точность (+)
- Площадь под прямой
- Аккуратность (+)
- Площадь под кривой (+)
- Правильность

XXXXI. Какие ограничения приходится накладывать на решение методом кластеризации?

- Использовать определённое число кластеров (+)
- Использовать ограниченную модель
- Использовать итеративный подход (+)
- Использовать метрики, которые позволяют отклонять заведомо «плохие» варианты (+)
- Использовать лишь некоторые метрики качества

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
Б3.01 ГИА

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное
учреждение высшего образования

«Ульяновский государственный технический университет»

Требования к разработке, оформлению и защите выпускных квалификационных работ

Методические указания

Составитель

И. В. Беляева

Ульяновск 2021

Оглавление

ВВЕДЕНИЕ

Магистерская диссертация является формой итогового контроля уровня теоретической и практической подготовки выпускника магистратуры и представляет собой самостоятельную и логически завершённую выпускную квалификационную работу, связанную с решением задач тех видов деятельности, к которым готовится магистрант в соответствии с требованиями ФГОС ВО, профессиональных стандартов и содержанием образовательной программы магистратуры по направлению подготовки 09.04.01 «Информатика и вычислительная техника».

В соответствии с учебным планом, магистранты ориентированы на научно-исследовательскую профессиональную деятельность, поэтому написание выпускной работы связано с решением конкретной научно-исследовательской задачи, а также проектированием и созданием автоматизированной информационной системы, реализующей и проверяющей решение этой задачи. При этом, как правило, созданная система имеет и самостоятельную практическую ценность.

Подготовка материалов диссертации осуществляется магистрантом прежде всего в ходе научно-исследовательской работы и преддипломной практики.

К защите выпускной квалификационной работы допускается лицо, успешно завершившее в полном объеме освоение основной образовательной программы по направлениям подготовки (специальности) высшего профессионального образования, разработанной высшим учебным заведением в соответствии с требованиями государственного образовательного стандарта высшего профессионального образования и успешно прошедшее все другие виды итоговых аттестационных испытаний.

При выполнении выпускной квалификационной работы обучающиеся должны показать свою способность и умение, опираясь на полученные

углубленные знания, умения и сформированные общекультурные и профессиональные компетенции, самостоятельно решать на современном уровне задачи своей профессиональной деятельности, профессионально излагать специальную информацию, научно аргументировать и защищать свою точку зрения.

Оформление выпускных квалификационных работ должно соответствовать действующим государственным стандартам.

1. ОБЩИЕ ПОЛОЖЕНИЯ ПО ВЫПОЛНЕНИЮ ВКР

В образовательном процессе, с учетом меняющихся стандартов, в которых возрастает роль самостоятельной работы студентов, под научно-исследовательской работой понимается самостоятельное выполнение и ведение проектов.

Исследование – это процесс изучения явления или предмета с целью выявления закономерностей его возникновения, развития, изменения. Этот процесс включает обобщение накопленного опыта, знаний и применение соответствующих инструментов и методов познания. Итог исследования - получение новых знаний и практических результатов.

Магистерская диссертация является завершенной научно-исследовательской работой или законченной и нашедшей практическое применение инженерной разработкой. Выполняется под руководством профессора или доцента соответствующей специальности содержит новое решение актуальной научной задачи, имеющей научное или существенное инженерное значение для информатики и вычислительной техники.

Основные научные или инженерные результаты, полученные автором магистерской диссертации в процессе ее выполнения, должны быть опубликованы в печатных изданиях в виде статей, тезисов, докладов конференций, симпозиумов и семинаров различного уровня (региональных, всероссийских, международных), в виде зарегистрированных программ или баз данных в Российском агентстве по правовой охране программ для ЭВМ и баз данных, а также в виде патентов (или поданных заявок на изобретение).

Магистерская диссертация является научным или научно-инженерным трудом, написанным на актуальную тему, утвержденную кафедрой.

Магистерская диссертация представляет собой квалификационную научную работу, имеющую внутреннее единство, содержащую совокупность научных и инженерных результатов, научных положений,

выдвигаемых автором для публичной защиты и свидетельствующих о личном вкладе автора диссертации в науку и практику в области информатики и вычислительной техники и его личных качествах как молодого ученого.

Основными целями выполнения и защиты ВКР являются:

1. Углубление, систематизация и интеграция теоретических знаний и практических навыков, полученных в период обучения, развитие умения оценивать и обобщать теоретические положения;
2. Овладение современными методами научного исследования, приобретение навыков самостоятельной аналитической работы;
3. Приобретение опыта работы в реальном проекте, применение полученных знаний при решении прикладных задач по направлению подготовки;
4. Приобретение навыков достижения результатов и их оценка при решении поставленных задач;
5. Получение навыков написания и публикации научных статей и выступлений на научных конференциях (желательно);
6. Оценка подготовленности к работе на предприятиях и компаниях соответствующего профиля и соответствия его деловых и профессиональных качеств компетенциям, определенным корпоративным стандартом (ответственности, стремлению к профессиональному и карьерному росту, работе в команде, творчеству и пр.);
7. Приобретение навыков публичной дискуссии и защиты научных идей, предложений и рекомендаций.

Полученные в ВКР результаты могут обладать элементами новизны (научной, прикладной, технологической) и должны обладать практической значимостью. Под новизной результата ВКР следует понимать отличительные от известных, полученных другими авторами, характеристики методов, алгоритмов и средств сбора, создания, преобразования, накопления, защиты, передачи и использования

информации, внедрения, эксплуатации и управления информационными системами, повышения эффективности и качества ИТ-проектов. Обоснование новизны может проводиться на моделях (алгоритмах, методиках) исследуемых процессов и систем. Результаты должны быть получены автором самостоятельно при решении актуальных теоретических или прикладных задач предприятий и/или учебно-научных подразделений УлГТУ, в которых выполняется работа.

ТЕМАТИКА РАБОТ

Тема магистерской диссертации должна соответствовать направлению подготовки магистров «Информатика и вычислительная техника», быть актуальной и максимально приближенной к решению реальных задач, а также содержать наиболее существенные признаки объекта исследования. Тема должна отвечать современным техническим требованиям, учитывать перспективы развития техники и технологии, и должна быть связана с планом основных научных работ выпускающей кафедры.

Темы магистерских диссертационных работ утверждаются на заседании кафедры, когда установлена их актуальность, научное и прикладное значение, наличие необходимых условий для ее выполнения в установленный срок и наличия должного научного руководства.

В магистерскую диссертацию включаются научные и инженерные положения автора, их теоретическое обоснование и (или) экспериментальные подтверждения, обоснование выбранной методике исследования и методики принятия инженерных решений, полученные результаты. Постановка задачи должна быть конкретной, вытекать из современного состояния вопроса и обосновываться анализом соответствующих научных и прикладных работ. Предложенные автором диссертации пути решения проблемы в целом и конкретных задач должны быть строго аргументированы и критически оценены по сравнению с известными решениями по всем аспектам, в том числе и по эффективности.

В диссертации (или в приложениях к ней) должны приводиться сведения, подтверждающие внедрение или практическое использование полученных автором магистерской диссертации научных и практических результатов.

ЭТАПЫ И ВЫПОЛНЯЕМЫЕ РАБОТЫ ПО ВКР

Основные этапы подготовки выпускной квалификационной работы приведены в таблице 1. По времени они привязаны к графику учебного процесса соответствующего уровня подготовки.

Таблица 1

Этап	**Выполняемые работы**
Практика	Сбор фактического материала по теме ВКР, обзор состояния вопроса и анализ предметной области
Подготовительный этап	Выбор и согласование с руководителем темы ВКР
	Подача заявления на кафедру о закреплении за студентом темы и руководителя ВКР
	Выход приказов о закреплении за студентами тем и руководителей ВКР
	Получение задания на ВКР от руководителя темы, составление плана работы
Выполнение ВКР	Систематизация материала по ВКР, подготовка обзора состояния вопроса, формулировка постановки задачи
	Проектирование и разработка содержательной части ВКР
	Оформление пояснительной записки, подготовка демонстрационных и раздаточных материалов
Подготовка к защите ВКР	Рассмотрение ВКР на кафедре и назначение рецензента
	Согласование ВКР с консультантами
	Получение отзыва научного руководителя
	Получение рецензии

Защита	Защита ВКР перед Государственной аттестационной комиссией
Передача материалов ВКР в архив кафедры	Передача сброшюрованной пояснительной записки и электронной копии материалов ВКР на оптическом диске в архив кафедры

ПОДГОТОВИТЕЛЬНЫЙ ЭТАП ВКР

Подготовку к выполнению ВКР рекомендуется начинать заранее. Имеет смысл во время производственных практик наметить предполагаемую тему ВКР, начать сбор фактического материала по ней, провести обзор состояния вопроса и анализ предметной области.

В соответствии с уставом УлГТУ студенту предоставляется право самостоятельного выбора темы ВКР с необходимым обоснованием целесообразности ее разработки.

Темы выпускных квалификационных работ должны быть актуальными, соответствовать современному состоянию и перспективам развития науки, техники и культуры.

После выбора темы каждый студент оформляет заявление по установленной форме (Приложение А)1 на имя заведующего кафедрой об утверждении темы ВКР под руководством конкретного специалиста, с которым согласован вопрос выполнения данной темы.

Руководителями могут быть наиболее опытные преподаватели и начные сотрудники УлГТУ, а также научные сотрудники и высоко квалифицированные специалисты других учреждений и предприятий.

Приказом проректора по учебной работе по представлению выпускающей кафедры утверждаются темы и руководители выпускных квалификационных работ.

Научный руководитель ВКР выдает студенту задание на выпускную квалификационную работу (Приложение Б).

Перед началом выполнения выпускной квалификационной работы студент должен составить календарный график работы на весь период с указанием очередности выполнения отдельных этапов и согласовать его с научным руководителем.

В установленные сроки студент отчитывается перед руководителем, который фиксирует степень готовности выпускной квалификационной работы.

Время, отводимое на написание и защиту выпускной квалификационной работы, определяется рабочим учебным планом направления обучения.

СОДЕРЖАНИЕ ВКР

Содержание выпускной квалификационной работы соответствует определенным ступеням высшего профессионального образования.

Выпускная работа бакалавра является результатом самостоятельного исследования или входит в состав научного комплекса как часть научно-исследовательских работ, выполненных кафедрой, с экспериментальными исследованиями или с решениями прикладных задач. В этом случае при распечатке документов приложений обозначение и название приложения необходимо исключить.В обязательном порядке должен быть отражен личный вклад автора в результаты работы научного коллектива.

Как исключение, в качестве выпускных работ бакалавров могут приниматься работы, имеющие реферативный характер, однако содержание такой работы должно в обязательном порядке включать обобщения и новые выводы, разработанные непосредственно автором, с приложением статей и публикаций по теме работы.

Бакалаврские работы могут основываться на обобщении выполненных курсовых работ и проектов и подготавливаться к защите в завершающий период теоретического обучения.

Дипломный проект является самостоятельной разработкой и решением конкретной комплексной инженерной задачи, включающей в себя обзор и критический анализ современного состояния вопроса, выбор и обоснование способа (метода) решения поставленной задачи, проектирование разрабатываемого устройства (блока) или программного комплекса (базы данных), разработку конструкций блока или устройства, технико-экономическое обоснование разработки и расчет экономического эффекта от ее внедрения, вопросы технологии производства разрабатываемого устройства (блока) или технологии программирования, отладки и

документирования программного средства, вопросы обеспечения безопасности жизнедеятельности при эксплуатации разрабатываемого устройства (блока) или программного обеспечения [23].

Магистерская диссертация представляет собой самостоятельную и логически завершенную выпускную квалификационную работу, связанную с решением задач того вида (видов) деятельности, к которым готовится магистр (научно-исследовательской, научно-педагогической, проектной, опытно-конструкторской, технологической, исполнительской, творческой). Тематика выпускных квалификационных работ должна быть направлена на решение профессиональных задач:

- анализ и моделирование проектных решений; оптимизация и принятие проектных решений;
- разработка алгоритмов и программ для автоматизированных систем управления и проектирования;
- разработка математических моделей физических, технологических, экономических процессов;
- разработка структурных, функциональных, принципиальных схем и конструкций устройств вычислительной техники и другой электронной аппаратуры.

Выпускная квалификационная работа должна включать:

1. Пояснительную записку.
2. Демонстрационный материал.
3. Иллюстрационный материал.
4. Электронный носитель.

Пояснительная записка

Техническое оформление выпускных квалификационных работ должно соответствовать нижеизложенным требованиям и указанным в разделе 3 настоящих методических указаний.

Пояснительная записка (ПЗ) должна быть грамотно написана, аккуратно оформлена, сброшюрована в твердой обложке (ориентация листов книжная, листы сброшюрованы слева).

ПЗ переплетается вместе с приложениями.

На обложку папки должна быть наклеена этикетка (Приложение В) с размерами 145 × 95 мм, на которой должен быть указан соответствующий вид документа: «Выпускная работа бакалавра», «Дипломный проект»,

«Дипломная работа» или «Магистерская диссертация».

ПЗ магистерской диссертации должна содержать все иллюстративные материалы, которые выносятся в презентацию.

ПЗ должна быть подписана автором, научным руководителем, консультантом, рецензентом и утверждена заведующим кафедрой.

К ПЗ должны быть приложены без переплетения: задание на ВКР;

отзыв научного руководителя; рецензия.

Демонстрационный материал

В демонстрационном материале (презентации), подготовленном в MS PowerPoint, должны быть представлены в удобном для обозрения виде основные результаты, полученные при выполнении ВКР.

По возможности на слайды (плакаты) выносятся наиболее существенные иллюстрационные материалы, которые приведены в тексте пояснительной записки. Количество и содержание слайдов согласовывается с научным руководителем с учетом ограниченного времени доклада.

Для плакатов строгих требований к оформлению не устанавливается, не оформляются ограничительные рамки и штампы.

Слайды выполняются в альбомной ориентации (экран 4:3, параметры страницы 25,4×19,05 см), должны быть пронумерованы (в правом верхнем углу), снабжаться заголовками, которые располагают в верхней части листа и выравнивают по центру. Заголовки не нумеруются.

Презентация должна иметь лист заглавный, лист, представляющий цели и задачи работы, а также лист результатов работы.

На одном плакате допускается размещение нескольких материалов, объединенных общим заголовком и имеющих собственные подзаголовки, которые указывают непосредственно над этими материалами. Кроме изобразительной части, плакат может содержать, при необходимости, пояснительный текст.

В отличие от иллюстраций и таблиц, размещаемых в пояснительной записке, на плакатах не используются слова «таблица», «рис.». Использование термина «таблица» допустимо только в тексте самого заголовка (подзаголовка). При вынесении на плакат формул на том же листе необходимо размещать и расшифровку значений используемых в них символов и числовых коэффициентов.

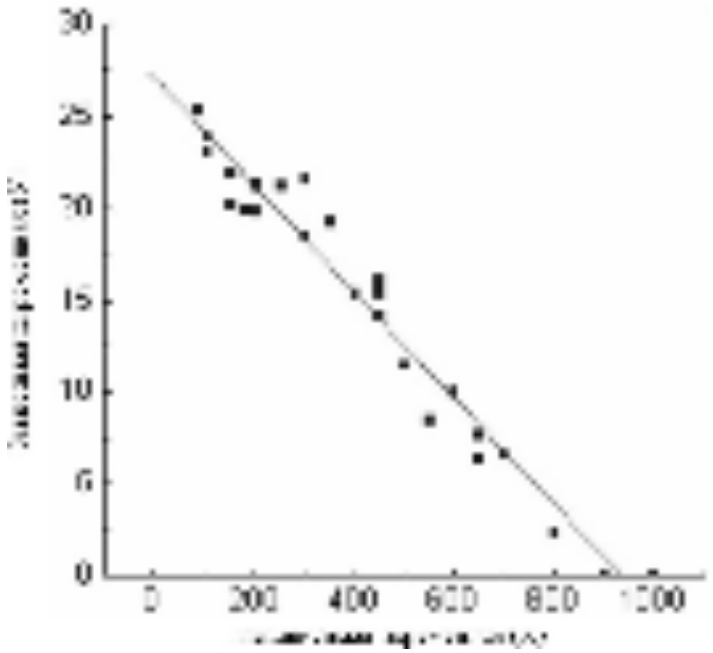
Размеры надписей и цифр на плакатах должны быть четкими и ясными, позволяющими их разглядеть с расстояния 3-4 метра.

Плакат может содержать цветные изображения и надписи, но использование цвета должно быть всегда целесообразным и функциональным [22].

При подготовке слайдов рекомендуется учитывать также требования ГОСТ 2.051-2006, ГОСТ 2.601-2006, ГОСТ 2.605-68 и Р 50-77-88.

Графики и диаграммы Основные правила выполнения диаграмм, изображающих функциональную зависимость двух или более переменных величин в системе координат, устанавливают рекомендации Р50-77-88.

При оформлении графиков оси (абсцисс и ординат) вычерчиваются сплошными линиями. На концах координатных осей стрелок не ставят (рис.1). Числовые значения масштаба шкал осей координат пишут за



пределами графика (левее оси ординат и ниже оси абсцисс). По осям координат должны быть указаны условные обозначения и размерности отложенных величин в принятых сокращениях. На графике следует писать только принятые в тексте условные буквенные обозначения. Надписи, относящиеся к кривым и точкам, оставляют только в тех случаях, когда их немного, и они являются краткими. Многословные надписи заменяют цифрами, а расшифровку приводят в подрисуночной подписи.

Алгоритмы и тексты программ

Требования к алгоритмам и текстам программ определяются комплексом стандартов «Единая система программной документации».

В ВКР по рассматриваемой специальности наиболее применимы следующие государственные стандарты.

ГОСТ 19.005-85 «Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения».

Стандарт устанавливает условные графические обозначения элементов и структур Р-схем, а также правила их выполнения автоматическим и (или) ручным способами.

Р-схема (R-chart) – нагруженный по дугам ориентированный граф, изображаемый с помощью вертикальных и горизонтальных линий и состоящий из структур (подграфов), каждая из которых имеет только один вход и один выход.

На одном листе может располагаться одна или несколько без переноса Р-схем, каждая из которых может сопровождаться текстом, записываемым до и (или) после нее.

Р-схемы вместе с сопровождающими текстами Р-схем в программных документах могут оформляться в виде иллюстраций, приложений или располагаться в разрыве между строками текста документа без нумерации.

Расстояние между Р-схемой и сопровождающим ее текстом, а также между Р-схемами должно быть больше одного интервала между строками ЗАПИСЕЙ на элементах Р-схем.

Расстояние между Р-схемой и текстом документа должно быть больше одного интервала между строками текста документа.

Р-схемы и сопровождающие их тексты при ручном изготовлении должны быть выполнены черными чернилами, пастой или тушью, иметь одинаковую толщину линий и шрифт, соответствующий ГОСТ 2.304-81.

Специальные знаки (*, #, круглые скобки), используемые при изображении Р-схем, должны по высоте не превышать 1, 5h , где h – максимальная высота строки ЗАПИСЕЙ на элементах Р-схемы.

Расстояние между ЗАПИСЯМИ, расположенными одна под другой на разных дугах одной Р-схемы, должно быть больше одного интервала между строками ЗАПИСЕЙ на элементах Р-схем.

Квадратная скобка в комментарии должна охватывать текст комментария.

Расстояние сверху и снизу от текста комментария должно быть больше одного интервала между строками текста комментария.

ГОСТ 19.106-78 «Требования к программным документам, выполненным печатным способом» допускает включение в документ частей текста программы, оформляемых в соответствии с правилами языка, на котором написан текст программы.

Для выделения отдельных понятий допускается изменять интервалы между словами, а также печатать отдельные слова или части текста шрифтом, отличным от печати основного текста, например:

UNCATLG – указывает, что запись каталога, относящаяся к исходному набору данных, должна быть исключена. ТО = устройство = список – указывает носители данных, на которые осуществляется... ABC3-91 СИНТАКСИЧЕСКАЯ ОШИБКА ПРИЧИНА. Указанный в сообщении... ДЕЙСТВИЯ СИСТЕМЫ. Задание не выполняется...ДЕЙСТВИЯ ПРОГРАММИСТА.

ГОСТ 19.401-78 «Текст программы. Требования к содержанию и оформлению».

Основная часть документа должна состоять из текстов одного или нескольких разделов, которым даны наименования.

Каждый из этих разделов реализуется одним из типов символической записи, например:

символическая запись на исходном языке; символическая запись на промежуточных языках; символическое представление машинных кодов и т.п.

В символическую запись разделов рекомендуется включать комментарии, которые могут отражать, например, функциональное назначение, структуру.

ГОСТ 19.402-78 «Описание программы».

Основная часть документа должна состоять из вводной части и следующих разделов:

–функциональное назначение;

–описание логики.

В зависимости от особенностей программы допускается введение дополнительных разделов.

Во вводной части документа приводится информация общего характера о программе (полное наименование, обозначение, применение программы и т.п.).

В разделе «Функциональное назначение» указывают назначение программы и приводят общее описание функционирования программы и сведения об ограничениях на применение, а также указывают типы электронных вычислительных машин и устройств, которые используются при работе.

В разделе «Описание логики» указывают:

–описание структуры программы и ее основных частей;

–описание функций составных частей и связей между ними;

–сведения о языке программирования;

–описание входных и выходных данных для каждой из составных частей;

–описание логики составных частей (при необходимости следует составлять описание схем программ).

При описании логики программы необходима привязка к тексту программы.

ГОСТ 19.502-78 «Описание применения. Требования к содержанию и оформлению».

Текст документа должен состоять из следующих разделов:

–назначение программы;

–условия применения;

–описание задачи;

–входные и выходные данные.

В разделе «Назначение программы» указывают назначение, возможности программы, ее основные характеристики, ограничения, накладываемые на область применения программы.

В разделе «Условия применения» указывают условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам, и другим программам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера и т. п.).

В разделе «Описание задачи» должны быть указаны определения задачи и методы ее решения.

В разделе «Входные и выходные данные» должны быть указаны сведения о входных и выходных данных.

ГОСТ 19.701-90 «Схемы алгоритмов, программ данных и систем. Условные обозначения и правила выполнения».

Стандарт распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем, и устанавливает правила выполнения схем, используемых для отображения различных видов задач обработки данных и средств их решения.

В настоящем стандарте определены символы, предназначенные для использования в документации по обработке данных, и приведено руководство по условным обозначениям для применения их в:

–схемах данных;

–схемах программ;

–схемах работы системы;

–схемах взаимодействия программ;

–схемах ресурсов системы.

Примеры названных схем приводятся в указанном ГОСТе.

Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

Большинство символов задумано так, чтобы дать возможность включения текста внутри символа. Формы символов, установленные настоящим стандартом, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры,

влияющие на соответствующую форму символов. Символы должны быть, по возможности, одного размера. Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация. Зеркальное изображение формы символа обозначает одну и ту же функцию, но не является предпочтительным.

Минимальное количество текста, необходимого для понимания функции данного символа, следует помещать внутри данного символа. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока.

Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным. В случаях, когда необходимо внести большую ясность в схему (например, присоединениях), на линиях используются стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

ГОСТ 24.211-82 «Требования к содержанию документа «Описание алгоритма».

Документ «Описание алгоритма» предназначен для описания последовательности действий и логики решения задачи (комплекса задач) в АСУ.

Документ разрабатывают на алгоритм вычисления для задачи (комплекса задач) или автоматизированной функции в целом. Допускается дополнительно разрабатывать (заимствовать) документы на отдельные части алгоритма, при этом в документе на алгоритм в целом описывают только взаимодействие этих частей.

Документ должен содержать следующие разделы:

–назначение и характеристика;

–используемая информация;

–результаты решения;

–математическое описание;

–алгоритм решения;

–требования к контрольному примеру.

Алгоритмом должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения задачи.

При изложении алгоритма следует использовать условные обозначения реквизитов, сигналов, граф, строк со ссылкой на соответствующие массивы и перечни сигналов.

Алгоритм представляют графически (в виде схемы), в виде текста или таблиц решений.

Изложение алгоритма в виде схемы представляют в соответствии с требованиями ГОСТ 19.701-90 и, при необходимости, дополняют текстовой частью.

В расчетных соотношениях (формулах) должны быть использованы обозначения реквизитов, приведенные при описании их состава.

Наименования и условные обозначения показателей и реквизитов, формируемых алгоритмом, приводят в текстовой части, если они не указаны в описании постановки задачи (или при описании реквизитов, содержащихся в массивах).

Описание алгоритма в виде текста приводят в соответствии с требованиями ГОСТ 24.301-80.

Соотношения для контроля вычислений на отдельных этапах выполнения алгоритма приводят в виде равенств и неравенств. При этом указывают контрольные соотношения, которые позволят выявить ошибки, допущенные в процессе счета, и принять решение о необходимости отклонений от нормального процесса вычислений (продолжении работы по одному из вариантов алгоритма).

Чертежи

Чертежи необходимо выполнять в соответствие с требованиями ЕСКД. При этом:

ГОСТ 2.109-73 ЕСКД определяет общие требования к чертежам,

ГОСТ 2.104-2006 ЕСКД определяет правила выполнения основных надписей к чертежам,

группа ГОСТ 2.3XX определяет общие правила выполнения чертежей,

группа ГОСТ 2.4XX определяет правила выполнения чертежей отдельных видов.

В частности:

ГОСТ 2.413-72. ЕСКД. Правила выполнения конструкторской документации изделий, изготовляемых с применением электрического монтажа,

ГОСТ 2.414-75. ЕСКД. Правила выполнения чертежей, жгутов, кабелей и проводов,

ГОСТ 2.417-91. ЕСКД. Платы печатные,

Необходимо учитывать также требования следующих стандартов: ГОСТ 2.004-88. ЕСКД. Общие требования к выполнению конструкторских и технологических документов на печатающих и графических устройствах вывода ЭВМ, ГОСТ 2.051-2006. ЕСКД. Электронные документы. Общие положения, ГОСТ 2.052-2006. ЕСКД. Электронная модель изделия. Общие положения, ГОСТ 2.053-2006. ЕСКД. Электронная структура изделия. Общие положения.

При разработке чертежей, входящих в техническую документацию на АСУ, следует использовать ГОСТ 24.304-82. СТД АСУ. Требования к выполнению чертежей.

Иллюстрационный (раздаточный) материал

Иллюстрационный (раздаточный) материал распечатывается на бумаге формата А4 с альбомной ориентацией и односторонней печатью с расчетом на каждого члена ГАК.

Содержание материала должно полностью соответствовать изображениям на слайдах презентации.

Все листы раздаточного материала должны быть скреплены или помещены в пластмассовый скоросшиватель.

Электронный носитель

Электронный носитель является неотъемлемой частью выпускной квалификационной работы и должен включать:

- файл пояснительной записки (формат doc или pdf);
- презентацию (формат doc или ppt).

Тексты программ и чертежи (при наличии) должны быть представлены в форматах использованных программных средств разработки.

В качестве оптического носителя используется диск без возможности перезаписи (CD-R или DVD-R).

Если совокупный объем файлов превышает емкость выбранного носителя, они могут быть заархивированы.

На контейнер диска должна быть наклеена этикетка (Приложение Г) с размерами 130 × 95 мм.

Ответственность за полноту и содержание записанных на диск файлов несет студент. Без комплекта документов на диске студент до защиты не допускается.

2. СТРУКТУРА И СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ ВКР

Пояснительная записка к ВКР бакалавров и специалистов, магистерская диссертация магистрантов являются по существу научно техническими документами, которые содержат систематизированные данные о научно-исследовательской работе, описывают состояние научно-технической проблемы, процесс и/или результаты научного исследования. И в этой связи при их подготовке можно учитывать требования ГОСТ 7.32- 2001 «Отчет о научно-исследовательской работе».

Отсюда рекомендуемая структура пояснительной записки (диссертации) может быть следующей:

Титульный лист

Содержание

Аннотация

Определения, обозначения и сокращения

Введение

Основная часть

Заключение

Список литературы

Приложения

Объем и структура пояснительной записки ВКР окончательно определяются по усмотрению исполнителя тематикой работы и согласовываются с научным руководителем. В этот объём не входят приложения.

ТИТУЛЬНЫЙ ЛИСТ

Титульный лист выполняется в соответствии с приложением **Б** и отличается только названием выпускной квалификационной работы:

- пояснительная записка к выпускной бакалаврской работе;
- пояснительная записка к дипломной работе (проекту);
- магистерская диссертация.

СОДЕРЖАНИЕ

Содержание включает введение, наименование всех разделов, подразделов, пунктов (если они имеют наименование), заключение, список литературы и наименование приложений с указанием номеров страниц, с которых начинаются эти элементы в документе. Оно составляется в соответствие со структурой текста (рубрикация, заголовки, нумерация), оформленной согласно п. 3.2. настоящих указаний. Слово «Содержание» записывают в виде заголовка (симметрично тексту) прописными буквами.

Наименования, включенные в содержание, записывают строчными буквами, начиная с прописной буквы.

В содержание включаются заголовки только тех структурных частей записки, которые расположены после него. Заголовки одинаковых ступеней рубрикации следует располагать друг под другом. Заголовки каждой последующей ступени смещают на три-пять знаков вправо по отношению к заголовкам предыдущей ступени. Все заголовки должны состоять из прописных букв или начинаться с прописной буквы – в соответствии с тем, как они оформлены в тексте.

Последнее слово каждого заголовка соединяют отточием с соответствующим ему номером страницы в правом столбце.

Целесообразно формировать содержание средствами текстового редактора на основе стилей заголовков. При этом все указанные выше рекомендации могут быть выполнены автоматически. Междустрочный интервал должен быть 18 пт.

Пример оформления «Содержания» пояснительной записки приведен в приложении Д.

АННОТАЦИЯ

Аннотация – краткая характеристика документа с точки зрения его назначения, содержания, вида, формы и других особенностей.

Общие требования к аннотации определяются ГОСТ 7.9-95.

В частности, в аннотации не допускается применение не общепринятых терминов и сокращений.

Образец оформления аннотации приведен в приложении Ж.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Структурный элемент «Определения» содержит определения, необходимые для уточнения или установления терминов, используемых в ВКР с соответствующими разъяснениями, если в документе принята специфическая терминология.

Перечень определений начинают со слов: «В настоящей ПЗ (или диссертации) применяют следующие термины с соответствующими определениями».

Структурный элемент «Обозначения и сокращения» содержит перечень обозначений и сокращений, применяемых в названных документах.

Запись обозначений и сокращений проводят в порядке приведения их в тексте с необходимой расшифровкой и пояснениями.

Перечень обозначений и сокращений, условных обозначений, символов, единиц физических величин и терминов должен располагаться столбцом. Слева в алфавитном порядке приводят сокращения, условные обозначения, символы, единицы физических величин и термины, справа – их детальную расшифровку.

Допускается определения, обозначения и сокращения приводить в одном структурном элементе «Определения, обозначения и сокращения».

ВВЕДЕНИЕ

Введение должно содержать оценку современного состояния решаемой научно-технической проблемы, основание и исходные данные для разработки темы, обоснование необходимости проведения НИР, сведения о планируемом научно-техническом уровне разработки, о патентных исследованиях и выводы из них, сведения о метрологическом обеспечении НИР. Во введении должны быть показаны актуальность и новизна темы, связь данной работы с другими научно-исследовательскими работами.

ОСНОВНАЯ ЧАСТЬ

Основная часть пояснительной записки к ВКР бакалавров и специалистов или магистерской диссертация в общем случае может иметь следующую структуру:

1. Постановочно-обзорная часть.
2. Проектная часть.
3. Реализация проекта.

4. Экономический раздел.

5. Безопасность и экологичность проекта.

ЗАКЛЮЧЕНИЕ

Заключение должно содержать:

- краткие выводы по результатам выполнения ВКР;

- оценку полноты решений поставленных задач;

- разработку рекомендаций и исходных данных по конкретному использованию результатов ВКР;

- оценку технико-экономической эффективности внедрения;

- оценку научно-технического уровня выполненной ВКР в сравнении с лучшими достижениями в данной области.

Пример оформления заключения ВКР приведен в приложении К.

СПИСОК ЛИТЕРАТУРЫ

В конце ПЗ обычно приводят список литературы, которая была использована при его составлении, который подготавливают согласно ГОСТ 7.1-2003.

Наиболее распространены следующие способы расположения материала в списке: алфавитный, систематический, по разделам выпускной квалификационной работы, хронологический и в порядке упоминания в тексте. Для выпускной квалификационной работы используют, как правило, алфавитный принцип расположения материалов.

При алфавитном расположении литература группируется по алфавиту фамилий авторов и заглавий книг и статей, раздельно в русском и латинском алфавитах, работы авторов-однофамильцев – по алфавиту инициалов. Общие требования и правила составления библиографического описания электронного ресурса устанавливает ГОСТ 7.82-2001.

При этом сокращение слов и словосочетаний на иностранных европейских языках библиографической записи следует выполнять в соответ- ствии с ГОСТ 7.11-2004, а на русском языке в соответствие с ГОСТ 7.12-93.

Список использованной литературы входит в основной объем работы.

Пример оформления списка литературы для различных источников приведен в приложении **Л**.

ПРИЛОЖЕНИЯ

Материал, дополняющий текст документа, допускается помещать в приложениях. Приложениями могут быть, например, графический материал, таблицы большого формата, расчеты, описания аппаратуры и приборов, описания алгоритмов и программ задач, решаемых на ЭВМ, и т. д. Размер шрифта текста программ в приложении 10 пт.

Приложение оформляют как продолжение данного документа на последующих его листах или выпускают в виде самостоятельного документа. Каждое приложение следует начинать с новой страницы с указанием наверху посередине страницы слова «Приложение» и его обозначения, а под ним в скобках для обязательного приложения пишут слово «обязательное», а для информационного – «рекомендуемое» или «справочное».

Приложение должно иметь заголовок, который записывают симметрично относительно текста с прописной буквы отдельной строкой.

Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением букв Е, З, Й, О, Ч, Ь, Ы, Ъ. После слова «Приложение» следует буква, обозначающая его последовательность.

Допускается обозначение приложений буквами латинского алфавита, за исключением букв I и O.

В случае полного использования букв русского и латинского алфавитов допускается обозначать приложения арабскими цифрами.

Если в документе одно приложение, оно обозначается «Приложение А».

Приложения должны иметь общую с остальной частью документа сквозную нумерацию страниц.

Все приложения должны быть перечислены в содержании документа (при наличии) с указанием их номеров и заголовков.

Приложения размещают по мере появления на них ссылок в тексте.

Если в качестве приложения используется реальный документ или бланк, его вкладывают в записку без изменений. Листы, на которых он размещен, включают в общую нумерацию, но не нумеруют. При необходимости отдельные элементы документа могут быть забелены (белилами типа «штрих»). Приложения, состоящие из таких документов, должны идти после всех остальных приложений. Их обозначения и наименования приводятся в только содержании записки.

Описания программных документов в приложениях должны выполняться в соответствии со следующими стандартами:

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению. ГОСТ 19.402-78 Описание программы.

ГОСТ 19.502-78 Описание применения. Требования к содержанию и оформлению.

ГОСТ 19.503-79 Руководство системного программиста. Требования к содержанию и оформлению.

ГОСТ 19.504-79 Руководство программиста. Требования к содержанию и оформлению.

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению.

ГОСТ 19.506-79. ЕСПД. Описание языка. Требования к содержанию и оформлению.

ГОСТ 24.204-80. СТД АСУ. Требования к содержанию документа «Описание постановки задачи».

ГОСТ 24.211-82 Требования к содержанию документа «Описание алгоритма».

Требования к содержанию документов, разрабатываемых при создании автоматизированных систем (АС), установлены РД50-34.698-90, а также ГОСТ 34.602-89.

Виды и комплектность документов регламентированы ГОСТ 34.201-89.

3. ОФОРМЛЕНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ ВКР

Оформление элементов пояснительной записки ВКР необходимо выполнять с учетом требований следующих государственных стандартов:

ГОСТ 2.105-95 «Единая система конструкторской документации. Общие требования к текстовым документам»,

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе»,

ГОСТ 19.106-78 «Единая система программной документации. Требования к программным документам, выполненным печатным способом»,

ГОСТ 24.301-80 «Система технической документации на АСУ. Общие требования к выполнению текстовых документов»,

РД 50-34.698-90 «ЕКС АС. Требования к содержанию документов». ОСТ 29.115–88. «Оригиналы авторские и текстовые издательские.

Общие технические требования».

Кроме того, много полезных сведений по оформлению технической документации можно найти на сайте *http://author-it.ru/.*

Ниже изложены требования с учетом названных источников.

ФОРМАТИРОВАНИЕ ТЕКСТА. ПЕЧАТАНИЕ ЗНАКОВ

Пояснительная записка ВКР является комплексным документом, включающим текст, таблицы, формулы, рисунки, графики, диаграммы, схемы, фотографии, чертежи и т. п. элементы, поясняющие текстовую часть.

Страницы текста ПЗ, включенные в нее иллюстрации и таблицы должны соответствовать формату А4 по ГОСТ 9327-60.

Все листы пояснительной записки, выполняются без рамки, кроме пояснительной дипломного проекта и чертежей.

ПЗ выполняется на принтере на одной стороне листа белой бумаги формата А4.

При выполнении текстовой части работы на компьютере текст должен быть оформлен в текстовом редакторе *Word for Windows*.

Шрифт основного текста: *Times New Roman,* обычный, размер 14 пт. Межсимвольный интервал: обычный. Междустрочный интервал: полуторный. Абзацный отступ обычно устанавливается равным 1,25 см.

Текст следует печатать, соблюдая следующие размеры полей: верхнее – 25 мм, левое – 25 мм, нижнее – 20 мм и правое – 15 мм.

Текст должен быть выровнен по ширине листа. Перед набором текста устанавливается автоматический режим переносов (кроме переносов в заголовках разделов, подразделов и пунктов, а также в названиях рисунков, таблиц и приложений).

Отдельные слова или словосочетания могут быть выделены стилем шрифта (курсивом, полужирным начертанием, подчеркиванием), если на них требуется акцентировать внимание. В случае необходимости могут выделяться одно или несколько предложений. Но злоупотреблять этими средствами не следует.

Если в тексте приводятся надписи, используемые, к примеру, в интерфейсе программного обеспечения, их выделяют стилем шрифта без употребления кавычек. Например: «…в меню **Файл**», «… предназначена кнопка **Выполнить расчет**». Но тексты сообщений, наименования режимов (и т. д.) обрамляются кавычками, например, «… выводится сообщение«Расчет выполнен». Следует применять только угловые кавычки (« »).

Обычные кавычки могут использоваться лишь в английских текстах.

Фамилии, названия предприятий и другие имена собственные приводятся в тексте записки на языке оригинала. Допускается транскрибировать имена собственные или переводить их на русский язык (за исключением фамилий) с добавлением при первом упоминании оригинального названия. Опечатки, описки и графические неточности, обнаруженные в процессе выполнения документа, допускается исправлять подчисткой или закрашиванием белой краской и нанесением на том же

месте исправленного текста (графики) машинописным способом или черными чернилами, пастой или тушью рукописным способом. При этом не должно быть повреждений листов, помарок и следов прежнего текста.

При выполнении текста необходимо соблюдать равномерную плотность, контрастность и четкость изображения по всему документу. В ПЗ должны быть четкие, не расплывшиеся линии, буквы, цифры и знаки.

Не следует использовать для оформления текста какой-либо другой цвет, кроме чёрного

Правила печатания знаков. Знаки препинания (точка, запятая, двоеточие, точка с запятой, многоточие, восклицательный и вопросительный знаки) от предшествующих слов пробелом не отделяют, а от последующих отделяют одним пробелом.

Дефис (короткая черточка) применяется в основном для разделения частей сложных слов и никогда не отделяется пробелами.

Тире (длинная черточка) – знак препинания, используемый в предложениях. Тире всегда отделяется пробелами с двух сторон, но не переносится так, чтобы с него начиналась новая строка (поэтому перед тире лучше ставить «неразрывный» пробел, если это позволяет текстовый редактор).

Кавычки и скобки не отбивают от заключенных в них элементов.

Знаки препинания от кавычек и скобок не отбивают.

Знак *№* применяют только с относящимися к нему числами, между ними ставят пробел.

Знаки сноски (звездочки или цифры) в основном тексте печатают без пробела, а от текста сноски отделяют одним ударом (напр.: *слово*[1]*,* [1] *Слово*).

Знаки процента и промилле от чисел отбивают.

Знаки углового градуса, минуты, секунды, терции от предыдущих чисел не отделяют, а от последующих отделяют пробелом (напр.: 5º 17′′).

Знак градуса температуры отделяется от числа, если за ним следует сокращенное обозначение шкалы *(напр*., 15 ºС, но 15º *Цельсия*).

СТРУКТУРА ТЕКСТА (РУБРИКАЦИЯ, ЗАГОЛОВКИ, НУМЕРАЦИЯ)

Основную часть ПЗ следует делить на разделы, подразделы и пункты.

Пункты, при необходимости, могут делиться на подпункты. При делении текста отчета на пункты и подпункты необходимо, чтобы каждый пункт содержал законченную информацию.

Разделы должны иметь порядковую нумерацию в пределах всего текста, за исключением приложений.

Пример – 1, 2, 3 и т. д.

Номер подраздела или пункта включает номер раздела и порядковый номер подраздела или пункта, разделенные точкой.

Пример – 1.1, 1.2, 1.3 и т. д.

Номер подпункта включает номер раздела, подраздела, пункта и порядковый номер подпункта, разделенные точкой.

Пример - 1.1.1.1, 1.1.1.2, 1.1.1.3 и т. д.

После номера раздела, подраздела, пункта и подпункта в тексте точку не ставят.

Если раздел или подраздел имеет только один пункт, или пункт имеет один подпункт, то нумеровать его не следует.

Если текст ПЗ подразделяют только на пункты, их следует нумеровать, за исключением приложений, порядковыми номерами в пределах всего отчета.

Разделы, подразделы должны иметь заголовки. Пункты, как правило, заголовков не имеют. Заголовки должны четко и кратко отражать содержание разделов, подразделов.

Заголовки разделов, подразделов и пунктов следует печатать без точки в конце, не подчеркивая.

Если заголовок состоит из двух предложений, их разделяют точкой.

Переносить части слова заголовка или части обозначения на другую строку не допускается.

Наименование разделов записываются в виде заголовков (симметрично тексту) прописными буквами: шрифт *Times New Roman*, размер 14 пт, полужирный.

Наименование подразделов записываются в виде заголовков (с выравниванием по левому краю) прописными буквами: шрифт *Times New Roman*, размер 14 пт, обычный.

Наименование пунктов записываются в виде заголовков (с выравниванием по левому краю) строчными буквами (кроме первой прописной), шрифт *Times New Roman*, размер 14 пт, полужирный.

Расстояние между заголовком и текстом, между заголовками раздела и подраздела должно быть равно одному интервалу (см. приложение Т).

Каждый раздел текстового документа рекомендуется начинать с нового листа (страницы).

Структурные части записки «Аннотация», «Содержание», «Список использованных сокращений и обозначений», «Введение», «Заключение», «Список использованных источников» не нумеруют.

Для выделения объединенных по смыслу частей длинного текста он разбивается на абзацы. Число предложений в абзаце обычно колеблется от двух до пяти-шести. Абзацный отступ должен быть одинаковым в пределах всей записки.

Номера листов (страниц) проставляют, начиная с первого листа, следующего за титульным листом, в верхней части листа (над текстом, посередине), *за исключением ПЗ дипломного проекта, где номера листов указывают в основной надписи.*

Титульный лист включают в общую нумерацию страниц отчета. Номер страницы на титульном листе не проставляют.

Иллюстрации и таблицы, расположенные на отдельных листах, включают в общую нумерацию страниц отчета.

Иллюстрации и таблицы на листе формата А3 учитывают, как одну страницу.

Нумерация страниц отчета и приложений, входящих в состав отчета, должна быть сквозная.

ИЗЛОЖЕНИЕ ТЕКСТА

Изложение содержания пояснительной записки должно быть кратким и четким, исключающим возможность неверного толкования.

В документах должны применяться научно-технические термины, обозначения и определения, установленные соответствующими стандарта ми, а при их отсутствии – общепринятые в научно-технической литературе.

Пояснительная записка является техническим документом, поэтому следует избегать употребления в ее тексте обращений от первого лица (не «я выбрал», а «было выбрано»), оборотов разговорной речи, предложений с восклицательными знаками и т. п. При изложении обязательных требований следует применять слова «должен», «следует», «необходимо», «обеспечить» и т. п., при изложении менее категоричных положений – слова «могут быть», «как правило», «при необходимости», «в случае» и т. д.

В тексте документа не допускается:

- применять обороты разговорной речи, техницизмы, профессионализмы;
- применять для одного и того же понятия различные научно-технические термины, близкие по смыслу (синонимы), а также иностранные слова и термины при наличии равнозначных слов и терминов в русском языке;
- применять произвольные словообразования;
- применять сокращения слов, кроме установленных правилами русской орфографии и соответствующими государственными стандартами;
- сокращать обозначения единиц физических величин, если они употребляются без цифр, за исключением единиц физических величин в голов- ках и боковиках таблиц и в расшифровках буквенных обозначений, входящих в формулы и рисунки.

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

При оформлении ПЗ допускаются сокращения слов в тексте и надписях под иллюстрациями по ГОСТ 2.316-68. Дополнительные сокращения, принятые в документе и не входящие в ГОСТ 2.316-68, следует приводить в разделе «Определения, обозначения и сокращения».

Сокращение русских слов и словосочетаний в тексте – по ГОСТ 7.12-93.

В тексте документа, за исключением формул, таблиц и рисунков, не допускается:

- применять математический знак минус (-) перед отрицательными значениями величин (следует писать слово «минус»);

- применять знак «диаметра» для обозначения диаметра (следует писать слово «диаметр»). При указании размера или предельных отклонений диаметра на чертежах, помещенных в тексте документа, перед размерным числом следует писать знак «диаметр»;

- применять без числовых значений математические знаки, например:

> (больше), < (меньше), = (равно), >= (больше или равно), <= (меньше или равно), =/ (не равно), а также знаки N (номер), % (процент).

При необходимости применения условных обозначений, изображений или знаков, не установленных действующими стандартами, их следует пояснять в тексте или в перечне обозначений.

Используемые сокращения должны соответствовать правилам грамматики, а также требованиям государственных стандартов.

Однотипные слова и словосочетания везде должны либо сокращаться, либо нет (напр.: *в 1919 году и XX веке* или *в 1919 г. и XX в.*; *и другие, то есть* или *и др., т.е.*).

Существует ряд общепринятых графических сокращений: Сокращения, употребляемые самостоятельно: *и др., и пр., и т.д., и т.п.* Употребляемые только при именах и фамилиях*: г-н, т., им., акад., др., доц., канд.физ.-мат.наук, ген., чл.-кор.* Напр.: *доц. Иванов И.И.*

Слова, сокращаемые только при географических названиях: *г., с., пос., обл., ул., просп*. Например: *в с. Н. Павловка*, но: *в нашем селе*.

Употребляемые при ссылках, в сочетании с цифрами или буквами:

гл.5, п.10, подп.2а, разд.А, с.54 – 598, рис.8.1, т.2, табл.10 – 12, ч.1.

Употребляемые только при цифрах: *в., вв., г., гг., до н.э., г.н.э., тыс., млн, млрд, экз., к., р.* Например: *20 млн. р., 5 р. 20 к.*

Используемые в тексте сокращения поясняют в скобках после первого употребления сокращаемого понятия. Обозначение, встречающееся в тексте в первый раз, указывается в скобках, сразу за его расшифровкой, например, «... может применяться генетический алгоритм (ГА)». Далее по тексту обозначение употребляется уже без скобок. Допускается приводить расшифровку одного и того же обозначения в каждом из разделов записки. В пояснительной записке собственные сокращения, как правило, вводить не требуется, но могут использоваться общепринятые сокращения, применяемые при записи единиц измерения, а также грамматические сокращения, в том числе:

– после перечисления: и т. д. (и так далее), и т. п. (и тому подобное), и др. (и другие), и пр. (и прочие);

– при пояснении: т. е. (то есть);

– при ссылках: см. (смотри), ср. (сравни), напр. (например);

– при цифровом обозначении веков и годов: в. (век), вв. (века), г. (год), гг. (годы).

Запрещается применять сокращенные обозначения единиц измерения (в том числе денежных), если они употребляются без цифр, за исключением использования сокращений в таблицах и расшифровках формул. Сокращения типа «и др.» ставятся только перед знаком препинания или закрывающей скобкой, в середине текста их нужно записывать полностью. В конце части общеупотребительных сокращений (м, г) принято точку не ставить.

ФИЗИЧЕСКИЕ ВЕЛИЧИНЫ

В пояснительной записке следует применять стандартизованные единицы физических величин, их наименования и обозначения в

соответствии с ГОСТ 8.417-2002. В качестве обозначений предусмотрены буквенные обозначения и специальные знаки, напр.: *20.5 кг*, *438 Дж/(кг/К)*, *36ºС*. При написании сложных единиц комбинировать буквенные обозначения и наименования не допускается. Наряду с единицами СИ, при необходимости, в скобках указывают единицы ранее применявшихся систем, разрешенных к применению.

Представление в ПЗ данных о свойствах веществ и материалов проводятся по ГОСТ 7.54-88.

При отборе численных данных в первую очередь необходимо использовать официально утвержденные нормативные документы: государственные и отраслевые стандарты, таблицы стандартных и рекомендуемых справочных данных, стандарты и рекомендации СЭВ и международных организаций и источники, содержащие критически оцененные численные данные (научные публикации, массивы банков данных).

В ПЗ следует включать сведения о свойствах веществ и материалов из источников, появляющихся в отечественной и зарубежной литературе. При этом следует разграничивать численные данные, полученные авторами и заимствованные ими из других источников.

Применение в одном документе разных систем обозначения физических величин не допускается.

В тексте документа числовые значения величин с обозначением единиц физических величин и единиц счета следует писать цифрами, а числа без обозначения единиц физических величин и единиц счета от единицы до девяти – словами.

Примеры:

3.1.1. Провести испытания пяти труб, каждая длиной 5 м.

3.1.2. Отобрать 15 труб для испытаний на давление.

Единица физической величины одного и того же параметра в пределах одного документа должна быть постоянной. Если в тексте приводится ряд числовых значений, выраженных в одной и той же единице физической

величины, то ее указывают только после последнего числового значения, например 1,50; 1,75; 2,00 м.

Если в тексте документа приводят диапазон числовых значений физической величины, выраженных в одной и той же единице физической величины, то обозначение единицы физической величины указывается после последнего числового значения диапазона.

Примеры:

1. От 1 до 5 мм.
2. От 10 до 100 кг.
3. От плюс 10 до минус 40 °С.
4. От плюс 10 до плюс 40 °С.

Недопустимо отделять единицу физической величины от числового значения (переносить их на разные строки или страницы), кроме единиц физических величин, помещаемых в таблицах, выполненных машинописным способом.

Приводя наибольшие или наименьшие значения величин следует применять словосочетание «должно быть не более (не менее)». Приводя допустимые значения отклонений от указанных норм, требований, следует применять словосочетание «не должно быть более (менее)». Например, массовая доля углекислого натрия в технической кальцинированной соде должна быть не менее 99,4 %.

Числовые значения величин в тексте следует указывать со степенью точности, которая необходима для обеспечения требуемых свойств изделия, при этом в ряду величин осуществляется выравнивание числа знаков после запятой.

Округление числовых значений величин до первого, второго, третьего и т. д. десятичного знака для различных типоразмеров, марок и т.п. изделий одного наименования должно быть одинаковым. Например, если градация толщины стальной горячекатаной ленты 0,25 мм, то весь ряд толщин ленты должен быть указан с таким же количеством десятичных знаков, например 1,50; 1,75; 2,00.

Дробные числа необходимо приводить в виде десятичных дробей, за исключением размеров в дюймах, которые следует записывать 1/4; 1/2.

При невозможности выразить числовое значение в виде десятичной дроби, допускается записывать в виде простой дроби в одну строчку через косую черту, например, 5/32; (50A -4C)/(40B + 20).

Буквенные обозначения единиц счета и измерений физических величин применяют в тексте только при числовых значениях и записывают без точки (например: 5 шт, 10 кг).

Наименование единиц счета и измерений физических величин, употребляемые в тексте без числовых значений, следует записывать без сокращений (например: длина приведена в миллиметрах).

ЗНАКИ И ЧИСЛА. ЧИСЛИТЕЛЬНЫЕ

Знаки №, §, %, +, - и т. д. следует применять только при числовых значениях. В тексте эти знаки следует писать словами. Математические знаки («+», «-», «=», «>», «<» и др.) используются только в формулах, в тексте их следует писать словами.

Римские цифры следует употреблять только для обозначения сорта (категории) изделия, кварталов года, полугодия. В остальных случаях применяют арабские цифры.

Многозначные числа пишут арабскими цифрами и разбивают на классы (напр.: 13 692). Не разбивают четырехзначные числа и числа, обозначающие номера.

Числа должны быть отбиты от относящихся к ним наименований (напр.: 25 *м).* Числа с буквами в обозначениях не разбиваются (напр.: *в пункте 2б*). Числа и буквы, разделенные точкой, не имеют отбивки (напр.: 2.13.6).

Основные математические знаки перед числами в значении положительной или отрицательной величины, степени увеличения от чисел не отделяют (напр.: -15, ×20).

Для обозначения диапазона значений употребляют один из способов: многоточие, тире, знак ÷, либо предлоги от … до … . По всему тексту следует придерживаться принципа единообразия.

Сложные существительные и прилагательные с числами в их составе рекомендуется писать в буквенно-цифровой форме (напр.: *150-летие*, *30-градусный*, *25-процентный*).

Стандартной формой написания дат является следующая: 20.03.93 г. Возможны и другие как цифровые, так и словесно-цифровые формы: *20.03.1993 г., 22 марта 1993 г., 1 сент. 1999 г.*

Все виды некалендарных лет (бюджетный, отчетный, учебный), т. е. начинающихся в одном году, а заканчивающихся в другом, пишут через косую черту: *В 1993/94 учебном году. Отчетный 1993/1994 год.*

Представление дат и времени дня определяется по ГОСТ 7.64-90.

Количественные числительные записываются цифрами, если они являются многозначными, и словами, если они однозначны. Например, «десять модулей», но «25 строк». При количественных числительных, записанных арабскими цифрами, падежные окончания не пишутся, если числительные сопровождаются существительными. Например, не «12-ти рублей», а «12 рублей».

Порядковые числительные пишутся либо словами («седьмой», «двадцать первый»), либо арабскими цифрами. Для выбора способа записи целесообразно использовать то же правило, что и для количественных числительных. При записи цифрами числительные имеют падежные окончания. При перечислении нескольких порядковых числительных падежное окончание ставится только один раз. Например, «уравнения 2 и 3-й степени». При записи римскими цифрами порядковые числительные падежных окончаний не имеют. Например, «XXI век», а не «XXI-й век».

При размещении в записке цифрового материала должны использоваться только арабские цифры, за исключением нумерации кварталов, полугодий и т. п., которые традиционно обозначаются римскими цифрами.

Числа с размерностью следует писать цифрами, а без размерности – словами. Например: «не более 2,5 сек.», «время уменьшится в два с половиной раза».

Если приводится ряд величин одной и той же размерности, то единица измерения указывается только после последнего числа. Аналогично, знаки номера, градуса, процента пишутся только один раз, соответственно, при первой или последней цифре. Например: «№ 2, 3, 5, 9», «7; 10; 12».

Для величин, составляющих диапазон, единица измерения пишется только один раз при второй цифре, к примеру: «3-5 см», «от 3 до 5 см».

Недопустимо отделять единицу измерения от числового значения (переносить ее на следующую строку).

ПЕРЕЧИСЛЕНИЯ

При изложении большого числа однородных фактов в тексте можно использовать перечисления (списки), причем сразу после заголовка (без вводного предложения, которое предшествует перечислению) они не допускаются. Перед перечислением ставится двоеточие. Все элементы перечисления должны грамматически подчиняться вводному предложению.

Перечисления могут быть маркированными, нумерованными или буквенными.

Элементы маркированных списков с текстовыми фрагментами из нескольких предложений следует начинать с прописной буквы и в конце фрагментов ставить точку.

При мелких, например, однострочных элементах перечислений, их следует начинать со строчной буквы и заканчивать точкой с запятой.

Если элементы перечисления не имеют внутренних знаков пунктуации, например, содержат по одному слову или слову с определением, их можно разделять запятой.

Хотя при использовании текстового редактора роль маркера может выполнять произвольный символ, в пояснительной записке для поддержания строгого стиля оформления следует использовать только тире.

В нумерованных списках с крупными текстовыми фрагментами используются арабские цифры с точкой, после которой текст начинается с прописной буквы и завершается точкой. Если перечисление состоит из коротких элементов, они нумеруются арабскими цифрами со скобкой, начинаются со строчной буквы и разделяются точкой с запятой или запятой. Использование римских цифр не допускается. Аналогично, в буквенных списках в первом случае роль идентификатора элемента списка играют прописные буквы с точкой после них, а во втором – строчные буквы со скобкой.

Перечисления, элементы которых не содержат отдельных предложений, могут включаться в состав абзаца. Например, «оперативная память в системах с массовым параллелизмом имеет трехуровневую структуру:

1. кэш-память процессора,
2. локальная оперативная память узла,
3. оперативная память других узлов».

В маркированных списках рекомендуется использовать одинаковые маркеры по всей работе.

ТАБЛИЦЫ

Объемный цифровой материал, а также многомерный текстовой материал должен оформляться в виде таблиц. Эта форма дает лучшую наглядность и сравнимость показателей.

Таблицу, в зависимости от ее размера, помещают под текстом, в котором впервые дана ссылка на нее, или на следующей странице, а при необходимости, в приложении к документу.

Допускается помещать таблицу вдоль длинной стороны листа документа.

Название таблицы, при его наличии, должно отражать ее содержание, быть точным и кратким.

Название таблицы следует помещать над таблицей слева, без абзацного отступа в одну строку с ее номером через тире (рис. 2). Заголовок следует

выполнять строчными буквами. Прописными должны печататься заглавные буквы и аббревиатуры.

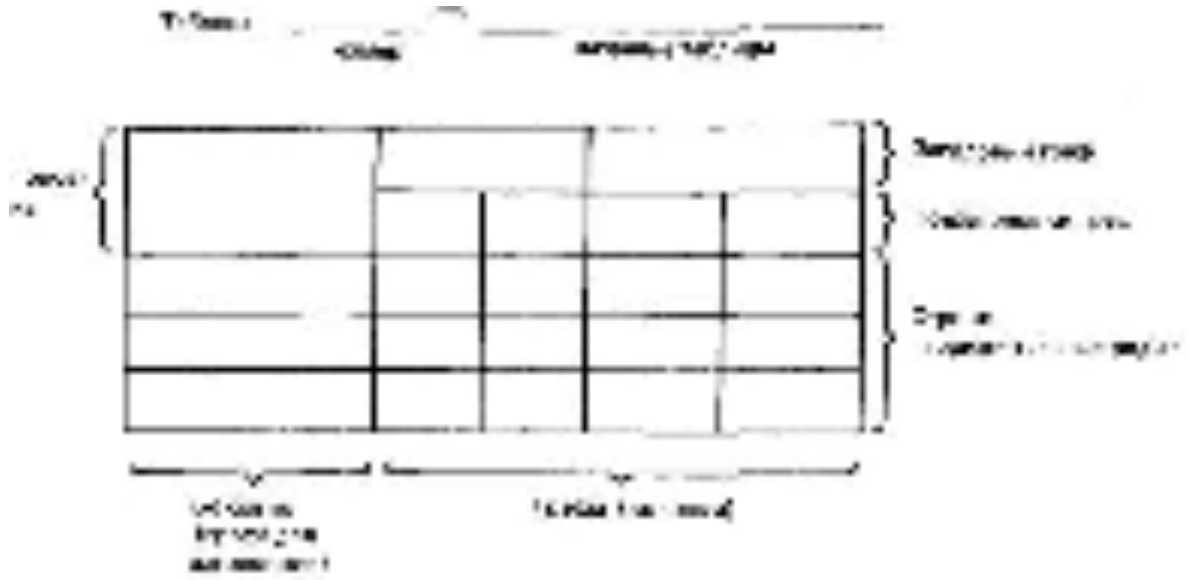


Рис.2

При переносе части таблицы на ту же или другие страницы название помещают только над первой частью таблицы.

Таблицы, за исключением таблиц приложений, следует нумеровать арабскими цифрами сквозной нумерацией.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Если в документе одна таблица, она должна быть обозначена «Таблица 1» или «Таблица В.1», если она приведена в приложении В. Дополнительный интервал перед этой строкой должен составлять 12 пт. При наличии тематического заголовка его помещают над таблицей в следующей строке и выравнивают по центру. Точка в конце заголовка таблицы не ставится. Дополнительный начальный интервал для абзаца, следующего после таблицы, должен быть равен 18 пт.

Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой.

На все таблицы документа должны быть приведены ссылки в тексте документа, при ссылке следует писать слово «таблица» с указанием ее номера.

Заголовки граф и строк таблицы следует писать с прописной буквы, а подзаголовки граф – со строчной буквы, если они составляют одно предложение с заголовком. В конце заголовков и подзаголовков точка не ставится. Обозначение единицы измерения, общей для всех данных в графе или строке, следует указывать через запятую вслед за заголовком графы или строки.

Заголовки и подзаголовки граф указывают в единственном числе. Таблица со всех сторон, как правило, ограничивается линиями. «Шапка» таблицы также должна быть отделена линией от остальной части таблицы. Горизонтальные и вертикальные линии, разграничивающие строки и столбцы таблицы, допускается не проводить, если это не затрудняет изучение таблицы.

Использовать диагональные разделяющие линии в таблице недопустимо. Заголовки и подзаголовки граф выравниваются по центру соответствующей графы. Они, как правило, записываются параллельно строкам таблицы, но также допускается их перпендикулярное расположение, если параллельная запись затруднительна.

Заголовки граф, как правило, записывают параллельно строкам таблицы. При необходимости допускается перпендикулярное расположение заголовков граф.

Головка таблицы должна быть отделена линией от остальной части таблицы.

Высота строк таблицы должна быть не менее 8 мм.

Графу «Номер по порядку» в таблицу включать не допускается.

При необходимости нумерацию показателей, параметров или других данных порядковые номера следует указывать в первой графе (боковике) таблицы непосредственно перед их наименованием, не используя отдельную графу.

Заменять кавычками повторяющиеся в таблице цифры, математические знаки, знаки процента и номера, обозначение марок материалов и типоразмеров изделий, обозначения нормативных документов не допускается.

Если параметры одной графы имеют одинаковые значения в двух и более строках, то допускается объединение соответствующих ячеек данной графы в одну ячейку и проставление параметра один раз. То же относится и к одинаковым значениям параметра в одной строке.

При отсутствии отдельных данных в таблице следует ставить прочерк (тире).

Если все показатели, приведенные в графах таблицы, выражены в одной и той же единице измерения, то ее обозначение следует помещать непосредственно над таблицей справа, например, «Длительность в секундах». При делении таблицы на части, это обозначение указывается над каждой ее частью.

Сноски к таблицам располагают непосредственно под таблицей. Например:

Таблица 2 - Наборы данных, используемые для распечатки

Назначение	Стандартное имя	Используемое устройство
Для информационной распечатки	SSSSSSS[1)]	Печатающее устройство[2)]
Для распечатки во время выполнения программы	PPPPPPPP	Печатающее устройство[2)]

[1)] Имя SSSSSSS должно быть задано при настройке операционной системы.

[2)] Для уменьшения простоев центрального процессора из-за операций ввода-вывода может быть использована магнитная лента.

Если таблица текстовая, то текст в ячейках таблицы всегда начинается с прописной буквы без точки в конце. Однако все промежуточные знаки препинания проставляются, в том числе и точки между предложениями. Текстовое значение записывается на уровне первой строки наименования

показателя. Числовое значение показателя проставляется на уровне последней строки наименования показателя. При этом числа должны располагаться так, чтобы цифры одного разряда во всей графе были расположены точно друг под другом. Числовые величины в одной графе должны иметь одинаковое количество десятичных знаков.

Если строки или графы таблицы выходят за формат листа, таблицу делят на части, помещая одну часть под другой. И в каждой части таблицы повторяют ее шапку и боковик (первую графу). При этом во второй и последующих частях допускается замена шапки и боковика таблицы соответственно номерами граф и строк. Также нумерация граф или строк производится, если на них имеются ссылки в тексте. Номера граф задаются отдельной строкой сразу после шапки таблицы, номера строк – первой графой.

Если таблица занимает несколько листов пояснительной записки, то в начале каждого листа, на котором продолжается таблица, рекомендуется добавлять строку с записью «Продолжение табл.», выровненной по правому краю, и указывать номер таблицы, например, «Продолжение табл. 6.1». Название таблицы при этом не повторяется.

ИЛЛЮСТРАЦИИ

Все имеющиеся в работе иллюстрации (чертежи, графики, схемы, компьютерные распечатки, диаграммы, фотоснимки и пр.) именуются рисунками.

Количество иллюстраций должно быть достаточным для пояснения излагаемого текста.

Чертежи, графики, диаграммы, схемы, иллюстрации, помещаемые в ПЗ, должны соответствовать требованиям государственных стандартов.

Перечень допускаемых сокращений слов в графических документах установлен в ГОСТ 2.316-2008.

Иллюстрации могут быть расположены в тексте документа и (или) в приложениях.

Иллюстрации следует располагать в ПЗ непосредственно после текста, в котором они упоминаются впервые, или на следующей странице.

Иллюстрации, если их в данном документе более одной, за исключением иллюстрации приложений, следует нумеровать арабскими цифрами сквозной нумерацией.

В приложениях иллюстрации нумеруются в пределах каждого приложения в порядке, установленном для основного текста документа. Допускается не нумеровать мелкие иллюстрации, размещенные непосредственно в тексте и на которые в дальнейшем нет ссылок.

Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст). Слово «Рисунок» и наименование помещают после пояснительных данных и располагают следующим образом: Рисунок 1 - Детали прибора.

Слово «рисунок», его номер и наименование рисунка помещают ниже изображения и пояснительных данных, по центру (относительно рисунка).

Тематический заголовок (наименование) помещают над иллюстрацией, подрисуночный текст – под ней. Номер иллюстрации помещают под поясняющими данными.

Текст, обтекающий иллюстрацию, должен отстоять от нее на расстоянии не менее 8 мм. Дополнительный интервал между текстом (сверху) и иллюстрацией, между подписью иллюстрации и текстом (снизу) – 12 пт, между иллюстрацией и ее подписью – 6 пт.

Размеры иллюстраций не должны превышать формата страницы с учетом полей. Если ширина рисунка больше 8 см, то его располагают симметрично посередине. Если его ширина менее 8 см, то рисунок, как правило, располагают с краю, в обрамлении текста. Допускается размещение нескольких иллюстраций на одном листе. Сложные иллюстрации могут выполняться на листах формата А3 и больше со сгибом для размещения в пояснительной записке.

Фотоснимки размером меньше формата А4 должны быть наклеены на стандартные листы белой бумаги.

Рисунки, фотографии, вставляемые в ПЗ, должны быть без пометок, карандашных исправлений, пятен, трещин и загибов.

На все иллюстрации должны быть даны ссылки в ПЗ. При ссылках на иллюстрации следует писать «... в соответствии с рисунком 2» при сквозной нумерации и «... в соответствии с рисунком 1.2» при нумерации в пределах раздела.

В тексте, где идет речь о теме, связанной с иллюстрацией, помещают ссылку либо в виде заключенного в круглые скобки выражения (рис. 3.1) либо в виде оборота типа «…как это видно на рис. 3.1».

Ссылки на ранее упомянутые иллюстрации дают с сокращенным словом «смотри», например, «см. рис. 12».

Иллюстрации должны быть расположены так, чтобы их было удобно рассматривать без поворота работы или с поворотом по часовой стрелке.

При переносе объемной иллюстрации на следующий лист (в исключительных случаях) ее наименование не повторяется, однако ниже указывается номер иллюстрации со словом «продолжение», например, «Продолжение рис. 3.1».

Иллюстрации, размещаемые на отдельном листе, могут иметь как вертикальную, так и горизонтальную ориентацию.

Иллюстрации могут быть в компьютерном исполнении, в том числе и цветные.

Иллюстрации должны быть вставлены в текст одним из следующих способов:

- либо командами ВСТАВКА-РИСУНОК (используемые для вставки рисунков из коллекции, из других программ и файлов, со сканера, созданные кнопками на панели рисования, автофигуры, объекты Word Art, а также диаграммы). При этом все иллюстрации, вставляемые как рисунок, должны быть преобразованы в формат графических файлов, поддерживаемых Word;

- либо командами ВСТАВКА-ОБЪЕКТ. При этом необходимо, чтобы объект, в котором создана вставляемая иллюстрация, поддерживался редактором Word стандартной конфигурации.

Примеры оформления различных иллюстраций в ПЗ показаны в приложениях Ф, Х, Ц [23], Ш к данным методическим указаниям.

С дополнительными требованиями к иллюстрациям рекомендуется ознакомиться в п.п. 6.10 – 6.25 ГОСТ 2.601-2006.

ФОРМУЛЫ

В формулах в качестве символов следует применять обозначения, установленные соответствующими государственными стандартами. Все расчеты представляются в системе СИ. Размерность одного и того же параметра в пределах одного документа должна быть постоянной.

Формулы и расчеты должны органически вписываться в текст, не разрывая его грамматической структуры. Формулы следует располагать на середине строки, а связывающие их слова «где», «следовательно», «откуда», «находим», «определяем» – в начале строк.

Небольшие и не имеющие особого значения формулы можно размещать непосредственно в строке текста, а объемные, достаточно важные формулы, на которые будут делаться ссылки, следует выделять в отдельную строку. В этом случае формулы размещают с абзацного отступа или выравнивают по центру, отделяя от текста снизу и сверху дополнительным интервалом в 6 пт.

Формулы лучше размещать отдельными строками, но небольшие формулы можно размещать по тексту. Если формула имеет высоту до 1,5 интервала и в тексте на нее нет ссылок, то она может быть помещена в текст. Формулы, следующие одна за другой и не разделенные текстом, отделяют запятыми.

Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Для удобства чтения следует между формулой (группой формул) и текстом установить интервал, равный одному интервалу текста.

Формулы, следующие одна за другой и не разделенные текстом, отделяют друг от друга запятой или точкой с запятой, которые ставят за формулами до их номера. При написании формул, не помещающихся по ширине листа, их разделяют на две, три и более строк. Переносы формул со строки на строку осуществляются в первую очередь на знаках отношения (=; ≠; ≥, ≤ и т. п.), во вторую – на знаках сложения и вычитания, в третью – на знаке умножения. При переносе эти знаки повторяются в конце и начале строк. При переносе формулы на знаке умножения применяют знак «×». Не допускаются переносы на знаке деления (:). Переносить на другую строку допускается только самостоятельные члены формулы. Не допускается при переносе разделение показателей степени, выражений в скобках, дробей, а также выражений, относящихся к знакам корня, интеграла, суммы, логарифма, тригонометрических функций и т. п.

Пояснения символов и числовых коэффициентов, входящих в формулу, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него.

Пояснение для того или иного символа или коэффициента не дается, если оно уже было сделано ранее в этом разделе (для другой формулы или в тексте).

Формулы, на которые в дальнейшем придется ссылаться, следует нумеровать. Все формулы нумеруются арабскими цифрами, номер ставят с правой стороны листа на уровне формулы в круглых скобках (номер формулы выравнивается по правому краю текста). Нумерация формул в разделе начинается с единицы.

Номер формулы состоит из 2-х частей, разделенный точкой, например (3.1), первая часть выделена под номер раздела, вторая часть – номер формулы. Допускается нумерация формул в пределах пояснительной записки. При переносе формулы номер ставят напротив последней строки в

край текста. Если формула помещена в рамку, номер помещают вне рамки против основной строки формулы. В приложениях формулы нумеруются в пределах каждого приложения с добавлением буквенного обозначения приложения.

Производные от приведенной ранее основной формулы можно обозначать, используя номер основной формулы с добавлением строчной буквы русского алфавита, которая пишется слитно с номером, например, (6.1а).

Группа формул, объединенных фигурной скобкой, имеет один номер, помещаемый точно против острия скобки.

В пределах всего документа или его частей, в случае деления документа на части, формулы имеют сквозную нумерацию.

Ссылки в тексте на порядковый номер формулы дают в скобках, например: «в формуле (3)», «…из формулы (3.1) следует…».

При делении документа на части номер части ставится перед порядковым номером формулы и отделяется от последней точкой, например:

«в формуле (1.4)».

В конце формулы и в тексте перед ней знаки препинания ставят в соответствии с правилами пунктуации.

Формулы, которые нельзя корректно представить в текстовом виде (многоуровневые, использующие операцию суммирования и т. д.), должны создаваться средствами редактора формул, встроенного в используемый текстовый редактор или внешнего.

Для подготовки формул в пояснительной записке может быть использован встроенный редактор формул MS Word.

При использовании других текстовых редакторов (например, группы TeX) следует стремиться к тому, чтобы высота символа в формуле и его начертание совпадали со шрифтом основного текста пояснительной записки.

В формулах английские и латинские буквы набираются курсивом, русские и греческие – прямым шрифтом.

Допускается также вписывать формулы в текст пояснительной записки от руки черной тушью (по ГОСТ 2.304) чертежным шрифтом высотой не менее 2,5 мм.

Применение машинописных и рукописных символов в одной формуле не допускается.

ПРИМЕЧАНИЯ, ПРИМЕРЫ И СНОСКИ

Примечания используются, если необходимы пояснения или справочные данные к содержанию текста, таблицы или иллюстрации. Их следует помещать непосредственно после материала, к которому они относятся. Примечания не должны содержать требования.

Слово «примечание» пишется с прописной буквы с абзацного отступа и выделяется полужирным стилем шрифта. После слова «примечание» ставится точка и приводится текст примечания, начинающийся с прописной буквы. Текст примечаний допускается печатать только через один интервал. Для записи примечаний целесообразно использовать кегль шрифта

12. Например:

Примечание. Текст примечания.

Одно примечание не нумеруется. Если примечаний несколько, то они записываются со следующей строки и нумеруются арабскими цифрами. Например:

Примечания:

1. Текст первого примечания.

2. Текст второго примечания.

Примеры полезны в тех случаях, когда они могут пояснить приводимый в тексте записки материал или способствуют более краткому его изложению.

Примеры, если они не являются составной частью предложения или абзаца, размещают, оформляют в целом так же, как и примечания, однако кегль шрифта, как правило, не уменьшают.

Необходимые пояснения к тексту документа могут оформляться *сносками.* Шрифт – Times New Roman. Размер шрифта текста сносок – 12 пт.

Абзацный отступ отсутствует. Строки в тексте сносок разделяются полуторным интервалом.

Сноска обозначается цифрой со скобкой, вынесенными на уровень линии верхнего обреза шрифта, например: «печатающее устройство[2)]...» или «бумага[5)]».

Если сноска относится к отдельному слову, знак сноски помещается непосредственно у этого слова, если же к предложению целом, то в конце предложения.

Если необходимо пояснить отдельные данные или данные, относящиеся к началу крупного текстового фрагмента, то их следует помечать надстрочным знаком сноски, используя арабские цифры или знаки «*», причем применение более четырех «*» не допускается.

Текст сноски располагают в конце страницы и отделяют от основного текста страницы короткой тонкой горизонтальной линией с левой стороны.

Как правило, текстовый редактор предоставляет средства для поддержки формирования сносок и работы с ними.

ССЫЛКИ

Ссылки в пояснительной записке могут быть как внешними (относиться к использованным источниками), так и внутренними (ссылаться на части текста самой записки).

На каждый литературный источник в тексте работы обязательно должна быть хотя бы одна ссылка.

Внешняя ссылка представляет собой номер источника по списку использованных источников, заключаемый в квадратные скобки. Использование номера источника без квадратных скобок не допускается. Можно ссылаться сразу на несколько источников. При необходимости, ссылка может быть указана с точностью до страницы в источнике. Например, при ссылке на один источник используется запись вида [21] или [21, с. 10], на несколько – [21, 30, 33-35].

Ссылаться следует на источник в целом или его разделы и приложения. Ссылки на подразделы, пункты, таблицы и иллюстрации не допускаются, за исключением подразделов, пунктов, таблиц и иллюстраций данного документа.

Ссылки на отдельные подразделы, пункты и иллюстрации другого документа не допускаются. Допускаются ссылки внутри документа на пункты, иллюстрации и отдельные подразделы.

При ссылках на стандарты и технические условия указывают только их обозначение, при этом допускается не указывать год их утверждения при условии полного описания стандарта в списке использованных источников в соответствии с ГОСТ 7.1-2003.

Ссылка указывает, что излагаемые положения, факты, рассуждения не принадлежат автору дипломного проекта, а заимствованы им. С другой стороны, применение ссылок придает материалу больший вес и убедительность. Использование заимствованного материала без ссылки на источник недопустимо.

Цитата, включаемая в текст записки, выделяется кавычками и снабжается ссылкой на источник (например: «программное обеспечение – это совокупность программ системы обработки данных и программных документов, необходимых для эксплуатации этих программ» [7, с. 18]). При использовании или описании мнений, суждений других авторов в своей работе необходимо также указывать номер литературного источника и номер страницы, где излагаются используемые материалы, (например, Иванов И. И. под программным обеспечением понимает совокупность программ системы обработки данных и программных документов, необходимых для использования данных программ [7, с. 18]). При ссылке, кроме номера источника, обязательно указывается страница, с которой взята цитата.

Если цитата полностью воспроизводит одно или несколько предложений цитируемого текста, то она начинается с прописной буквы. Если же она органически входит в состав авторского предложения, то начинается со строчной буквы, даже если в источнике использовалась прописная. При

цитировании допускается делать пропуски, обозначая их многоточием, если смысл цитаты не искажается.

При ссылках на составные части и элементы пояснительной записки указывают их номера. Например: «как описано в разд. 1», «см. п. 2.1.5», «как показано на рис. 3.1», «(рис. 3.1)», «в табл. 6.2».

Ссылки на номер формулы дают в круглых скобках, к примеру, «в формуле (6.1)».

Первую ссылку обычно делают по типу (на примере таблицы) «приведены в табл. 6.2» или «(табл. 6.2)». Повторные ссылки дают с сокращенным словом «смотри», например, «см. табл. 6.2». При значительной удаленности такой ссылки целесообразно также указывать номер листа, к примеру, «см. табл. 6.1, с. 57».

4. ПОДГОТОВКА К ЗАЩИТЕ И ЗАЩИТА ВКР

Студент-выпускник несет ответственность за все технические решения, принятые в выпускной квалификационной работе и за ее оформление.

В период подготовки ВКР ему необходимо:

1. Оформить пояснительную записку, а также электронный носитель.

2. Подготовить графические материалы (демонстрационный и раздаточный материал).

3. Заранее проверить соответствие темы работы (диплома) в приказе на кафедре (до буквы).

4. Заранее проверить соответствие данных руководителя работы (диплома) на кафедре.

5. Проверить, все ли зачеты, экзамены, курсовые проставлены в зачетной книжке за весь период обучения, и сдать зачетку в деканат.

6. Взять данные у рецензента для заключения договора на оплату выполненной им работы.

7. Ознакомиться с графиком защит ВКР на кафедре и записаться в аудитории 304.

Далее законченная выпускная квалификационная работа, подписанная студентом и консультантами на титульном листе (Приложения Б), представляется руководителю. После просмотра и одобрения выпускной квалификационной работы научный руководитель подписывает ее и вместе со своим письменным отзывом (Приложение **М**) [9] представляет заведующему кафедрой.

Заведующий кафедрой на основании этих материалов решает вопрос о допуске выпускной квалификационной работы к защите, делая об этом соответствующую запись на титульном листе. В случае, если заведующий кафедрой не считает возможным допускать студента к защите выпускной квалификационной работы, этот вопрос рассматривается на заседании кафедры с участием руководителя. Протокол заседания кафедры представляется через деканат факультета на утверждение проректору по учебной работе УлГТУ.

Выпускные квалификационные работы, выполненные по завершении профессиональных образовательных программ подготовки специалистов и магистров и допущенные заведующими кафедрой к защите, подлежат обязательному рецензированию высококвалифицированным специалистом университета или внешней организации, которых определяет руководство кафедры.

Заведующий выпускающей кафедрой знакомит с рецензией студента-выпускника и направляет выпускную квалификационную работу с рецензией в ГАК для защиты.

В день защиты студенту необходимо иметь:

• пояснительную записку, подписанную дипломником, руководителем проекта, консультантами по соответствующим разделам диплома, рецензентом, заведующим кафедрой;

• графические материалы (презентацию);

• зачетную книжку с отметкой деканата и с допуском руководителя;

• отзыв руководителя;

• рецензию на ВКР с указанием оценки (Приложение Ю) [9];

• акт о внедрении (если таковой имеется);

• раздаточный материал (распечатанный на каждого члена ГАК);

• паспорт;

• сведения о рецензентах и сторонних руководителях для оплаты выполненной ими работы.

Примечания:

1. Фамилия, имя и отчество дипломника на всех документах должны соответствовать паспортным данным. Изменение фамилии должно быть оформлено приказом по университету.

2. Справка для оплаты руководителю дипломного проекта заполняется только для лиц, не работающих преподавателями УлГТУ.

Защита ВКР проводится в соответствии с Положением о Государственных аттестационных комиссиях высших учебных заведений.

Расписание работы ГАК доводится до общего сведения студентов не позднее, чем за месяц до начала защит ВКР.

Защита ВКР производится публично на заседаниях Государственной аттестационной комиссии (ГАК).

Во время защиты председатель или секретарь ГАК объявляет тему ВКР, приводит необходимые сведения о выпускнике и предоставляет ему слово для доклада.

Во время доклада (10-15 мин) студент обосновывает необходимость разработки, доказывает правильность принятых решений и выводов, демонстрирует работу программной системы на компьютере.

После доклада выпускнику задаются вопросы и выслушиваются его ответы.

После этого предоставляется слово руководителю (или зачитывается его заключение), зачитывается рецензия, выпускнику предоставляется возможность ответить на высказанные замечания.

Итоги защит подводятся на закрытом заседании членов ГАК с участием руководителей (консультантов) ВКР. Оценки принимаются большинством голосов членов ГАК, участвующих в заседании. Результаты защиты выпускной работы определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». При оценке работы учитывается качество выполнения и оформления выпускной работы, уровень защиты работы и ответов на вопросы, мнение руководителя.

Затем приглашаются защитившиеся выпускники и другие, присутствовавшие на защите лица. Председатель ГАК объявляет всем присутствующим итоги защит, полученные оценки, отмечает отличившихся студентов и их работы, сообщает о рекомендациях, которые дает комиссия.

5. ПЕРЕЧЕНЬ СТАНДАРТОВ, РЕКОМЕНДУЕМЫХ К ИСПОЛЬЗОВАНИЮ ПРИ ВЫПОЛНЕНИИ ВКР

По текстовым документам:

1. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам.
2. ГОСТ 2.316-2008. Правила нанесения надписей, технических требований и таблиц на графических документах.
3. ГОСТ Р 6.30-2003. Унифицированная система организационно-распорядительной документации. Требования к оформлению документов.
4. ГОСТ 7.1-2003. Библиографическое описание документа. Общие требования и правила составления.
5. ГОСТ 7.9-95. Реферат и аннотация. Общие требования.
6. ГОСТ 7.11-2004. Сокращения слов и словосочетаний на иностранных европейских языках в библиографическом описании.
7. ГОСТ 7.12-93. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила.
8. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Общие требования и правила оформления.
9. ГОСТ 7.54-88. Система стандартов по информации, библиотечному и издательскому делу. Представление численных данных о свойствах веществ и материалов в научно-технических документах. Общие требования
10. ГОСТ 7.64-90. Система стандартов по информации, библиотечному и издательскому делу. Представление дат и времени дня. Общие требования.
11. ГОСТ 7.82-2001. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления.
12. ГОСТ 8.417-2001. Единицы физических величин.

13. ГОСТ 19.106-78. ЕСПД. Требования к программным документам, выполненным печатным способом.

14. ГОСТ 24.301-80. СТД АСУ. Общие требования к выполнению текстовых документов.

15. ОСТ 29.115-88. Оригиналы авторские и текстовые издательские. Общие технические требования.

16. ГОСТ 9327-60. Бумага и изделия из бумаги. Потребительские форматы.

По алгоритмам, программам ЭВМ и программным документам:

1. РД50-34.698-90. ЕКС АС. Требования к содержанию документов.

2. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.

3. ГОСТ 19.106-78. ЕСПД. Требования к программным документам, выполненным печатным способом.

4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.

5. ГОСТ 19.402-78. ЕСПД. Описание программы.

6. ГОСТ 19.502-78. ЕСПД. Описание применения. Требования к содержанию и оформлению.

7. ГОСТ 19.503-79. ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.

8. ГОСТ 19.504-79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

9. ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

10. ГОСТ 19.506-79. ЕСПД. Описание языка. Требования к содержанию и оформлению.

11. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ данных и систем. Условные обозначения и правила выполнения.

12. ГОСТ 24.204-80. СТД АСУ. Требования к содержанию документа «Описание постановки задачи».

13. ГОСТ 24.211-82. СТД АСУ. Требования к содержанию документа «Описание алгоритма».

14. ГОСТ 34.201-89. ЕКС АС. Виды, комплектность и обозначение документов при создании АС.

15. ГОСТ 34.602-89. ЕКС АС. Техническое задание на создание АС.

По иллюстрациям:

16. ГОСТ 2.051-2006. ЕСКД. Электронные документы. Общие положения.

17. ГОСТ 2.601-2006. ЕСКД. Эксплуатационные документы.

18. ГОСТ 2.605-68. ЕСКД. Плакаты учебно-технические. Общие технические требования.

19. Р 50-77-88. ЕСКД. Правила выполнения диаграмм.

По чертежам:

1. ГОСТ 2.004-88. ЕСКД. Общие требования к выполнению конструкторских и технологических документов на печатающих и графических устройствах вывода ЭВМ.
2. ГОСТ 2.051-2006. ЕСКД. Электронные документы. Общие положения.
3. ГОСТ 2.052-2006. ЕСКД. Электронная модель изделия. Общие положения.
4. ГОСТ 2.053-2006. ЕСКД. Электронная структура изделия. Общие положения.
5. ГОСТ 2.104-2006. ЕСКД. Основные надписи.
6. ГОСТ 2.109-73. ЕСКД. Основные требования к чертежам.
7. ГОСТ 2.301-68. ЕСКД. Форматы.
8. ГОСТ 2.302-68. ЕСКД. Масштабы.
9. ГОСТ 2.303-68. ЕСКД. Линии.

10. ГОСТ 2.304-81. ЕСКД. Шрифты чертежные.
11. ГОСТ 2.306-68. ЕСКД. Обозначения графические материалов и правила их нанесения на чертежах.
12. ГОСТ 2.307-68. ЕСКД. Нанесение размеров и предельных отклонений.
13. ГОСТ 2.316-2008. ЕСКД. Правила нанесения надписей, технических требований и таблиц на графических документах.
14. ГОСТ 2.317-69. ЕСКД. Аксонометрические проекции.
15. ГОСТ 2.321-84. ЕСКД. Обозначения буквенные.
16. ГОСТ 2.413-72. ЕСКД. Правила выполнения конструкторской документации изделий, изготовляемых с применением электрического монтажа.
17. ГОСТ 2.417-91. ЕСКД. Правила выполнения чертежей печатных плат.
18. ГОСТ 24.304-82. СТД АСУ. Требования к выполнению чертежей.

СПИСОК ЛИТЕРАТУРЫ

1. Приказ Министерства образования и науки РФ от 19 сентября 2017 г. N 918 "Об утверждении федерального государственного образовательного стандарта высшего образования - магистратура по направлению подготовки 09.04.01 Информатика и вычислительная техника" (с изменениями и дополнениями) Редакция с изменениями N 1456 от 26.11.2020
2. Положение о порядке подготовки и защиты выпускных квалификационных работ по образовательным программам высшего образования (бакалавриат, специалитет, магистратура). Утверждено ректором УлГТУ, Ульяновск 24.02.21
3. Правила оформления рукописей для издания в УлГТУ : Основные положения / составитель М. В. Теленкова. – 4-е изд. исправ. и доп. – Ульяновск : УлГТУ, 2011. – 47 с.
4. Общие требования к выполнению выпускной квалифика- ционной работы студентов по кафедре «Вычислительная техника» : методические указания / сост. В. Н. Арефьев. – Ульяновск : УлГТУ, 2012. – 82 с.

Приложение А
(*обязательное*)
Пример оформления заявления

Заведующему кафедрой
«Вычислительная техника»
Святову К.В.
Студента _____
(номер курса прописью)
курса ФИСТ группы
_____
(название группы)
_____
(Фамилия Имя Отчество)
_____
_____

ЗАЯВЛЕНИЕ

Прошу утвердить тему выпускной квалификационной работы бакалавра (магистра) _____
(НАЗВАНИЕ ТЕМЫ)

_____

_____

Руководитель проекта: _____
(Фамилия, Имя, Отчество, должность)

Личная подпись
руководителя проекта

Личная подпись студента

_____ _____

_____ _____
(Дата) (Дата)

Приложение Б

(*обязательное*)

Пример оформления титульного листа ПЗ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет_____Информационных систем и технологий
Кафедра _____Вычислительная техника

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. кафедрой

_____ /____Святов К. В.____/

подпись инициалы, фамилия
«___»_____ 2021 г.

_____Магистерская диссертация_____
Вид ВКР (дипломный проект (работа) / бакалаврская работа / магистерская диссертация)

Тема «Разработка библиотеки шаблонов автоматных моделей для решения прикладных задач»___
_____

Обучающийся _____ /_____Горшков Л.И. _____/
подпись инициалы, фамилия
Обозначение ВКР ___МД-УлГТУ-09.04.01-17/445-2021____ Группа ИВТИСмд-21
для технических направлений подготовки (специальностей)

Направление подготовки (специальность) 09.04.01 профиль «Информатика и вычислительная техника»
_____

Руководитель ВКР _____ /___Лылова А.В.___ /
подпись, дата инициалы, фамилия

Ульяновск 2021г.

Приложение В
(*обязательное*)
Пример оформления задания на ВКР

МИНИСТЕРСТВО НАУКИ И ВЫШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет *ИСТ* Кафедра *Вычислительная техника*

09.04.01 «Информатика и вычислительная техника»
(шифр и наименование направления)

УТВЕРЖДАЮ
Заведующий кафедрой
«Вычислительная техника»
(наименование кафедры)
к.т.н., доцент *Святов К.В.*
(ученая степень, ученое звание, Ф.И.О.)

(подпись)

«_____

ЗАДАНИЕ
на выполнение *магистерской диссертации*
(наименование выпускной квалификационной работы)

Обучающемуся *Горшкову Даниилу Александровичу* курса 2 группы ***ИВТИСмд-21***
очной формы обучения

Тема ***«Автоматизация тестирования алгоритмических процессов»***

утверждена приказом по Университету от «_28_»_____12____2020г. № 1185

Срок сдачи студентом *магистерской диссертации* «_10_»__06_2020г.
(наименование ВКР)

* Название темы указывается в точном соответствии с приказом.

Исходные данные:

Подсистема должна:

- *описывать прецеденты (методы и данные) по средствам работы как одного агента, так и взаимодействия между собой группы агентов;*
- *поддерживать изменение окружающего мира;*
- *выводить данные о прецеденте, за который отвечает запущенный агент.*

Перечень вопросов, подлежащих разработке, или краткое содержание

магистерской диссертации
(наименование ВКР)

- *техническое задание;*
- *проектирование системы;*
- *реализация системы;*

Перечень иллюстративного материала:

Общая структура системы, диаграмма вариантов использования, диаграмма последовательности, таблицы для представления прецедентов, экранные формы системы, алгоритм работы системы и межагентного взаимодействия, агентная платформа JADE, пример работы агента с ре- альным прецедентом.

Календарный график работы над ВКР на весь период

№ этапа	Содержание этапа	Срок выполнения
1	Получение задания на магистерскую диссертацию. Проработка задания	
2	Написание рукописи пояснительной записки	
3	Отработка графической части проекта	
4	Оформление, согласование и утверждение проекта	
5	Предзащита проекта	

Консультанты по работе, с указанием относящихся к ним разделов проекта

Раздел	Консультант	Подпись, дата
		Задание выдал

Дата выдачи задания «01» февраля 2020г.

Руководитель д.т.н., профессор *Негода В.Н.* _____
(ученая степень, ученое звание, Ф.И.О.)

Задание принял к исполнению «01» февраля 2020г.

Подпись обучающегося _____

Приложение Г

(*обязательное*)

Пример оформления этикетки для ПЗ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

(вид выпускной квалификационной работы)

«Автоматизация тестирования алгоритмических процессов»

Напрввление 09.04.01
Обучающийся: Горшков Д.А.
Группа: ИВТИСмд-21
Руководитель: Негода В.Н.

УЛЬЯНОВСК 2021

Приложение Д
(*справочное*)
Пример оформления раздела «Содержание»

СОДЕРЖАНИЕ

Типовое оглавление магистерской ВКР по созданию средств автоматизации строится на основе конкретного примера – разработке темы «Автоматизация процесса сбора и обработки эпидемиологических данных». Для многих ВКР магистрантов кафедры ВТ значительная часть заголовков может быть получена путем замены словосочетания «процесса сбора и обработки эпидемиологических данных» на такое, которое отражает содержание автоматизируемого в ВКР процесса.

2.3. Разработка логико-алгебраической модели процессов сбора и обработки эпидемиологических данных

2.4. Разработка алгоритмических моделей сбора и обработки эпидемиологических данных

3. Проектирование средств сбора и обработки эпидемиологических данных

3.1. Анализ требований к средствам сбора и обработки эпидемиологических данных

3.2. Разработка спецификаций, представляющих эпидемиологические данные

3.3. Разработка структурно-функциональной организации средств сбора и обработки эпидемиологических данных

3.3.1. Разработка диаграммы компонентов

3.3.2. Разработка диаграммы классов

3.4. Проектирование аспектов поведения средств сбора и обработки эпидемиологических данных и их пользователей

3.4.1. Разработка диаграммы деятельности

3.4.2. Разработка диаграммы последовательности

3.5. Прототипирование средств сбора и обработки эпидемиологических данных

4. Экспериментальное исследование созданных средств

4.1. Формулировка замыслов экспериментов

4.2. Планирование экспериментов

4.3. Создание средств поддержки реализации планов экспериментов

4.4. Экспериментирование и анализ результатов

Приложение Ж
(*справочное*)
Пример оформления аннотации в ПЗ АННОТАЦИЯ

Магистерская диссертация Лапшова Юрия Александровича на тему: *«Комплекс средств управления прерываниями в корпоративных средах коллективного проектирования».* Кафедра ВТ УлГТУ, 2011г.
Научный руководитель д.т.н., профессор Соснин П.И.

Диссертация содержит 151 страницу, включающую 44 иллюстрации, 16 таблиц, 2 приложения и список информационных источников из 102 наименований.

Ключевые слова: человеческие прерывания, коллективное проектирование, управление прерываниями, прерывания интеллектуального процессора, псевдокодовые методики.

В диссертации представлены результаты исследований и разработок комплекса средств для управления прерываниями в человеко-компьютерном взаимодействии в процессе работы в средах коллективного проектирования.

Для организации наиболее рационального процесса проектирования разработан метод управления прерываниями, основанный на использовании псевдокодового интерпретатора методик и вопросно-ответных протоколов задач, существенно упрощающий процесс разработки проекта.

Предусмотрена его практическая реализация в вопросно-ответной среде WIQA.Net.

Приложение К
(*справочное*)
Пример оформления заключения в ПЗ

ЗАКЛЮЧЕНИЕ

В данной работе были проведены предпроектные исследования в рамках разработки системы распознавания речи. Во время проведения аналитического обзора произведен анализ требований к средствам распознавания речи, так же было выяснено, что ни одна из уже существующих систем распознавания речи не удовлетворяет предъявленным требованиям.

В ходе аналитического обзора произведен предварительный выбор алгоритмов распознавания речи и была разработана структурно-функциональная схема распознавания речи. После чего был произведен выбор методов и средств для дальнейшего макетирования.

В ходе выполнения работы произведено макетирования ряда алгоритмов, выделение речевых признаков, разработана дискретная скрытая марковская модель для подсистемы синтеза лексических единиц, реализован алгоритм синтеза лексических единиц из последовательности базисных речевых единиц, произведена оценка влияния типа базисной речевой единицы на качество распознавания, созданы средства поддержки разработки скрытых марковских моделей для словарей произвольной длины и так далее.

Итогом работы является качественный задел для магистерской диссертации по тематике распознавания речи. По данной работе были опубликованы две статьи [62], [63].

Приложение Л
(*обязательное*)
Примеры библиографического описания для списка литературы

Описание книги:

с одним, двумя или тремя авторами

1. Платонов, С. Ф. Лекции по русской истории / С. Ф. Платонов. – СПб. : Кристалл, 1998. – 838 с.
2. Алтунин, А. Е. Модели и алгоритмы принятия решений в нечетких условиях : монография / А. Е. Алтунин, М. В. Семухин. – Тюмень: Издательство Тюменского государственного университета, 2000. – 352 с.
3. Леонтьев, К. Б. Закон «Об авторском праве и смежных правах» в схемах / К. Б. Леонтьев, О. В. Сенотов, В. В. Терлецкий. – М. Бератор пресс, 2003. – 139 с.

с четырьмя и более авторами

1. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г. К. Вороновкий [и др.]. – Харьков: Основа, 1997. – 112 с.

Описание учебника, учебного пособия, методических указаний:

1. Мухин, В. И. Исследования систем: учебник. – М. : Экзамен, 2006. – 479 с.
2. Курганов, С. А. Анализ установившихся режимов линейных электрических цепей методом схемных определителей: учебное пособие / С. А. Курганов, В. В. Филаретов. – Ульяновск: УлГТУ, 2002. – 148 с.
3. Основы теории управления: метод. указания к практ. занятиям / сост. Арефьев В. Н. – Ульяновск : УлГТУ, 2000. – 79 с.

Описание многотомного издания:

документ в целом

1. Всемирная история экономической мысли: в 6 т. / Моск. гос. ун-т им.М. В. Ломоносова ; редкол. : В. Н. Черковец (гл. ред.) [и др.]. – М.: Мысль, 1987. – 6 т.

отдельный том

1. Бондарев, Б. В. Курс общей физики: учебное пособие для студен- тов высш. техн. учеб. заведений : в 3 кн. / Б. В. Бондарев, Н. П. Калашни- ков, Г. Г. Спирин. – М.: Высшая школа, 2003. Кн. 1: Механика. – 2003. – 352 с.

2. Дарвин, Ч. Сочинения. В 12 т. Т. 3. О происхождении видов путем естественного отбора или сохранении благоприятствуемых пород в борьбе за жизнь / Ч. Дарвин. – М.:Л. : Изд-во АН СССР, 1939. – 832 с.Описание стандартов:

ГОСТ 7.53–2001. Издания. Международная стандартная нумерация книг. – М.: Изд-во стандартов, 2002. – 3 с.

Описание диссертации:

1. Вишняков, И. В. Модели и методы оценки коммерческих банков в условиях неопределенности: дис. канд. экон. наук: 08.00.13: защищена

12.2.2 : утв. 24.06.02 / Вишняков Илья Владимирович. – М., 2002. – 234 с.

Описание отчета о научно-исследовательской работе:

12.2.2.1. Формирование генетической структуры стада: отчет о НИР (про- межуточ.): 42-44 / Всерос. науч.-исслед. ин-т животноводства; рук. Попов В. А. – М., 2001. – 75 с. – Исполн.: Алешин Г. П., Ковалева И. В., Латышев Н. К., Рыбакова Е. И. Стриженко А.А. – № ГР 01840051145. – Инв. № 04534333943.

Описание статьи (главы) из книги или другого разового издания:

1. Вавинов, С. В. Параметризованный жадный алгоритм построения статических расписаний / С. В. Вавинов, В. А. Костенко // Методы и средства обработки информации: тр. всерос. науч. конф., Москва, 1-3 октября 2003 г. – М. : Издательский отдел факультета ВМиК МГУ, 2003. – С. 323-328.

Описание статьи из периодического или продолжающегося издания:

1. Крюков, В. А. Разработка параллельных программ для вычислительных кластеров и сетей / В. А. Крюков // Информационные технологии и вычислительные системы. – 2003. – № 1-2. – С. 42-61.

Описание изданий на иностранном языке:

1. Kitainik, L. Fuzzy Decision Procedures with Binary Relations. Towards a Unified Theory / L. Kitainik. – Boston : Kluwer, 1993. – 254 pp.
2. Ernst, R. Hardware-Software Cosynthesis for Micro-Controllers / R. Ernst, J. Henkel, Th. Benner // IEEE Design & Test Magazine. – 1993. – Vol. 10. – № 4. – pp. 64-75.

Описание электронного ресурса:

ресурс локального доступа

1. Цветков, В. Я. Компьютерная графика: рабочая программа [Электронный ресурс]: для студентов заоч. формы обучения геодез. и др. специальностей. – Электрон. дан. и прогр. – М.: МИИГАиК, 1999. – 1 дискета. –№ гос.регистрации 0329900020.

ресурс удаленного доступа

1. Костенко, В. А. Синтез структур вычислительных систем реального времени с использованием генетических алгоритмов [Электронный ресурс] / В. А. Костенко, Р. Л. Смелянский, А. Г. Трекин // Программирование. – 2000. – № 5. – Режим доступа: http://lvk.cs.msu.su/

Приложение М
(*справочное*)
Шаблон отзыва руководителя на ВКР

МИНИСТЕРСТВО НАУКИ И ВЫШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет *ИСТ* Кафедра *Вычислительная техника*

ОТЗЫВ РУКОВОДИТЕЛЯ

на _____
(наименование выпускной квалификационной работы ВКР)
обучающегося _____ группы _____
(фамилия, имя, отчество)

Тема ВКР _____
_____

На отзыв представлено___ листов чертежей и пояснительная записка на __листах

Актуальность и новизна ВКР

Оценка содержания и хода выполнения ВКР

Положительные стороны ВКР

Замечания к ВКР

Рекомендации по внедрению

Рекомендуемая оценка ВКР

Дополнительная информация для ГЭК

Уровень освоения компетенций _____
Высокий, средний, низкий

Руководитель _____/ _____/
должность, ученая степень,учёное звание подпись инициалы фамилия
«_____» _____20 г.

Приложение Н

(*справочное*)

Шаблон оформления рецензии на ВКР

МИНИСТЕРСТВО НАУКИ И ВЫШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра «Вычислительная техника»

РЕЦЕНЗИЯ

на_____
(наименование выпускной квалификационной работы ВКР)

Направление: 09.04.01 «Информатика и вычислительная техника»

обучающегося_____группы_____
(фамилия, имя, отчество)

Выполненную на тему:_____

Актуальность и новизна ВКР:

Оценка содержания ВКР

Положительные стороны ВКР

Замечания ВКР

Практическая значимость (рекомендована к внедрению, внедрена, находиться на стадии опытной эксплуатации) подтверждается актом о внедрении_____

Рекомендуемая оценка ВКР

_____

Дополнительная информация для ГЭК_____
(по внедрению, развитию и т.д.)

_____

Уровень освоения компетенций_____
Высокий, средний, низкий

Рецензент_____/ _____/
должность, ученая степень,учёное звание подпись инициалы фамилия

«____»_____20 г.

Учебное издание

Требования к разработке, оформлению и защите выпускных квалификационных работ

Методические указания

Составитель БЕЛЯЕВА Ирина Владимировна

Редактор

Подписано в печать Формат 60×84/16.
Усл. печ. л. 4,88. Тираж 50 экз. Заказ .

Ульяновский государственный технический университет,
432027, г. Ульяновск, ул. Сев. Венец, д. 32.

Типография УлГТУ, 432027, г. Ульяновск, ул.Сев. Венец, д. 32.

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
ФТД.01-Психология и педагогика высшей школы

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

ПЕДАГОГИКА И ПСИХОЛОГИЯ ВЫСШЕЙ ШКОЛЫ

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ
к проведению лекционных занятий

в соответствии с учебным планом по направлению подготовки (специальности)	09.04.03 Прикладная информатика
профиль (программа / специализация)	Искусственный интеллект и бизнес-аналитика

Основные вопросы, освещаемые на лекциях

Раздел, тема учебной дисциплины (модуля), содержание темы
Раздел 1. Психология высшей школы
Тема 1. Предмет и задачи курса «психология высшей школы». Предмет, цели, задачи, функции психологии высшей школы. Место дисциплины в системе наук. Становление и перспективы развития.
Тема 2. Особенности развития личности студента. Психолого-педагогические особенности одаренных студентов. Социализация личности студента. Адаптация личности студента, ее трудности и последствия. Адаптация к учебной деятельности в вузе.
Тема 3. Профессиональное становление. Факторы профессионального становления. Противоречия профессионального становления. Стадии и кризисы профессионального становления.
Тема 4. Лидерство в организации. Феномен лидерства. Психологическое содержание понятия «лидерство». Стили лидерства. Лидерство и руководство. Гендерные аспекты организационного руководства и лидерства. Методика формирования команды. Организация межличностных, групповых и организационных коммуникаций.
Раздел 2. Педагогика высшей школы
Тема 5. Педагогика высшей школы. Предмет, задачи, категории педагогики высшей школы. Принципы и методы педагогического исследования.
Тема 6. Приоритетные стратегии и тенденции развития высшего образования. Современные стратегии модернизации высшего образования в России и за рубежом.
Тема 7, 8. Формы организации обучения в вузе: традиции и инновации. Трехмерная модель систематики форм организации обучения. Вузовская лекция. Игры. Семинары и конференции. Самостоятельная работа студентов. Проектно-творческая деятельность. Дистанционное обучение. Авторские технологии обучения. Научно-исследовательская работа студентов. УИР как часть профессиональной подготовки студентов. Формы организации НИР в вузе. Защита интеллектуальной собственности.

Раздел 1. Психология высшей школы

Тема 1. Предмет и задачи курса «Психология высшей школы».

1.1. Предмет, цели, задачи, функции психологии высшей школы.

1.2. Место дисциплины в системе наук.

1.3. Становление и перспективы развития.

1.1. Предмет, цели, задачи, функции психологии высшей школы.

Концепция высшего образования в современной России предполагает, что человек, гражданин должен стремиться повышать свой образовательный уровень постоянно, хорошо разбираться в различных профессиональных сферах, людях, их отношениях. Самообразование и мотивация на достижение успеха - вот главные ценности профессионала нынешнего времени. Трудно выстраивать комфортные взаимоотношения с людьми в повседневной жизни, а также налаживать профессиональные коммуникативные каналы с коллегами, клиентами или партнерами, не обладая минимумом научно достоверных сведений по психологии, а руководствуясь лишь житейскими, обывательскими представлениями.

Предметом психологии высшей школы выступают психологические закономерности и условия эффективности процессов обучения и воспитания в высшей школе. В этом совокупном предмете можно выделить ряд частных предметов изучения и отдельных проблем: психологическая структура учебной деятельности в высшей школе; формирование и функционирование познавательных процессов учащихся (профессионального восприятия, мышления, памяти, внимания), профессиональных способностей, черт личности, умений и навыков; возрастные психические и психофизиологические особенности студентов; дифференциально-психологические характеристики учащихся, которые необходимо учитывать в процессе обучения и воспитания, и методы их диагностики; социально-психологические закономерности формирования студенческого и преподавательского коллектива; психологические аспекты педагогического общения; психологические конфликты в студенческой среде и между студентом и преподавателем (виды и способы их разрешения) и др.

Психология высшей школы так же изучает место и роль психических процессов, состояний, свойств, опыта, их проявление, развитие и функционирование в деятельности студентов, преподавателей и руководителей вузов.

Психологию высшей школы интересуют прежде всего функционирование, изменение, развитие и формирование психики студентов, психологические особенности деятельности преподавателей.

Цель изучения «Психология высшей школы»: овладение методологическими основами и инструментариями психологии высшей школы, повышение образованности молодых специалистов в вопросах психологии, а также формирование умений анализировать основные достижения и тенденции развития психологии высшей школы и использовать их в своей практической и профессиональной деятельности.

Основные задачи:

• ознакомление с основами психологической науки, ее возможностями в успешном решении проблем жизни и профессиональной деятельности, возникающих перед каждым человеком и человеческими общностями;

• достижение научного понимания студентами основ психологической и педагогической реальностей, их проявлений и влияний в жизни и деятельности людей;

• раскрытие роли и возможностей психологии в самореализации и самоутверждении человека;

• ознакомление с психологическими и педагогическими основами деятельности в условиях современного российского образовательного и профессионального пространства,

способствование развитию у них элементов государственного мышления и активной гражданской позиции;

• психологическая подготовка студентов к предстоящей профессиональной деятельности;

• содействие гуманитарному развитию студентов, их психологического мышления, наблюдательности, культуры их отношения к людям, общения и поведения;

• выработка навыков по использованию методов психологии в повышении студентами личной образованности, воспитанности, в освоении учебных программ, повышении профессионального мастерства, овладении различными психологическими техниками;

• формирование устойчивого интереса к продолжению работы по повышению своей психологической подготовленности в рамках повседневной и профессиональной деятельности.

Функции «Психологии высшей школы» как учебной дисциплины:

– Образовательно-мировоззренческая – постоянное расширение знаний студентов о человеке, об обществе, о себе, собственных мотивах, особенностях и ориентирах для будущей деятельности. Благодаря реализации этой функции общежитейский опыт дополняется и расширяется научно достоверными, систематизированными знаниями, которые помогают более четко сформировать взгляды на жизнь, убеждения, надежные опоры на жизненном пути;

– Воспитательно-мобилизующая – выражается в мощном вкладе, который вносят психология высшей школы в процесс гуманизации студента и преподавателя ВУЗа. Т.е. благодаря этому люди по-другому, глубже и обстоятельнее, начинают воспринимать других людей, студентов и преподавателей как своего, так и других ВУЗов. Также происходит более достоверная оценка себя, своих достоинств и недостатков, понимание новых возможностей и побуждения к самосовершенствованию. Формируется постоянное стремление к самовоспитанию, самообразованию и в конечном счете к самосовершенствованию.

– Жизненно-практическая – заключается в определённой установке на использование многих конкретных знаний, принципов, методик и рекомендаций психологии в своей жизни, в образовательном учреждении, в семье, среди людей, на досуге, в трудных ситуациях и пр.; в стремлении преподавателя не просто четко реализовывать свои профессиональные обязанности, но выстроить эффективные коммуникативные каналы со студенческим обществом и воспринимать каждого студента как значимую личность. Все это позволит значительно уменьшить число промахов, неудач, конфликтов, связанных с профессиональной деятельностью и повысить успешность при самоутверждении и самореализации;

– Профессионально-прикладная – связана с наполнением специалиста психологическими знаниями, навыками и умениями, нужными для профессиональной деятельности, так как любая такая деятельность связана с контактами в социальной сфере на разных уровнях;

– Развивающая – всё вышесказанное о функциях, уже свидетельствует о широком развивающем влиянии изучения психологии высшей школы, как на личность студента, так и на профессиональные навыки преподавателя. Усвоение знаний, выработка практики их применения в образовательном процессе неразрывно связано с элементами такого специфического качества, как психологическое и педагогическое мышление. Содержание, формы и методы преподавания учебной дисциплины предусматривают также целенаправленное профессиональное развитие отдельных качеств: наблюдательности, памяти, внимания и др.

Изучение психологии высшей школы строится эффективно, если реализуются все функции.

1.2. Место дисциплины в системе наук.

Связь психологии высшей школы с психологией очевидна. Психология - наука о закономерностях, механизме и фактах психической жизни человека. Психологи изучают устойчивое и повторяющееся в индивидуальном поведении. В центре внимания - проблемы восприятия, памяти, мышления, обучения и развития человеческой личности, усвоения нового знания на различных этапах возрастного развития. И если на начальных этапах взросления достаточно было возрастной психологии, то на этапе вузовского и поствузовского обучения центральную роль начинает играть психология ВУЗа. Именно изучение проблем обучения, воспитания, усвоения нового знания выработка новых навыком на основе этого знания и применение их в своей профессиональной деятельности в рамках ВУЗа позволило выделиться психологии высшей школы как самостоятельной дисциплине.

Так же психология высшей школы связана с социологией, как с учением о человеческом поведении, оформленном в групповую жизнь, включающую, с одной стороны, коллективные действия, мотивы и поступки, а с другой - способы, какими люди придают значение своему жизненному опыту (саморефлексия). Если говорить о процессе обучения в ВУЗе, то здесь важен не столько студент как отдельный индивид, но вся совокупность студенческих коллективов, их взаимосвязь как друг с другом, так и преподавательским коллективом. Именно студенческий коллектив и его взаимосвязь с преподавателями и является предметом рассмотрения Психологии высшей школы.

Трудно переоценить для психологии высшей школы значение тесных связей с философией и естествознанием. Философия обосновывает общие подходы к пониманию человека, раскрывает понятие личности студента, причины и цели его активности, уровни детерминации поведения. Философский анализ взаимосвязи человека и общества, лежащий в основе этических концепций, раскрывается психологией в правилах и нормах воспитания. Естествознание показывает, какие объективные параметры лежат в основе психолого-педагогических закономерностей, например, как связана динамика работы нервной системы с индивидуальными особенностями.

Психология высшей школы и педагогика - две науки, изучающие процесс обучения в целом, включая и Вузовское обучение, в неразрывном единстве. Такое тесное сотрудничество происходит в рамках действия такой перспективной отрасли научного знания как педагогическая психология. Она изучает условия и закономерности формирования психических процессов под воздействием образования и обучения. Педагогическая психология стала сферой совместного изучения взаимосвязей между воспитанием, обучением и развитием подрастающих поколений на различных этапах взросления (Б.Г. Ананьев). Например, одной из многих важнейших педагогических проблем является проблема несоответствия ожиданий преподавателей усвоения учениками (студентами) нового материала с теми профессиональными навыками, которыми они оперируют в своей профессиональной деятельности. Проще говоря, почему учебный материал не был усвоен или был усвоен неправильно. В связи с этой проблемой складывается предмет педагогической психологии, исследующей закономерности усвоения, учения. На основе сложившихся научных представлений формируются техника, практика учебно-педагогической деятельности, обоснованные со стороны психологии закономерностей процессов усвоения. Вторая педагогическая проблема возникает, когда осознается различие обучения и развития в образовательной системе. Предмет исследования в этом случае - закономерности развития интеллекта, личности, способностей, вообще человека. Данное направление педагогической психологии разрабатывает практику не обучения, а организации развития. Еще одной проблемой становится проблема взаимоотношения преподавателя и студента, а также роль педагогического коллектива в воспитании студента. Здесь на первый план выходят вопросы воспитательной деятельности ВУЗа, государственной политики образования, а также личная мотивация педагога и студента в рамках самовоспитания и саморазвития.

1.3. Становление и перспективы развития

Становление «Психологии высшей школы» как научной дисциплины неразрывно связано с генезисом таких наук как психология и педагогика. Становление педагогической научной мысли, неразрывно связана с деятельностью таких ученых как Яна Амоса Коменского, Жан-Жака Руссо, Иоганна Песталоцци, Иоганна Гербарта, Адольфа Дистервега, К.Д. Ушинского, П.Ф. Каптерева, чьи труды положили начало развитию педагогической теории и целенаправленной организации обучения как процесса. Сама же Психология высшей школы начала формироваться как самостоятельная наука в рамках педагогической психологии только в конце XIX в.

Первоначальный этап становления науки был посвящен таким вопросам как: связь развития, обучения и воспитания; творческая активность обучаемого, способности ученика и их развитие, роль личности преподавателя, организация обучения и многие другие. Однако это были только первые попытки научного осмысления этого процесса. Хотя уже и здесь в достаточной мере раскрываются такие проблемы как соотношение творчества, обучения и саморазвития (Песталоцци).

Так же анализируются вопросы методологического единства обучения и воспитания, роли преподавателя в этом процессе. Так Дистервегу принадлежит тезис о главенствующей роли педагога, учителя в образовательном процессе. Он рассматривает учебный процесс как единство ученика - обучаемого субъекта, учителя, изучаемого предмета и условий обучения.

Огромен вклад в разработку основ психологии высшей школы П.Ф. Каптерева. В его трудах были рассмотрены педагогические проблемы учительского труда и подготовки учителя, проблемы эстетического развития и воспитания и многие другие. Образовательный процесс, по мнению автора, представляет собой «выражение внутренней самодеятельности человеческого организма», развитие способностей и др.

Существенен вклад в становление педагогической психологии и психологии высшей школы С.Т. Шацкого (1878-1934), разработавшего целостную концепцию гуманизации и демократизации воспитания в процессе социализации человека. Ему принадлежит авторство модели педагога, в которой соединены обобщенные требования к его личности и профессиональной компетентности как к субъекту социально-педагогической деятельности.

Следующий этап (конец XIX в. - середина XX в.) проходит на фоне интенсивного развития экспериментальной психологии, создания и разработки конкретных педагогических систем.

Начало этого неразрывно связано с идеями, представленными в работах А.П. Нечаева, А. Бине и Б. Анри, М. Оффнера, Э. Меймана, В.А. Лайя, в исследованиях Г. Эббингауза, Ж. Пиаже, А. Валлона, Дж. Дьюи, С. Фрэне, Э. Клапереда. Экспериментальное изучение особенностей поведения научения (Дж. Уотсон, Э. Толмен, Э. Газри, К. Халл, Б. Скиннер), развития детской речи (Ж. Пиаже, Л.С. Выготский, П.П. Блонский, Ш. и К. Бюлер, В. Штерн и др.), развитие специальных педагогических систем Вальдорфской школы, школы М. Монтессори.

Особое значение имеет развитие, начиная с работ Ф. Гальтона, тестовой психологии, психодиагностики. Благодаря исследованиям А. Бине, Б. Анри, Т. Симона во Франции и Дж. Кэттелла в Америке это позволило найти действенный механизм (при взаимодействии тестов достижения и тестов способностей) не только контроля знаний и умений обучающихся, но и управления подготовкой учебных программ, учебным процессом в целом. Как отмечает М.В. Гамезо, в этот период в Европе образовался ряд лабораторий при школах. Так, в Германии возникла лаборатория Э. Меймана, в которой для решения учебных и воспитательных задач использовались приборы и методики, созданные в лабораториях университетов. В 1907 г. Мейман публикует книгу «Лекции по экспериментальной психологии», где дает обзор работ по экспериментальной дидактике. В Англии вопросами экспериментального изучения типологических особенностей

школьников занимался известный детский психолог Дж. Селли, который в 1898 г. опубликовал работу «Очерки по психологии детства». Во Франции А. Бине основал при одной из школ Парижа экспериментальную детскую лабораторию. В лаборатории изучались физические и душевные способности ребенка, а также методы преподавания учебных дисциплин.

О самостоятельности психологии высшей школы как науки, свидетельствует не только использование тестовой психодиагностики, исследовательских программ на основе ВУЗов, экспериментально-педагогических систем и программ, но и попытки научной рефлексии образовательного процесса, его строгого теоретического осмысления.

Основанием для выделения третьего этапа развития педагогической психологии и психологии высшей школы служит создание психологических теорий обучения, т.е. разработка теоретических основ педагогической психологии. Так, в 1954 г. Б. Скиннер выдвинул идею программированного обучения, а в 60-х годах Л.Н. Ланда сформулировал теорию его алгоритмизации. Затем В. Оконь, М.И. Махмутов построили целостную систему проблемного обучения. Это, с одной стороны, продолжило разработку системы Дж. Дьюи, полагавшего, что обучение должно идти через решение проблем, а с другой - соотносилось с положениями О. Зельца, К. Дункера, С.Л. Рубинштейна, А.М. Матюшкина и др. о проблемном характере мышления, его фазности, о природе возникновения каждой мысли в проблемной ситуации (П.Г. Блонский, С.Л. Рубинштейн). В 50-е годы появились первые публикации П.Я. Гальперина и затем Н.Ф. Талызиной, в которых излагались исходные позиции теории поэтапного формирования умственных действий, впитавшей в себя основные достижения и перспективы педагогической психологии. В это же время разрабатывается теория развивающего обучения, описанная в работах Д.Б. Эльконина, В.В. Давыдова на основе общей теории учебной деятельности (сформулированной этими же учеными и развиваемой А.К. Марковой, И.И. Ильясовым, Л.И. Айдаровой, В.В. Рубцовым и др.). Развивающее обучение нашло свое отражение и в экспериментальной системе Л.В. Занкова.

В этот же период С.Л. Рубинштейн в «Основах психологии» дал развернутую характеристику учения как усвоения знаний. Усвоение с разных позиций детально разрабатывалось далее Л.Б. Ительсоном, Е.Н. Кабановой-Меллер и др., а также в работах Н.А. Менчинской и Д.Н. Богоявленского (в рамках концепции экстериоризации знаний). Появившаяся в 1970 г. книга И. Лингарта «Процесс и структура человеческого учения» и в 1986 г. книга И.И. Ильясова «Структура процесса учения» позволили сделать широкие теоретические обобщения в этой области.

Заслуживает внимания возникновение принципиально нового направления - суггестопедии, суггестологии Г.К. Лозанова (60-70-е годы). Его основой является управление педагогом неосознаваемыми обучающимися психическими процессами восприятия, памяти с использованием эффекта гипермнезии и суггестии. В дальнейшем был разработан метод активизации резервных возможностей личности (Г.А. Китайгородская), группового сплочения, групповой динамики в процессе такого обучения (А.В. Петровский, Л.А. Карпенко).

В настоящее время психология высшей школы стала неотъемлемой частью новой концепции непрерывного обучения, существующей в рамках Болонского процесса и активно внедряющегося в Российское образование. Именно изучение психологических основ поведения студентов, их мотивации и особенности взаимодействия с преподавателем способствует достижению основных целей непрерывного образования, стремления к постоянному самообучению и самосовершенствованию, а также усвоению основных компетенций и использованию навыков, полученных как в ВУЗе, так в процессе поствузовского образования.

Вопросы для повторения:

1. Каковы предмет, цели и задачи «Психологии высшей школы»?

2. С какими науками психология высшей школы связана наиболее тесно?

3. Какое влияние на становление и развитие психологии высшей школы оказало становление и развитие педагогической психологии?

4. Каковы перспективы развития психологии высшей школы в современном образовательном пространстве?

5. Почему необходимо изучать психологию высшей школы в ВУЗе?

Тема 2. Особенности развития личности студента.

2.1. Социализация личности студента.

2.2. Психолого-педагогические особенности одаренных студентов.

2.3. Адаптация личности студента, ее трудности и последствия. Адаптация к учебной деятельности в вузе.

2.1. Социализация личности студента.

Понятие «студент» с латинского языка на русский язык переводится как усердно работающий, занимающийся, т.е. овладевающий знаниями. При этом в самом понятии «студент» заключен смысл личностных качеств человека, его индивидуальность, желание трудиться и работать. Студент как индивид определенного возраста, прошедший пубертатный период, и как личность еще формирующаяся характеризуется с трех аспектов:

1) психологического, который помогает охарактеризовать совокупность психологических процессов, состояний и свойств личности индивида;

2) социального, где можно заметить анализ межличностных отношений, качества, появляющиеся вследствие принадлежности студента к определенной социальной группе, национальности и т.д.;

3) биологического, где находит свое применение учет типа высшей нервной деятельности, строения анализаторов, безусловных рефлексов, инстинктов, физической силы, телосложения, черт лица, цвета кожи, глаз, роста и т.д.

Изучение трех аспектов в совокупности раскрывает качества и возможности студента как индивида, его возрастные и личностные особенности. Все этапы личного развития в процессе появления статуса студента можно подразделить на 3 этапа, в каждом из которых существуют разные личностные установки и цели (таблица 1).

Таблица 1. Этапы развития личности студента

Этапы саморазвития	Критерии личностного развития в обучении	Принципы н личностные установки	Личностная цель	Зоны развития мыслительной деятельности
Первый: самопознание, самоутверждение	Активность, обучаемость	1.Что я умею и смогу делать? 2. Я должен понять и выучить программы	Выявление своих возможностей при активной учебной деятельности	Зона актуального развития
Второй: самосовершенствование, самовоспитание	Активность, обучаемость и рефлексивность	1. Приобрести уверенность в себе	Активное и уверенное проявление себя и	Зона ближайшего развития

		2. Я должен стать лучше	своих возможностей в учебной деятельности	
Третий: самоактуализация, саморазвитие	Рефлексивность, творческость и креативность	Я должен сделать отличную работу, чтобы показать свою способность и уважать себя	Умение реализовать свои возможности и способности в создании своего продукта	

Процесс поступления в высшее учебное заведение, процесс становления школьника студентом укрепляет его веру в личностные силы и личные способности, помогает возродить надежду на полноценную и интересную жизнь в купе с тесным взаимодействием с остальными. Вместе с тем на 2 и 3 курсах чаще всего появляется вопрос о правильности решения выбранного учебного заведения, а также специальности, профессии. К концу 3 курса полностью решается вопрос о профессиональном личностном определении индивида. Зачастую замечаются сдвиги в настроении студентов - от восторженного в первые полгода учебы в вузе до скептического в процессе узнавания режима внутри вуза, его системы преподавания, отдельных преподавателей, общественной атмосферы, коллектива и т.п.

2.2. Психолого-педагогические особенности одаренных студентов.

Одаренность – это системное, развивающееся в течение жизни качество психики, которое определяет возможность достижения человеком более высоких, незаурядных результатов в одном или нескольких видах деятельности по сравнению с другими людьми.

Одаренный студент – это студент, который выделяется яркими, очевидными, иногда выдающимися достижениями (или имеет внутренние предпосылки для таких достижений) в том или ином виде деятельности.

Уровень, качественное своеобразие и характер развития одаренности – это всегда результат сложного взаимодействия наследственности (природных задатков) и социокультурной среды.

Признаки одаренности проявляются в реальной деятельности обучающегося и могут быть выявлены на уровне наблюдения за характером его действий. Признаки явной (проявленной) одаренности зафиксированы в ее определении и связаны с высоким уровнем выполнения деятельности.

В качестве примера признаков одаренности можно привести:

Особый тип организации знаний одаренного студента:

- высокая структурированность;
- способность видеть изучаемый предмет в системе разнообразных связей;
- свернутость знаний в соответствующей предметной области при одновременной их готовности развернуться в качестве контекста поиска решения в нужный момент времени;

Признаки мотивационного аспекта поведения одаренного обучающегося:

1. Повышенная избирательная чувствительность к определенным сторонам предметной действительности (знакам, звукам, цвету, техническим устройствам, растениям и т. д.) либо определенным формам собственной активности (физической, познавательной, художественно-выразительной и т. д.), сопровождающаяся, как правило, переживанием чувства удовольствия.

2. Повышенная познавательная потребность, которая проявляется в ненасытной любознательности, а также готовности по собственной инициативе выходить за пределы исходных требований деятельности.

3. Ярко выраженный интерес к тем или иным занятиям или сферам деятельности, чрезвычайно высокая увлеченность каким-либо предметом, погруженность в то или иное дело. Наличие столь интенсивной склонности к определенному виду деятельности имеет своим следствием поразительное упорство и трудолюбие.

4. Высокая требовательность к результатам собственного труда, склонность ставить сверхтрудные цели и настойчивость в их достижении, стремление к совершенству.

Психологические особенности студентов, демонстрирующих одаренность, могут рассматриваться лишь как признаки, сопровождающие одаренность, но не обязательно как факторы, ее порождающие.

Блестящая память, феноменальная наблюдательность, способность к мгновенным вычислениям и т. п. сами по себе далеко не всегда свидетельствуют о наличии одаренности.

Критерии одаренности:

– «степень сформированности одаренности»: актуальная; потенциальная.

Актуальная одаренность — это психологическая характеристика обучающегося с такими наличными (уже достигнутыми) показателями психического развития, которые проявляются в более высоком уровне выполнения деятельности в конкретной предметной области по сравнению с возрастной и социальной нормами.

Потенциальная одаренность — это психологическая характеристика обучающегося, который имеет лишь определенные психические возможности (потенциал) для высоких достижений в том или ином виде деятельности, но не может реализовать свои возможности в данный момент времени в силу их функциональной недостаточности. Развитие этого потенциала может сдерживаться рядом неблагоприятных причин (трудными семейными обстоятельствами, недостаточной мотивацией, низким уровнем саморегуляции, отсутствием необходимой образовательной среды и т. д.). Потенциальная одаренность проявляется при благоприятных условиях, которые обязан создать педагог.

– «форма проявления»: явная; скрытая.

Явная одаренность проявляется в деятельности обучающегося достаточно ярко и отчетливо (как бы «сама по себе»), в том числе и при неблагоприятных условиях.

Скрытая одаренность проявляется в атипичной, замаскированной форме, она не замечается окружающими. В результате возрастает опасность ошибочных заключений об отсутствии одаренности такого студента.

2.3. Адаптация личности студента, ее трудности и последствия. Адаптация к учебной деятельности в вузе.

Адаптация - означает приспособление. Адаптация студента к вузу – это не только «голое» приспособление к новой вузовской образовательной деятельности, сколько приобщение его к новой вузовской среде, которая дает новые возможности для реализации своих функций. Процесс адаптации неизбежно проходит через череду противоречий требований предъявляемой вузовской средой и возможностями (способностями) личности соответствовать им. Преодоление этих противоречий, сопровождается изменением поведения, деятельности личности, переосмыслением ранее полученного опыта, приобретением новых ценностей, навыков и умений. Социологические и психолого-педагогические подходы к определению вузовской адаптации различаются. В частности, социальная адаптация личности студента к вузу, как представителя определенной социальной группы, предполагает его активное взаимодействие в социуме. Средствами успешного достижения активности служат общее образование и профессиональная подготовка. Психолого-педагогическая адаптация студента, как объекта (субъекта) учебно-воспитательного процесса вуза и прежде всего конкретного курса, разумеет

приспособление личности студента к новым условиям учебной деятельности, выработку оптимальной модели поведения для целенаправленного функционирования.

Показателями успешной адаптации являются высокий статус студента в коллективе и психологическая удовлетворенность этим коллективом в целом. Речь идет о приспособлении студента к условиям учебного процесса и окружения «без ощущения внутреннего дискомфорта и без конфликта со средой». Показателями низкой адаптации - неудовлетворенность учебным трудом, его организацией, коллективом, неформальными связями, бытом и др., что приводит к низким показателям учебы, а в некоторых случаях и к отчислению из вуза.

Важным моментом процесса адаптации студентов-первокурсников является выработка активно-положительного отношения к будущей профессии и скорейшее приспособление к ритму учебной и учебно-профессиональной деятельности.

Структура вузовской адаптации зависит от конкретного исследования. Наиболее часто рассматривают: социальную или социально-психологическую (отражает изменение социальной роли студента, усвоение норм и традиций, сложившихся в вузе), психофизиологическую (ломка прежнего динамического стереотипа, формирование новых установок, навыков и привычек), психологическую (отражает перестройку мышления, речи, внимания, памяти, восприятия, воли, способностей), профессиональную (вхождение в профессиональную среду, усвоение норм и ценностей) и учебную (деятельностную) (отражает приспособление к учебному ритму, методам и формам работы, приобщение к учебному труду, то есть является дидактической адаптацией) - виды адаптации.

Деление адаптации на структурные компоненты является достаточно условным, так как «человек адаптируется как целостная структура, как организм (физиологическая адаптация); как индивид (психологическая адаптация); как личность (социальная адаптация); как субъект труда (профессиональная адаптация); как субъект учебной деятельности (учебная адаптация)».

Таким образом, условно выделяемые компоненты представляют собой единую систему - вузовскую адаптацию.

Выделяют первичную адаптацию - период начального включения студента в учебный процесс, в среду вуза и в коллектив, и вторичную адаптацию последующий период профессионального становления.

Начальный этап вузовской адаптации характеризуется освоением студентами новой информации (об условиях среды, деятельности) претерпевают изменения их представления о предстоящей деятельности и об особенностях новой социальной среды. Как следствие, изменяется самооценка и уровень притязаний личности. На следующем этапе новые представления о профессиональной деятельности и среде приводят к коррекции: норм, ценностей, структуры опыта, направленности на себя, на свою деятельность и свое социальное окружение.

Выделяют пять этапов адаптации:

1) Подготовительный этап адаптации. Он состоит в аккумулировании релевантной информации о предметных и социальных условиях предстоящей деятельности. Он может протекать в одной из двух форм (активно-целенаправленной или пассивной) в зависимости от индивидуально-психологических свойств и мотивационной сферы личности.

2) Этап стартового психического напряжения. Он связан с состоянием нервно-психического переживания подготовительных действий (событий) и первоначального вхождения в новые условия профессиональной деятельности. Это время внутренней мобилизации психических и психофизиологических ресурсов человека, обеспечивающей необходимые предпосылки для функционирования в новых условиях.

3) Этап острых психических реакций входа, на котором студент начинает ощущать воздействие изменившихся факторов предметной и социальной среды на себе (например, впервые столкнувшись с необходимостью принятия решения в новых условиях). Он

характеризуется переживанием состояния фрустрации, вызывающим конструктивные или деструктивные реакции.

4) Этап завершающего психического напряжения. Переход к этому этапу происходит в случае благоприятного развития адаптационного процесса. Этап, своеобразной подготовки психики человека к актуализации прежних режимов функционирования, привычных способов поведения. Такая подготовка связана с предстоящим возвращением к привычной жизни.

5) Этап острых психических реакций выхода. Состоит из комплекса эмоциональных и поведенческих реакций, связанных с вхождением в уже знакомую среду обитания и профессиональной деятельности.

1-3 этапы приходятся на 1-ый и 2-ой курсы обучения. В этот период происходит формирование определенных способов поведения (конструктивных реакций) личности, позволяющих ей справляться с адаптационными трудностями и успешно овладевать новой деятельностью. К конструктивным реакциям можно отнести переоценку ситуации и адекватное замещение способа удовлетворения потребностей, к деструктивным - агрессию и избегание решения возникшей проблемы.

Как известно, процесс адаптации на первом курсе обучения у студентов связан со следующими факторами, например:

• недостаточная психологическая готовность к будущей профессии;

• отсутствие повседневного контроля за выполнением заданий (ослабевает саморегуляция и самоконтроль учащегося);

• неумение работать с первоисточниками, словарями.

• неумение вести конспект.

• налаживание быта при условии жизни в общежитии;

• отрицательные переживания, связанные с уходом старого школьного коллектива.

Процесс адаптации у студентов проходит по следующим формам:

• формальная (приспособление студентов к структуре высшей школы);

• общественная (приспособление студентов к студенческому коллективу);

• дидактическая (приспособление студентов к новым формам обучения).

Никитиной К.А. выделяет низкий, средний и высокий уровни адаптированности в которых отражены показатели физиологического, психологического и социально-психологического видов адаптации.

Низкий уровень адаптированности характеризуется повышенной утомляемостью, низким и ниже среднего уровнями соматического здоровья, несогласованностью взаимной ответственности при выполнении коллективной работы, наличием отрицательного эмоционального состояния тревожности, студенты несамостоятельны, не умеют выстраивать межличностные отношения, не готовы к обучению в вузе, испытывают сложности в усвоении информации.

Для среднего уровня адаптированности характерны: средний уровень соматического здоровья, наличие мотивов самостоятельной работы и построения межличностных отношений, но при этом недостаточная способность включаться в жизнь новых социальных групп, готовность преодолевать трудности, наличие волевых качеств.

Высокий уровень адаптированности предполагает наличие выше среднего и высокого уровня соматического здоровья, сформированность умений и навыков практической работы, умения самостоятельно учиться, удовлетворенность общением с сокурсниками, преподавателями и низким уровнем тревожности.

Адаптация студентов к новым условиям обучения и общения на разных курсах имеет свои отличительные черты. Па 1 курсе, поведение студента отличается высокой степенью конформизма, отсутствует дифференцированный подход к своей роли. На 2 - период напряженной учебы, процесс адаптации к среде завершается. На 3 - происходит углубление профессиональных знаний и умений, сужается сфера интересов в связи с конкретизацией своей будущей профессии. На 4 - переоценка ценностей, связанная с практической

деятельностью в профессии, студенты отходят от коллективной формы общения, идет перестройка на будущие профессиональные и семейные установки.

Важным является вопрос о типологии студентов в процессе вузовской адаптации. Здесь рассматривая адаптационные типы людей на основании способности стратегического или комплексного решения проблем жизнедеятельности, выделяют: прогрессивно-творческий, адаптивно-репродуктивный, адаптивно-активационный и адаптивно-деформирующий типы.

Прогрессивно-творческий тип - для него характерна постановка широкого спектра перспективных, обобщенных задач оптимизации труда и собственного саморазвития.

Адаптивно-репродуктивный тип - для него характерно упрощение актуальной ситуаций и сведение их к известному из прошлого опыту, здесь в основе проблематизации лежит стереотипизация.

Адаптивно-активационный тип - для него характерно вместе с использованием стереотипизации использовать проблематизацию, искусственно усложняющая ситуацию, создавая квазипроблему.

Адаптивно-деформирующий тип - для него характерно невозможность комфортного существования «вне точки» своей профессии, что является следствием неправомерной аксеологии субъектов средств, условий и т.п. своей профессиональной деятельности.

Важным вопросом в процессе успешного обучения студента считается освоение специфики учебы в вузе, исключающее ощущение внутреннего дискомфорта и ограничивающее возможность появления конфликта с окружающим миром. В начале процесса обучения формируется студенческий коллектив, определяются имеющиеся навыки и возможности рациональной организации интеллектуального труда, осознается призвание к выбранной профессии, формируется оптимальный режим трудовой деятельности, создается структура работы по личному образованию и воспитанию профессионально значимых качеств индивида.

Можно заметить, что система обучения в вузе имеет массу отличий от школьной системы, потому как в школе учебный процесс построен таким образом, что у ученика постоянно и регулярно прививается желание к занятиям, в противном случае появится масса двоек. В вузовской же системе все обстоит иначе: лекции, лекции, лекции. Во время семинаров тоже, как оказалось, можно не постоянно готовиться, не надо на каждое занятие учить, решать и запоминать. Как результат часто формируется мнение о кажущейся легкости обучения в высшем учебном заведении в первом семестре, создается перспектива все наверстать и освоить информацию перед сессией, появляется беспечное отношения к учебе. Однако сессия ставит все мнения на правильные места и меняет беспечность индивида на озадаченность.

Адаптация студентов к учебному процессу в вузе заканчивается в конце 2- го - начале 3-го учебного семестра.

Главной задачей работы с первокурсниками считается процесс формирования и применения на практике методов рационализации и оптимизации свободного времени и самостоятельной работы. Уже сформированная система контроля за самостоятельной работой студентов с помощью семинарских, практических и лабораторных занятий включает в себя и пассивность, и уклонение студентов от выполнения соответствующих требований. Для формирования тактики и стратегии, позволяющей создать оптимальную адаптацию индивида к системе вузовского обучения, необходимо знать жизненные цели и интересы индивида, структуру доминирующих мотивов, уровень притязаний, личную самооценку, возможность индивидуальной и личной регуляции поведения и т.д.

Адаптация и ее протекание для каждого индивида происходит по-разному. Индивиды, обладающие уже трудовым стажем, проще и быстрее адаптируются к новым условиям студенческой жизни. Основная цель студенческой группы заключается в формировании подходящих условий для совокупной оптимальной деятельности всех индивидов в отдельности.

Еще одним важным аспектом в понимании важности адаптации у индивида в вузе состоит в объяснении основных черт индивида на каждом курсе.

На первом курсе индивид только учится приобщаться к студенческим формам коллективной жизни. Его поведение характеризуется чрезмерно высокой степенью конформизма; у индивида на первом курсе отсутствует дифференцированный подход к своим ролям как личности.

На втором курсе обычно происходит самая напряженная учебная деятельность индивида. У него как у второкурсника с особой силой включены все формы и обучения, и воспитания. К данному периоду процесс адаптации к внутривузовской среде чаще всего завершен.

Третий и четвертый курс характеризуется углубленным изучением специализации, увеличением интереса к выбранному направлению и научной работе как отражение дальнейшего развития. Индивид уже вполне сформирован как личность, к этому моменту пересмотрены уже все жизненные ценности и приоритеты, что помогает установить правильные установки на последующий род его деятельности.

Процесс нахождения спутника жизни играет значительную роль, оказывая непосредственное влияние и на успеваемость, и на общую направленность личности студента. Однако ошибочно считать, что в нахождении второй половинки есть негативное влияние. Интимные отношения часто вовлекают индивида в увеличение желания учиться, рабочему настроению, а также и в увеличение творческой деятельности.

Вопросы для повторения:

1. Что включает в себя адаптация индивида?
2. Какие виды адаптации выделяют?
3. Какие три аспекта характеристики личности студента существуют?
4. Как можно охарактеризовать юность с точки зрения развития личности?
5. Какие основные трудности появляются в процессе адаптации индивида к вузу?
6. Дайте определение понятия «адаптация» с точки зрения физиологии и медицины, педагогики, социологии и психологии.
7. Вузовская адаптация. Каковы показатели высокой и низкой вузовской адаптации?
8. Перечислите компоненты, выделяемые различными авторами в структуре вузовской адаптации.
9. Какие адаптационные типы людей вам известны?
10. Дайте описание уровней адаптированности студентов.

Тема 3. Профессиональное становление.

3.1. Профессиональное становление. Факторы профессионального становления.

3.2. Стадии и кризисы профессионального становления. Противоречия профессионального становления.

3.1. Профессиональное становление. Факторы профессионального становления.

Изменения, происходящие с личностью в процессе подготовки, овладения профессиональной деятельностью и самостоятельного выполнения, приводят к становлению ее как специалиста и профессионала.

Методологической основой профессиональной психологии является концепция профессионального становления личности (ПСЛ). Ее суть заключается в том, что, выбирая и осваивая профессию, личность изменяется: обогащается ее направленность, расширяются опыт и компетентность. Профессиональное становление личности сопровождается кризисами, конфликтами и деструктивными изменениями. Темп и траектория этого процесса детерминируется биологическими и социальными факторами, собственной активностью личности, а также случайными обстоятельствами, жизненно важными событиями и профессионально обусловленными инцидентами.

Изменение личности в процессе освоения мира профессий отражается в понятии *«профессиональное становление»,* которое характеризует индивидуально своеобразный путь (траекторию) личности, большую часть онтогенеза человека: с начала формирования профессиональных намерений до завершения профессиональной жизни.

Для решения проблем профессионального становления личности, обусловленных необходимостью профессионального самоопределения и выбора профессии, профессионального образования и повышения квалификации, профессионального роста и карьеры, профессиональной адаптации и достижения вершин профессионализма, прежде всего, требуется определиться с ведущими смыслообразующими понятиями.

К ним относятся, прежде всего: «профессиональное становление», «профессиональное развитие» и «профессионализация» личности. Понятия эти не тождественны.

Профессиональное становление личности - движение личности в профессионально-образовательном пространстве и времени профессиональной жизни. Это индивидуально своеобразный путь (траектория) личности с начала формирования профессиональных интересов и склонностей до окончания активной профессиональной деятельности.

Профессиональное развитие - это изменение психики в процессе освоения и выполнения профессионально-образовательной, трудовой и профессиональной деятельности. Объектом развития выступает субъект деятельности. Факторами, детерминирующими его развитие, являются социально-экономическая ситуация и ведущая деятельность. Профессиональное развитие человека происходит при его взаимодействии с миром профессий.

Профессионализация - «формообразование» субъекта, адекватного содержанию и требованиям профессиональной деятельности.

Существует множество внешних и внутренних факторов, в течение длительного времени оказывающих влияние на профессионализацию личности.

В качестве основных внешних факторов, оказывающих влияние на профессиональное развитие личности, ученый указывает 1) семью, школу, церковь; 2) социально-классовую позицию; 3) этнос, расу, национальные особенности индивида; 4) общую культуру (т.е. те нормы, традиции, которые сложились в обществе). Профессиональные интересы развиваются в результате взаимодействия между родителями и их детьми. Выбор карьеры индивидом отражает желание удовлетворить потребности, которые не были удовлетворены родителями в детстве. Формирование жизненных стереотипов происходит прежде всего в течение нескольких первых лет детства.

I.Существует три различных стереотипа отношений между родителями и детьми:

1 - характеризуется эмоциональной концентрацией на детях. Существуют две формы этого стереотипа: гиперопека, при которой родители слишком много делают для детей и поддерживают их зависимость; сверх требовательность, родители концентрируются на достижениях детей. В этом случае у детей вырабатывается потребность в постоянной обратной связи и поощрении. Они часто выбирают карьеры, которые обеспечивают общественное признание.

2 – отстраненность от детей. Существуют два экстремальных проявления данного стереотипа: пренебрежение родительскими обязанностями (для удовлетворения потребностей детей прилагается чрезвычайно мало усилий); отказ от воспитания (не предпринимается никаких усилий, чтобы удовлетворить потребности ребенка). Дети, в таких условиях, во взрослой жизни концентрируются на карьере, представляющей научный и технический интерес, находя в этом удовлетворение. Они более склонны иметь дело с предметами и идеями.

3 – принятие детей. Может быть непреднамеренным или более активным проявлением любви. Здесь поощряется независимость детей. Дети, выросшие в таких семьях, обычно выбирают карьеру, уравновешивающую личностные и социальные аспекты жизни, например, карьеру учителя или консультанта.

Существуют три типа косвенного влияния семьи на профессиональное становление детей: 1) социальный уровень семьи; 2) усвоение ценностей, предпочтений семейного окружения; 3) ретрансляционная функция семьи. В этом случае семейное окружение осуществляет оценку информации, исходящей из СМИ, и транслирует ее ребенку

Школа оказывает целенаправленное воздействие на профессиональное определение личности, поскольку это ее основная цель. Для выполнения данной функции школа имеет специальные средства (способы воздействия) и специалистов (учителей). Здесь закладываются интересы, формируются и развиваются способности. Если в семье влияние на профессиональное становление ребенка носит стихийный характер, то школа обязана заниматься этим системно.

Церковь с точки зрения профессионализации личности осуществляет в основном воспитательную задачу, формируя в личности понимание необходимости трудиться.

II. В качестве следующего внешнего фактора, оказывающего влияние на профессиональное развитие личности, является социально-классовая позиция. Ребенок, выросший в семье рабочих, вряд ли сможет осуществить свою профессиональную деятельность в банковской системе.

III. Этнос, раса, национальные особенности индивида. В связи с этим идет дифференциация сфер деятельности (неквалифицированный труд) в зависимости от цвета кожи.

IV. Общая культура, нормы и традиции, сложившиеся в обществе. В данном случае можно привести пример норм и традиций, сложившихся на Востоке, когда профессия обязательно передается по мужской линии от отца к сыну

Достаточно большое количество разнообразных факторов оказывает воздействие на профессиональное становление личности извне. Однако еще большее количество факторов, связанных с индивидуально-психологическими, личностными особенностями индивида, влияют на профессионализацию субъекта деятельности «изнутри». Данную категорию механизмов воздействия принято обозначать как «внутренние факторы».

3.2. Стадии и кризисы профессионального становления. Противоречия профессионального становления.

Согласно эвристической модели профессионально-образовательного пространства, предложенной Э.Ф. Зеером, профессиональное становление личности представляется в нем траекторией, которая формируется как результат совместного взаимодействия трех факторов:

- возрастные изменения, обусловливающие периодизацию развития личности;
- система непрерывного образования;
- ведущая профессионально ориентированная деятельность.

Данные факторы являются координатами профессионально-образовательного пространства (рис. 1).

Для визуального представления взаимодействия трех факторов, определяющих профессиональное становление личности, можно допустить их направленное, последовательное изменение, выделив уровни их выраженности. Такое допущение позволяет представить возможные траектории профессионального становления личности (рис. 1). Кратко остановимся на характеристике данных факторов.

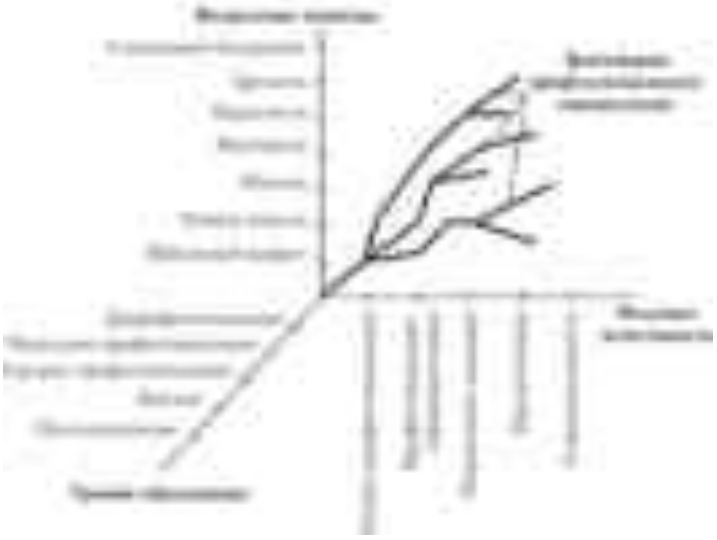


Рисунок 1. Модель профессионально-образовательного пространства

Возрастные изменения человека в течение длительного периода онтогенеза являются важным фактором профессионального становления личности. Особое значение в обеспечении и поддержке профессионального становления имеет система непрерывного профессионально ориентированного образования. Общее образование является ведущим фактором развития личности и предпосылкой успешного профессионального становления.

Также базовым, ключевым фактором профессионального становления личности является ведущая деятельность, которая формирует ее отношения с социально-экономической средой, общение с окружающими, институализирует социальную ситуацию развития.

Для каждой стадии профессионального становления характерна практически одна ведущая деятельность. Многоаспектность выполняемой личностью деятельности обогащают процесс профессионального становления. Считается, что для периода взрослости дифференциация ведущей деятельности отсутствует.

В качестве основания дифференциации онтогенеза может использоваться активность личности. В зависимости от уровня психической (социально и профессионально обусловленной) активности личности можно выделить следующие уровни социально-профессиональной активности: нормативно заданная; адаптивная; надситуативная; сверхнормативная.

Эти уровни активности имеют место на всех возрастных стадиях, но для каждого возрастного периода можно выделить преобладающий уровень психической активности, который и определяет характер ведущей деятельности. Нормативно заданная активность определяет учебно-профессиональную и профессионально-образовательную ведущую деятельность, адаптивная - нормативно одобряемую профессиональную деятельность (или воспроизводящую), надситуативная - продуктивную (высококвалифицированную), сверхнормативная - творческую самодеятельность. Последняя ведущая деятельность не имеет предела совершенствования.

Согласно модели (рис.1) прямолинейные участки индивидуальной траектории профессионального становления личности представляют собой усредненный вектор ее эволюционного развития. В силу нелинейного характера совместного действия факторов профессионально-образовательного пространства и других воздействий среды, оказывающих влияние на субъект деятельности, могут наблюдаться случайные отклонения вокруг вектора профессионального становления личности. Их существование обусловлено случайными внешними обстоятельствами, личностными кризисами, неудовлетворенностью собой, пресыщением рутинной деятельностью и т.п. Они постоянно нарушают эволюционный характер профессионального становления личности.

Ограниченность и предопределенность траекторий профессионального становления личности обусловлена социальной структурированностью общественного воспроизводства. При определенных внешних обстоятельствах и уровне развития структуры личности отклонения могут служить «спусковым механизмом» для перехода или переключения человека на новую траекторию профессионального становления. Области, где происходит изменение вектора развития, являются критическими точками или точками перестройки. В них происходит «излом» траектории, появляются несколько вариантов новых траекторий профессионального становления личности. Периоды между критическими точками называются стадиями профессионального становления.

Развитие профессионального становления является неустойчивым, неупорядоченным. Не все стадии периодизации поочередно сменяют друг друга, некоторые стадии могут даже отсутствовать. При переходе от одной стадии профессионального становления личности к другой происходит смена социальной ситуации развития, изменяется содержание ведущей деятельности, возникает новая социальная роль, профессиональное поведение и, конечно, трансформируется личность. В критических точках траектории профессионального становления поведение личности под воздействием внешних факторов становится неустойчивым и может продолжиться по одной из нескольких альтернативных новых стадий развития субъекта деятельности. Нарушение эволюционного развития может инициироваться одним из факторов профессионально-образовательного пространства: возрастными изменениями, социально-экономической ситуацией, ростом уровня профессионального образования и квалификации, перестройкой способов выполнения деятельности, а также случайным стечением обстоятельств.

Профессионально обусловленные изменения порождают субъективные и объективные трудности, межличностные и внутри- личностные конфликты. Развертывание этих психологических проблем приводит к кризисам профессионального становления.

Субъективное переживание кризисов способствует возникновению критических моментов, так называемых точек «раздвоения», которые побуждают личность к поиску новых путей реализации профессиональной биографии.

Профессиональное становление охватывает длительный период жизни человека - 35-40 лет. Поэтому возникает необходимость разделения данного процесса на периоды, или стадии. Цикличность процесса профессионального становления человека отражается в его периодизации, которая позволяет систематизировать, обобщить и согласовать многочисленные эмпирические факты и частные закономерности. Периодизацию процесса профессионального становления проводят по двум группам содержательных и формальных классификационных признаков, соответственно:

- качественные изменения (новообразования), характерные для каждого периода;
- длительность периодов в определенных временных единицах измерения.

В настоящее время наибольшую известность в России приобрела периодизация жизненного пути профессионала, предложенная Е.А. Климовым, который предлагает в профессиональном становлении личности вычленять следующие стадии, или фазы:

- оптация - период выбора профессии и пути ее приобретения;
- адаптация - вхождение в профессию и привыкание к ней;
- фаза интернала - приобретение профессионального опыта;
- мастерство - квалифицированное выполнение трудовой деятельности;
- фаза авторитета-достижение наивысшей квалификации;
- наставничество - передача опыта молодому поколению.

А.К. Маркова в качестве критерия для выделения этапов становления профессионала берет уровни профессионализма личности. Она различает пять уровней и девять этапов:

- допрофессионализм - этап первичного ознакомления с профессией;
- профессионализм состоит из трех этапов: адаптации к профессии, самоактуализации в ней и свободного владения профессией в форме мастерства;

• суперпрофессионализм также состоит из трех этапов: свободное владение профессией на уровне творчества, овладение рядом смежных профессий, самопроектирование себя как личности;

• непрофессионализм - выполнение труда по профессионально искаженным нормам на фоне деформации личности;

• послепрофессионализм - завершение профессиональной деятельности.

За рубежом широкое признание получила периодизация Дж. Сьюпера, выделившего пять основных этапов профессиональной зрелости (в некоторых изданиях - от 4 до 6 этапов):

1) рост - развитие интересов и способностей (до 14 лет);

2) исследование - апробация своих сил (14-25 лет);

3) утверждение — профессиональное образование и упрочение своих позиций в обществе (25-44 года);

4) поддерживание - создание устойчивого профессионального положения (45-64 года);

5) спад - уменьшение профессиональной активности (после 65).

В онтогенетических моделях профессионального становления и реализации субъекта деятельности в основном используется следующая временная структура периодизации:

1) стадии или уровни. *Стадия* - определенная ступень в развитии. Это самые длительные временные интервалы между критическими точками индивидуальной траектории профессионального. Для выделения стадий профессионального становления личности используют социальную ситуацию и уровень реализации ведущей деятельности, а также факторы профессионально образовательного пространства;

2) периоды или этапы. Стадии по своему психологическому содержанию являются неоднородными и в свою очередь могут делиться на отдельные периоды или этапы. Основанием для выделения является уточнение ситуации профессионального развития и конкретизация задач профессионального становления. В общем случае можно выделить три нормативных периода в составе стадии: адаптация или завершение решения задач развития предыдущей стадии, период решения основной задачи профессионального становления данной стадии, подготовка к переходу на новую стадию развития.

3) фазы. Структурными элементами периода являются фазы. Данный элемент периодизации должен определяться основаниями, выделяющего его в рамках нормативных задач профессионального становления.

Обобщив различные подходы к периодизации профессионального становления личности, Э.Ф. Зеер предлагает выделить следующие стадии (табл. 2):

Таблица 2 - Стадии профессионального становления личности

Название стадии и возраст	Социальная ситуация	Ведущая деятельность	Основные психологические новообразования на данной стадии
1. Аморфная оптация (0-12 лет)	Влияние родителей, родственников и учителей	Сюжетно-ролевые игры и учеба в школе, занятия в кружках и секциях	Зарождение профессионально ориентированных интересов и склонностей
2. Оптация (12-16 лет)	Завершающий период детства. Поиск своего места в мире профессий и в жизни	Учебно-профессиональная. Развитие познавательных и	Профессиональные намерения. Выбор пути профессионального образования, учебно-

		профессиональных интересов	профессионального самоопределения
3. Профессиональная подготовка (16-23 года)	Поступление в профессиональное учебное заведение. Новые социальные роли, взаимоотношения, социальная независимость	Профессионально-познавательная, ориентированная на получение конкретной профессии	Профессиональная подготовленность, профессиональное самоопределение, готовность к самостоятельному труду
4. Профессиональная адаптация (18-25 лет)	Новая система отношений в разновозрастном производственном коллективе	Профессиональная деятельность на нормативно репродуктивном уровне	Освоение новой социальной роли, самостоятельная профессиональная деятельность
5. Первичная профессионализация (23-27 лет)	Новая система отношений к окружающей действительности	Стабильная профессиональная деятельность	Профессиональная позиция, интегративные профессионально значимые констелляции
6. Вторичная профессионализация (27-33 года)	Стабилизация профессиональной активности, высокий уровень профессиональной деятельности	Дальнейшее повышение квалификации, выработка собственной профессиональной позиции, высокое качество и производитель-ность труда	Профессиональный менталитет, идентификация с профессиональным сообществом, профессиональная мобильность, корпоративность, гибкий стиль и высококвалифицирован-ная деятельность
7. Профессиональное мастерство (33-55 лет)	Этой стадии достигают не все, а только обладающие творческой потенцией, развитой потребностью в самоосушествлении и самореализации	Высокая профессиональная и социальная активность, поиск новых способов деятельности и взаимоотношений, стремление выйти за пределы себя	Творческая профессиональная деятельность, подвижные интегративные психологические новообразования, само проектирование своей деятельности и карьеры, вершина профессионального развития

Переход между стадиями сопровождается кризисами профессионального становления.

Кризис определяется как резкий. крутой перелом, тяжелое переходное состояние. Каждый кризис свидетельствует о завершении одного этапа жизни, развитии человека и о начале перехода к следующему или о начале поиска такового. Любой психологический кризис сопровождается трудными психическими состояниями и тяжелыми эмоциональными переживаниями.

Анализируя различные подходы и детерминанты, вызывающие кризисы, Э.Ф. Зеер классифицирует их на шесть типов: 1) нормативные, 2) психического развития и 3) профессионального становления, 4) ненормативные, 5) критические, 6) невротические). Первые три типа психологических кризисов, имеющих относительно выраженный хронологический характер, объединяются в группу возрастных, или нормативных. А вторые три типа - в группу жизненных, имеющих ненормативный, вероятностный характер.

Нормативный - значит изменяющий траекторию развития в соответствии с нормами профессионального становления личности, а ненормативный - имеющий случайный характер, не связанный с поступательным движением к овладению профессией.

Возрастные кризисы переживают все люди, но уровень их выраженности не всегда приобретает характер конфликта. Преобладающая тенденция возрастных кризисов конструктивная, развивающая личность.

Жизненные кризисы во многом случайны. Они возникают вследствие стечения обстоятельств. Выход из таких кризисов проблематичен. Иногда он бывает деструктивным, и тогда общество получает циников, маргиналов, бомжей, алкоголиков, самоубийц.

Типологию этих групп кризисов можно представить в виде следующей схемы (рис. 2).

Рисунок 2 - Типология кризисов

Под кризисами профессионального становления понимаются непродолжительные по времени периоды (до года) кардинальной перестройки профессионального сознания, деятельности и поведения личности, изменения вектора ее профессионального развития. Кризисы приводят к переориентации на новые цели, коррекции и ревизии социально-профессиональной позиции, подготавливают смену способов выполнения деятельности, ведут к изменению взаимоотношении с окружающими людьми, а в отдельных случаях - к смене профессии.

Основываясь на концепции профессионального становления личности, кризисы можно определить как резкие изменения вектора ее профессионального развития. Непродолжительные по времени, они наиболее ярко проявляются при переходе от одной стадии профессионального становления к другой. Кризисы протекают, как правило, без ярко выраженных изменений профессионального поведения. Однако происходящая перестройка смысловых структур профессионального сознания могут привести и к более кардинальным изменениям, вплоть до смены места работы и профессии.

К факторам, детерминирующим кризисы профессионального развития, относятся:

- сверхнормативная профессиональная активность, которая может выразиться в переходе на новый образовательно-квалификационный либо творческий уровень выполнения деятельности;

- возросшая социально-профессиональная активность личности вследствие ее неудовлетворенности своим социальным и профессионально-образовательным статусом;
- социально-экономические условия жизнедеятельности человека: ликвидация предприятия, сокращение рабочих мест, неудовлетворительная зарплата, переезд на новое местожительство и др.;
- возрастные психофизиологические изменения: ухудшение здоровья, снижение работоспособности, ослабление психических процессов, профессиональная усталость и т.д.;
- полная поглощенность профессиональной деятельностью;
- изменения жизнедеятельности (смена местожительства; перерыв в работе, связанный с уходом за малолетними детьми; «служебный роман» и т.п.).

Кризисные явления нередко сопровождаются нечетким осознанием недостаточного уровня своей компетентности и профессиональной беспомощностью. Иногда наблюдаются кризисные явления при уровне профессиональной компетентности, более высоком, чем требуется для выполнения нормативной работы. Как следствие возникает состояние профессиональной апатии и пассивности.

Проанализируем кризисы профессионального развития личности.

- Предкритическая фаза обнаруживается в неудовлетворенности существующим профессиональным статусом, содержанием деятельности, способами ее реализации, межличностными отношениями. Эта неудовлетворенность не всегда отчетливо осознается, но проявляется в психологическом дискомфорте на работе, раздражительности, недовольстве организацией, оплатой труда, руководителями и т.п.
- Критическая фаза отличается осознанной неудовлетворенностью реальной профессиональной ситуацией. Намечаются варианты ее изменения, проигрываются сценарии дальнейшей профессиональной жизни, усиливается психическая напряженность. Противоречия усугубляются, и возникает конфликт, который становится ядром кризисных явлений. Конфликт сопровождается рефлексией, ревизией учебно-профессиональной ситуации, анализом своих возможностей и способностей.
- Разрешение конфликта приводит кризис в посткритическую фазу. Способы разрешения конфликтов могут иметь конструктивный, профессионально-нейтральный и деструктивный характер.

Конструктивный выход из конфликта предполагает повышение профессиональной квалификации, поиск новых способов выполнения деятельности, изменение профессионального статуса, смену места работы и переквалификацию. Такой путь преодоления кризисов требует от личности проявления сверхнормативной профессиональной активности, совершения поступков, которые прокладывают новое русло для ее профессионального развития.

Профессионально-нейтральное отношение личности к кризисам приводит к профессиональной стагнации, равнодушию и пассивности. Личность стремится реализовать себя вне профессиональной деятельности: в быту, различного рода хобби, садоводстве и т.п.

Деструктивные последствия кризисов выражаются в нравственном разложении, профессиональной апатии, пьянстве, безделье.

В рассматриваемой нами концепции профессионального становления личности выделены следующие стадии этого процесса: оптация, профессиональное образование и подготовка, профессиональная адаптация, первичная и вторичная профессионализация и мастерство. Согласно определению кризисов, переход от одной стадии к другой порождает нормативные кризисные явления. Рассмотрим их психологические особенности, следуя логике профессионального становления.

1. Профессиональное становление личности начинается со стадии оптации, когда происходит смена ведущей деятельности с учебно-познавательной на учебно-профессиональную. Кардинально изменяется социальная ситуация развития, порождающая

неизбежное столкновение желаемого будущего и реального настоящего, которое приобретает характер кризиса учебно-профессиональной ориентации.

Старшеклассники, продолжившие учебу в 10-11 классах, переживают этот кризис в 16-17 лет, перед завершением школьного образования. Ядром кризиса является необходимость выбора способа получения профессионального образования или профессиональной подготовки. Деструктивное разрешение кризиса приводит к ситуативному выбору профессиональной подготовки или профессии, выпадению из нормальной социальной сферы.

2. На стадии профессиональной подготовки многие учащиеся и студенты переживают разочарование в получаемой профессии. Возникает недовольство отдельными учебными предметами, появляются сомнения в правильности профессионального выбора, падает интерес к учебе. Наблюдается кризис профессионального выбора. Как правило, он отчетливо проявляется в первый и последний годы профессионального обучения. За редким исключением этот кризис преодолевается сменой учебной мотивации на социально-профессиональную.

3. После завершения профессионального образования наступает стадия профессиональной адаптации. Первые недели и месяцы самостоятельной работы вызывают большие трудности. Но они не становятся фактором возникновения кризисных явлений. Основная причина психологическая - несовпадение реальной профессиональной жизни со сформировавшимися представлениями и ожиданиями. Это несоответствие вызывает кризис профессиональных экспектаций (ожиданий).

Переживание этого кризиса выражается в неудовлетворенности организацией труда, его содержанием, должностными обязанностями, производственными отношениями, условиями работы и зарплатой.

Возможны два варианта разрешения кризиса:

- конструктивный: активизация профессиональных усилий по скорейшей адаптации и приобретению опыта работы;
- деструктивный: увольнение, смена специальности; неадекватное, некачественное, непродуктивное выполнение профессиональных функций.

4. Возникает на завершающей стадии первичной профессионализации, после 3-5 лет работы. При отсутствии перспектив профессионального роста личность испытывает дискомфорт, психическую напряженность, появляются мысли о возможном увольнении, смене профессии. Кризис профессионального роста может временно компенсироваться разного рода непрофессиональными, досуговыми видами деятельности, бытовыми заботами или же кардинально решаться путем ухода из профессии. Стабилизация же всех сторон профессиональной жизни способствует профессиональной стагнации личности: смирению и профессиональной апатии. Стагнация может длиться годами, иногда до ухода на пенсию.

5. Вторичная профессионализация. Кардинально перестраиваются социально-профессиональные ценности и отношения. Ведущая деятельность на этой стадии характеризуется индивидуальным стилем и элементами творчества. Во многих случаях качественное и высокопродуктивное выполнение деятельности приводит к тому, что личность перерастает свою профессию. Сформировавшееся к этому времени профессиональное самосознание подсказывает альтернативные сценарии дальнейшей карьеры, и не обязательно в рамках данной профессии. Противоречия между желаемой карьерой и ее реальными перспективами приводят к развитию кризиса профессиональной карьеры. При этом серьезной ревизии подвергается «Я-концепция», вносятся коррективы в сложившиеся производственные отношения. Возможные сценарии выхода из кризиса: увольнение, освоение новой специальности в рамках той же профессии, переход на более высокую должность. Одним из продуктивных вариантов является переход на следующую стадию профессионального становления стадию мастерства.

6. Стадия мастерства характеризуется творческим и инновационным уровнем выполнения профессиональной деятельности. Движущим фактором дальнейшего профессионального развития личности становится потребность в самореализации, которая нередко приводит к неудовлетворенности собой, окружающими людьми. Кризис социально-профессиональной самоактуализации, - это душевная смута, бунт против себя. Продуктивный выход из него - новаторство, изобретательство, стремительная карьера, социальная и профессиональная сверхнормативная активность. Деструктивные варианты разрешения кризиса - увольнение, конфликты, профессиональный цинизм, алкоголизм, создание новой семьи, депрессия.

7. Кризисный характер для многих работников приобретает и предпенсионный период. Уход на пенсию означает сужение социально-профессионального поля и контактов, снижение финансовых возможностей. Острота протекания кризиса утраты профессиональной деятельности зависит от характера трудовой деятельности (работники физического труда переживают его легче), семейного положения и здоровья.

8. Социально-психологическое старение. Проявляется в ослаблении интеллектуальных процессов, повышении или снижении эмоциональных переживаний. Отмечается пристрастие к морализированию и осуждению поведения молодежи, четко прослеживается противопоставление своего поколения поколению, идущему на смену. Неудовлетворенность современной жизнью обусловлена укорочением жизненной перспективы. Данное беспокойное и тревожное состояние личности определяется как кризис социально-психологической адекватности.

Ненормативные кризисы - это кризисы, обусловленные ненормативными, случайными событиями. Предсказать эти кризисы невозможно, они сугубо индивидуальны и возникают вследствие стечения обстоятельств. К ненормативным кризисам относятся:

1) Кризисы невротического характера, которые появляются вследствие внутриличностных изменений, а именно перестройки сознания, различных бессознательных впечатлений, иррациональных тенденций;

2) Жизненные кризисы, которые связанны с изменения индивидуальной биографии человека. В большинстве случаев следствием данных событий является перестройка сознания и поведения у человека;

3) Критические кризисы, которые связаны с трагическими событиями в жизни человека. Эти кризисы инициируют сильнейшие эмоциональные переживания, перестройку ценностей, потерю прежнего смысла жизни и поиск нового, изменения поведения и сознания.

Таким образом, ненормативные кризисы - это сугубо индивидуальный спектр событий в жизни каждого конкретного человека, спрогнозировать который крайне затруднительно.

Условием преодоления развивающегося кризиса может стать определение областей приложения накопленного опыта, в которых можно получить подтверждение своей полезности, нужности. Главное - насытить жизнь активной деятельностью.

Решающее значение в возникновении кризисов на первых стадиях профессионального становления имеют *объективные факторы*: смена ведущей деятельности, кардинальное изменение социальной ситуации. На последующих стадиях все большую роль играют субъективные факторы: изменение «Я-концепции», перестройка профессионального сознания, возрастание уровня притязаний и самооценки, проявление потребности в самоутверждении и самоосуществлении. Продуктивное выполнение деятельности приводит к тому, что профессионализм личности перерастает саму деятельность.

Общие моменты в определении кризиса. Во-первых, кризис обязательный этап развития, относительно непродолжительный во времени по сравнению со стабильными этапами развития личности. Во-вторых, исход кризиса, может быть либо «положительным» для личности, т. е. происходит переход на качественно новый уровень развития; либо «отрицательным», т.е. деструктивная, неадаптивная линия развития личности. В-третьих,

кризис является следствием «конфликта» между личностью и наличными условиями развития социальной ситуации, возникшее противоречие порождает кризис.

Вопросы для повторения:

1. Что понимают под профессиональным становлением личности?
2. Какие факторы влияют на профессиональное становление личности?
3. Что представляет собой профессиональное становление личности в модели профессионально-образовательного пространства?
4. Как определяются стадии профессионального становления личности по Зееру Э.Ф.?
5. Приблизительно какой период жизни человека охватывает его профессиональное становление?
6. Что такое периодизация профессионального становления?
7. Сколько стадий периодизации профессионального становления и какие предложены Климовым Е.А.?
8. Какие типы кризисов личности выделяет Зеер Э.Ф.?
9. Какой тип кризиса лежит в основе кризиса профессионального становления?
10. Что понимается под кризисом профессионального становления?
11. Какие выделяют виды нормативных кризисов профессионального становления?

Тема 4. Лидерство в организации.

4.1. Феномен лидерства. Психологическое содержание понятия «лидерство».
4.2. Стили лидерства.
4.3. Лидерство и руководство.
4.4. Гендерные аспекты организационного руководства и лидерства.
4.5. Методика формирования команды. Организация межличностных, групповых и организационных коммуникаций.

4.1. Феномен лидерства. Психологическое содержание понятия «лидерство».

Английское leader имеет ряд сходных значений: руководитель, вождь, глава, командир, которые с известной степенью условности можно обобщить как «некто главный» или даже «самый главный». В то же время этимология этого английского слова восходит к lead - путь, дорога. Исходя из этого понятие «лидерство» часто трактуется как «идущий впереди», «указывающий дорогу». Лидерство – процесс межличностного влияния, обусловленный реализацией ценностей, присущих членам группы и направленный на решение стоящих перед группой целей.

Понятие «лидер» в психологии определяют следующим образом:

1) Лидер — член группы, обладающий наибольшим ценностным потенциалом, который и обеспечивает ему ведущее влияние в группе. В малой группе лидер может выступать как организатор, мотиватор деятельности, он может быть наиболее отзывчивым или влиятельным.

2) Лидер – член группы, за которым она признает право принимать ответственные решения в значимых для нее ситуациях, т.е. авторитетная личность, реально играющая центральную роль в организации совместной деятельности и регулировании взаимоотношений в группе.

Классификация лидеров:

- по содержанию деятельности: вдохновитель; исполнитель;
- по характеру деятельности: универсальный; ситуативный;
- по направленности деятельности: эмоциональный; деловой.

4.2. Стили лидерства. Лидерство и руководство. Гендерные аспекты организационного руководства и лидерства.

Модель лидерства как научного управления, при котором руководителя интересует не сам работник, а наиболее оптимально устроенная среда, заменились в середине двадцатых годов на модель «человеческих взаимоотношений». В модели «человеческих взаимоотношений» роль руководителя интерпретируется по типу «лидер – ведомый».

В соответствие с различными теориями лидерства проводили исследования на определение стилей лидерства, которые разделили на Х-теории или Теории Y.

Теория Х, в соответствии которой лидер должен обладать чертами диктатора, основана на следующих представлениях:

- люди обычно не любят работать и стараются уклониться от своих обязанностей;
- поэтому работников нужно заставлять трудится, манипулировать ими, угрожать и наказывать, чтобы добиться стоящих перед организацией целей;
- люди хотят быть направляемыми, стремясь к защите и избегая ответственности.

Теория Y исходит из прямо противоположных представлений:

- людям нравится работать, и для многих в труде скрыт источник удовлетворения;
- большинство работников руководствуются самодисциплиной и нуждаются в угрозах, они также заинтересованы в выполнении общих целей;
- многие из них не только избегают ответственности, но и стремятся к ней;
- способности к творчеству в решении организационных пробоем присущи не только лидерам или руководителям;
- поощрение является лучшим способом для вдохновения людей к выполнению задач, стоящих перед организацией.

В соответствие с теорией Y и лидер Y должен быть чувствителен к запросам и нуждам работников, прислушиваться к их предложениям по поводу улучшения работы в организации.

Теории Х и Y обозначают крайние полюса стиля лидерства, характеризуемые в терминах авторитарного или демократического стиля.

Наиболее известные исследования в данном ключе были выполнены под руководством К.Левина и Липпета. Авторы выделили 3 основных стиля лидерства:

1) авторитарный стиль;

2) демократический стиль;

3) попустительский стиль.

При авторитарном (автократическом) стиле лидер принимает решения единолично, определяя и регламентируя всю деятельность подчиненных, не давая им возможность проявить инициативу.

При попустительском стиле лидер вообще избегает принимать какие-либо решения, не участвуя в этом процессе и предоставляя подчиненным полную свободу действий.

При демократическом стиле лидер вовлекает сотрудников в процесс принятия решений, используя групповую дискуссию, стимулируя их активность и разделяя вместе с ним ответственность за принятие решений.

В настоящее время предпринимаются попытки переназвать их соответственно: директивный стиль, коллегиальный стиль, разрешительный стиль.

Описанное лидерство содержит в себе две стороны: содержательную и техническую. Содержательная сторона включает в себя решения, предлагаемые лидером, а техническая – способы этих решений. Г.М. Андреева предлагает рассматривать лидерские стили, анализируя их с 2 указанных сторон, на основе следующей таблицы (табл. 3).

Таблица 3 - Содержательные и технические характеристики стилей лидерства по Левину и Липпету

Стиль	Формальная (техническая)	Содержательная сторона

Авторитарный	Деловые, краткие распоряжения. Запреты без снисхождения, с угрозой. Четкий язык, неприветливый тон. Похвала и порицание субъективны. Эмоции не принимаются в расчет. Показ приемов – не система. Позиция лидера – вне группы.	Дела в группе планируются заранее во всем объеме. Определяются лишь непосредственные цели, дальние – неизвестны. Голос руководителя – решающий.
Демократический	Инструкция в форме предложений. Не сухая речь, товарищеский тон. Похвала и порицание – с советами. Распоряжение и запреты с дискуссиями. Позиция лидера – внутри группы.	Мероприятия планируются не заранее, а в группе. За реализацию предложений отвечают все. Все разделы работы не только предлагаются, но и обсуждаются.
Попустительский	Тон – конвенциональный. Отсутствие похвалы и порицаний. Никакого сотрудничества. Позиция лидера – незаметно в стороне от группы.	Дела в группе идут сами собой. Лидер не дает указаний. Разделы работы складываются их отдельных интересов или исходят от нового лидера.

Г.М. Андреева подчеркивает, что любая схема не может охватить все стороны и проявления стиля лидерства, однако исследователи пытаются усложнить ее или ввести новые понятия, например, новые типы лидеров: лидер-организатор, лидер-эрудит, лидер-инициатор, лидер – генератор эмоционального настроя, лидер эмоционального притяжения, лидер-умелец. При этом проблема заключается в том, что феномен лидерства в настоящий момент описан еще неопределенно, в том числе нет четкого разделения между понятиями «лидер» и «руководитель».

С точки зрения Г.М. Андреевой, самым большим упрощением является мнение, согласно которому руководитель и лидер – обязательно одно лицо.

Что касается эффективности того или иного стиля лидерства: в группе руководимой лидером с демократическим стилем руководства, уровень общей удовлетворенности работой наивысший, также, как и стремление к творчеству, в группе лидера с авторитарным стилем продуктивность самая высокая, по сравнению с другими группами. Попустительский стиль приводит к беспорядкам и конфликтам, что сказывается на снижении объема и качества выполняемой работы. Выбор стиля лидерства во многом зависит от содержания той или иной деятельности.

В целом, предпочтительным является демократический стиль, однако, когда речь идет о выполнении «простой» работы за короткий срок силами группы со средним уровнем образования более эффективным является авторитарное лидерство. Попустительский стиль продуктивен, когда речь заходит о выполнении сложной работы с неопределенными целями, которую выполняет креативная и образованная группа.

Основу теорий лидерства Ф.Фидлера составляют следующие понятия: руководитель, «руководитель, ориентированный на задачу», «руководитель, ориентированный на межличностные отношения». При этом особую роль в анализе продуктивности того или иного лидера играет «наименее предпочитаемый сотрудник» (он особо «мешает» руководителю, ориентированному на задачу).

В рамках данной модели была разработана Шкала Измерения Предпочтений Сотрудников. Лидеры с высокими оценками характеризуются в позитивно-окрашенных тонах (имеют более близкую дистанцию в отношении с работниками, активно обсуждают

ход выполнения заданий с сотрудниками). С низкими оценками характеризуются негативно и оцениваются как ориентированные на задание. Для личностно-ориентированных лидеров оптимальным является средний, сдержанный уровень контроля. Предметно-ориентированные лидеры были гораздо эффективнее в условиях или слишком высокого, или слишком низкого уровня контроля. Стиль личностно-ориентированного лидера следует назвать демократическим, а ориентированного на результат – авторитарным или практикующим либерально-попустительский стиль.

4.3. Лидерство и руководство.

Лидер может быть одновременно руководителем группы, но может и не быть им. Руководителя целенаправленно избирают, а чаще назначают, он отвечает за положение дел в возглавляемом коллективе, обладает официальным правом поощрять или наказывать участников совместной деятельности.

Лидер выдвигается стихийно, не обладает никакими властными полномочиями, на него не возложены никакие официальные обязанности (табл. 4). Лидерство и руководство имеет много общего в области использования инструментов воздействия на людей, в процессе управления группой, в стремлении к результату.

Таблица 4 - Сравнительный анализ лидерства и руководства

Основания сравнения	Лидерство	Руководство (по Парыгину)
Социальная роль	Регуляция межличностных отношений	Регуляция официальных отношений
Сферы влияния	Микросреда	Макросреда - связь со всей системой общественных отношений
Рождение	Стихийно возникает	Назначается, выбирается и т.п.
Устойчивость	Нестабильно	Стабильно
Наличие инструментов власти	Не может применить санкций	Может применить санкции: наказать или наградить
Сфера ответственности	Непосредственное принятие решения, личная ответственность	Сложный многоступенчатый процесс принятия решения, разделение ответственности
Сфера влияния	Малая сфера действия	Широкая сфера действия

4.4. Гендерные аспекты организационного руководства и лидерства.

Лидерство – процесс, присущий как мужчинам, так и женщинам, долгое время в социокультурном пространстве и социальных практиках считался «мужской сферой».

В современном мире участие женщин в управленческой деятельности имеет тенденцию ко все большему расширению: женщины занимают руководящие позиции во множестве организаций как в некоммерческом, так и в коммерческом секторах, в том числе в больших международных корпорациях, они становятся мэрами городов, занимают посты министров и глав правительств, возглавляют государства. Много девушек возглавляет и молодежные организации — профсоюзные организации студентов, волонтерские организации и политические.

Однако если мы сравним их количество с мужчинами, которые занимают руководящие должности, женщины окажутся в явном меньшинстве. Почему так происходит? Ответ прост: гендерные стереотипы отражают распределение мужчин и женщин в нашем обществе. Эти стереотипы все еще традиционно связывают женщин с домохозяйками, а мужчин с кормильцами и добытчиками. Женщины обычно описываются

ориентированными на поддержание хороших межличностных отношений, отзывчивыми и заботливыми, а мужчины — нацеленными на результат, уверенными в себе, напористыми и более независимыми. Именно поэтому традиционные стереотипы лидерства связаны с мужскими атрибутами и ассоциируются именно с мужчинами.

Действия же женщин, которые пытаются использовать активные, целенаправленные стили руководства (т. е. «вести себя как мужчины») воспринимаются негативно. Часто это приводит к тому, что одинаково квалифицированные женщины в одних и тех же руководящих позициях уступают по сравнению с мужчинами. «Вторичность» женщин в области лидерства объясняют «объективно». Так рождаются гендерные стереотипы относительно «естественности» подчинения и нелидерства женщин.

Рассмотрим данные стереотипы.

1. Стиль поведения.

Гендерный стереотип:

- мужской стиль – инструментальный (добытчик, кормилец, глава семьи, ответственен за дисциплину детей);
- женский – экспрессивный (хранительница домашнего очага, ориентирована на отношения, поддерживает теплый эмоциональный климат).

Фактически: Мальчики являются более личностно-ориентированными в отношении сверстников, а в отношении взрослых оба пола не отличаются по указанной ориентации. Стиль поведения представителя определенного пола не является врожденным, а задается обществом. Это соответствует исторически сложившемуся разделению ролей.

2. Эффективность деятельности, успешность.

Гендерный стереотип: Считается, что мужчины более продуктивны, чем женщины, в осуществлении деятельности. При этом имеются в виду либо все виды деятельности, либо те, что ценятся обществом – политика, наука, искусство, спорт и т.д. Отсутствие у женщин выдающихся успехов в этих областях объясняется обычно отсутствием у них соответствующих способностей к этим видам деятельности.

Фактически: Девочки успешнее мальчиков на протяжении всего школьного периода и больше интересуются успехами. Мужчины превосходят в деятельности, выполняемом индивидуально, и при решении задач, требующих ориентации на задачу, превосходство женщин - когда требуется интеракционный стиль.

3. Мотивация достижений и отношение к наградам за деятельность.

Гендерный стереотип:

Мужчины с детства имеют: большую потребность в достижениях; особенно в областях, связанных с неодушевленными предметами; мотивированы на успех ради успеха.

Женщины: имеют меньшую мотивацию достижений; эффективнее в областях взаимодействия людей; их усилия направлены не на успех, а обусловлены желанием нравится другим или избегать осуждения.

Фактически: Исследования, проведенные на детях, обнаружили либо отсутствие половых различий по уровню мотивации достижения, либо преимущество девочек. Однако в период взрослости меняется соотношение мотивации достижения у мужчин и женщин. Мужчины стимулируются обществом на успех, женщины либо не поощряются, либо поощряются негативно. В результате женщины отказываются от социальных достижений.

4. Конкурентность и кооперативность.

Гендерный стереотип: Мужчины нацелены на конкуренцию, а женщины – на сотрудничество.

Фактически: И мальчики и девочки демонстрируют либо одинаковую конкурентность, либо мальчики превосходят девочек по этому качеству, более остро реагируя на конкурентную ситуацию и соперников. У взрослых женщин имеется феномен «боязни успеха», успех связывается с негативными последствиями: потерей женственности, осуждением близких и т.п. Также у взрослых нет различий по параметру

сотрудничество: большая конкурентность мужчин не означает большей кооперативности у женщин.

5. Стремление к лидерству и мотивация власти.

Гендерный стереотип: Мужчинам приписывается наличие стремления к лидерству и мотивация власти, женщинам – отсутствие подобной мотивации.

Фактически: Девушки показывают больше стремлений в получении управленческих должностей в будущем, чем юноши.

6. Характеристики личности, способствующие и препятствующие лидерству

Гендерный стереотип: Мужчины воспринимаются как агрессивные и доминантные, женщины – подчиненные и тревожные.

Фактически: Женщины более тревожны, а мужчины превосходят женщин по доминантности и агрессивности, но это не дает им преимуществ в лидерстве, т.к. в современном обществе агрессия неприемлема, а на высшем уровне управления наблюдается переход к неагрессивному лидерскому стилю.

7. С каким полом связывается лидерство.

Гендерный стереотип: Лидером должен быть мужчина.

Фактически: В детстве мальчики делают больше попыток лидировать, чем девочки. В организациях менеджерские должности чаще принадлежат мужчинам, но доля женщин растет.

Существующие гендерные стереотипы лидерства не всегда подтверждаются при изучении лидеров. Встречающаяся меньшая эффективность женщин-лидеров может объясняться: а) влиянием стереотипов; б) мотивацией; в) отношением общества; г) ролевым конфликтом у женщины; д) методологическими проблемами исследования.

4.3. Методика формирования команды. Организация межличностных, групповых и организационных коммуникаций.

Команда – это небольшая группа людей, стремящихся к достижению общей цели, постоянно взаимодействующих и координирующих свои усилия. Работа в командах является средством повышения эффективности деятельности организации. Огромное количество информации, быстро изменяющаяся внешняя среда, конкуренция и т.п. – все это затрудняет процесс управления в организации в целом. В этих условиях большое значение имеет использование команд, которые более гибки и мобильны, и способны быстро реагировать на сигналы, посылаемые окружением.

Чтобы деятельность команд была эффективной, необходимо обеспечить разнообразие ролей в команде:

1. Специалисты по решению задач. Их роль - достигать целей, стоящих перед командой. Черты, характерные для членов команды, играющих эту роль:

- [illegible]
проблем.

- [illegible]ию среди членов команд по поводу решения проблем, оценивают полученные идеи.

- [illegible]
проблеме.

- [illegible]
ограничений выстраивают целостную картину.

- [illegible]

2. Члены команды, осуществляющие социально-эмоциональную поддержку. Их роль состоит в удовлетворении эмоциональных потребностей членов команды. Черты, характерные для членов команды, играющих эту роль:

- [illegible]
похвалы.

- [illegible]

[illegible]
[illegible]енным мнением ради поддержания гармонии в команде.

Если большинство членов команды склонны к исполнению этой роли, то члены команды получают высокое индивидуальное удовлетворение, но, как правило, за счет снижения эффективности действия.

Если же большинство членов команды склонны к исполнению роли «специалистов по решению задач», то такая команда оказывается очень эффективна, но только в течение короткого отрезка времени, однако в долгосрочной перспективе у членов таких команд снижается степень удовлетворения от работы, и, следовательно, снижается эффективность.

3. Члены команды, играющие двойную роль. Такие люди совмещают в себе две вышеописанные роли: выполняют поставленную перед командой задачу и удовлетворяют эмоциональные потребности членов команды. Обычно люди, способные играть двойную роль, становятся лидерами команд.

4. Члены команды, играющие роль стороннего наблюдателя. Такие люди обычно держатся отстраненно от повседневной жизни команды, активно не участвуют ни в решении задач, ни в создании положительного эмоционального климата. Однако такие люди очень полезны в критический момент, поскольку видят проблемы команды как бы «со стороны» и часто дают нетривиальную «обратную связь».

Руководители не должны забывать, что команда должна быть хорошо сбалансирована, в ней должен присутствовать весь «спектр» ролей.

Согласно другой классификации, помимо содержательных оснований (конкретные специальности, опыт, квалификация и т.п.), в команде должны быть представлены определенные типы людей:

- Доводящий до конца. Как правило, об успехе команды судят по окончательным результатам ее работы. Доводящие до конца завершают все, что начинают и неохотно предпринимают что-либо там, где есть сомнения по поводу того, что удастся довести дела до завершения. Они заботятся о завершении намеченного и настаивают на этом даже тогда, когда энтузиазм всех остальных членов команды уже исчерпан. Их присутствие не дает команде тратить время впустую на проекты, которые не могут быть доведены до конца.

- Возмутитель спокойствия. Возмутители спокойствия всегда выступают как побудители к действию, и если команда склонна к бездействию или самодовольству, то присутствие Возбудителя спокойствия выведет ее из этого состояния.

- Действующий. Основным качеством Действующих, отражающим их установки и характер, является дисциплинированность. Будучи дисциплинированными по своей сути, они упорядочено подходят к любой поручаемой им работе. Среди их отличительных качеств также следует назвать: организованность, сознательность, приверженность обязательствам, серьезное отношение к любому делу, надежность, практичность, терпимость к окружающим.

- Коллективист. Представители этой роли оказывают «смягчающее» воздействие на команду: их присутствие улучшает моральный климат и повышает степень сотрудничества между членами команды.

- Мыслитель. Основное назначение Мыслителя в команде – привнесение новых и оригинальных идей. Как правило, Мыслители действуют в одиночку, обдумывая различные варианты. Им свойственен самоуглубленный, аналитический подход к решению проблем.

- Оценивающий. Представители этой роли ярко не проявляют себя в команде до тех пор, пока не приходит время принятия важных решений. Представители этой роли в команде обладают высоким интеллектуальным уровнем, высокими показателями критичности мышления, особенно это касается их способности выдвигать контраргументы.

- Председатель. Основной залог успеха Председателя – его личностные качества. Прежде всего, по своей натуре он склонен доверять людям и принимать их такими, какие они есть, без проявлений ревности или подозрительности. В качестве противовеса этому

качеству он должен уметь доминировать в команде и быть приверженным целям и задачам команды, что усиливает и морально обосновывает его доминирующую позицию. Председатель – это хороший лидер для сбалансированной команды, перед которой стоят сложные и многогранные проблемы, требующие эффективного распределения ролей в команде.

- Исследователь ресурсов. Это еще один член команды, ориентированный на предложение новых идей. Однако, способ генерации идей Исследователями ресурсов и сам характер предлагаемых ими идей отличны от Мыслителей. Они склонны не столько сами предлагать оригинальные идеи, сколько «подбирать» фрагменты идей окружающих и развивать их. Исследователи ресурсов особо искусны в изучении ресурсов за пределами команды.

Этапы развития команды:

– [illegible] узнают и принимают друг друга, формулируют задачи группы. Преобладает аура вежливости, взаимоотношения членов группы отличаются осторожностью. Все члены будущей команды «размахивают визитками», то есть стараются подчеркнуть свои прошлые мнимые и реальные заслуги перед человечеством.

– [illegible] «ясно», кто есть кто, и они начинают самоопределяться в команде. Члены группы конкурируют за обладание более высоким статусом, за относительное влияние, дискутируют о направлениях развития. Группа испытывает внешнее давление, между ее участниками складываются достаточно напряженные отношения. На сцену выходят лидеры «первой волны». Они уверены в себе, опытны, настойчивы, громко говорят и всегда знают «точно», что надо делать.

– [illegible] движение к общим целям, устанавливаются равновесие конкурирующих сил и групповые нормы, определяющие поведение ее членов, сотрудничество членов команды становится все более эффективным. В это время может произойти смена лидеров, на сцену могут выйти лидеры «второй волны». Они внешне менее эффектны, но люди чувствуют себя с ними более уверенными и раскованными.

– [illegible] способна решать самые сложные задачи, каждый ее член исполняет несколько функциональных ролей. На этом этапе команде присущи все те качества, которые мы сформулировали в виде списка тринадцати характеристик.

– [illegible] трансформируются) самые успешные команды, интенсивные социальные отношения их участников постепенно сходят на нет. Но это скорее оптимистичное наблюдение, нежели пессимистичное. Люди устают друг от друга, поэтому возникающие новые крупные проекты и идеи (если это подлинно инновационная организация), собирают людей под новое знамя, предлагая иные конфигурации отношений и новые вызовы времени.

В организациях различают две крупные группы коммуникаций – формальные и неформальные коммуникации. Формальные коммуникации осуществляются между элементами формальной структуры организации – межуровневые коммуникации (нисходящие и восходящие), горизонтальные коммуникации (между подразделениями одного уровня в иерархии организации), коммуникации «руководитель – подчиненный», «руководитель – рабочая группа». Неформальные коммуникации связаны с неформальными группами и неслужебными вопросами, а также с распространением слухов о служебных вопросах. Обычным средством осуществления формальных коммуникаций является письменная и устная речь. При осуществлении коммуникаций следует принимать во внимание наличие «барьеров непонимания» (семантического, стилистического, логического, фонетического, барьера авторитета и др.) и овладевать методами их преодоления. В неформальных коммуникациях силен эмоциональный фактор.

Общее для всех типов коммуникаций внутри организаций – это то, что в организациях они имеют место между людьми, являющимися исполнителями разнообразных ролей, представителями разнообразных групп интересов, живыми сложными большими системами, обладающими всеми общесистемными свойствами и массой уникальных, индивидуальных свойств, качеств, особенностей. Отношения между организациями опосредованы людьми, представляющими организацию и выражающими ее интересы, поэтому и здесь важен человеческий фактор, который необходимо учитывать и проявлением которого следует управлять. Коммуникации в организациях имеют место между людьми, являющимися исполнителями разнообразных ролей, представителями разнообразных групп интересов, живыми сложными большими системами, обладающими всеми общесистемными свойствами и массой уникальных, индивидуальных свойств, качеств, особенностей.

Межличностные коммуникации - это коммуникации, которые осуществляются преимущественно на психическом уровне, под сильным влиянием эмоционального аспекта, в основе психология и социальная психология.

Групповые коммуникации - коммуникации внутри групп, межличностные и статусные, формальные, ролевые, и между группами, отражаемые в общении личностей, персонифицирующих или представляющих группы, а также личностей, идентифицирующих себя с группами. Это коммуникации, происходящие в переплетении формальных и неформальных, рациональных и эмоциональных, социальных и служебных и т.п. отношениях.

Коммуникации внутри групп связаны с достижением групповых результатов и трудовыми отношениями, но испытывают влияние межличностных отношений. Эффективные групповые коммуникации в решающей степени способствуют превращению группы в продуктивно работающую команду. Развал команды зачастую является следствием нарушения нормальных коммуникаций, усиления и преобладания личностных неприязненных отношений над деловыми.

Организационные коммуникации – это либо только формальные связи, определяемые административными актами, официальными организационно-распорядительными документами, т.е. связи, необходимые для исполнения организационных заданий, закрепления разделения и кооперации труда, его коммуникативного обеспечения, либо вся совокупность коммуникаций всех видов в организациях и между ними.

Вопросы для повторения:

1. В каких случаях уместно авторитарное лидерство? Какие плюсы оно имеет?
2. Выделите основные элементы трансформационного лидерства. Какие из них позволяют организации лучше развиваться и реагировать на изменения и подвижность внешней среды?
3. Дайте определение команде.
4. Перечислите и опишите командные роли.
5. Какова роль лидера в команде?
6. Раскройте содержание этапов развития команды.
7. Раскройте понятие коммуникация в организации.
8. Перечислите и охарактеризуйте этапы развития команды.

Раздел 2. Педагогика высшей школы

Тема 5. Педагогика высшей школы.

5.1. Предмет, задачи, категории педагогики высшей школы.

5.2. Принципы и методы педагогического исследования.

5.1. Предмет, задачи, категории педагогики высшей школы.

«Педагогика высшей школы – область знания, выражающего основные научные идеи, дающие целостное представление о закономерностях и существенных связях в учебно-познавательной, научной, воспитательной, профессиональной подготовке и всестороннем развитии студентов»

В первую очередь, нужно отметить, что педагогика высшей школы – это отрасль, раздел общей педагогики, а точнее будет сказать, профессиональной педагогики, изучающей закономерности, осуществляющей теоретическое обоснование, разрабатывающей принципы, технологии воспитания и образования человека, ориентированного на конкретно-профессиональную сферу действительности.

Предметом изучения педагогики высшей школы является лишь один этап в профессиональном становлении – процесс обучения и воспитания специалистов с высшим образованием.

Таким образом, будем понимать под педагогикой высшей школы – отрасль (раздел) общей (профессиональной) педагогики, изучающую основные составляющие (закономерности, принципы, формы, методы, технологии, содержание) образовательного процесса в вузе, а также особенности и условия (требования к процессу взаимодействия преподавателя и студента, требования к личности преподавателя и студента и др.) эффективного осуществления профессиональной подготовки будущего специалиста.

Приведем задачи профессиональной педагогики, которые можно отнести к задачам педагогики высшей школы как общее к частному. В них входят:

1. Разработка теоретико-методологических основ профессионального образования и методик проведения исследований в профессиональной педагогике.

2. Обоснование сущности, аспектов и функций профессионального образования.

3. Изучение истории развития профессионального образования и педагогической мысли.

4. Анализ современного состояния и прогнозирование развития профессионального образования в нашей стране и за рубежом.

5. Выявление закономерностей профессионального обучения, воспитания и развития личности.

6. Обоснование образовательных стандартов и содержания профессионального образования.

7. Разработка новых принципов, методов, систем и технологий профессионального образования.

8. Определение принципов, методов и способов управления профессионально-педагогическими системами, мониторинга профессионально-образовательного процесса и профессионального развития обучающихся.

Кроме этого можно выделить задачи педагогики вышей школы в практической области:

1. Формирование у преподавателей высшей школы умений и навыков методически обоснованного проведения всех видов учебной, научной и воспитательной работы.

2. Установление связи обучения, профессиональной подготовленности и формирование у студентов устойчивых навыков проведения исследовательской работы на основе этой связи.

3. Преобразование учебного процесса в процесс развития самостоятельного, творческого мышления.

4. Формирование, развитие, проявление педагогического мастерства с целью мобилизации студентов на разнообразные творческие действия.

5. Анализ социально-педагогического фактора, законов и особенностей формирования у студентов педагогических знаний, умений, навыков, педагогического сознания.

6. Вооружение педагогов психологическими знаниями.

Использование содержания педагогики высшей школы в качестве программы действий по организации и проведению многообразных видов педагогической деятельности.

К категориальному аппарату педагогики высшей школы, помимо общепедагогических, можно отнести профессионально-педагогические категории, такие как:

Профессиональное образование – процесс и результат профессионального развития личности посредством научно-организованного профессионального обучения и воспитания.

Профессиональное обучение – процесс и результат овладения обучающимися профессиональными знаниями, умениями и навыками.

Профессиональное воспитание – процесс и результат формирования профессионально важных качеств (различают общие и специальные ПВК).

Профессиональное развитие – развитие личности как субъекта профессиональной деятельности.

Профессиональное становление – результат профессионального развития: разряд, категория, класс, должность, степень, звание и др.

5.2. Принципы и методы педагогического исследования.

Научные исследования в области педагогики высшей школы представляют собой специфический вид познавательной деятельности, в ходе которой с помощью разнообразных методов выявляются новые, прежде не известные стороны, отношения, грани изучаемого объекта.

Любое научное исследование осуществляется в соответствии с теми или иными методологическими установками. Методология характеризует подход исследователя к анализу действительности. Она проявляется в его замысле, методике и результатах.

По своему характеру и содержанию исследования в области педагогики высшей школы разделяются на фундаментальные, прикладные и разработки.

Фундаментальные исследования призваны разрешать задачи стратегического характера. Их основными отличительными признаками являются: теоретическая актуальность, выражающаяся в выявлении закономерностей, принципов или фактов, имеющих принципиально важное значение, концептуальность, историзм, критический анализ научных положений и установок, новизну и научную достоверность полученных результатов. Однако главным критерием является решение перспективной задачи подготовки специалистов, а также те теоретические выводы, которые вносят серьезные изменения в логику развития самой науки.

Основными признаками прикладных исследований являются: приближенность их к актуальным запросам практики; сравнительная ограниченность выборки исследования; оперативность в проведении и внедрении результатов и др. Прикладные исследования опираются на исследования фундаментальные, которые вооружают их общей ориентацией, теоретическими и логическими знаниями, помогают определить наиболее рациональную методику исследования. В свою очередь, прикладные исследования дают ценный материал для фундаментальных исследований.

К разработкам в педагогике высшей школы относят методические рекомендации по тем или иным вопросам обучения и воспитания студентов, инструкции, методические средства и пособия. Они опираются на прикладные исследования и передовой

педагогический опыт. Отличительными чертами являются: целеустремленность, конкретность, определенность и сравнительно небольшой размер.

Специфический вид научно-педагогического исследования - изучение, обобщение и внедрение в практику передового опыта обучения и воспитания. Особенность исследований состоит в том, что они, как правило, вплетены в конкретную практику вузов и доступны каждому преподавателю.

Педагогический опыт - явление многоплановое, динамичное, противоречивое. С одной стороны - это живой, реальный педагогический процесс, повседневная педагогическая практика, а с другой - отражение этого процесса в сознании педагога - система его знаний, навыков и умений, привычек и личностных качеств, приобретенных в процессе обучения и воспитания людей.

В совершенствовании педагогического процесса вуза важная роль принадлежит передовому современному опыту и опыту педагогов-новаторов. Его отличают: высокая действенность, стабильность результатов, репрезентативность и перспективность. Он создается, как правило, на основе новой педагогической идеи и вместе с тем порождает новую идею. Главной целью изучения передового опыта учебно-воспитательной работы в учебных заведениях является повышение эффективности педагогического процесса, достижение более высокого уровня подготовки специалистов.

Применительно к исследованиям в области педагогики высшей школы системный подход предполагает изучение педагогического процесса вуза или его элементов как целостной системы. Исследуемая система, в свою очередь, состоит из множества элементов, каждый из которых сам является сложной системой. Между элементами системы существуют сложные связи и зависимости, совокупность которых составляет структуру данной системы. Система в целом и ее элементы выполняют различные функции, определенным образом связанные с ее структурой и внешним миром. Системный подход дает возможность моделировать изучаемые явления и исследовать их в состоянии развития и в разных условиях.

Системный подход предполагает многоуровневое и многоплановое изучение объекта, в ходе которого строится не одна, а ряд его моделей, отражающих объект на разных уровнях и срезах. При этом возможен синтез этих моделей в новой, целостной, обобщающей модели. Как показывает практика, диапазон системного подхода в педагогических исследованиях широк. Он позволяет разрабатывать как общие проблемы педагогики, так и многие вопросы методики воспитания и дидактики.

Исторически в педагогике сложились и активно используются следующие методы: теоретический анализ, наблюдение, эксперимент, беседа, письменный опрос (анкетирование), изучение результатов деятельности обучаемых и воспитуемых, педагогической документации, сравнительно-исторический метод. В педагогических исследованиях начали применяться математические и кибернетические методы. Имеются новые методы такие как: аналогия, формализация, моделирование.

Методология требует, чтобы методы исследования были адекватны цели и содержанию предмета науки. Особо следует сказать о математических методах и формализации. Их применение в педагогических исследованиях в соответствии с требованиями методологии и учетом своеобразия педагогических явлений оправданно и дает положительные результаты.

Педагогические исследования складываются из нескольких этапов. Основными из них являются:

– выбор и обоснование темы, предварительная разработка замысла и рабочего плана исследования;

– изучение литературных источников, соответствующей документации, предварительное ознакомление с опытом, уточнение условий обстановки, построение гипотезы, формулировка задач, разработка методики исследования;

– изучение педагогической практики, сбор фактического материала в целях проверки гипотезы;

– теоретический анализ добытого фактического материала;

– проверка выводов и рекомендаций;

– оформление результатов исследования;

– внедрение результатов исследования в повседневную практику.

Методика исследования педагогического процесса в целом и его отдельных сторон всегда индивидуальна. Она определяется замыслом, характером исследования, зависит от условий и имеющихся средств для намеченной работы. В ходе исследования она непрерывно совершенствуется.

Для педагогики высшей школы в настоящее время особенно важны экспериментальные исследования, позволяющие активно вмешиваться в изучаемое явление, вносить новые элементы в педагогический процесс в целях его совершенствования. Педагогическому эксперименту присущи следующие черты:

- преднамеренное, строго продуманное внесение в изучаемое явление чего-то принципиально нового в соответствии с задачами исследования и в целях проверки гипотезы;

- организация учебно-воспитательной деятельности, позволяющей видеть связи между явлениями и их взаимовлияние;

- проверка и контроль эксперимента, его сравнение с другими экспериментами, решающими аналогичную задачу с иных позиций;

- систематическая проверка количественных и качественных изменений, проведение контрольных срезов и, если требуется, внесение корректив; использование при необходимости вариационной статистики;

- объективный количественный и качественный анализ полученных результатов, теоретические обобщения, научные выводы и рекомендации.

Эффективность педагогического эксперимента состоит в том, что он позволяет создавать новый опыт в точно учитываемых условиях. Его успех во многом зависит от обстоятельной разработки и оригинальности гипотезы, от умелого выбора экспериментальных и контрольных групп (учебных групп, курсов, факультетов, вузов), четкости планирования, непременного учета всех основных условий и факторов, проверяемых в исследовании, их влияния на ход и результаты обучения, воспитания, развития человека. С особой тщательностью следует провести сравнение и сопоставление учебной деятельности и результатов экспериментальных и контрольных групп.

Результаты исследования оформляются в виде отчета о научной работе, научного доклада, статьи, брошюры, диссертации или монографии.

Оформлением результатов исследования не заканчивается работа над выдвинутой проблемой. Это должно быть внедрение обоснованных рекомендаций в повседневную практику вузов.

Таким образом, научные исследования в области педагогики высшей школы осуществляются в соответствии с методологическими принципами многими методами. Важнейшие требования методологии - постоянное стремление к внедрению результатов исследований в повседневную деятельность вузов и этим самым дальнейшее повышение качества подготовки будущих специалистов.

Вопросы для повторения:

1. Что составляет методологические основы педагогики высшей школы?
2. Раскройте сущность и содержание основных методов, используемых в педагогики высшей школы.
3. Раскройте сущность и содержание понятий «методология» и охарактеризуйте ее основные уровни.
4. Назовите предмет изучения педагогики высшей школы.

5. Каковы основные направления исследований в области педагогики высшей школы?

Тема 6. Приоритетные стратегии и тенденции развития высшего образования.

6.1. Современные стратегии модернизации высшего образования в России и за рубежом

Образование является одним из основных системообразующих институтов общества, реализующих широкий спектр общественно-значимых функций и находящихся под влиянием происходящих общественных трансформаций. Особенностью образования в современном мире является то, что оно одновременно выступает одним из самых консервативных институтов, сохраняющих и воспроизводящих традиционные формы и отношения, а с другой, – оно все более становится центром воспроизводства наиболее значимых инноваций и передовых практик, определяющих перспективы развития общества. Ряд глобальных трендов развития современного образования определяется общими мировыми тенденциями и находится под влиянием мировых общественных проблем.

Глобализация образования. Образование встроено в процесс всемирной экономической, политической, культурной интеграции и унификации, развертывающийся в последние десятилетние во всем мире. Проявлением этого является всеобщая унификация знания, в результате чего происходит выход национальных образовательных систем за пределы государственных границ, интернационализация образования и формирование единого мирового образовательного пространства и рынка образовательных услуг. Глобализация образования проявляется в гармонизации страновых систем образования между собой, унификации уровней образования и квалификационных рамок, открытости и трансграничности образования, возможности получать его из любой точки мира. В последнее время новый мощный импульс расширению глобализации образования дают информационные технологии и цифровизация образования, разрушающие национальные границы образования в принципе, и позволяющие говорить о формировании единого мирового цифрового образовательного пространства, определяющего новые конкурентные условия для всех игроков образовательного рынка.

Массовизация образования. Массовизация образования стала глобальным трендом образования в последние пятьдесят лет в связи расширением социальных функций государства, обеспечившего доступ к нему широких слоев населения, что привело к превращению образования из элитного в массовое. Влияние может оказать обратная сторона массовизации образования, выражающаяся в снижении его качества, определенной дискредитации образования, особенно более высоких ступеней, что уже приводит к снижению спроса на высшее образование среди населения разных стран. Это может стать угрозой финансовой стабильности университетов. Поэтому можно предположить, что новым драйвером станет идея «нового элитного» образования, которое будет ориентироваться на ограниченный круг людей и вернет принцип элитарности в школы и вузы, а реализовываться он будет на уникальном экспертном уровне очного (оффлайн) образования.

Демократизация образования. Демократизация образования проявляется в реализации и расширении прав каждого человека на образование, возможностей для самоорганизации и права выбора обучающих и обучающихся в образовательном процессе, поливариативности способов образовательной деятельности, многообразии образовательных систем и форм получения образования. Важным проявлением демократизации образования во всем мире является сокращение государственных функций в регулировании образования, развитие общественного управления, самоуправления и автономии образовательных организаций. Одной из современных форм демократизации образования явилось появление феномена массовых открытых образовательных курсов, которые выложили ведущие университеты мира на открытых цифровых платформах для

широкого пользователя без всяких ограничений. Это позволило университетам преодолеть все институциональные границы, существовавшие в образовании, продвинуть себя в мировом образовательном пространстве, а образовательный контент сделать максимально доступным из любой точки мира любому пользователю.

Технологизация образования. Технологизация образования, вылившаяся в настоящее время в «цифровую революцию», стала ведущим трендом развития образования. Сначала информационные, а теперь цифровые технологии кардинальным образом изменили образовательный ландшафт и конфигурацию, способствовали появлению новых сущностей в образовании. Из межличностного коммуникативного процесса оно, по сути, превратилось в технологический процесс, зависимый от использования развивающихся стремительными темпами информационных технологий. За последние несколько лет возникли принципиально новые образовательные онлайн-проекты. Также появился ряд проектов и платформенных решений в таких областях как управление учебным процессом, оценка и сертификация результатов обучения, социальные сети для преподавателей и студентов, исследователей и работодателей и т.д.

Образование в современной экономике рассматривается не как затратная сфера наряду с социальной помощью, пенсионной системой, госаппаратом, обороной и безопасностью, а как инвестиционная сфера, определяющая темпы и качество экономического роста. Значение человеческого капитала еще более возросло в XXI веке в условиях увеличения роли знаний и инноваций в экономике и усиления неопределенности. Поэтому в последние десятилетия ключевым элементом человеческого капитала становится интеллектуальный капитал, который представляет собой способность генерировать и осваивать инновации. Он приобретает характер решающего фактора для модернизации экономики, перехода к новым технологическим укладам и для ответа на вызовы глобальной конкуренции. Растет спрос населения на высшее образование, что также способствует росту инвестиций в данную сферу. Определенным отражением этого является рост стоимости высшего образования.

Непрерывность и пожизненность образования. Среди общемировых тенденций образования особо следует выделить быстрое развитие непрерывного образования (образования на протяжении всей жизни). Необходимость этого обусловлена не только ускорившимися процессами технико-технологического и информационного прогресса, но и особенностями социально-экономического и демографического развития. Институционализация непрерывного образования осуществляется в разных формах и на разных уровнях. Оно формируется как в вертикальной (образование по уровням в течение всей жизни), так и в горизонтальной (параллельное обучение на программах разного уровня, самообразование) плоскостях. Реагируя на изменяющиеся потребности рынка труда, сфера образования все более приобретает многоуровневый и многоформатный характер. Особо бурное развитие получает неформальное образование, которое активно начинает конкурировать с формальным образованием, а дополнительное с основным. Для университетов появляется угроза оказаться в арьергарде этих процессов в том случае, если не удастся диверсифицировать спектр реализуемых образовательных услуг и выстроить у себя систему непрерывного образования для различных сегментов рынка труда и образовательных потребностей граждан разных возрастов на основе современных цифровых технологий. Поэтому вузы вынуждены не просто своевременно реагировать на эти процессы, а обеспечивать опережающие реакции, создавая конкурентоспособные условия для обучения на протяжении всей жизни Растущая интернационализация одинаково характерна как для школьного, так и для университетского образования во всем мире. Студенты из-за рубежа – это наиболее мобильные молодые люди, финансово обеспеченные и обладающие большими способностями и талантом, что позволяет им поступать в ведущие вузы мира. Именно за таких людей сегодня развернулась серьезная мировая конкуренция, которую зачастую ведут уже не отдельные университеты, а образовательные консорциумы или даже страны. На этом фоне усиливается роль

международных образовательных стандартов и рейтингов, как в школьном образовании (PISA, PIRLS, TIMSS — наиболее известные международные системы оценки навыков школьников), так и в высшем образовании (TOEFL, рейтинги мировых университетов, Болонский процесс и система унификации результатов образования). Указанные глобальнее тренды развития образования определяют общий контекст, в котором разворачиваются основные тенденции развития высшего образования в мире и в России.

Основные направления стратегического развития ведущих университетов мира.

В настоящее время под влиянием глобальных трендов развития образования активно идут процессы трансформации высшей школы. В условиях нарастающей конкуренции на мировом образовательном пространстве перед университетами мира встают более серьезные задачи, чем раньше. Они вынуждены конкурировать не только в учебной и научной работе, но и в сфере создания инноваций, влияния на экономический рост, в решении основных мировых проблем. В соответствии с этим выстраиваются основные направления стратегического развития ведущих университетов мира, которые находят отражение в их стратегических документах, программах развитиях, модельных решениях. В качестве наиболее значимых: Эпоха гринфилда в образовании. Исследование SEDeC. Центр образовательных разработок Московской школы управления Сколково.

Глобализация университетского образования и науки на основе цифровизации. Формирование глобального образовательного электронного пространства, создание нового типа открытых информационных ресурсов «без границ», наиболее известным из которых является Coursera (Free Online Courses From Top Universities) – это самый значимый инновационный тренд трансформации высшего образования в мире, меняющий представление о возможностях и формате деятельности университета в целом. По существу, это – одна из первых информационно-образовательных моделей реализации идеи мета-университета (Meta University) – глобальной сети консорциума университетов и корпораций. Целью такого университета должно стать решение глобальных проблем, выходящих за рамки региональных и национальных приоритетов – здравоохранения, экологии, международного взаимодействия и ряда других. ИКТ-форматы все в большей степени становятся реальностью образовательного процесса в западной традиции.

Коммерциализация научных идей. Современные ведущие университеты располагают собственными центрами трансфера технологий, технопарками, в частности, предоставляющими студентам возможности организации стартапов, малых фирм с целью разработки технологической продукции, ее последующего патентования и маркетингового продвижения. Фактически студенты создают рабочее место сами в кампусах университетов, совмещая исследовательскую деятельность с образовательным процессом и бизнесом. Так происходит постепенная и преимущественная приватизация университетской системы бизнесом, все больше превращающим научные технологии в коммерческий продукт. Что касается юникорнов (unicorn – англ. единорог) - стартап, оценка рыночной стоимости которого превышает 1 млрд дол.), то по оценкам экспертов Стэнфордский университет - самое популярное место, где учились основатели юникорнов - 63 лучших предпринимателя мира.

Личностно-ориентированное обучение студентов. Данная стратегия означает диверсификацию образования путем обеспечения индивидуального подхода к обучающемуся с целью раскрытия потенциала каждого студента. Образовательное учреждение все в большей степени соответствует электронному каталогу товаров и услуг, где можно одним кликом мыши выбрать необходимый образовательный контент. Таким образом, у студентов есть возможность индивидуального выбора образовательной программы.

Служение обществу. Направлено на построение партнерских отношений между подразделениями университета и за пределами кампуса, создание атмосферы инклюзивности, благополучия и вежливости. В рамках направления могут реализовываться следующие инициативы: - стратегическое корпоративное и общественное взаимодействие:

разработка эффективных моделей взаимодействия с бизнесом и сообществом, гражданами; использование отношений с корпоративными партнерами для максимального создания рабочих мест; - здоровье и оздоровление: повышение участия в инициативах, которые направлены на поддержание физического и психического здоровья научно-педагогического сообщества, персонала, студентов университета; построение взаимовыгодных деловых партнерств для борьбы с неравенством в отношении оздоровления местного населения; - осуществление положительного влияния на мир: постоянное совершенствование процесса оказания услуг, мониторинг расширения спектра услуг студентам, преподавателям, населению; рост вовлеченности выпускников в деятельность университета (Университета Говард, США).

Повышение производительности труда и эффективности затрат. Направление ставит своим результатом повышение эффективности и результативности деятельности университета за счет вложения инвестиций в обновленные технологии и системы в целях содействия автоматизации процессов. Основной задачей является эффективная работа на всех уровнях организации. В рамках данного направления могут реализовываться следующие инициативы: - операционное превосходство и соответствие: оптимизация ключевых процессов, процедур, ресурсов университета, сокращение времени завершения основных процессов на 30%; регулярное проведение мониторинга эффективности и результативности административных, академических процессов для обеспечения соответствия со стандартами качества, установленными нормами; - инфраструктура университета и её устойчивость: снижение потребления электроэнергии на 20%; определение приоритетов в сфере строительства и реконструкции зданий университета с учетом его научно-технического развития; - обслуживание клиентов и взаимодействие: проведение обучения по обслуживанию клиентов в рамках всего университета в целях улучшения качества обслуживания студентов, сотрудников и внешних клиентов; вовлечение всех сотрудников университета в общественную жизнь, включая обсуждение новых идей, процессов развития университета, повышения производительности труда, эффективного управления активами университета (Университет Говард, США).

Достижение финансовой устойчивости. Направлено на достижение финансовой устойчивости университета за счет диверсификации доходов, оптимизации процессов, обеспечения прозрачной отчетности и надежного управления персоналом. Целью является повышение качества подготовки финансовой отчетности, улучшение финансового состояния университета. В рамках данного направления развития могут реализовываться следующие инициативы: - стратегическое планирование долгосрочных инвестиций; - увеличение эндаумент-фонда в целях роста поддержки студенческих стипендиальных программ и инвестиций в инфраструктуру; - сбор средств (фандрайзинг): создание инфраструктуры для устойчивой корпоративной и частной благотворительности в целях проведения активной и успешной кампании по накоплению капитала; сбор средств согласно основным приоритетам и целям университета; повышение уровня участия выпускников университета в сборе средств; увеличение числа частных доноров на 10% ежегодно (Университет Говард, США).

Финансирование университета в биткоинах. Университет Никосии на Кипре стал первым в мире учебным заведением, принимающим оплату за обучение в криптовалюте (2013 г.). По мнению финансового директора университета Христоса Влахоса, биткоины позволят облегчить распространение финансовых услуг в тех регионах мира, где не хватает развитой банковской сети. Согласно информации с официального сайта Университета Никосии, оплата в биткоинах осуществляется и в настоящее время. Некоторые американские вузы 12–18 месяцев назад начали приобретать криптовалюту на различных биржах, включая Coinbase. Зачастую средства поступают в университетские фонды в форме благотворительных пожертвований. На них финансируется образовательная и исследовательская деятельность. Также цифровые активы используются в качестве инвестиций. Самый крупный фонд оказался у Гарвардского университета.

Рост рынка управления онлайн-программами университетов. Аутсорсинг в сфере управления онлайн-образованием. OPM (Online Program Management) модель. Развитие онлайн-программ и курсов университетами, растущее внедрение технологий, увеличение количества мобильных устройств являются одними из основных факторов, которые, как ожидается, будут стимулировать рост рынка управления онлайн-программами. Все больше университетов используют бизнес-модель OPM. Сейчас можно отметить резкий рост числа университетов, которые начали передавать управление своими онлайн-программами сторонним поставщикам. Фактически онлайн управление программами (OPM) — это набор услуг для образовательного учреждения от зачисления учащегося до маркетинга, от повышения осведомленности потенциальных студентов до их зачисления на первый семестр в университете и удержания их на протяжении всего периода обучения. Что касается высшего образования, учреждения, которые сосредоточены на разработке или расширении своих онлайн программ, должны принять твердое решение о том, использовать ли стороннюю организацию OPM (Online Program Management) или пытаться создать и реализовать собственными силами. Наряду с бизнес-моделью, в которой операторы управления программами занимаются всеми аспектами разработки и реализации программ, появилась альтернатива «а ля карт», позволяющая учреждениям выбирать компоненты управления программами для передачи на аутсорсинг. Стратегические задачи и передовые практики ведущих университетов мира позволяют определять основные ориентиры развития высшего образования на глобальном и национальных уровнях.

3. Тенденции развития российского образования в сравнении с мировыми образовательными системами. Общий уровень образования населения России В современном мире экономическое благосостояние страны и ее граждан тесно связано с образованием. Это подтверждается тем, что между величиной валового внутреннего продукта (ВВП) на одного жителя и долей населения со средним профессиональным и высшим образованием, что соответствует третичному образованию по Международной стандартной классификации образования (МСКО 11), имеется статистически значимая положительная связь. Население Российской Федерации — одно из наиболее образованных в мире, уступая Ирландии и Канаде.

Третичное образование. Интересная особенность была отмечена в нашей стране: в среднем, молодежь в нашей стране получают образование раньше, чем сверстники за рубежом. При этом, «…если охват образованием населения младшей возрастной группы в России значительно превышает показатели развитых стран, то начиная с 23-24 лет мы уже отстаем от средних значений…, а в возрасте 27-28 лет доля студентов в общей численности российского населения опускается на уровень экономически менее развитых, чем Россия, стран». Отсюда, возникают проблемы при трудоустройстве выпускников. Еще одно существенное отличие участия российских молодых людей в третичном образовании от сложившихся в мире моделей заключается в том, что значительная - бóльшая, чем в других странах, - их часть учится по программам СПО, готовящим специалистов среднего звена (эквивалент «коротких программ» третичного образования МСКО). Кроме того, в нашей стране гораздо выше доля студентов, обучающихся по неочной форме.

Структура выпуска по программам. В структуре выпуска по программам третичного образования обращает на себя внимание значительное преобладание в России выпускников по инженерным специальностям. Можно предположить, что это реакция на существенное снижение выпуска по промышленным специальностям. Еще одно направление, по которому доля выпускников в России превышает средние значения по развитым странам, — услуги. По всем остальным областям знаний доля выпускников в России ниже. Особенно велико это отставание в таких областях, как естественные науки и математика, здравоохранение и социальная защита, искусство и гуманитарные науки.

Привлекательность российского образования. На долю России приходится 6,8% мирового рынка третичного образования (процент иностранных студентов, обучающихся в стране в общей численности иностранных студентов в мире). По этому показателю наша

страна уступает только США, Соединенному Королевству и Австралии. Но если рассматривать уровни третичного образования по отдельности, то обнаруживается, что высокие показатели Российской Федерации обеспечены главным образом привлечением иностранных студентов на программы бакалавриата. По доле студентов на магистерских программах мы уступаем, помимо перечисленных, еще Франции и Германии, а по доли аспирантов опускаемся на 14-е место. Иными словами, к нам едут учиться в основном по наиболее простым программам. Если в вузах стран ОЭСР студенты из других стран, входящих в эту организацию, составляют 27%, то доля иностранных студентов из стран ОЭСР в России составляет только 1%. Среди иностранных студентов России 4% — из Китая, 2% — из Индии, а 43% — из соседних с Россией стран. Следует отметить также, что Россия — нетто-экспортер образования, то есть число приехавших к нам иностранных студентов больше, чем число молодых россиян, обучающихся за рубежом. Соотношение иностранных студентов в России и россиян, обучающихся за рубежом, составляет 5:1. Половина российских студентов за рубежом приходится на пять стран: Германию (17%), Чехию (10%), США (9%), Соединенное Королевство (7 %) и Францию (6%).

Расходы и финансирование в образовании. Уровень расходов в начальном и среднем образовании широко варьируется по странам. Средняя величина расходов на 1 школьника по странам ОЭСР составляет 10 тыс. долл. в год на 1 ученика. В России этот показатель ниже более чем в два раза (4,2 тыс. долл.). В высшем образовании размах вариации еще шире: от 7,1 тыс. долл. на студента в год в Латвии до 52 тыс. долл. в Люксембурге при среднем значении по странам ОЭСР 16,5 тыс. долл. Россия и здесь существенно отстает от других стран: в нашей стране расходы на 1 студента вуза составляют 9,5 тыс. долл. в год. Участие государства в расходах на дошкольное образование варьируется по странам очень широко: от 98% в Люксембурге до менее 50% в Великобритании и Японии. При этом наблюдается, хотя и слабо выраженная, тенденция: чем выше уровень экономического развития страны, тем большую долю расходов на дошкольное образование берет на себя государство. В России доля государства в расходах на дошкольное образование составляет 88%. Финансирование третичного образования в Российской Федерации на 64% происходит за счет государственных источников, 23% средства домохозяйств, около 12% - прочие негосударственные источники, 1% - международное финансирование. В России доля расходов на оплату труда в среднем образовании равна 83% от текущих расходов, в третичном - 67%, что близко к средним показателям развитых стран, которые составляют 80% и 66% соответственно. Но важно отметить, что по отдельным странам этот показатель варьируется значительно: от 62% (Чехия) до 95% (Колумбия) в среднем образовании и от 57% (Италия) до 100% (Колумбия) в третичном образовании.

Вопросы для повторения:

1. Как вы понимаете, что такое глобализация образования?
2. В чем минусы массовизации образования?
3. Что такое технологизация образования?
4. Как вы понимаете непрерывность и пожизненность образования?
5. Перечислите тенденции развития российского образования в сравнении с мировыми образовательными системами.
6. Перечислите основные направления стратегического развития ведущих университетов мира.

Тема 7, 8. Формы организации обучения в вузе: традиции и инновации.

7.1. Трехмерная модель систематики форм организации обучения.

7.2. Вузовская лекция. Игры. Семинары и конференции. Самостоятельная работа студентов. Проектно-творческая деятельность. Дистанционное обучение. Авторские технологии обучения.

8.1. Научно-исследовательская работа студентов.

8.2. УИР как часть профессиональной подготовки студентов. Формы организации НИР в вузе.

8.3. Защита интеллектуальной собственности.

7.1. Трехмерная модель систематики форм организации обучения.

В высшей школе сегодня успешно применяют такие формы организации обучения, как: лекции, семинары, лабораторные работы, практические занятия, специальные и функциональные (деловые) игры, теоретические (научно-практические) конференции, тестирование, консультации, подготовка рефератов, индивидуальные контрольные собеседования, самостоятельная работа обучаемых, производственная практика, курсовые работы (проекты, задачи), презентации, выпускные квалификационные ра-боты и многие другие, обусловленные творческим подходом педагогических коллективов вузов к организации и проведению образовательного процесса.

В настоящее время с появлением в вузах современных информационных средств, мультимедийного и интерактивного оборудования стремительно развиваются такие организационные формы, как дистанционное и открытое обучение, обучение в компьютерных классах и лабораториях, обучение с помощью «Кейс технологий» и т. п.

Лекция и самостоятельная работа студентов остаются основными формами обучения. Форма организации обучения выбирается с учетом целей, особенностей содержания учебного материала, адекватных им методов и средств обучения, места и времени проведения занятий.

Цель лекции, как правило, в основном ориентирована на формирование знаний, а не практических умений.

Практические занятия имеют характерную особенность в целях, поскольку направлены на развитие практических умений и навыков студентов, а также особенности в выборе методов преподавания и учения.

Форма организации обучения - это целостная системная характеристика процесса обучения с точки зрения особенностей взаимодействия преподавателя и студента, соотношения управления и самоуправления, особенностей места и времени обучения, количества студентов, целей, средств, содержания, методов и результатов обучения.

Многомерный анализ существующих форм организации обучения позволяет представить их в виде трехмерной модели систематики форм организации обучения (рис. 3).

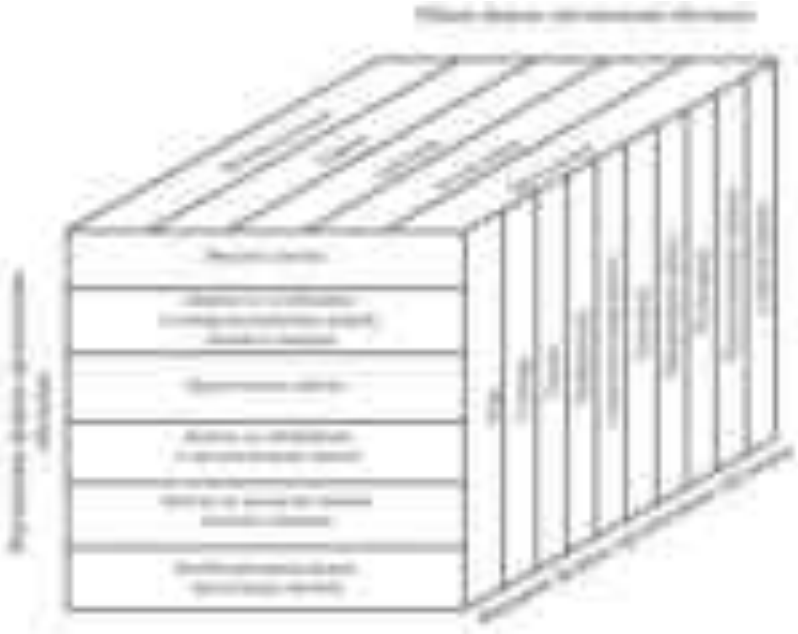


Рисунок 3 - Трехмерная модель систематики форм организации обучения

Внешняя форма организации обучения - это лекция, семинар, конференция, самостоятельная работа, игра, экскурсия, психодрама и др. Если взять цели, содержание, методы, средства, соотношение педагогического управления и самоуправления студентов, то достаточно видоизменить хотя бы один элемент, как видоизменяется внешняя форма организации обучения. Важное значение для понимания особенностей и возможностей эффективного функционирования той или иной формы организации обучения имеет ее структура, которая характеризует специфику ее внутренней организации, отражающей способы взаимодействия ее элементов. Основанием, для выделения внутренней формы организации обучения, является структурное взаимодействие элементов с точки зрения доминирующей цели обучения.

В основу общих форм организации обучения положены характеристики особенностей коммуникативного взаимодействия как между преподавателем и студентами, так и между самими студентами. При парном обучении преподаватель общается с двумя студентами, которые в свою очередь активно взаимодействуют. При групповом обучении общение преподавателя осуществляется с группой студентов из трех и более человек, которые в свою очередь имеют свои общие цели учебной деятельности и осуществляют активное взаимодействие как между собой, так и непосредственно с преподавателем. При коллективной форме обучения студенты рассматриваются как целостный коллектив, имеющий своих лидеров — руководителей из среды студентов. Общие цели, задачи и активное взаимодействие между всеми студентами обеспечивают достаточно высокий уровень их сплоченности и взаимопонимания в процессе коллективной учебной деятельности. При индивидуальной форме организации обучения преподаватель адаптирует степень сложности, трудности заданий, оказывает помощь с учетом знаний, умений и личностных качеств студентов. При фронтальной форме обучения преподаватель работает со студентами всей группы, следит, чтобы в едином темпе студенты продвигались к единой цели.

7.2. Вузовская лекция. Игры. Семинары и конференции. Самостоятельная работа студентов. Проектно-творческая деятельность. Дистанционное обучение. Авторские технологии обучения.

Выделяют следующие типы лекций: информационная, проблемная, лекция-визуализация, лекция вдвоем, лекция с заранее запланированными ошибками, лекция-пресс-конференция. Их цель - переход от простой передачи информации до активного освоения содержания обучения с одновременным запуском механизмов теоретического мышления и всей структуры психических функций. В данном процессе происходит усиление социального контекста в формировании профессионально важных качеств специалиста.

Информационная лекция – способ передачи готовых знаний обучающимся посредством монологической формы общения. Незаменим при передаче большого объема информации. Структура: вступление (формулировка темы, цели, изложение плана, характеристика рекомендуемой литературы, связь предыдущих лекций с новым материалом), основная часть (изложение содержания в соответствии с планом), заключение (подведение итога, возможности использования информации в практической деятельности, ответы на вопросы слушателей).

Проблемная лекция – тип лекции, на котором процесс познания студентов приближается к поисковой, исследовательской деятельности. При этом обеспечивается достижение трех основных целей: усвоение студентами теоретических знаний; развитие теоретического мышления; формирование познавательного интереса к содержанию учебного предмета и профессиональной мотивации. Задача преподавателя заключается не столько в передаче информации, сколько в развитии научного знания и способов их разрешения. Новое знание вводится как неизвестное для студентов. Студент не просто перерабатывает информацию, а переживает ее усвоение как субъективное открытие еще

неизвестного для себя знания. Учебный материал представляется в форме учебной проблемы, которая фиксирует некоторое противоречие (научные проблемы). Неизвестным является ответ на вопрос, который разрешает противоречие, переживаемое студентом как интеллектуальное затруднение. Для проблемного изложения отбираются узловые разделы курса, которые являются важными для будущей профессиональной деятельности. С помощью особых методических приемов (постановка проблемных и информационных вопросов, выдвижение гипотез, обращение к студентам «за помощью» и др.) преподаватель побуждает студентов к совместному размышлению, дискуссии. Лекции проблемного характера дополняются семинарскими занятиями, которые организуются как дискуссии.

Лекция-визуализация является результатом принципа наглядности, содержание которого меняется под влиянием данных психолого-педагогической науки, форм и методов активного обучения. Подготовка лекции-визуализации преподавателем состоит в перекодировании, переконструировании учебной информации по теме лекции в визуальную форму для предъявления студентам через технические средства обучения (схемы, рисунки, чертежи, презентации и т.п.). Чтение лекции-визуализации сводится к развернутому комментированию преподавателем подготовленных визуальных материалов, полностью раскрывающих тему данной лекции. Здесь важна определенная визуальная логика и ритм подачи материала. Важно учитывать цвет, графический дизайн, сочетание словесной и наглядной информации, дозировка подачи материала, мастерство и стиль общения преподавателя с аудиторией. Данный тип лекции лучше всего использовать на этапе введения студентов в новый раздел, тему, дисциплину. Основная трудность данного типа лекции состоит в выборе и подготовке средств наглядности, дидактически обоснованной режиссуре процесса ее чтения с учетом психофизиологических возможностей студентов, уровня образования и профессиональной принадлежности.

Телелекция – это один из видов лекций, опирающийся на современные аудио-, видеосредства и коммуникационные технологии обучения. Она может быть проведена без обратной связи, т.е. сначала записывается и затем тиражируется. Можно использовать и обратную связь, если лекция проводится в телестудии и применяется телефонная связь со слушателями, которые могут задать лектору, интересующий их вопрос.

Лекция вдвоем – процесс моделирования реальных профессиональных ситуаций, обсуждение теоретических вопросов с разных позиций двумя специалистами (теоретиком и практиком, сторонником и противником того или иного решения). В процессе «лекции вдвоем» создается проблемная ситуация, развертывается система доказательств, обосновывается конечный вариант совместного решения. «Лекция вдвоем» особенно эффективна в случаях, когда целями обучения выступают формирование теоретического мышления, воспитание убеждений, необходимость развития у студентов умений оперативно анализировать профессиональные ситуации, выступать в роли экспертов, отбирать неверную или неточную информацию.

Лекции с заранее запланированными ошибками. Преподаватель закладывает в содержание лекции определенное количество ошибок содержательного, методического или поведенческого характера. Список таких ошибок преподаватель приносит на лекцию и предъявляет их студентам в конце. Лектор строит изложение таким образом, чтобы ошибки были тщательно «замаскированы». Задача студентов заключается в том, чтобы по ходу лекции отмечать в конспекте замеченные ошибки и назвать их в конце лекции. На разбор ошибок отводится 10–15 минут. В ходе этого разбора даются правильные ответы на вопросы – преподавателем студентам или совместно. Лекцию с запланированными ошибками целесообразно применять в качестве контроля знаний, диагностики трудностей усвоения материала.

Лекция-пресс-конференция близка к соответствующей форме профессиональной деятельности с некоторыми изменениями. Назвав тему лекции, преподаватель просит слушателей письменно задать ему вопрос по данной теме. Каждый слушатель должен в течение 2-3 минут сформулировать наиболее интересующий его вопрос, написать на

бумажке и передать преподавателю. Затем лектор в течение 3-5 минут сортирует вопросы по их смысловому содержанию и начинает читать лекцию. Изложение материала строится не как ответ на каждый заданный вопрос, а в виде связного раскрытия темы, в процессе которого формулируются соответствующие ответы. В завершение лекции преподаватель проводит оценку вопросов как отражение знаний и интересов слушателей. Данный тип лекции лучше всего проводить в начале изучения темы или раздела (выявить круг интересов и потребностей обучаемых, степень их готовности к работе, отношение к работе), в середине (на привлечение внимания студентов к узловым моментам содержания учебного предмета, уточнение представлений преподавателя о степени усвоения материала, систематизацию знаний студентов, коррекцию выбранной системы лекционной и семинарской работы по курсу) и в конце (подведение итогов лекционной работы, определение перспектив развития усвоенного содержания в последующих разделах). Лекцию данного рода можно провести и по окончании всего курса с целью обсуждения перспектив применения теоретических знаний на практике как средства решения задач освоения материала последующих учебных дисциплин, средства регуляции будущей профессиональной деятельности.

Педагогическая (дидактическая) игра — это такая форма организации обучения, воспитания и развития личности, которая осуществляется педагогом на основе целенаправленно организованной деятельности студентов, которая изначально мотивирована на успех, осуществляется по специально разработанному сценарию и правилам, максимально опирается на самоорганизацию обучающихся; воссоздает или моделирует опыт человеческой деятельности и общения. На рисунке 4 показана классификация игр.

По целевой ориентации среди педагогических игр могут быть выделены: дидактические (они позволяют организовать различные виды учебной деятельности; сформировать познавательные и практические умения, углубить знания); воспитывающие (ориентированные на воспитание нравственных, эстетических, коммуникативных, волевых и других качеств личности); контролирующие (они одновременно или специально могут выполнять и функции контрольно-оценочной деятельности).

Рисунок 4 – Педагогические (дидактические) игры

Семинарская форма обучения. Главная цель – обеспечить студентам возможность практического использования теоретических знаний в условиях, моделирующих форм деятельности научных работников, предметный и социальный контексты этой деятельности. На семинарском занятии студенты должны научиться выступать в роли докладчиков и оппонентов, владеть умениями и навыками постановки и решения интеллектуальных проблем и задач, доказательства и опровержения, отстаивания своей точки зрения, демонстрации достигнутого уровня теоретической подготовки.

По большинству учебных дисциплин семинарские занятия целесообразно проводить в форме дискуссий, руководимых преподавателем. На семинаре отрабатываются важнейшие темы и разделы учебной программы. Широко распространено также обсуждение рефератов или докладов, подготовленных студентами.

Наибольшее распространении в последнее время получают спец-семинары – семинары исследовательского типа с независимой от лекционного курса тематикой, целью которых является углубленное изучение отдельных научно-практических проблем, с которыми столкнется будущий специалист. Спец-семинар, руководимый обычно крупным специалистом, приобретает характер научной школы. На семинарские занятия выносятся узловые темы курса, усвоение которых определяет качество профессиональной подготовки; вопросы, наиболее трудные для понимания и усвоении. Проработка этих тем осуществляется не в условиях индивидуальной (выступление студентов «по очереди»), а в условиях коллективной работы, обеспечивающей активное участие в ней каждого студента. Содержание семинарских занятий должно отражать принцип проблемности, быть методической основой для развертывания дискуссии, творческого применения имеющихся знаний.

Семинар-дискуссия организуется как процесс диалогического общения участников, в ходе которого происходит формирование практического опыта совместного участия в обсуждении и разрешении теоретических проблем, теоретико-практического мышления будущего специалиста. Особенностью семинарского занятия как формы коллективной творческой работы является возможность равноправного и активного участия каждого студента в обсуждении теоретических позиций, предлагаемых решений, в оценке их правильности и обоснованности. Преподаватель заранее должен ознакомить студентов с правилами ведения дискуссии, возможными ролями. Это целесообразно сделать на предшествующих семинарам проблемных лекциях с использованием метода микродискуссии. По окончании семинара-дискуссии преподаватель может сделать общие выводы, подвести итоги, оценить вклад каждого и группы в целом в решение проблемы семинара.

Семинар с использованием «сократовского» метода обучения - формой совместной творческой деятельности преподавателя и студентов. Их суть – в самостоятельном определении обучающимися основных понятий, в раскрытии сущности и закономерностей изучаемого явления и процессов путем последовательной постановки преподавателем вопросов и поиска ответов студентами. Данный метод обучения требует кропотливой самостоятельной подготовки студентов и преподавателя к занятию, в ходе которой у обучающихся формируются исследовательские умения и навыки (умение ставить проблемные вопросы, анализировать проблемные ситуации, выдвигать гипотезы – предполагаемые ответы и т.д.). Использование «сократовской» беседы в ходе семинара позволяет выявить пробелы в знаниях студентов, повышает интерес к изучаемой дисциплине, способствует активному усвоению знаний, формирует и развивает навыки самостоятельной работы и ведения беседы.

Семинар с использованием метода конкретных ситуаций. На семинарском занятии преподавателем создаются конкретные ситуации, взятые из профессиональной деятельности специалистов. От студентов требуются глубокий анализ ситуации и решение поставленной задачи. Ситуационная задача может иметь несколько вариантов решения, которые окажутся приемлемыми в данной ситуации, что требует от специалиста умения

выбрать из них наиболее оптимальные. В практике применения метода анализа конкретных ситуаций обычно используются следующие виды ситуаций: ситуация-иллюстрация (демонстрация конкретного примера из практики, в котором проявляются способы действия должностных лиц, типовые алгоритмы решения задач, эффективность использования методов и приемов руководства и т.д.); ситуация-упражнение не может быть разрешена без обращения студентов к специальным источникам информации, литературе и справочникам. Обучающий эффект обеспечивается деятельностью всех участников семинара по анализу и решению ситуационных задач; ситуация-проблема заключает в себе проблемную задачу, которая стоит перед профессиональной практикой. Она может предъявляться студентам в виде текста, видеофрагмента, доклада, набора документов, отражающих состояние какого-либо объекта, процесса, события, или в форме выступления приглашенных специалистов перед студентами.

Практическое занятие – это, как правило, решение прикладных задач, образцы которых были даны на лекциях. Практические занятия преследуют следующие цели: помочь обучающимся систематизировать, закрепить и углубить теоретические знания; научить студентов приемам решения практических задач, способствовать овладению умениями и навыками в выполнении расчетов, графических и других видов заданий; научить студентов работать с книгой, документацией и схемами, пользоваться справочной литературой и прикладными программами; выработать у студентов умения учиться самостоятельно, т.е. овладевать способами и приемами самообразования и самоконтроля. Структура практического занятия может быть различной: вначале сам преподаватель демонстрирует способы решения определенного класса задач, а затем организует упражнения по решению подобных задач; студенты сразу приступают к самостоятельному решению задач, но при необходимости преподаватель дает пояснения, консультации.

Лабораторные занятия – одна из форм практической работы студентов, в которой путем проведения экспериментов осуществляются углубление и закрепление теоретических знаний, формирование умений и навыков в интересах профессиональной подготовки. Основными структурными элементами лабораторной работы являются: постановка темы и целей занятия; проверка уровня теоретических знаний, необходимых для работы; ознакомление студентов с содержанием лабораторной работы; групповое выполнение лабораторной работы; консультация преподавателя в процессе работы; обсуждение полученных результатов членами рабочей группы; письменный или устный отчет о выполненной работе; контроль и оценка результатов лабораторной работы.

В вузах применяют следующие виды лабораторных занятий: фронтальный (одновременное выполнение работы всеми студентами), по циклам (работы делятся на несколько циклов, соответствующих определенным разделам лекционного курса), индивидуальный (студенты могут одновременно работать над различными темами) и смешанный (комбинированный) тип.

Подготовка студентов к лабораторной работе осуществляется в часы самостоятельной работы с использованием учебников, конспектов лекций и методических материалов. Проведению лабораторного занятия предшествует сдача студентами коллоквиума (от лат. collogvium – беседа) – собеседования преподавателя со студентами по поводу предстоящей лабораторной работы, проверка глубины усвоения теоретического материала. Лабораторные занятия заканчиваются защитой результатов работы и полученных результатов.

Самостоятельная работа - планируемая, организационно и методически направляемая познавательная деятельность студентов, осуществляемая без прямой помощи преподавателя для достижения образовательных целей. Ядром самостоятельной работы является познавательная (учебная, научная, производственная) задача, предлагаемая студентам. Самостоятельная работа реализуется: непосредственно в процессе аудиторных занятий – на практических и семинарских занятиях, при выполнении лабораторных работ; в контакте с преподавателем вне рамок расписания – на консультациях по учебным

вопросам, в ходе творческого сотрудничества в рамках научно-исследовательских работ, при ликвидации задолженностей, выполнении индивидуальных заданий и т.д.; в библиотеке, дома, в общежитии, на кафедре при выполнении студентом творческих и учебных задач.

Рекомендуются следующие виды заданий: текущая работа с лекционным материалом, предусматривающая проработку конспекта лекций и учебной литературы; поиск и обзор литературы и электронных источников информации по индивидуально заданной проблеме курса; изучение материала, вынесенного на самостоятельную проработку; решение задач; подготовка к лабораторным занятиям; подготовка к практическим и семинарским занятиям; практикум по учебной дисциплине с использованием программного обеспечения; написание реферата по заданной проблеме; выполнение расчетно-графической работы; подготовка к контрольной работе или коллоквиуму; подготовка к зачету, экзамену; выполнение курсовой работы или проекта; участие в научных студенческих конференциях и семинарах; аналитический разбор научной публикации по заранее определенной преподавателем теме; анализ статистических и фактических материалов по заданной теме, проведение расчетов, составление схем и моделей на основе статистических материалов. В качестве форм контроля самостоятельной работы могут быть: тестирование; проверка контрольных работ; доклад по самостоятельно изученной теме; веерный экспресс-опрос; отчет по результатам выполненного проекта.

Проектно-творческая деятельность. В процессе проектно-творческой деятельности студентов применяются самые разнообразные методы (эксперимент, моделирование, мозговой штурм и др.). Проектно-творческая деятельность студентов - это одна из форм самостоятельной работы студентов, направленная на решение учебных и (или) научных проблем, творческих (исследовательских) задач и заданий, выполнение (решение) которых осуществляется студентом преимущественно самостоятельно на основе педагогических методов и средств проблемного и эвристического обучения.

Проектно-творческая деятельность студентов имеет большую вариативность и по некоторым критериям ее удается классифицировать:

1. По доминирующему методу выполнения проекта: исследовательские; творческие; практико-ориентированные; теоретические; информационные и др.

2. По количеству участников: индивидуальные (личностные); парные; групповые; коллективные.

3. По содержанию деятельности: учебные; научные; практические.

4. По продолжительности выполнения: краткосрочные; среднесрочные; долгосрочные.

5. По степени вовлеченности организаций: внутривузовские; межвузовские; международные.

В проектно-творческой деятельности студентов можно выделить несколько этапов.

I этап. Самоопределение, самоактуализация, мотивация. На нем идет определение с выбором темы, проблемы, над которыми студент хотел бы работать.

II этап. Организационное и информационное обеспечение. Организация, создание реальных условий для доступа студента к необходимой справочной, учебной, научной литературе, включая и использование Интернета, позволяющие ему «войти» в проблему, познакомиться с базовой информацией, расширить и углубить свои знания по предложенной теме, проблеме.

III этап. Выдвижение предположений, формулирование гипотез, идей, разработка проекта. Выполнение творческого, исследовательского задания на этом этапе происходит индивидуально или с небольшой поддержкой преподавателя.

IV этап. Планирование. Происходит более детальное планирование выполнения проекта, конкретизация целей и задач в разработке проекта.

V этап. Сбор дополнительной информации и выполнение проекта. Собирается, систематизируется, анализируется дополнительная информация, проверяется ранее

выдвинутая гипотеза, систематизируются и анализируются данные, которые ее подтверждают или опровергают.

VI этап. Оформление результатов выполненного проекта. На этом этапе происходит не просто оформление результатов проектной деятельности студентов, но и их более целостное и глубокое осмысление.

VII этап. Защита проекта. Осуществляется публичная защита проекта, дается общая оценка результативности проектно-творческой деятельности студента.

На всех этапах проектно-творческой деятельности студента возможны консультации и помощь преподавателя, которые варьируются как по содержанию, так и по форме в зависимости от мотивации и творческого потенциала студента.

Очень важно, чтобы «защита проектов», их оценка происходили на основе четко выделенных критериев: новизны, оригинальности, обоснованности, системности и глубины проработки проблемы теоретической (или) практической значимости. Защиту проекта лучше всего проводить в форме презентации проекта с использованием схем, таблиц и других средств наглядности.

В условиях вузовской практики наибольшее внимание уделяется курсовым и дипломным проектам.

Дистанционное обучение. Это форма индивидуального обучения, в процессе которого осуществляется их погружение в интерактивную технотронную обучающую среду, обеспечивающую полный контроль студента, индивидуальное планирование учебного процесса и тестирования на расстоянии из единого центра дистанционного обучения. Дистанционное обучение - это одна из форм заочного обучения, плюс общение с преподавателем через Интернет, в ходе которого студент получает учебные материалы и задания на свой компьютер, выполняет тесты и контрольные работы и отправляет их преподавателю. В процессе дистанционного обучения изучается теоретическая часть, выполняются практические задания и решаются контрольные работы, которые затем отсылаются преподавателю по электронной почте. В процессе обучения возможно общаться и задавать вопросы преподавателю по e-mail. При дистанционном обучении Вы имеете возможность сами выбирать последовательность изучения предметов и темп работы. Можете решать, сколько времени потратить на изучение того или иного курса. «Дистанционный» студент получает комплект материалов сразу при зачислении на занятия. В такой комплект входят не только учебники, но и тексты лекций, практикумы, задания для самостоятельной работы на разных носителях — традиционных бумажных, CD, аудио- и видеоносителях.

Вместе с новыми методами и технологиями обучения дистанционное обучение привносит в теоретическую педагогику и образовательную практику новые понятия и термины, в первую очередь к ним относятся:

— виртуальный класс (группа);
— поддержка обучения (поддержка студентов);
— учебные телекоммуникационные проекты;
— обратная связь;
— диалоговая технология;
— компьютерная связь;
— телеконференция;
— координатор, модератор, фасилитатор телекоммуникационного проекта (телеконференции).

Под виртуальным классом (группой) понимается общность студентов, взаимодействие между которыми при совместном выполнении ими учебных заданий происходит по компьютерным сетям.

Под поддержкой обучения (или поддержкой обучаемого) понимают любые материалы, информацию, поступающую от преподавателя к студенту, находящемуся в другой географической точке.

Учебный телекоммуникационный проект - совместная (коллективая) деятельность студентов, направленная на достижение некоторой модельной цели, которая носит не учебный характер и моделирует цель какой-либо научной или производственной деятельности. Важными отличительными чертами учебного телекоммуникационного проекта являются:

— его временная определенность и ограниченность (от двух недель до трех месяцев);

— использование компьютерных телекоммуникационных сетей и программных средств для обмена информацией между всеми участниками проекта, которые часто образуют виртуальную или квазивиртуальную группу;

— необходимость четкой организации деятельности студентов, которая устанавливается координатором проекта.

Обратная связь в дистанционном обучении — поток информации от педагога к дистанционному студенту на стадии оценивания педагогом деятельности учащегося, его продвижения и успехов и несущая реакцию педагога на успехи студентов, оценку его деятельности (одобрение или неодобрение).

Диалоговая технология — конфигурация программного обеспечения, оборудования, а также межличностного взаимодействия и деятельности, обеспечивающая свободное общение.

Телеконференция — способ обмена текстовыми сообщениями с некоторыми сообществами заинтересованных в этом людей.

Компьютерная связь — совокупность способов использования компьютеров и телекоммуникационных сетей в качестве инструментов для организации связи. Компьютерная связь включает в себя: электронную почту, которая позволяет направлять сообщения в почтовые ящики пользователей сети; телеконференции, которые позволяют направлять сообщения всем участникам одновременно; доступ к удаленным информационным источникам, например, библиотечным ресурсам, базам данных, серверам.

Авторские технологии обучения. При разработке авторских технологий следует учитывать, что главное в арсенале преподавателя - это он сам. Его голос, жесты, доброе, заботливое отношение к студентам, стремление соприкоснуться с духовным миром каждого студента и открыть свой, всякий раз неожиданно загадочный, новый мир.

Большинство преподавателей при разработке собственных, более гибких авторских технологий опираются не на какую-то одну, а несколько дидактических (педагогических) концепций, выстраивая их с учетом специфики и приоритетности решаемых задач обучения, воспитания и развития личности. Приведем девять этапов или «слагаемые любой педагогической технологии», выделенные в результате исследования В.П. Беспалько:

1 — анализ будущей деятельности студента;

2 — определение содержания обучения на каждой ступени обучения;

3 — проверка степени нагрузки студента и расчет необходимого времени при заданном способе построения учебного процесса;

4 — выбор организационных форм обучения и воспитания, наиболее благоприятных для реализации намеченного дидактического процесса;

5 — подготовка материалов (текстов, ситуаций) для осуществления мотивационного компонента дидактического процесса;

6 — разработка системы учебных упражнений, нацеленных на усвоение предметов с заданными показателями качества;

7 — разработка материалов (тестов) для объективного контроля за качеством усвоения студентами знаний и действий соответственно целям обучения и критериям оценки степени усвоения;

8 — разработка структуры и содержания учебных занятий, нацеленных на эффективное решение образовательных и воспитательных задач;

9 — апробация проекта на практике и проверка завершенности учебно-воспитательного процесса (достижения цепей с показателями усвоения К>0,7), коррекция проекта».

8.1. Научно-исследовательская работа студентов

Формы участия студентов в научно-исследовательской работе могут быть сведены к двум направлениям: учебно-исследовательской работе студентов, проводимой в учебном процессе и внесенной в учебные планы; учебно-исследовательской работе студентов, не связанной или косвенно связанной с учебным процессом. Эти направления и составляют систему научно-исследовательской работы студентов.

Научно-исследовательская работа студентов – одна из важнейших форм учебного процесса. Научные лаборатории и кружки, студенческие научные общества и конференции – все это позволяет студенту начать полноценную научную работу, найти единомышленников, с которыми можно посоветоваться и поделиться результатами своих исследований. Написание рефератов, курсовых, дипломных работ невозможно без проведения каких-то, пусть самых простых, исследований. Более глубокая научно-исследовательская работа, заниматься которой студента не обязывает учебный план, охватывает далеко не всех. Для тех же, кто проявляет интерес к творчеству и поиску, в вузах специально организуется дополнительная научно-исследовательская и творческая работа.

Внеучебная, вне сетки расписания, работа включает в себя большое многообразие видов деятельности студентов: участие в научных кружках; подготовку рефератов, докладов, сообщений; выступление с ними на факультетских и других научных конференциях; выполнение исследований по хозяйственным договорам или госбюджетной тематике в составе научных коллективов преподавателей; участие в студенческих конкурсах научно-исследовательских работ, олимпиадах, выставках студенческого творчества; подготовку статей в научные журналы, сборники научных работ, периодическую печать и др.

Часто старт студента в науку начинается именно с участия в работе кружка или иного студенческого научного объединения. Целями любого студенческого научного объединения являются развитие у молодежи творческого мышления через изучение методологических основ научной работы, освоение научной методики, способов и приемов изложения материала и обработки результатов научного исследования.

Демонстрация получаемых научных результатов обычно выходит за рамки собственно кружковой работы в виде выступлений с докладами и сообщениями по итогам научных исследований, участие в научных дискуссиях и т.д.

Существуют следующие виды студенческих научных объединений: студенческий предметный или тематический научный кружок, научная проблемная или творческая группа, научно-исследовательская лаборатория и др.

Научный кружок (тематический или предметный) - объединение студентов, аспирантов и других заинтересованных лиц, основанное на общности интересов, взглядов, идей с целью совместного научного творчества. Как правило, основными видами научных работ в таких объединениях являются: составление аннотаций по научной литературе и написание рефератов, овладение навыками проведения эксперимента и обработки результатов, проектирование и изготовление наглядных пособий, подготовка сообщений и выступлений на семинарах и конференциях и т.д. Предметный научный кружок чаще всего организуется при работе со студентами младших курсов по изучаемым ими учебным предметам и является первой ступенькой в «царство науки», и поэтому задачи перед его участниками ставятся несложные. Чаще всего это подготовка докладов и рефератов, которые потом заслушиваются на заседаниях кружка или на научной конференции.

Проблемные кружки. Все выше сказанное можно отнести и к проблемным, но следует учесть и некоторые отличия. Проблемный кружок может объединять собой студентов разных факультетов и курсов, а также, если при вузе имеются таковые, колледжей и лицеев.

Во главу угла может быть поставлена проблема, которой занимается научный руководитель кружка, или любая другая по его выбору. Проблемные кружки предполагают встречи с людьми, которые сталкиваются с проблемами, выбранными для рассмотрения, на работе и в быту, проведение различных викторин и КВН. Проблемный кружок может сочетать в себе элементы научного кружка, лаборатории и т.д.

Проблемные студенческие лаборатории. В них принимают участие студенты второго курса и старше. В рамках проблемных студенческих лабораторий осуществляются различные виды моделирования, изучение и анализ реальных документов, программ, деловых игр, а также практическая помощь предприятиям. Работа в такой лаборатории предполагает не столько изучение и анализ литературы, сколько постановку эксперимента, создание чего-то нового, способности студента к коллективной работе.

Научно-исследовательская лаборатория - студенческая группа, проводящая учебные исследования и научные эксперименты. В ней осуществляются различного вида пробы, опыты, моделирование, создание чего-то нового, изучение и анализ документов, проводятся деловые игры и т.д. Работа в лаборатории предполагает наличие определенного запаса знаний и навыков. Еще одной отличительной чертой лаборатории является преобладание коллективных форм работы над индивидуальными. Если в кружке студент отвечает, как правило, только за себя, то в лаборатории его тема исследования включена в общую тему и от правильности решения частных задач зависят общие результаты работы.

В лаборатории темы более конкретные, как правило, имеющие выход на практику. Поэтому студенты имеют больше возможностей быть приглашенными на работу в организации, выступающие заказчиками исследований. Студенческие лаборатории, работающие по научной хоздоговорной теме кафедры, получают от нее не только моральную поддержку, но и материальное вознаграждение.

Участие в научно-практических конференциях. Включают в себя не столько теоретические научные доклады, сколько обсуждение путей решения практических задач. Проходят на территории завода, управляющего органа, с которым вуз поддерживает отношения. Они способствуют установлению тесных дружеских связей между вузом и предприятиями, помогают студентам учиться применять изученную теорию на практике.

Участие в научно-производственных структурах, временных творческих коллективах преподавателей кафедры, в бюджетных и внебюджетных научных исследованиях, в том числе включенных в планы НИР университета. Деятельность таких коллективов осуществляется под руководством ведущего по данному научному направлению преподавателя, чаще всего профессора, доктора наук. Каждый член такого научного объединения работает на постоянной, возможно даже частично платной, основе имеет научного консультанта и строго индивидуальное задание. Отвечая за себя, член такого коллектива понимает, что от его вклада зависит результат общего большого научного проекта.

Тьюторство. Оно предполагает, что студент в течение всего периода обучения в вузе последовательно разрабатывает определенную тему под руководством одного преподавателя. При такой организации обучения студенты наиболее полно осваивают методы и специфику научной деятельности, приобретают навыки работы в научных коллективах и организациях, а их научные руководители отбирают для себя потенциальных аспирантов.

Любая научно-исследовательская работа, независимо от того, в какой организационной форме она осуществляется, проходит через строго определенные этапы, последовательность и взаимосвязь которых отражает «технологию научно-исследовательской работы»: постановка проблемы, определение объекта исследования; выбор и обоснование темы исследования; определение целей и задач исследования; выбор методов исследования; сбор и обработка информации об объекте исследования; построение модели функционирования объекта познания и его многоаспектное изучение с

применением различных методов исследования; оформление результатов исследования и их защита.

Выбор объекта исследования определяется объективными факторами, такими как его значимость, наличие нерешенной проблемы, ее актуальность, новизна и перспективность, и субъективными факторами – жизненным опытом, склонностями, интересами исследователя, научным руководителем и др.

От доказательства актуальности темы логично перейти к формулировке цели исследования, а также указать на конкретные задачи, которые предстоит решать в соответствии с этой целью (изучить, проанализировать, установить, выяснить, построить, обосновать, доказать и т.д.).

Первый этап научного исследования – выбор методов, которые служат инструментом в добывании фактического материала, что является необходимым условием достижения цели исследования. Второй этап – это проведение самого исследования с помощью выбранных методов и описание этого процесса, в котором освещаются методика, техника и результаты исследования. Третий этап научного исследования – обсуждение его результатов, которое ведется на заседаниях соответствующих кафедр, где дается предварительная оценка теоретической или практической значимости проведенной работы. Результаты исследования оформляются в виде научно-исследовательского отчета, а также докладываются студентом на научном семинаре или научно-практической конференции.

8.2. УИР как часть профессиональной подготовки студентов. Формы организации НИР в вузе

Источниками получения информации для студентов, кроме лекций преподавателя, являются учебники, учебные и методические пособия, научная литература (монографии, журналы), средства массовой информации и т.д.

Одним из основных видов самостоятельной учебно-воспитательной деятельности студентов является работа с книгой. Эффективность этой работы зависит от уровня сформированности таких умений, как: работа в библиотеке с каталогами и подбор литературы по определенной теме (проблеме); чтение и анализ текста; выделение узловых элементов информации; составление плана и конспекта по прочитанному тексту; цитирование; подготовка доклада к семинарскому занятию или к студенческой конференции; составление рецензии; оформление реферата.

Знакомство с книгой начинается с чтения аннотации (краткая характеристика печатного издания), включающая: сведения о целях, структуре и содержании работы, об авторе и достоинствах печатного издания, о тех, кому она предназначена. Помещается на обороте титульного листа книги.

Составление плана информационного текста. План текста – это перечень узловых вопросов, отражающих структуру его содержания (например, перечень вопросов, приведенных к каждой главе).

Конспектирование – это сжатое и последовательное письменное изложение содержания прочитанного. Нужно для того, чтобы: переработав любую информацию, передать ее в сокращенном виде; выделить в письменном тексте самое необходимое и нужное для решения учебной или исследовательской задачи; создать модель проблемы; упростить запоминание текста, облегчить овладение специальными терминами; накопить информацию для написания более сложной работы (доклада, реферата, курсовой, дипломной работы).

Цитирование. Цитата – это точная выдержка (часть текста) из какого-либо литературного источника. Она служит подтверждением выдвинутых автором положений и приводится в кавычках, точно по тексту оригинала (первоисточника). Пропуск слов, предложений, абзацев при цитировании обозначается многоточием; не допускается объединение в одной цитате нескольких отрывков, взятых из разных мест.

Библиографическая ссылка приводится с полной информацией о первоисточнике с указанием его номера из библиографического списка использованной литературы и страницы.

Рецензирование. Рецензия - статья, содержащая в себе критический обзор какого-либо научного или художественного произведения, либо отзыв на научную работу (диссертацию, монографию, учебник и т.д.). Она раскрывает содержание рецензируемой работы, дает критическую оценку работе.

Написание реферата (реферирование). Реферат – это сжатое изложение основной информации первоисточников на основе ее смысловой переработки. Он позволяет представить содержание печатных изданий в обобщенном виде. Реферат как разновидность учебной исследовательской работы студента должен включать: обоснование актуальности темы, цель, задачи исследования, анализ литературы по проблеме со ссылкой на первоисточники, основное содержание, заключение, список литературы.

Курсовые работы призваны приобщать студентов к исследовательской работе над проблемами, которые разрабатываются преподавателями; учитывать разнообразие интересов студентов в области выбранной специальности, а также тематику исследовательской работы на факультете. После того, как тема курсовой работы выбрана и согласована с научным руководителем, составляется календарный план, в котором определяются сроки выполнения курсовой работы.

Дипломные работы. Дипломная работа – одна из основных форм выпускных квалификационных работ, предусмотренных в качестве аттестационных испытаний. Тематика дипломных работ определяется вузом. Студенту предоставляется право выбора темы, он может также предложить свою тему с обоснованием целесообразности ее разработки. Дипломная работа проверяется научным руководителем, а затем после исправления ошибок направляется на рецензирование (кроме бакалавриата). В рецензии отмечаются: актуальности темы; полнота и обстоятельность изложения поставленной проблемы; эффективность использования выбранных методов для решения проблемы; достижение поставленной цели; практическая значимость результатов.

8.3. Защита интеллектуальной собственности.

Многообразие видов интеллектуальной деятельности обуславливает многообразие форм её результатов – объектов интеллектуальной собственности (ОИС). Выделяют две сферы возникновения ОИС: 1) научно-техническую и производственную; 2) гуманитарную (рис. 5). Разделяют ОИС на 3 группы в зависимости от институтов права, регулирующих правоотношения в связи с их созданием и использованием: 1) объекты авторского права и смежных прав; 2) объекты промышленной собственности (объекты патентного права); 3) производственные секреты (ноу-хау). Отличительная особенность авторского права в том, что охрана прав распространяется в отношении формы произведения, а не его содержания. В произведениях живописи, литературы охрана предоставляется не сюжету, а форме, в которой он выражен. Защищается не сюжет книги, а словесная (литературная) форма его выражения. Соответственно, один и тот же сюжет может быть использован и писателем, и киносценаристом. В отношении программ для ЭВМ охрана распространяется на совокупность команд, но не на решаемые программой задачи и алгоритмические процедуры, которые она реализует. Наоборот, для объектов промышленной собственности (изобретений, полезных моделей) и производственных секретов (ноу-хау) приобретаемые права распространяются в отношении их содержания и, как правило, не зависят от конкретной формы реализации.

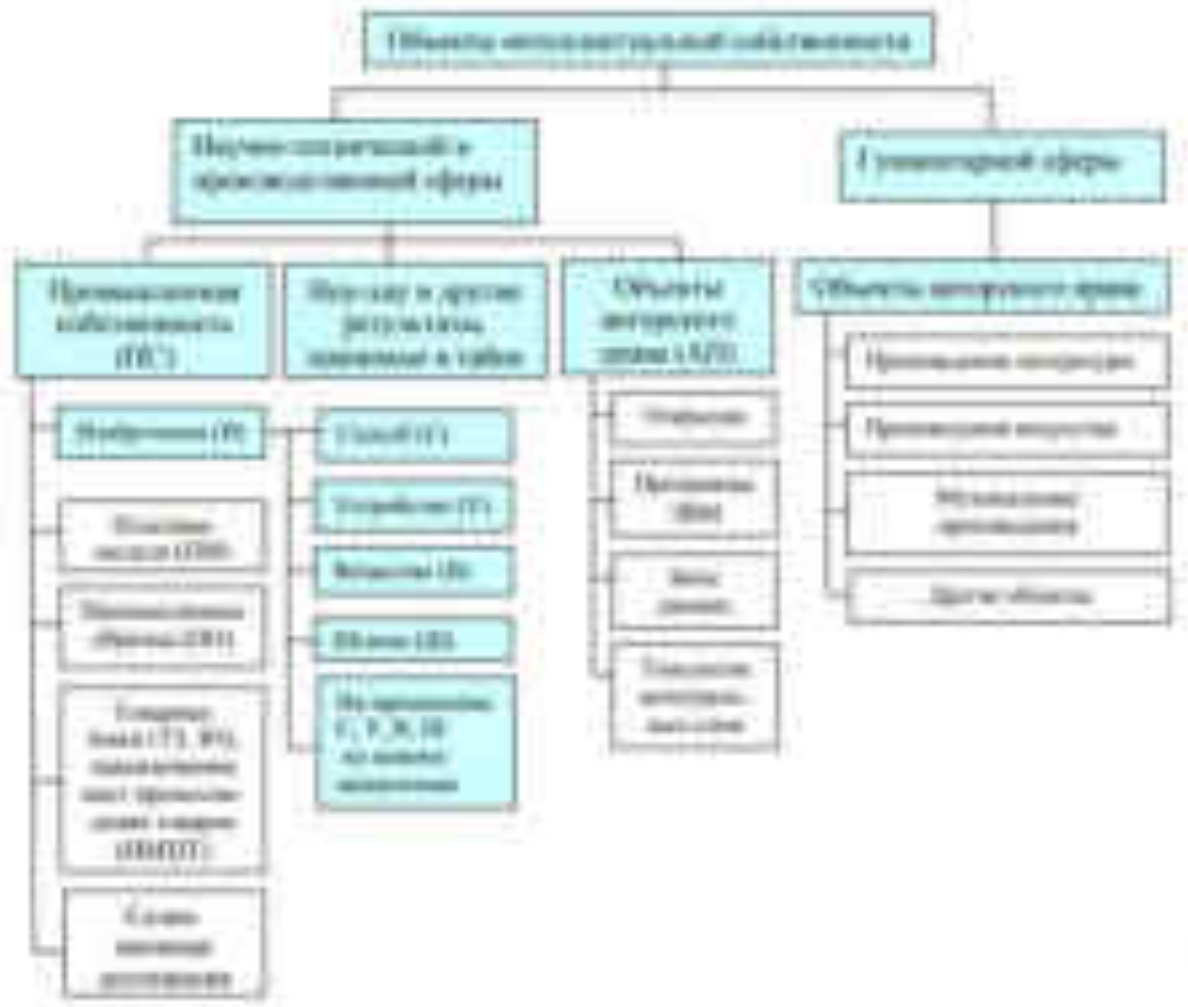


Рисунок 5 – Классификация объектов интеллектуальной собственности

Объекты промышленной собственности (ПС) составляют наиболее сильную подсистему ОИС (рис. 5). Охрана объектов ПС возникает только после признания их патентным ведомством патентоспособными и выдачи охранного документа – патента или свидетельства. Выдаче предшествует специальная экспертиза. Весьма специфично осуществляется охрана прав на ноу-хау: государство гарантирует обладателю ноу-хау защиту от незаконного использования этих сведений третьими лицами, но при условии, что: 1) эта информация имеет действительную или потенциальную коммерческую ценность в силу неизвестности её третьим лицам; 2) к этой информации нет свободного доступа на законном основании; 3) обладатель информации принимает надлежащие меры к охране ее конфиденциальности. Таким образом, пока выполняются эти условия, существует ноу-хау и существует охрана ИС в отношении этого ноу-хау. Следовательно, формой охраны ноу-хау является сохранение его в тайне.

Объекты авторского права (АП) - произведения (совокупность идей, мыслей и образов, получивших в результате творческой деятельности автора свое выражение в доступной для восприятия человеческими чувствами конкретной форме, допускающей возможность воспроизведения) науки, литературы и искусства независимо от назначения, а также от способа его выражения. Объектом АП следует считать не просто работу автора и не идеи, выраженные им, а произведение как комплекс идей и образов, получивших свое выражение в готовом труде, как индивидуальное и неповторимое творческое отражение объективной действительности. В ГК РФ определены следующие виды произведений: литературные произведения; драматические и музыкально-драматические произведения, сценарные произведения; хореографические произведения и пантомимы; музыкальные

произведения с текстом или без текста; аудиовизуальные произведения; произведения живописи, скульптуры, графики, дизайна, графические рассказы, комиксы и другие произведения изобразительного искусства; произведения декоративно-прикладного и сценографического искусства; произведения архитектуры, градостроительства и садово-паркового искусства, в т.ч. в виде проектов, чертежей, изображений и макетов; фотографические произведения и произведения, полученные способами, аналогичными фотографии; географические, геологические и другие карты, планы, эскизы и пластические произведения, относящиеся к географии, топографии и к другим наукам; другие произведения.

К объектам АП также относятся программы для ЭВМ, которые охраняются как литературные произведения. Каждый из объектов АП может быть классифицирован по многочисленным подвидам – по их внешним формам, жанрам и сферам применения. Так, литературные произведения могут быть художественного (1), научного (2), учебного (3) характера. Надо помнить, что АП не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования. Спорным в теории и практике является вопрос о включении в число объектов АП формул. К числу объектов, не охраняемых АП, относятся, прежде всего, те из них, которые не обладают хотя бы одним из признаков произведения науки, литературы и искусства. Если в ходе проделанной работы достигнут чисто технический результат, он также АП не охраняется. К ним относятся телефонные справочники, расписания движения, адресные книги и т.п. при условии, что составителем не применена оригинальная схема изложения справочных данных. Наряду с подобными объектами существуют произведения, обладающие всеми необходимыми для охраны признаками, но не охраняемые АП в силу прямого указания закона. К их числу относятся следующие четыре категории произведений: 1) произведения, срок охраны которых истек; 2) официальные документы, их официальные переводы, а также государственные символы и знаки; 3) произведения народного творчества; 4) сообщения о событиях и фактах, имеющие информационный характер. Значение имеет деление произведений на обнародованные и необнародованные, опубликованные и неопубликованные. АП охраняются и те, и другие. Однако если необнародованные произведения неприкосновенны и ни при каких условиях не могут быть использованы без согласия их авторов, то обнародованные произведения в исключительных, предусмотренных законом случаях, могут быть использованы заинтересованными лицами без согласия авторов и даже вопреки их возражениям. Аналогичные различия есть между опубликованными и неопубликованными произведениями. Под опубликованием в законе понимается выпуск в обращение экземпляров произведения, т.е. изготовление и выпуск в обращение копий произведения, изготовленных в любой материальной форме. Оно должно быть совершено с согласия автора. Произведения подразделяются на оригинальные, производные, составные. Практическое значение этой классификации – в том, что для создания и использования производных произведений надо получить разрешение обладателей авторских прав на те произведения, которые станут основой для производных. Оригинальным является такое произведение, все основные охраняемые элементы которого созданы самим автором. В производном (зависимом) произведении заимствованы охраняемые элементы чужого произведения.

Вторым условием возникновения авторских прав на такое произведение является соблюдение его создателем прав автора произведения, подвергшегося переводу, переделке, аранжировке или другой переработке. Помимо производных произведений к объектам АП также относятся сборники (энциклопедии, антологии, базы данных) и другие составные произведения, представляющие собой результат творческого труда по подбору или расположению материалов. Значение для определения авторских правомочий и режима использования произведения оказывает признание его служебным (произведения,

созданные в порядке выполнения служебных обязанностей или служебного задания работодателя).

Одним из объектов АП является программа для ЭВМ. Под ней понимается объективная форма представления совокупности данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств с целью получения определенного результата. К числу программ для ЭВМ относят также подготовительные материалы, полученные в ходе ее разработки, и порождаемые ею аудиовизуальные отображения. Авторские права на все виды программ для ЭВМ (в т.ч. на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код, охраняются так же, как авторские права на произведения литературы. Охрана не распространяется на идеи и принципы, лежащие в основе программ, баз данных и топологий, в т.ч. на языки программирования.

Однотипны и личные права (право авторства, право на имя и право на неприкосновенность). имущественные права на программы, базы и топологии принадлежат как их создателям (авторам), так и их наследникам, а также другим физическим или юридическим лицам, получившим исключительные права в силу закона или договора. АП на программы для ЭВМ возникает с момента их создания и воплощения в объективной форме. Вместе с тем обладатель всех имущественных прав на программу вправе по своему желанию непосредственно либо через своего представителя зарегистрировать этот объект в Роспатенте. Использование программ для ЭВМ третьими лицами (пользователями) осуществляется, как правило, по договору с правообладателями. Допускается свободная перепродажа или передача иным способом права собственности либо иных прав на экземпляр программы или базы данных после первой продажи или другой передачи права на этот экземпляр. Лицу, правомерно владеющему экземпляром программы, разрешено свободно манипулировать ее данными, в т.ч. адаптировать их – вносить изменения, необходимые для функционирования программы на технических устройствах пользователя, а также осуществлять ее запись и хранение в памяти ЭВМ. Законный обладатель вправе изготавливать копию программы для архивных целей и для замены правомерно приобретенного и впоследствии утерянного, испорченного или ставшего непригодным к использованию оригинала. При определенных условиях обладатель экземпляра программы для ЭВМ может также ее декомпилировать – воспроизвести и преобразовать объектный код в исходный текст.

Интеллектуальная собственность научно-технической и производственной сфер – это изобретения (И), полезные модели (ПМ) и промышленные образцы (ПО), охрана которых осуществляется в рамках ГК РФ и подзаконных актов. Правовая охрана предоставляется на основании процедуры государственной регистрации, в ходе которой соответствующие РИД проверяются на охранную способность (патентоспособность). На РИД, признанный патентоспособным Роспатентом выдается официальный документ – патент, удостоверяющий исключительное право, авторство и приоритет И, ПМ либо ПО. Срок его действия исчисляется со дня подачи первоначальной заявки в Роспатент и составляет: для И – 20 лет, для ПМ – 10 лет, для ПО – 15 лет. Срок действия патента на ПМ может быть продлен по заявлению патентообладателя на срок, указанный в заявлении, но не более 3 лет, на ПО – на срок, указанный в заявлении, но не более 10 лет. По истечении срока действия исключительного права И, ПМ или ПО переходят в общественное достояние и могут использоваться любым лицом без чьего-либо согласия или разрешения и без выплаты вознаграждения за использование. В соответствии со ст. 1349 ГК РФ объектами патентных прав не могут быть: а) способы клонирования человека; б) способы модификации генетической целостности клеток зародышевой линии человека; в) использование человеческих эмбрионов в промышленных и коммерческих целях; г) иные решения, противоречащие общественным интересам, принципам гуманности и морали.

Ключевым понятием патентного права является патентоспособность – совокупность свойств технического решения, без наличия которых оно не может быть признано

изобретением на базе действующего законодательства (в нашем случае – России). Статья 1350 ГК РФ, определяет три условия патентоспособности изобретения – новизну, изобретательский уровень и промышленную применимость. В ГК РФ есть указание на объекты, которые не могут выступать в качестве изобретения. Так, не являются изобретениями: 1) открытия; 2) научные теории и математические методы; 3) решения, касающиеся только внешнего вида изделий и направленные на удовлетворение эстетических потребностей; 4) правила и методы игр, интеллектуальной или хозяйственной деятельности; 5) программы для ЭВМ; 6) решения, заключающиеся только в представлении информации. Не предоставляется правовая охрана в качестве изобретения: 1) сортам растений, породам животных и биологическим способам их получения, за исключением микробиологических способов и продуктов, полученных такими способами; 2) топологиям интегральных микросхем. Срок действия исключительного права на изобретение (согласно ст. 1363 ГК РФ) составляет 20 лет.

Лицензионный договор – это соглашение, по которому обладатель исключительного права на объект интеллектуальной собственности (лицензиар) предоставляет или обязуется предоставить другой стороне (лицензиату) право использования этого объекта в предусмотренных договором пределах. Сторонами лицензионного договора, т.е. лицензиаром — обладателем исключительного права и лицензиатом — временным пользователем объекта интеллектуальной собственности, могут быть любые субъекты гражданских прав при соблюдении правил о право- и дееспособности.

Предметом лицензионного договора является право использования определенного объекта интеллектуальной собственности определенными в договоре способами, и лицензиат вправе использовать объект только в пределах тех прав и теми способами, которые предусмотрены договором. Договор должен быть заключен в письменной форме, если Кодексом не предусмотрено иное.

Вопросы для повторения:

1. Определите цели и содержание научно-исследовательских работ студентов.
2. Каковы функции участия студентов в исследовательских работах?
3. Раскройте содержание этапов исследовательских работ.
4. Каковы особенности различных форм их организации?
5. Какие задачи решаются советом вуза по исследовательским работам?
6. Назовите основные группы ОИС и раскройте состав этих групп.
7. В чем состоит отличие особенностей объектов авторского права от объектов промышленной собственности?
8. Что стоит за словом «патент» и что не может быть объектом патентных прав?

Министерство науки и высшего образования российской федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Ульяновский государственный технический университет

Методические рекомендации

Дисциплина (модуль):
ФТД.02 Информационная безопасность в профессиональной деятельности

Уровень образования: магистратура

Квалификация: магистр

г. Ульяновск, 2021г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Информационная безопасность в профессиональной деятельности

Профиль подготовки
Искусственный интеллект в автоматизации проектирования

Квалификация выпускника
Магистр

Формы обучения
очная

г. Ульяновск, 2021

Тема 1.1. Информационная безопасность

1. Информация. Определение, особенности, виды информации.

2. Компрометация информации. Базовые критерии информационной безопасности. Конфиденциальность, целостность, доступность.

3. Информационная безопасность. Определение и структура ИБ. Подходы к обеспечению и управлению ИБ. Классификация способов защиты информации

Информация – это философская категория, в зависимости от контекста обозначающая:

- смысл/содержание формы;
- данные, содержащиеся в хранилищах;
- сигналы, передающиеся по линиям связи;
- сведения (сообщения, данные) независимо от формы их представления (ФЗ № 149-ФЗ, ст. 2);
- множество строк из определённого алфавита (Теория формальных грамматик).

Какую бы форму ни принимала информация, она обладает следующими особенностями:

- нематериальна, следовательно: неотделима от носителя, неисчерпаема, не локализована в пространстве;
- упорядочивает хаос (энтропия).

Различают следующие виды информации:

- вербальная («мягкая»): носитель – канал передачи, форма – язык;
- невербальная («твёрдая»): носитель – хранилище, форма – код;
- смешанная – сочетание различных видов;

- комплексная – наложение различных видов.

С точки зрения защиты информации ее можно классифицировать по типам:

12. открытая – незащищённая;

- конфиденциальная – предприняты меры по защите;
- публичная – предприняты меры по подготовке к публикации.

К определённому типу информацию относит обладатель информации – лицо, самостоятельно создавшее информацию либо получившее на основании закона или договора право разрешать или ограничивать доступ к информации, определяемой по каким-либо признакам (ФЗ № 149-ФЗ, ст. 2).

В зависимости от обладателя информации и типа защищенности выделяют следующие категории информации (табл. 1).

Таблица 1.

Категории информации

Индивид	**Организация**	**Государство**
Конфиденциальная	Секретная	Тайная
Открытая	Рабочая	Свободная
Публичная	Декларированная	Официальная

Всякая информация обладает определённой ценностью. Ценность (полезность) информации можно оценить со стороны:

- полезности (результативности) – что можно сделать с её помощью;
- правильности (полноты, точности) – соответствует ли она действительности;
- своевременности (актуальности) – можно ли использовать её.

Компрометация информации – негативные последствия от угроз для информации. Выделяют основные виды компрометации информации, связанные со снижением её ценности с определённой стороны:

- утечка – информация неисчерпаема, но утрачивается ее результативность (к);
- искажение – информация не уничтожима, но утрачивается полнота-точность (ц);
- потеря – информация не локализована, но утрачивается актуальность (д);

Информационная безопасность должна обеспечивать всестороннюю защиту ценности имеющейся информации.

Критерии информационной безопасности – основные (базовые) направления защиты ценности информации. Критерии могут быть использованы для определения приоритетов, выбора средств защиты, и оценки защищенности (табл. 2).

Таблица 2.

Модель CIA (Confidentiality-Integrity-Availability)

Критерий	**Определение**	**Операция**
Конфиденциальность Confidentiality	Легальное использование	Чтение, r (передача)
Целостность Integrity	Отсутствие вмешательства	Запись, w (модификация)
Доступность Availability	Беспрепятственный доступ	Активация x (преобразование)

Дополнительно выделяют (производные) критерии:

18. секретность (са) – защита данных и канала передачи (защита от перехвата);

- неотказуемость (ci) – возможность установить авторство (подтверждение подлинности);
- сохранность (ia) – соответствие и готовность к использованию (сохранность улик);
- идентичность (cia) – соответствие целям (от происхождения, до применения).

На основании критериев защиты формируются конкретные требования к защите:

- требования обеспечения конфиденциальности:
  1. неразглашение - требования к защите от утечек,
  2. категорирование - требования к разделению информации по степени защиты (виды информации),
  3. скрывание - требования к мерам по засекречиванию наличия информации,
  4. ответственность - требования к соответствию законодательству (назначение ответственных, поиск виновных),
- требования обеспечения целостности:
  1. сохранность - требования к хранению (что, место, время),
  2. неизменность - требования к отсутствию вмешательства,
  3. корректность - требования к проверкам (кто, когда, как),
  4. неотказуемость - требования к контролю изменений,
- требования обеспечения доступности:
  1. разграничение - разделение информации по способу обращения (роли-операции),
  2. производительность - количество одновременно/за период обрабатываемых запросов,
  3. надежность – степень сохранения возможности выполнения требуемых функций.

Надежность можно оценить количественно по формулам:

(Д(время обещанной доступности) – П(время простоя)) / И(интервал: 24ч, 7д, 31д) × 100%,

MTTF(средняя наработка до отказа) / (MTTF + MTTR(среднее время до восстановления)) × 100%.

Или качественно:

- Низкая.
- Средняя.
- Высокая доступность – наиболее распространённый уровень, ожидаемый пользователями, при котором система или приложение доступны в обозначенные требованиями дни и часы без незапланированных простоев, а о запланированных остановках в работе объявлено заранее.
- Непрерывный режим работы (continuous operations) -- система доступна 24 часа в сутки 7 дней в неделю без запланированных простоев.
- Постоянная доступность (continuous availability) -- сочетание высокой доступности с непрерывным режимом работы, система доступна 24 часа в сутки 7 дней в неделю без запланированных или незапланированных простоев.

Информационная безопасность (ИБ) – процесс защиты конфиденциальности, целостности и доступности информации.

Для защиты информации по каждому направлению существуют следующие методы:

– шифрование – конфиденциальность;

- хеширование – целостность;
- аутентификация, авторизация, аккаунтинг (протокол AAA) – доступность;
- стеганография – секретность;
- цифровая подпись – неотказуемость;
- цифровое архивирование – сохранность.

Применение этих способов уменьшает потерю ценности информации, но не предотвращает её, так как информационная система организации нуждается в постоянной защите от негативных внешних факторов (опасности).

Опасность – возможность пострадать от угрозы. Состоит из угрозы, наличия уязвимости и возможного вреда.

Безопасность – процесс нейтрализации угроз, уязвимостей, вреда. Составные элементы процесса обеспечения безопасности:

28. неприступность – отсутствие угроз;

- защищенность – невосприимчивость к угрозам;
- надёжность – отсутствие отрицательных последствий.

Обеспечение информационной безопасности – применение необходимых и достаточных мер по защите информации от угроз. Основные направления:

31. защита от атак;

- устранение уязвимостей;
- борьба с последствиями.

Области обеспечения информационной безопасности:

34. теоретическая – на основе закономерностей и опыта:

    2. принципы безопасности;

    3. модели безопасности;

- нормативно-правовая – на основе юридической ответственности и требованиях:

    1. правовые акты;

    2. нормативные документы;

- организационно-режимная – на основе правил и порядка:

    1. регламенты;

    2. режимные меры;

- техническая – на основе техники и автоматизации:

  1. инженерно-технические средства;

  2. программно-технические средства.

Используемые меры защиты информации необходимо контролировать, обслуживать, обновлять.

Управление информационной безопасностью – деятельность по созданию и поддержанию системы защиты информации, включающая: планирование архитектуры системы, выбор и применение средств защиты информации, выявление и реагирование на инциденты.

Комплексное управление информационной безопасностью называют системой управления информационной безопасностью.

Контрольные вопросы и задания

Чем информация отличается от данных, знаний?
Откуда берётся информация?
Оцените стоимость информации одной из сегодняшних новостей.
В чем разница между критериями, требованиями, свойствами информационной безопасности?
Что такое информационная безопасность?
Проанализируйте с точки зрения конфиденциальности, целостности и доступности информационную безопасность ваших конспектов.
Приведите примеры для каждого способа обеспечения информационную безопасность?
Чем управление информационной безопасностью отличается от обеспечения информационной безопасности?

1.2. Риски информационной безопасности

1. Понятие риска. Определение и структура риска. Термины риск-менеджмента.

2. Классификация угроз, уязвимостей, последствий. Особенности рисков ИБ.

3. Управление рисками. Процесс риск-менеджмента: анализ, оценка, обработка.

Современный системный подход к управлению информационной безопасностью основан на понятии риска и методах управлении им.

Риск – это:

- следствие влияния неопределенности на достижение поставленных целей;
- вероятностно-стоимостная оценка потерь;
- сочетание вероятности и последствий наступления неблагоприятных событий;
- неопределённое событие или условие, которое в случае возникновения имеет позитивное или негативное воздействие на репутацию компании, приводит к приобретениям или потерям в денежном выражении;
- вероятностные последствия (отрицательные и положительные);
- угрозы + уязвимости + последствия.

Риск, его структура и способы управления им изучаются в дисциплине «риск-менеджмент».

Термины риск-менеджмента, касающиеся информационной безопасности (ГОСТ Р ИСО/МЭК 31000-2019 Менеджмент риска):

- объект (актив) – непосредственно связанное с целями (деятельностью) (информация, ПО, устройство);
- источник риска – объект или деятельность, которые самостоятельно или в комбинации с другими обладают возможностью вызывать повышение риска (хакеры, вредоносы, сбои);

- событие (инцидент) – возникновение или изменение специфического набора условий (нарушение конфиденциальности, целостности и доступности);
- правдоподобность (появления события) – характеристика возможности и частоты появления события (уязвимость информационной безопасности (недекларированные возможности ПО, ошибки, скрытые каналы);
- последствие (consequence) – результат воздействия события на объект (последствия нарушения конфиденциальности, целостности и доступности (утечка, искажение, отказ);
- угроза – событие без последствий.

Существует множество подходов к классификации риска. Рассмотрим некоторые из них.

Классификация источников угроз:

1. По вероятности:
    1. случайные;
    2. неумышленные;
    3. преднамеренные.
- По источнику:
    1. стихийные – природа;
    2. техногенные – аппаратные/программные средства;
    3. антропогенные – человек-нарушитель:
        i. Сотрудники:
            1. халатный;
            2. некомпетентный;
            3. излишне компетентный (активный);
            4. вредитель;
            5. инсайдер.

2. Хакеры:
    1. любители;
    2. исполнители;
    3. мстители.
3. Криминал:
    1. вымогатели;
    2. вандалы;
    3. воры;
    4. мошенники.

- По направленности:
    1. нарушение конфиденциальности;
    2. нарушение целостности;
    3. нарушение доступа.

Причины уязвимостей:

1. обстоятельства;
- невнимательность;
3) попустительство;
4) совпадение;
5) злой умысел.

Классификация последствий:

6) по ущербу:
а) без последствий;

б) незначительные;

в) некритичные;

г) критичные, серьёзные;

д) опасные;

е) катастрофические.

7) по реакции:

а) игнорируемые;

б) учитываемые;

в) обрабатываемые;

г) приоритетные.

Для выявления совокупности условий и факторов (угрозы, уязвимости), которые приводят или могут привести к риску, а так же для их классификации и анализа составляется модель угроз. Общим для различных моделей угроз является: составление перечня угроз; определение границ и условий реализации угроз, оценка уровня опасности.

Что бы уменьшить количестве анализируемых угроз рассматривают только актуальные для конкретной организации угрозы. Определить актуальность угрозы помогает ландшафт угроз – совокупность наиболее распространённых и опасных угроз для определенных активов, определенного вида систем, типовой организации.

Основными требованиями к источникам данных по рискам являются: регулярность появления данных, их объективность и значимость. Все источники данных по ИБ можно разделить на 3 группы: отчеты IT-компаний, содержимое международных и национальных баз, публикации информационно-аналитических центров (табл. 3.).

Таблица 3.

Источники данных по рискам

Источник	**Открытость**	**Многосторонность**	**Масштаб**
Статистика киберугроз от Касперского	+	-	+
ESET Threat Intelligence	-	-	+
Microsoft Security Intelligence Report	+	+	-***
Отчеты CISCO по ИБ	+*	+	+***

Отчеты Positive Technologies	+	+	-
Отчеты Kaspersky ICSystems CERT	+	-**	+
Ежегодный обзор уязвимостей Flexera	+*	-	+
Отчеты Internet Storm Center	+	-	+
База уязвимостей CVE	+	-**	+
Отчеты CWE	+	-**	+
Аналитические отчеты US-CERT	+	+**	-***
Топ-10 сообщества OWASP	+	-	+
БДУ ФСТЭК	+	-**	-
Данные по киберпреступлениям Statista	-	+	+
Сообщество BISA	+*	+	-
Аналитическая центр компании InfoWatch	+*	+	-
Подписки по безопасности и приватности IDC	-	+	+***
Исследования Ponemon Institute	+	+	+***
Данные Федеральной торговой комиссии США	+	+	-
Портал databreaches	+	-**	+***
«Anti-Malware.ru»	+	+**	+

Деятельность по управлению риском – это непрерывный процесс, в котором можно выделить ключевые этапы. Процесс риск-менеджмента представлен на рис. 1 [1].

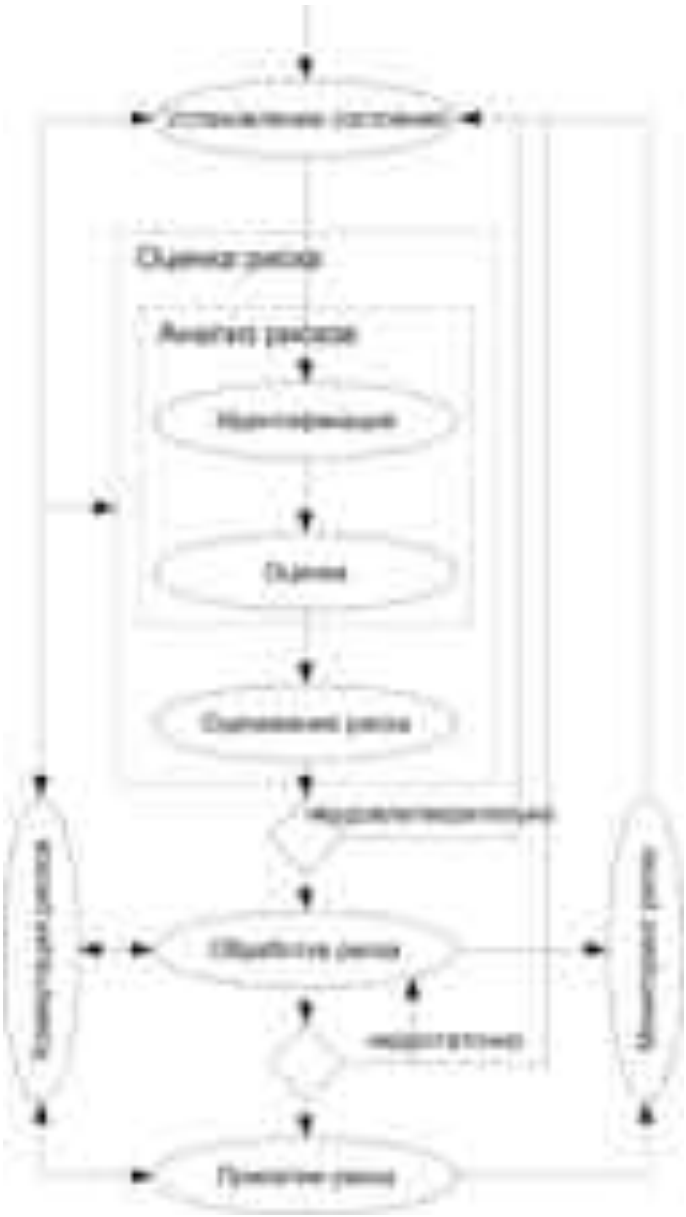


Рис. 1. Процесс риск-менеджмента

Установление состояния включает:

- определение текущей ситуации (контекста);
- установление приоритетов и стратегии;
- формулирование целей и задач;
- согласование критериев оценки.

Виды критериев:

8) количественные – величины;

9) качественные – метки;

10) полуколичественные – шкалы.

В процессе оценки рисков обычно придерживаются следующего порядка:

5. идентификация активов – составление списка активов, которые необходимо защищать;

- оценка стоимости активов – определение стоимости актива с учётом его роли в деятельности организации;
- идентификация уязвимостей – составление списка уязвимостей активов, условий их возникновения;
- идентификация угроз для активов – составление списка источников угроз с учетом их актуальности для защищаемых активов;
- оценка уязвимостей – определение вероятности возникновения инцидента с учетом имеющихся уязвимостей;
- вероятностная оценка последствий – определение последствий стоимости инцидента с учетом стоимости активов;
- оценка стоимости контрмер – определение стоимости средств защиты, мер по устранению узявимостей, профилактики инцидентов.

При проведении оценки риска важно идентифицировать все риски. Рекомендуется следовать одному из проверенных методов оценки риска, которые можно классифицировать по принципу оценки [2]:

11) экспертные – на основе мнения людей-экспертов (структурированный анализ «что-если» (SWIFT));

12) аналитические (логико-вероятностные) – на основе расчётов (анализ причин и последствий);

13) статистические – на основе имеющихся данных (деревья решений);

14) модельно-расчетные – сочетание предыдущих на основе симуляций (метод Монте-Карло).

После оценки риски разбиваются на группы по способу воздействия (обработки):

15) нейтрализация – устранение источника риска;

16) избежание (предотвращение, уклонение) – отказ от деятельности, обусловливающей риск;

17) уменьшение – изменение вероятности или возможности;

18) компенсация (снижение) – реагирование на последствия;

19) передача (перенос) – делегирование обработки риска другой стороне;

20) принятие (сохранение, но не игнорирование) – готовность терпеть последствия;

Также к способам обработки риска относят:

21) коммуникацию – обмен информацией о риске;

22) мониторинг – наблюдение за инцидентами.

Управление рисками опирается на инфрастуктуру риск-менеджмента – систему организации и поддержки процессов. Цель создания инфраструктуры – обеспечить необходимый уровень оценки и обработки риска.

Контрольные вопросы и задания

Что такое риск?
Опишите структуру риска.
Сформулируйте определение риска ИБ.
Приведите примеры современных угроз, уязвимостей, последствий ИБ.
Опишите процесс риск-менеджмента.
Как оценить риск несдачи экзамена?
Приведите примеры способов обработки рисков.

Тема 1.3. Шифрование

1. Криптология. Цели и задачи криптографии и криптологии. Шифрование и расшифровывание.

2. Шифры. Принципы и способы шифрования. Типы шифров.

3. Атаки на шифры. Классификация способов атак на шифры.

4. Цифровая подпись. Виды, принцип создания. Удостоверяющий центр.

Криптология – наука, изучающая методы шифрования и дешифрования. Подразделяется на криптографию (науку обеспечения конфиденциальности при хранении и передаче данных) и криптоанализ (науку об уязвимостях криптографических методов).

Шифрование – это:

- криптопреобразование данных;
- обратимое преобразование информации для сокрытия от недопущенных лиц (в общем смысле).

Шифрование обеспечивает:

- конфиденциальность – доступ только по ключу;
- перманентность – обнаружение вмешательства;
- идентифицируемость – только владельцы ключа.

Российский стандарт шифрования ГОСТ Р 34.12-2015 (Кузнечик) определяет следующие термины.

Открытый текст (plaintext) – незашифрованная информация.

Шифр (cipher) – криптографический метод, используемый для обеспечения конфиденциальности данных, включающий алгоритм зашифрования и алгоритм расшифрования.

Зашифрование (encryption) – обратимое преобразование данных с помощью шифра, которое формирует шифртекст из открытого текста.

Ключ (key) – изменяемый параметр в виде последовательности символов, определяющий криптографическое преобразование.

Итерационный ключ (roundkey) – последовательность символов, вычисляемая в процессе развёртывания ключа шифра и определяющая преобразование на одной итерации блочного шифра.

Шифртекст (ciphertext) – данные, полученные в результате зашифрования открытого текста с целью скрытия его содержания.

Расшифрование (decryption) – операция, обратная к зашифрованию.

Принципы шифрования:

- *принцип независимости* – защита информации, а не ее носителя;

2) *принцип обратимости* – должен существовать эффективный способ восстановить исходное сообщение;

3) *принципы Керкгоффса* – чем меньше секретов, тем выше безопасность:

а) система должна быть физически, если не математически, невскрываемой;

б) нобходимо, чтобы не требовалось сохранение системы в тайне; попадание системы в руки врага не должно причинять неудобств;

в) хранение и передача ключа должны быть осуществимы без помощи бумажных записей; корреспонденты должны располагать возможностью менять ключ по своему усмотрению;

г) система должна быть пригодной для сообщения через телеграф;

д) система должна быть легко переносимой, работа с ней не должна требовать участия нескольких лиц одновременно;

е) от системы требуется, учитывая возможные обстоятельства её применения, чтобы она была проста в использовании, не требовала значительного умственного напряжения или соблюдения большого количества правил;

4) *принцип нулевого разглашения* – предоставлять только необходимую информацию.

5) *принцип трудозатрат* – на преодоление защиты необходимы большие затраты.

Защита конфиденциальности с помощью шифрования основана на криптопреобразовании сообщения (рис. 2).

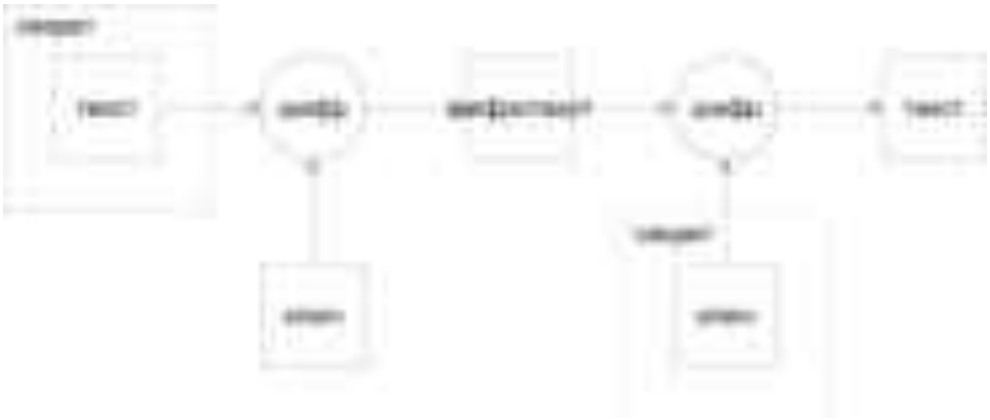


Рис. 2. Процессы шифрования и расшифровывания

Способы шифрования:

6) перестановки – изменение порядка символов сообщения;

7) подстановки – моноалфавитная или полиалфавитная замена символов сообщения;

8) динамическое – изменение ключа;

9) квантовое – изменение сообщения при попытке перехватить.

Типы шифров по работе с данными:

10) Блочные – шифрование по блокам в порядке определяемом режимом. Режимы блочных шифров:

а) *electronic code book* (ECB) – режим электронной кодовой книги;

б) *cipher block chaining* (CBC) – режим сцепления блоков шифротекста;

в) *propagating cipher block chaining* (pCBC) – режим распространяющегося сцепления блоков шифра;

4) *cipher feed back* (CFB) – режим обратной связи по шифротексту;

д) *output feed back* (OFB) – режим обратной связи по выходу;

е) *counter mode* (CTR) – режим сцепки за шифрованным значением счётчика.

Особенности:

а) низкая скорость работы – несколько раундов, перекрёстные связи;

б) размножение ошибки – невозможность исправления.

11) Поточные – шифрование последовательно по символам, с учётом позиции: Особенности:

а) шаблонность – одинаковое преобразование;

б) коррелированность выходного потока с потоком ключа – предсказуемый и неравномерный ключевой поток.

Типы шифров по работе с ключами:

12) Симметричные – $К_ш = К_р$ (по степени секретности). Проблема: передача ключа.

13) Асимметричные – $К_ш < К_р$ (по степени секретности). Проблема: низкая скорость работы.

14) Комбинированные – $К_{сим}(К_{ас})$. Асиметричный ключ шифруется симметричным шифром.

Шифрование не является 100%-м способом защиты информации. Клод Шеннон доказал существование абсолютно стойких шифров, при соблюдении требований: ключ используется один раз, ключ статистически надежен, длина равна или больше длины сообщения, исходное сообщение избыточно (оценка правильности расшифровки). На практике используются достаточно стойкие шифры – не вскрываемые в течении достаточного времени. Несоблюдение принципов, «слабости» реализации криптосистем могут стать целями атак на зашифрованный текст.

Проблемы шифрования:

15) соблюдение принципов;

16) сохранение секрета;

17) управление ключами.

Целью атаки может быть восстановление исходного сообщения или вычисление используемого ключа.

Классификация атак и методов «взлома» шифров:

1. *Пассивные* – основаны на возможности перехвата сообщения:

1) если есть только шифротекст – слишком мало информации;

2) если известно исходное сообщение:

а) *линейный криптоанализ*.

2. *Активные* – основаны на использовании той же криптосистемы:

1) если есть возможность повторно шифровать с искомым ключом:

а) *повтор сообщения;*

б) *дифференциальный криптоанализ;*

2) если есть возможность дешифровывать заданные сообщения:

а) *перебор (брутфорс);*

б) *алгебраический криптоанализ;*

в) *статистический криптоанализ;*

3. *Адаптируемые* – изменение метода в зависимости от результатов.

4. *Побочных каналов* – использование уязвимостей реализации

Современные шифры:

18) ГОСТ Р 34.12-2015 (Кузнечик);

19) 3DES;

20) AES;

21) Blowfish;

22) IDEA.

Контрольные вопросы и задания

Чем криптография отличается от шифрования?

Опишите принципы шифрования на примере замка и ключа.

Приведите примеры шифров.

Напишите, какой способ, тип, режим используются в конкретном шифре.

Какие проблемы может решить шифрование, а какие – не решает или создаёт?

Приведите примеры успешных атак на шифр.

Цифровая подпись

-
- Цифровая подпись – способ проверки неотказуемости авторства.
- В отличие от хеширования цифровая подпись формирует специальные зашифрованные данные, содержащие информацию о документе и авторе.
- Принцип работы цифровой подписи (рис. 10):

подписание – ассиметричное шифрование хеша документа закрытым ключом;

- проверка – расшифрование и сверка хеша документа с помощью открытого ключа.
-

- Рис. 10. Схема работы цифровой подписи

Ключевым элементом цифровой подписи является удостоверяющий центр (УД, центр сертификации, CA) -- тот, кому доверяют обе стороны, а открытый ключ широко известен. Задача центра сертификации — подтверждать подлинность ключей шифрования с помощью сертификатов электронной подписи.

Контрольные вопросы и задания

Чем цифровая подпись отличается от хеширования с аутентификацией.

Придумайте и проанализируйте свою систему цифровой подписи документов.

Тема 2.1. Иерархия нормативно-правовых документов по информационной безопасности.

1. Иерархия нормативно-правовых документов РФ

3. Виды тайн

2. Государственная система обеспечения информационной безопасности

4. Ответственность за нарушения в сфере ИБ

Иерархия нормативно-правовых документов:

- нормативно-правовые:
    1. конституция РФ,
    2. международные договоры и соглашения,
    3. законы РФ(кодексы, федеральные законы),
    4. указы и распоряжения Президента РФ,
    5. постановления и распоряжения Правительства РФ;
- нормативно-технические:
    1. технические регламенты,
    2. нормативно правовые акты федеральных органов исполнительной власти (приказы),
    3. стандарты государственные (национальные) и организации.

Основополагающие документы по ИБ:

- Конституция РФ

- Статья 23.

- 1. Каждый имеет право на неприкосновенность частной жизни, личную и семейную тайну, защиту своей чести и доброго имени.

- 2. Каждый имеет право на тайну переписки, телефонных переговоров, почтовых, телеграфных и иных сообщений. Ограничение этого права допускается только на основании судебного решения.
- Статья 29.4. Каждый имеет право свободно искать, получать, передавать, производить и распространять информацию любым законным способом. Перечень сведений, составляющих государственную тайну, определяется федеральным законом.
- Статья 29.5. Гарантируется свобода массовой информации. Цензура запрещается.
- Статья 24.1. Сбор, хранение, использование и распространение информации о частной жизни лица без его согласия не допускаются.
- Статья 42. Каждый имеет право на благоприятную окружающую среду, достоверную информацию о ее состоянии и на возмещение ущерба, причиненного его здоровью или имуществу экологическим правонарушением.
- Статья 44. Каждому гарантируется свобода литературного, художественного, научного, технического и других видов творчества, преподавания. Интеллектуальная собственность охраняется законом.
- Доктрина информационной безопасности РФ (Указ Президента от 05.12.2016 г. №646)
    - Национальные интересы в информационной сфере.
    - Основные информационные угрозы.
    - Стратегические цели и основные направления обеспечения.
    - Принципы обеспечения ИБ.
- ФЗ "Об информации, информационных технологиях и о защите информации" №149-ФЗ, 27.07.2006
    - ст.1.1. Настоящий Федеральный закон регулирует отношения, возникающие при:
    - 1) осуществлении права на поиск, получение, передачу, производство и распространение информации;
    - 2) применении информационных технологий;
    - 3) обеспечении защиты информации.

Классификация областей ИБ.

Указ Президента РФ от 06.03.1997 N188 "Об утверждении Перечня сведений конфиденциального характера".

Объекты защиты: информационные системы (ИС), автоматизированные системы управления (АСУ), информационно-телекоммуникационные сети (ИТКС).

- Публичная:
- общедоступная – к общедоступной информации относятся общеизвестные сведения и иная информация, доступ к которой не ограничен. [№149-ФЗ, ст.7.1.].
- Результаты интеллектуальной деятельности и средства индивидуализации -- творческим трудом которого создан такой результат (произведение или логотип) [ГК РФ, Ч.4, авторское право].
- Массовая информация -- предназначенные для неограниченного круга лиц печатные, аудио-, аудиовизуальные и иные сообщения и материалы; ["О средствах массовой информации" 27.12.1991 N2124-1, ст.2]. “О рекламе”.
- Государственная:
- Информация, содержащаяся в информационных системах общего пользования [№149-ФЗ] – информация, содержащаяся в государственных информационных системах (реестр Минкомсвязи или вводиться приказом гос.органа), а также иные имеющиеся в распоряжении государственных органов сведения и документы являются государственными информационными ресурсами (ГИС), … в целях реализации полномочий государственных органов и обеспечения обмена информацией между этими органами, а также в иных установленных федеральными законами целях [№149 ст.9,14].
- Государственная тайна -- защищаемые государством сведения в области его военной, внешнеполитической, экономической, разведывательной, контрразведывательной и оперативно-розыскной деятельности, распространение которых может нанести ущерб безопасности Российской Федерации ["О государственной тайне" от 21.07.1993 N 5485-1, ст.2].
- Тайна следствия и судопроизводства -- сведения, составляющие тайну следствия и судопроизводства, сведения о лицах, в отношении которых в соответствии с федеральными законами от 20 апреля 1995 г. N 45-ФЗ "О государственной защите судей, должностных лиц правоохранительных и контролирующих органов" и от 20 августа 2004 г. N 119-ФЗ "О государственной защите потерпевших, свидетелей и иных участников уголовного судопроизводства", другими

нормативными правовыми актами Российской Федерации принято решение о применении мер государственной защиты, а также сведения о мерах государственной защиты указанных лиц, если законодательством Российской Федерации такие сведения не отнесены к сведениям, составляющим государственную тайну [N188 Ук-Пр, п.2]. Сведения, содержащиеся в личных делах осужденных, а также сведения о принудительном исполнении судебных актов, актов других органов и должностных лиц, кроме сведений, которые являются общедоступными в соответствии с Федеральным законом от 2 октября 2007 г. N 229-ФЗ "Об исполнительном производстве" [N188 Ук-Пр, п.7].

- Служебная тайна -- служебные сведения, доступ к которым ограничен органами государственной власти в соответствии с Гражданским кодексом Российской Федерации и федеральными законами [N188 Ук-Пр, п.3].

- «Инфраструктура, которая при выведении из строя или разрушении приведет к катастрофическому и далеко идущему ущербу»(КИИ), научных и кредитно-финансовых организациях, а также предприятиях, работающих в стратегически важных для государства областях: оборонной, топливной и атомной промышленности, в сферах транспорта, энергетики, здравоохранения, связи, в ракетно-космической, горнодобывающей, металлургической и химической промышленности [№187-ФЗ от 26 июля 2017 г. «О безопасности критической информационной инфраструктуры Российской Федерации»].

- Коммерческая

- Коммерческая тайна (N188 Ук-Пр, п.5) -- сведения, связанные с коммерческой деятельностью, доступ к которым ограничен в соответствии с Гражданским кодексом Российской Федерации и федеральными законами). режим конфиденциальности информации, позволяющий ее обладателю при существующих или возможных обстоятельствах увеличить доходы, избежать неоправданных расходов, сохранить положение на рынке товаров, работ, услуг или получить иную коммерческую выгоду ["О коммерческой тайне" от 29.07.2004 N 98-ФЗ].

- Тайна изобретения, ноу-хау -- сведения о сущности изобретения, полезной модели или промышленного образца до официальной публикации информации о них [N188 Ук-Пр, п.6].

- Секрет производства, ноу-хау -- сведения любого характера (производственные, технические, экономические, организационные и другие) о результатах интеллектуальной деятельности в научно-технической сфере и о способах осуществления профессиональной деятельности, имеющие действительную или потенциальную коммерческую ценность вследствие неизвестности их третьим лицам,

если к таким сведениям у третьих лиц нет свободного доступа на законном основании и обладатель таких сведений принимает разумные меры для соблюдения их конфиденциальности, в том числе путем введения режима коммерческой тайны [ГК РФ Ч.4, секрет производства].

- Профессиональная тайна -- сведения, связанные с профессиональной деятельностью, доступ к которым ограничен в соответствии с Конституцией Российской Федерации и федеральными законами (врачебная, нотариальная, адвокатская тайна, тайна переписки, телефонных переговоров, почтовых отправлений, телеграфных или иных сообщений и так далее) [N188 Ук-Пр, п.4] -- «информация, которая становится доступной некоторому кругу лиц при осуществлении своих проф обязанностей».

- Банковская тайна -- тайна об операциях, о счетах и вкладах своих клиентов и корреспондентов. ["О банках и банковской деятельности" от 02.12.1990 N 395-1, ст.26].

- Частная

- Личная, семейная тайна.

- Персональная информация -- сведения о фактах, событиях и обстоятельствах частной жизни гражданина, позволяющие идентифицировать его личность (персональные данные), за исключением сведений, подлежащих распространению в средствах массовой информации в установленных федеральными законами случаях.[N188 Ук-Пр, п.1, "О персональных данных" от 27.07.2006 N 152-ФЗ].

- Тайна связи -- тайна переписки, телефонных переговоров, почтовых отправлений, телеграфных и иных сообщений, передаваемых по сетям электросвязи и сетям почтовой связи [N188 Ук-Пр, п.2, "О связи" от 07.07.2003 N126-ФЗ, ст.63].

ГСЗИ

Государственная система защиты информации представляет собой совокупность органов и исполнителей, используемой ими техники защиты информации, а также объектов защиты, организованная и функционирующая по правилам, установленным соответствующими правовыми, организационно-распорядительными и нормативными документами в области защиты информации. Так же является составной частью системы обеспе-

чения национальной безопасности Российской Федерации и призвана защищать безопасность государства от внешних и внутренних угроз в информационной сфере.

Функционирование государственной системы защиты информации осуществляется на основании законности:

- Конституция Российской Федерации
- ФЗ «О безопасности»
- ФЗ «О государственной тайне»
- ФЗ «Об информации, информатизации и защите информации»
- ФЗ «Об участии в международном информационном обмене»
- Доктрина информационной безопасности Российской Федерации
- Положение о государственной системе защиты информации в Российской Федерации от иностранных технических разведок и от утечки по техническим (утверждено Постановлением Совета Министров – Правительства Российской Федерации от 15 сентября 1993 г. №912–51)
- Указы президента Российской Федерации (№1085 от 16.8.2004 г.)
- Постановления правительства Российской Федерации
- Другие правовые акты федеральных органов власти в области защиты информации

Главными направлениями работ по защите информации являются:

- обеспечение эффективного управления системой защиты информации;
- определение сведений, охраняемых от технических средств разведки, и демаскирующих признаков, раскрывающих эти сведения;
- анализ и оценка реальной опасности перехвата информации техническими средствами разведки, несанкционированного доступа, разрушения (уничтожения) или искажения информации путем преднамеренных программно-технических воздействий в процессе ее обработки, передачи и хранения в технических средствах, выявление возможных технических каналов утечки сведений, подлежащих защите;

- разработка организационно-технических мероприятий по защите информации и их реализация;
- организация и проведение контроля состояния защиты информации.

Деятельность организуют следующие организации (рис.4):

Федеральная служба технического и экспортного контроля (ФСТЭК России) и ее территориальные органы (региональные управления в субъектах Российской Федерации)

Федеральные органы исполнительной власти, другие органы и организации Российской Федерации, руководящие работники которых входят в состав коллегии ФСТЭК России по должности (Минюст, Минобороны, МЧС, МВД, МИД, Минпромэнерго, Минэкономразвития, Минприроды, ФСО, ФСБ, СВР, ГУСП, РАН, ЦБР)

Структурные подразделения по защите информации федеральных органов исполнительной власти, других органов государственной власти и организаций Российской Федерации

Предприятия, проводящие работы с использованием сведений, отнесенных к информации ограниченного доступа, и их подразделения по защите информации

Научно-исследовательские организации по проблемам защиты информации

Организации-разработчики средств защиты информации, защищенных технических средств и средств контроля эффективности защиты информации

Предприятия, оказывающие услуги в области защиты информации

Организации Федерального агентства по техническому регулированию и метрологии (бывшего Госстандарта России), выполняющие работы по стандартизации в области защиты информации

Органы системы лицензирования деятельности в области защиты информации

Органы системы сертификации средств защиты информации

Органы системы аттестации объектов защиты по требованиям безопасности информации

Ответственность за нарушение ИБ

Виды ответственности [Конституция РФ 118.2.]: Судебная власть осуществляется посредством конституционного, гражданского, административного и уголовного судопроизводства.

Уголовный кодекс РФ:

- Статья 2. Задачи Уголовного кодекса Российской Федерации. 1. Задачами настоящего Кодекса являются: охрана прав и свобод человека и гражданина, собственности, общественного порядка и общественной безопасности, окружающей среды, конституционного строя Российской Федерации от преступных посягательств, обеспечение мира и безопасности человечества, а также предупреждение преступлений.
- Статья 44. Виды наказаний. Видами наказаний являются:
  - а) штраф;
  - б) лишение права занимать определенные должности или заниматься определенной деятельностью;
  - в) лишение специального, воинского или почетного звания, классного чина и государственных наград;
  - г) обязательные работы;
  - д) исправительные работы;
  - е) ограничение по военной службе;
  - ж) конфискация имущества;(утратил силу);
  - з) ограничение свободы;
  - з.1) принудительные работы;
  - и) арест;
  - к) содержание в дисциплинарной воинской части;
  - л) лишение свободы на определенный срок;
  - м) пожизненное лишение свободы;
  - н) смертная казнь.

- Глава 28. Преступления в сфере компьютерной информации
- Статья 272. Неправомерный доступ к компьютерной информации. 1. Неправомерный доступ к охраняемой законом компьютерной информации, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование компьютерной информации,
- Статья 273. Создание, использование и распространение вредоносных компьютерных программ. 1. Создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации,
- Статья 274. Нарушение правил эксплуатации средств хранения, обработки или передачи компьютерной информации и информационно-телекоммуникационных сетей. 1. Нарушение правил эксплуатации средств хранения, обработки или передачи охраняемой компьютерной информации либо информационно-телекоммуникационных сетей и оконечного оборудования, а также правил доступа к информационно-телекоммуникационным сетям, повлекшее уничтожение, блокирование, модификацию либо копирование компьютерной информации, причинившее крупный ущерб,
- Статья 274.1. Неправомерное воздействие на критическую информационную инфраструктуру Российской Федерации – для КИИ.
- Статья 146. Нарушение авторских и смежных прав.
- Статья 159.6. Мошенничество в сфере компьютерной информации.
- Статья 137. Нарушение неприкосновенности частной жизни.
- Статья 138. Нарушение тайны переписки, телефонных переговоров, почтовых, телеграфных или иных сообщений.
- Статья 283. Разглашение государственной тайны. Статья 283.1. Незаконное получение сведений, составляющих государственную тайну
-
- Кодекс РФ об административных правонарушениях:
- Статья 1.2. Задачи законодательства об административных правонарушениях. Задачами законодательства об административных правонарушениях являются защита

личности, охрана прав и свобод человека и гражданина, охрана здоровья граждан, санитарно-эпидемиологического благополучия населения, защита общественной нравственности, охрана окружающей среды, установленного порядка осуществления государственной власти, общественного порядка и общественной безопасности, собственности, защита законных экономических интересов физических и юридических лиц, общества и государства от административных правонарушений, а также предупреждение административных правонарушений.

- Статья 3.2. Виды административных наказаний. 1. За совершение административных правонарушений могут устанавливаться и применяться следующие административные наказания:
  - 1) предупреждение;
  - 2) административный штраф;
  - 3) возмездное изъятие орудия совершения или предмета административного правонарушения;(утратил силу. - Федеральный закон от 28.12.2010 N 398-ФЗ;)
  - 4) конфискация орудия совершения или предмета административного правонарушения;
  - 5) лишение специального права, предоставленного физическому лицу;
  - 6) административный арест;
  - 7) административное выдворение за пределы Российской Федерации иностранного гражданина или лица без гражданства;
  - 8) дисквалификация;
  - 9) административное приостановление деятельности;(п. 9 введен Федеральным законом от 09.05.2005 N 45-ФЗ)
  - 10) обязательные работы;(п. 10 введен Федеральным законом от 08.06.2012 N 65-ФЗ)
  - 11) административный запрет на посещение мест проведения официальных спортивных соревнований в дни их проведения.(п. 11 введен Федеральным законом от 23.07.2013 N 192-ФЗ)
  - 2. В отношении юридического лица могут применяться административные наказания, перечисленные в пунктах 1 - 4, 9 части 1 настоящей статьи.

- 3. Административные наказания, перечисленные в пунктах 3 - 11 части 1 настоящей статьи, устанавливаются только настоящим Кодексом.

-
- Гражданский кодекс РФ:
- Статья 2. Отношения, регулируемые гражданским законодательством. 1. Гражданское законодательство определяет правовое положение участников гражданского оборота, основания возникновения и порядок осуществления права собственности и других вещных прав, прав на результаты интеллектуальной деятельности и приравненные к ним средства индивидуализации (интеллектуальных прав), регулирует отношения, связанные с участием в корпоративных организациях или с управлением ими (корпоративные отношения), договорные и иные обязательства, а также другие имущественные и личные неимущественные отношения, основанные на равенстве, автономии воли и имущественной самостоятельности участников.
- Статья 12. Способы защиты гражданских прав. Защита гражданских прав осуществляется путем:
  - признания права;
  - восстановления положения, существовавшего до нарушения права, и пресечения действий, нарушающих право или создающих угрозу его нарушения;
  - признания оспоримой сделки недействительной и применения последствий ее недействительности, применения последствий недействительности ничтожной сделки;
  - признания недействительным решения собрания;(абзац введен Федеральным законом от 30.12.2012 N 302-ФЗ)
  - признания недействительным акта государственного органа или органа местного самоуправления;
  - самозащиты права;
  - присуждения к исполнению обязанности в натуре;
  - возмещения убытков;
  - взыскания неустойки;
  - компенсации морального вреда;
  - прекращения или изменения правоотношения;

- неприменения судом акта государственного органа или органа местного самоуправления, противоречащего закону;
- иными способами, предусмотренными законом.

Ответственность так же предусмотрена за нарушение технических регламентов и требований к лицензионным выдам деятельности.

Контрольные вопросы и задания

Зачем нужна иерархия нормативно-правовых документов?

Что такое конфиденциальная информация?

Какой государственный орган координирует деятельность по защите информации в Российской Федерации?

Чем отличается уголовная ответственность от административной?

Тема 2.2. Система обеспечения информационной безопасности организации

1. Архитектура системы обеспечения информационной безопасности
2. Политика информационной безопасности
3. Проектирование системы менеджмента информационной безопасности
4. Регламенты и правила информационной безопасности

Систему управления защитными мерами организации (рис. 5) называют:

1. СОИБ – система обеспечения информационной безопасности.
2. СУИБ – система управления информационной безопасности.
3. СМИБ – система менеджмента информационной безопасности.
4. ISMS – information security management system.

Принципы СУИБ:

1) многоуровневая защита – защита от различных видов атак;
2) многослойная (многоэшелонная) защита – подстраховка, дополнение, компенсация;
3) модульная архитектура – упрощение создания. Применение модулей обеспечивает:

а) стандартизированность,

б) тестируемость,

в) обновляемость;

4) компонентный подход – оптимальное распределение функций;
5) минимальное вмешательство человека – меньше зависимость от «человеческого фактора»;
6) прозрачность – обнаружение недостатков и «узких мест»;

7) секретная безопасность – скрытие недостатков, уязвимостей.

Типичная СУИБ состоит из:

- политика, архитектура,
- регламенты:
- тех.-/бизнес- процесс,
- контроль доступа к информации, инфраструктуре,
- управление программно-аппаратной инфраструктурой;
- регламент использования средств защиты;
- управление персоналом,
- служба безопасности,
- контроли: средства защиты, правила,
- мероприятия:
- жизненный цикл,
- режимы,
- реагирование.

По функциям СУИБ согласно стандартам Certified Information Systems Security Professional (CISSP) состоит из:

1) из подсистемы защиты периметра сети, включающую:

а) подсистему обеспечения безопасности межсетевых взаимодействий;

б) подсистему фильтрации контента и предотвращения утечки конфиденциальной информации;

2) подсистемы обнаружения и предотвращения атак, крупной частью которой является:

а) подсистема защиты от вредоносного ПО;

3) криптографической подсистемы;

4) подсистемы администрирования безопасности, включающую:

а) подсистему резервного копирования и восстановления данных;

б) подсистему установки обновлений ПО;

5) подсистемы мониторинга и аудита безопасности, содержащую:

а) подсистему контроля целостности данных.

В структуре СУИБ выделяют :

1. Руководящий комитет по надзору, в который входят:

1) исполнительное руководство,

2) владельцы данных,

3) владелец системы,

4) и сотрудник ответственный за безопасность.

2. Политика безопасности организации, в которую входят принципы, критерии.

3. Проект управления безопасностью, который регулирует правила применения защитных мер.

4. Проект управления персоналом, который регулирует правила работы сотрудников.

5. Проект управления ресурсами, который регулирует правила использования инфраструктуры.

6. Проект мониторинга (надзор), в который входят задачи наблюдение и оценка состояния.

7. Проект непрерывности бизнеса, который определяет действия при возникновении инцидентов.

8. Проект управления зрелостью, который регулирует совершенствование системы.

9. План создания и обслуживания СУИБ, который определяет жизненный цикл системы.

Политика безопасности организации – это:

- совокупность руководящих принципов, правил, процедур и практических приёмов в области безопасности, которые регулируют управление, защиту и распределение ценной информации;
- документ определяющий и регулирующий отношения между защитными мерами.

Виды политик безопасности:

1. По составу:

1) организационная – акцент на общую безопасность;

2) проблемная – акцент на существенные вопросы безопасности;

3) конкретная – акцент на конкретные системы.

2. По требованиям:

1) регулирующая – требования;

2) рекомендательная – рекомендации;

3) информирующая – общие положения безопасности.

Типовое содержание политики безопасности:

1) цели и задачи безопасности:

а) стратегические;

б) тактические;

в) оперативные;

2) требования и базисы:

а) законы и стандарты;

б) распределение ответственности;

3) модели безопасности и модели защиты;

4) архитектура системы защиты;

5) общая стратегия построения и использования СУИБ.

Проектирование СУИБ – важный этап защиты организации. Специалистами по ИБ сформулированы рекомендации, которых необходимо придерживаться при проектировании.

Подходы к проектированию:

1) сверху-вниз – построение СУИБ начинается с целей руководства;

2) снизу-вверх – построение СУИБ исходит от требуемых защитных мер;

3) горизонтальный – системный подход, отвечающий на основные вопросы:

а) зачем, т. е. определить цели;

б) почему, т. е. определить причины создания СУИБ (угрозы);

в) что, т. е. определить объект защиты;

г) кто, т. е. определить субъекты, ответственные за работу СУИБ;

д) где, т. е. определить место, на которое распространяется защита;

е) когда, т. е. определить время и режимы;

ж) чем и как, т. е. определить способы защиты;

з) сколько, т. е. определить стоимость СУИБ.

Разработка проекта состоит:

1) в назначении ответственных;

2) в выборе целей и приоритетов;

3) в определении необходимых действий;

4) в спецификации результатов (артефакты), которые необходимо получить.

Проект СУИБ разрабатываться в определённом порядке (плане). План проектирования СУИБ:

1. Определение целей.
2. Формирование политики безопасности и структуры проекта. Этот этап включает:

1) категорирование ресурсов;

2) оценка рисков;

3) права и обязанности;

4) нормативные акты;

5) составление плана и сметы создания СУИБ.

3. Выбор мер и средств защиты.
4. Регламентирование мер безопасности информационных процессов организации, составление правил и режимов работы организации.
5. Обеспечение непрерывности бизнес-процессов. На данном этапе составляются планы:

1) реагирования на инциденты;

2) восстановления от инцидентов;

3) расследования инцидентов.

6. Проведение аудита и тестирования.

В целом СУИБ может следовать каскадной или итеративной модели жизненного цикла.

Каскадная модель состоит из следующих этапов:

1) проектирование;

2) планирование;

3) внедрение;

4) аттестация;

5) эксплуатация;

6) ликвидация.

В итеративной модели неограниченное количество циклов, и каждый цикл проходит фазы:

1) PLAN – планирование действий;

2) DO – выполнение мероприятия;

3) CHECK – проверка полученных результатов;

4) ACT – анализ и принятие решений.

В курсах Certified Information Systems Security Professional (CISSP) жизненный цикл СУИБ включает следующее:

1. *Планирование и Организация*:

1) получение одобрения руководства;

2) создание руководящего комитета по надзору (oversight steering committee);

3) оценка бизнес-драйверов (люди, информация или задачи, которые обеспечивают реализацию бизнес-целей компании);

4) создание профиля угроз компании;

5) проведение оценки рисков;

6) разработка архитектуры безопасности на организационном, прикладном, сетевом и компонентном уровнях;

7) определение решений на каждом уровне архитектуры;

8) получение согласия руководства на дальнейшие действия.

2. *Реализация (внедрение)*:

1) распределение ролей и обязанностей;

2) разработка и внедрение политики безопасности, процедур, стандартов, базисов и различных руководств;

3) выявление критичных данных на этапах хранения и передачи;

4) реализация следующих проектов:

а) идентификация и управление активами;

б) управление рисками;

в) управление уязвимостями;

г) соответствие требованиям;

д) управление идентификацией и доступом;

е) управление изменениями;

ж) жизненный цикл разработки программного обеспечения;

з) планирование непрерывности бизнеса;

и) обучение и повышение осведомлённости;

к) физическая безопасность;

л) реакция на инциденты;

5) внедрение решений (административных, технических, физических) по каждому проекту;

6) разработка решений по аудиту и мониторингу для каждого проекта;

7) установка целей, соглашений об уровне обслуживания (SLA) и метрик по каждому проекту.

3. *Функционирование и Поддержка*:

1) соблюдение установленных процедур для обеспечения базисных уровней в каждом реализованном проекте;

2) проведение внутренних и внешних аудитов;

3) выполнение задач, намеченных в каждом проекте;

4) управление соглашениями об уровне обслуживания по каждому проекту.

4. *Мониторинг и Оценка*:

1) анализ лог-файлов, результатов аудита, собранных значений метрик и SLA по каждому проекту;

2) оценка достижения целей по каждому проекту;

3) проведение ежеквартальных встреч с руководящими комитетами;

4) совершенствование действий каждого этапа и их интеграция в фазу *Планирования и Организации*.

Регламенты информационной безопасности

Основные регламенты:

- регламент изменения аппаратно-программной конфигурации системы;
  - процедура выбора и компонентов;
  - процедура установки и модификации компонентов;
  - процедура обслуживания и тестирования;
  - права внесения изменений в конфигурацию аппаратно-программных средств;
  - процедура утилизации;
- регламент использования средств защиты:
  - инструкция по организации антивирусной защиты,

    - инструкция по организации защиты сети,
    - инструкция по применению криптографической защиты;
    - инструкция по защите отдельных узлов и каналов;
- регламент доступа к системе;
- регламент управления персоналом;
- регламент службы безопасности;
- регламент обеспечения непрерывности.

- Вопросы управления доступом:
    - К чему каждый пользователь должен иметь доступ?
    - Кто дает разрешение на доступ и сам доступ?
    - Как принимаются решения о доступе в соответствии с политиками?
    - Остается ли доступ у уволенных сотрудников?
    - Как поддерживать в порядке нашу динамичную и постоянно меняющуюся среду?
    - Каков процесс отзыва прав доступа?
    - Каким образом осуществляется централизованное управление правами доступа и их мониторинг?
    - Почему сотрудники должны помнить по восемь паролей?
    - Как нам централизовать доступ?
    - Как мы управляем доступом наших сотрудников, клиентов, партнеров?
    - Как мы можем убедиться, что мы соответствуем необходимому набору требований?

Правила управления доступом [15]:

- Запретить доступ к системам пользователям, не прошедшим аутентификацию, и анонимным учетным записям.
- Огранить и контролировать использование административных и иных привилегированных учетных записей.
- Блокировать учетную запись или вносить задержку после нескольких неудачных попыток регистрации.
- Удалять учетные записи уволенных сотрудников сразу же после их ухода из компании.
- Блокировать учетные записей, которые не использовались 30-60 дней.
- Внедрить строгие критерии доступа.
- Применять принцип «должен знать» и принцип минимальных привилегий.
- Отключить ненужные функции системы, службы и порты.
- Заменить пароли «по умолчанию» для встроенных учетных записей.
- Ограничить и контролировать правила глобального доступа.
- Убедиться, что названия учетных записей не раскрывают должностных обязанностей пользователей, которым они принадлежат.
- Удалить излишние правила использования ресурсов учетными записями и группами.
- Удалить из списков доступа к ресурсам излишние идентификаторы пользователей, учетные записи и роли.
- Организовать периодическую смену паролей.
- Установить требования к паролям (по длине, содержимому, сроку действия, распространению, хранению и передаче).
- Организовать журналирование системных событий и действий пользователей, а также периодический просмотр журналов.
- Обеспечить защиту журналов регистрации событий.
- Управление ключами:
    1. хранить только в закрытом виде,
    2. делать резервные копии,

3. передавать только безопасным способом,

4. ограничить срок действия ключа.

Управление персоналом

Основные роли:

- владельцы, руководство;
- персонал: служащие, сотрудники, специалисты, управляющие;
- охранники;
- клиенты, посетители, поставщики.

Принципы работы с посетителями, поставщиками, клиентами:

- разделение контролируемой территории на зоны по степени секретности;
- регламент посещения: часы, области, сопровождение;
- принцип «чистого» рабочего стола;
- множественные критерии выбора поставщиков;
- ответственность поставщика за поставляемое оборудование и работу;
- контроль выполняемых работ;
- мониторинг, анализ и проверка цепочек поставок;
- своевременное информирование клиентов, поставщиков об инцидентах.

Возможные последствия для организации от действий сотрудников:

- разглашение (Р) информации;
- утечка (У) информации;

- несанкционированный доступ (НСД): предоставление нелегального доступа, неправильное использование средств вычислительной техники.

- Основные причины, приводящие к нарушениям со стороны сотрудников:

- недостаточный уровень знания положений нормативных актов и внутренних организационно-распорядительных документов предприятия, регламентирующих деятельность по защите информации;

- слабый контроль со стороны руководителей всех уровней за состоянием защиты информации и эффективностью принимаемых мер по недопущению утечки этой информации;

- недостаточное внимание к вопросам организации работы с персоналом предприятия, изучению морально-деловых качеств сотрудников предприятия, допущенных к конфиденциальной информации;

- несвоевременное принятие эффективных и действенных мер по предотвращению разглашения персоналом предприятия конфиденциальной информации, а также мер по фактам нарушения норм и правил защиты информации сотрудниками предприятия.

Этапы работы с персоналом:

- найм,

- обучение,

- контроль,

- увольнение.

Принципы принятия на работу:

- уровень фильтрации зависит от выбираемой позиции;

- сотрудники это инвестиции;

- не нанять проще чем уволить;

- учет конфликтов интересов;

- раннее заключение договоров: договор о не разглашении конфиденциальной информации.

При принятии на работу проверяются:

- кандидат:
    1. характер,
    2. навыки,
    3. вредные привычки;
- резюме:
    1. образование,
    2. стаж,
    3. сертификаты;
- биография:
    1. долги,
    2. судимости,
    3. наркотики.

Формы обучения:

- допуск к работе:
    1. ознакомление с правилами допуска к работе,
    2. инструктажи на рабочем месте,
- повышение осведомленности:
    1. ознакомление с действующей политикой и регламентами (регламент действий пользователя), стандартами,
    2. семинары, вебинары, конференции и т.п.;
- повышение квалификации;

- переподготовка;
- стажировка.

Обучение включает в себя так же:

1. контроль сотрудника при выездных формах обучения;
2. оценка результатов;
3. получение подписей о прохождении.

Принципы контроля персонала:

- отчет снизу-вверх,
- контроль сверху-вниз,
- выявление необычных действий,
- информируемость.

Направления деятельности по предотвращению нарушений:

- изучение морально-деловых качеств сотрудников предприятия;
- повышение ответственности сотрудников всех категорий за сохранение в тайне доверенных по службе сведений конфиденциального характера, например: ротация обязанностей, разделение обязанностей, разделение знаний, двойное управление;
- проведение профилактической работы по предупреждению (исключению) утечки конфиденциальной информации путем ее разглашения;
- повышение уровня теоретических знаний и практических навыков сотрудников в вопросах защиты конфиденциальной информации – обучение;
- создание и поддержание устойчивого морально-психологического климата в коллективе предприятия;
- создание и применение системы стимулирования труда сотрудников, допущенных к конфиденциальной информации.

Методы мотивации:

- непосредственная:
    1. убеждение,
    2. внушение,
    3. агитация;
- властная:
    1. указание,
    2. приказ,
    3. распоряжение и др.;
- стимулирующая:
    1. моральная,
    2. материальная,
    3. трудовая.

Действия при увольнении:

- немедленное лишение прав доступа к системе;
- запрет свободного перемещения по территории организации;
- возврат имущества организации;
- наблюдение за увольняемым;
- договор о лояльности.

Контрольные вопросы и задания

Опишите систему управления защитой выбранной организации.

Чем план отличается от проекта?

Приведите примеры элементов структуры проекта.

Опишите процесс перехода от проекта СМИБ к плану его построения.

Что входит в минимальный набор (ядро) составляющий политику безопасности?

Как «заставить» сотрудника соблюдать правила безопасности?

Как регламентировать телефонные звонки и переписку сотрудников?

Допустимо ли требовать от уволенного сотрудника соблюдение конфиденциальной информации?

Тема 3.1. Средства защиты информации

1. Защита служб. Антивирусы.

2. Система восстановления. Резервные копии, Транзакции. RAID.

3. Контроль периметра. Сетевые экраны. Демилитаризованная зона (DMZ)

4. Средства мониторинга. Система обнаружения атак (IDS, IPS). Системы защиты от утечек (DLP).

Цель применения *программно-аппаратных средств защиты информации* (СЗИ) – защита информационных процессов.

Для повышения надёжности и эффективности СЗИ часть функций реализуется в виде аппаратных устройств.

По назначению СЗИ подразделяется:

1) на *средства администрирования* – внедрение СЗИ и управление их работой;
2) *средства контроля периметра* – создание и поддержание границы периметра;
3) *средства защиты служб* – обеспечение правильной работы информационных процессов;
4) *средства восстановления* – обеспечение непрерывной работы информационных процессов;
5) *средства мониторинга* – выявление и реагирование на инциденты;
6) *вспомогательные средства защиты* – реализация отдельных методов безопасности.

Рассмотрим наиболее значимые программно-аппаратные средства защиты информации.

Антивирусная защита

Вредоносная программа (зловред, malware) – код, осуществляющий негативное воздействие на состояние и работу системы.

Классификация зловредов:

1) вредоносные:

а) virus – вредоносный, саморазмножающийся код;

б) logic bomb – вредоносный код, запускающийся при определенных условиях;

в) worm – самостоятельно распространяющийся вредонос;

г) trojan – вредонос, распространяющийся обманом;

д) EICAR-Test-File (European Institute for Computer Antivirus Research) – 16-битный COM-файл, выводящий сообщение;

2) загрязняющие (вспомогательные):

а) rootkit – утилита, действующая в ядре операционной системы;

б) backdoor – средство скрытого несанкционированного доступа;

в) exploit – способ использования уязвимости системы, позволяющий внедрять вредоносный код;

г) shellcode – код предоставляющий доступ к командной консоли системы;

д) spyware – утилиты для незаконного сбора данных;

3) потенциально-вредоносные (нежелательные):

а) adware – «надоедливые», мешающие программы;

б) shovelware – бесполезные, самоустанавливающиеся программы;

в) вредоносные утилиты – специальные инструменты для проведения анализа и атак;

г) подозрительные упаковщики – утилиты для сокрытия зловредов.

Антивирусные технологии:

1) сигнатурный анализ – обнаружение по заданному шаблону;

2) эвристический анализ – обнаружение по критериям;

3) анализ поведения – обнаружение подозрительного поведения с помощью: эмуляции кода и песочницы;

4) анализ целостности – обнаружение вмешательства;

5) облачная проверка – проверка внешними экспертами и экспертными системами.

Режим работы антивирусных средств:

1) по требованию;

2) фоновый – во время выполнения операций;

3) по расписанию;

4) автоматический – при появлении внешних событий.

Параметры антивирусов:

1) расписание проверок: время, период, события;

2) области проверки: оперативная память, загрузочная область, диски, внешние хранилища;

3) объекты проверки: типы вредоносов, типы файлов, архивы, выполняемые действия;

4) способ лечения: удаление, очистка, карантин, уведомление;

5) расписание обновлений: время, период, события;

6) источники обновления: официальные, локальные;

7) самозащита: изменение настроек, остановка задач.

Современные антивирусные программы позволяют также контролировать активность программ и пользователя, содержат функции настройки и оптимизации системы.

Система резервного копирования

Система резервного копирования в соответствии с политикой, автоматически создает резервные копии данных и системы. В случае сбоев, после расследования инцидента, возможно восстановить испорченные данные и систему. Восстановление обычно происходит вручную, по требованию пользователя.

В регламенте резервного копирования определяется:

- Частота резервного копирования:
- периодическая;
- плановая;
- по событию.
- Принцип копирования:
  1. Физическое (клонирование) – копирование по блокам. Достоинства: возможен пропуск свободных блоков. Недостатки: сохраняются ненужные элементы, дефектные блоки.
  2. Логическое (копирование) – копирование рекурсивно по структуре данных. Достоинства: сохраняется структура, возможно пропустить ненужный элементы, возможно восстановление отдельного элемента. Недостатки: восстановление возможно только в такой же системе, требуется больше места на сохранение структуры и связей, занимает больше времени.
- Процесс копирования: на горячую – система продолжает работать, на холодную – система отключается во время копирования.
- Вид резервной копии:
  1. полная – содержит все данные, нужные для восстановления;
  2. разностная – содержит только новые и изменённые данные по сравнению с предыдущей полной копии;
  3. инкрементная – содержит новые и изменённые данные от другой инкрементной копии;
  4. декрементная – содержит старые и изменённые данные, по сравнению с полной текущей копией;

- Место хранения резервных копий:
  1. локально,
  2. на отдельном носителе,
  3. удаленно, по сети
  4. в облаке.
- Ротация резервных копий – схема хранения и обновления:
  1. одноразовая – хранится только одна копия;
  2. простая – хранится последовательность копий (износ хранилища);
  3. дед-отец-сын – периодически (раз месяц) создается полная копия (дед), по событию или раз в неделю создается дифференциальная копия (отец), регулярно, например один раз в день делается инкриментная копия (сын);
  4. хайнойская башня – одна полная, много инкрементных;
  5. 10 наборов - циклическая смена носителей;
- Дополнительные параметры: сжатие, шифрование.

Для повышения надежности хранения данный используется RAID (Redundant Array of Independent Disks) – избыточный массив независимых (самостоятельных, раньше было – дешовых) дисков). Несколько физических дисковых устройств объединяются в логический модуль для повышения отказоустойчивости и (или) производительности.

Паттерсон с коллегами из Беркли представили спецификации пяти уровней RAID, которые стали стандартом де факто:

- RAID 1 — зеркальный дисковый массив;
- RAID 2 — зарезервирован для массивов, которые применяют код Хемминга;
- RAID 3 — дисковый массив с выделенным диском чётности;
- RAID 4 — дисковый массив с чередованием и выделенным диском чётности;
- RAID 5 — дисковый массив с чередованием, в том числе данных чётности (нет диска, выделенного для хранения чётности — блоки чётности чередуются с блоками данных на каждом диске).

Межсетевой экран

Сетевой экран (firewall, brandmauer) это фильтрующий сетевой шлюз.

Цель применения – блокировка нежелательного сетевого трафика.

По уровню вмешательства, сетевые экраны классифицируют следующим образом:

1) канальный – мост с фильтрацией;
2) сетевой – пакетный фильтр, отдельно выделяют пакетный сетевой экран с отслеживанием состояний (statefull);
3) сеансовый – исключение прямых соединений;
4) приложений – проксирование соединений;
5) экспертный уровень – на нескольких уровнях, специализация.

По расположению сетевые экраны бывают:

1) периметровые – на границе с внешней сетью;
2) межсетевые – на сетевых устройствах;
3) персональные – на каждом компьютере.

Используя сетевые экраны, организуются специальные сетевые зоны, с определёнными правилами.

Демилитаризованная зона (DMZ) – изолированный сегмент сети. Виды зон:

1) «сэндвич» – между сетевыми экранами;
2) «тупик» – выделенный сегмент;
3) «остров» – на периметре сети.

Кроме ограничения трафика сетевые экраны, благодаря их расположению, могут осуществлять функции транзита и преобразования трафика.

Система выявления и предотвращения вторжений

Система выявления вторжений (IDS, intrusion detection system) – система детектирования и уведомления о подозрительных действиях.

Система предотвращения вторжений (IPS, intrusion prevention system) – это IDS с возможностью принимать действия по защите от атак.

Структура IDS:

1. Сенсоры – собирают трафик и данные о действиях.
2. Анализаторы – ищут подозрительные действия.
3. Административные интерфейсы – принимают сообщения от анализатора для последующей обработки.

Виды IDS:

1) host-based (HIPS/HIDS) – следят за состоянием компьютеров;
2) network-based (NIPS/NIDS) – следят за сетевым трафиком.

Методы обнаружения вторжений:

1. Сигнатурный (signature based) – ищет сигнатуры (признаки) атак, выделяют:

1) отслеживающий шаблоны (pattern matching) – анализирует данные (файлы, трафик), свидетельствующие об атаке;
2) отслеживающий состояние (stateful matching) – анализирует последовательность действий, приводящих в запрещённые состояния.

2. Основанный на аномалиях (anomaly based) – рассчитывает и сравнивают рейтинг «аномальности» данных с порогом, выделяют:

1) основанный на статистических аномалиях (statistical anomaly-based) – строит профиль «нормальной» деятельности;

2) основанный на аномалиях протоколов (protocol anomaly-based) – выделяет действия, эксплуатирующие уязвимости;

3) основанный на аномалиях трафика (traffic anomaly-based) – выделяет необычное (новое) использование системы.

3. Основанный на правилах (rule-based) или эвристический (heuristic-based) – экспертные системы, основанные на базе знаний (knowledge base), механизме логических выводов (inference engine) и программировании на основе правил (rule-based programming).

Система предотвращения утечек

Data Leak Prevention (DLP) – технологии предотвращения утечек конфиденциальной информации из информационной системы вовне.

DLP-системы строятся на анализе потоков данных, пересекающих периметр защищаемой информационной системы. Точками анализа являются: носители, сеть, потоки ввода-вывода.

Распознавание конфиденциальной информации в DLP-системах производится двумя способами:

1) анализ формальных признаков (например грифа, документа, специально введённых меток, сравнением хеш-функции);

2. анализ контента (содержимого).

Ошибки при работе DLP:

3. ложное срабатывание (более вероятны для 2-го способа проверки);

4. пропуск конфиденциальной информации (более вероятны для 1-го способа).

Проблемы использования DLP-систем:

5. шифрование, стеганография – для проверки необходимо дешифровать/найти данные;

6) право на частную жизнь – проверка трафика не может противоречить праву на частную жизнь и тайну связи.

Другие СЗИ

DDOS-prevention, криптоутилиты, сканеры уязвимостей, системы аутентификации, службы каталогов, SIEM-системы, программно-аппаратные замки, система резервного копирования.

Контрольные вопросы и задания

Существуют ли только аппаратные средства защиты информации?

Какие СЗИ используются в вашей организации для обеспечения конфиденциальности, целостности, доступности?

Является ли вредоносным объект с хеш-суммой d4b7b4b1aac1c51d1eec18bc5ca26fbb3053af5b9fee2f6ae0966b2981edbbcb?

Какие порты используются протоколом SSH?

Тема 3.2. Проверка информационной безопасности

1. Проверка информационной безопасности. Цели и задачи, способы оценки ИБ.

2. Аудит. Цели, принципы, виды аудита. Требования к аудитору.

3. Пентестинг. Методы и средства тестирования.

Лицензирование

Контроль деятельности организации в области ИБ осуществляется с помощью лицензирования.

ФЗ «О лицензировании отдельных видов деятельности» № 99-ФЗ, 04.05.2011:

- ст.2. Цели, задачи лицензирования отдельных видов деятельности и критерии определения лицензируемых видов деятельности. 3. К лицензируемым видам деятельности относятся виды деятельности, осуществление которых может повлечь за собой нанесение указанного в части 1 настоящей статьи ущерба и регулирование которых не может осуществляться иными методами, кроме как лицензированием.
- ст.3. Основные понятия, используемые в настоящем Федеральном законе: 2) лицензия - специальное разрешение на право осуществления юридическим лицом или индивидуальным предпринимателем конкретного вида деятельности (выполнения работ, оказания услуг, составляющих лицензируемый вид деятельности), которое подтверждается документом, выданным лицензирующим органом на бумажном носителе или в форме электронного документа, подписанного электронной подписью, в случае, если в заявлении о предоставлении лицензии указывалось на необходимость выдачи такого документа в форме электронного документа;
- ст.1. 2. Положения настоящего Федерального закона не применяются к отношениям, связанным с осуществлением лицензирования:

1. 1) использования атомной энергии;
2. 2) производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции;
3. 3) деятельности, связанной с защитой государственной тайны;
4. 4) деятельности кредитных организаций;

5. 5) деятельность по проведению организованных торгов;
6. 6) видов профессиональной деятельности на рынке ценных бумаг;
7. 7) деятельности акционерных инвестиционных фондов, деятельности по управлению акционерными инвестиционными фондами, паевыми инвестиционными фондами, негосударственными пенсионными фондами;
8. 8) деятельности специализированных депозитариев инвестиционных фондов, паевых инвестиционных фондов и негосударственных пенсионных фондов;
9. 9) деятельности негосударственных пенсионных фондов по пенсионному обеспечению и пенсионному страхованию;
10. 10) клиринговой деятельности;
11. 11) страховой деятельности.

- ст.12. Перечень видов деятельности, на которые требуются лицензии (57 видов):

1. 1) разработка, производство, распространение шифровальных (криптографических) средств, информационных систем и телекоммуникационных систем, защищённых с использованием шифровальных (криптографических) средств, выполнение работ, оказание услуг в области шифрования информации, техническое обслуживание шифровальных (криптографических) средств, информационных систем и телекоммуникационных систем, защищённых с использованием шифровальных (крипто графических) средств (за исключением случая, если техническое обслуживание шифровальных (криптографических) средств, информационных систем и телекоммуникационных систем, защищённых с использованием шифровальных (криптографических) средств, осуществляется для обеспечения собственных нужд юридического лица или индивидуального предпринимателя);

1. ПП "Об утверждении Положения о лицензировании деятельности по разработке, производству, распространению шифровальных (криптографических) средств..." №313, 16.04.2012

2. 2) разработка, производство, реализация и приобретение в целях продажи специальных технических средств, предназначенных для негласного получения информации;

1. ПП РФ "Об утверждении Положения о лицензировании деятельности по разработке, производству, реализации и приобретению в целях продажи специальных технических средств, предназначенных для негласного получения информации" №287, 12.04.2012

3. 3) деятельность по выявлению электронных устройств, предназначенных для негласного получения информации (за исключением случая, если указанная деятельность осуществляется для обеспечения собственных нужд юридического лица или индивидуального предпринимателя);

1. Постановление Правительства РФ "Об утверждении Положения о лицензировании деятельности по выявлению электронных устройств, предназначенных для негласного получения информации (за исключением случая, если указанная деятельность осуществляется для обеспечения собственных нужд юридического лица или индивидуального предпринимателя)" №314, 16.04.2012

4. 4) разработка и производство средств защиты конфиденциальной информации;

1. Постановление Правительства РФ "О лицензировании деятельности по разработке и производству средств защиты конфиденциальной информации" №171, 03.03.2012

5. 5) деятельность по технической защите конфиденциальной информации;

1. Постановление Правительства РФ "О лицензировании деятельности по технической защите конфиденциальной информации" №79, 03.02.2012

Ответственность за нарушения требований к лицензионной деятельности: обычно административная – штраф, конфискация, при крупном ущербе или получении крупного дохода – уголовная (171 УК РФ).

Сертификация

Требования к продукции и услугам устанавливается с помощью сертификации и техническим регулированием.

ФЗ "О техническом регулировании" №184-ФЗ, 27.12.2002:

1. ст.2. Основные понятия:

1.2. стандарт – документ, в котором в целях добровольного многократного использования устанавливаются характеристики продукции, правила осуществления и характеристики процессов проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации и утилизации, выполнения работ или оказания услуг. Стандарт также может содержать правила и методы исследований (испытаний) и измерений, правила отбора

образцов, требования к терминологии, символике, упаковке, маркировке или этикеткам и правилам их нанесения;

1.3. технический регламент – документ, который принят международным договором Российской Федерации, подлежащим ратификации в порядке, установленном законодательством Российской Федерации, или в соответствии с международным договором Российской Федерации, ратифицированным в порядке, установленном законодательством Российской Федерации, или федеральным законом, или указом Президента Российской Федерации, или постановлением Правительства Российской Федерации, или нормативным правовым актом федерального органа исполнительной власти по техническому регулированию и устанавливает обязательные для применения и исполнения требования к объектам технического регулирования (продукции или к продукции и связанным с требованиями к продукции процессам проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации и утилизации);

1.4. сертификация – форма осуществляемого органом по сертификации подтверждения соответствия объектов требованиям технических регламентов, положениям стандартов, сводов правил или условиям договоров;

1.5. сертификат соответствия - документ, удостоверяющий соответствие объекта требованиям технических регламентов, положениям стандартов, сводов правил или условиям договоров;

1.6. декларация о соответствии - документ, удостоверяющий соответствие выпускаемой в обращение продукции требованиям технических регламентов;

2. ст.5. Особенности технического регулирования в отношении оборонной продукции (работ, услуг), поставляемой по государственному оборонному заказу, продукции (работ, услуг), используемой в целях защиты сведений, составляющих государственную тайну или относимых к охраняемой в соответствии с законодательством Российской Федерации иной информации ограниченного доступа, продукции (работ, услуг), сведения о которой составляют государственную тайну, продукции, для которой устанавливаются требования, связанные с обеспечением безопасности в области использования атомной энергии, процессов проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации, утилизации, захоронения указанной продукции

2.2.2. Особенности технического регулирования в части разработки и установления обязательных требований государственными заказчиками, федеральными органами исполнительной власти, уполномоченными в области обеспечения безопасности, обороны, внешней разведки, противодействия техническим разведкам и технической защиты информации, государственного управления использованием атомной

энергии, государственного регулирования безопасности при использовании атомной энергии, в отношении продукции (работ, услуг), указанной в пункте 1 настоящей статьи, а также соответственно процессов ее проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации, утилизации, захоронения устанавливаются Президентом Российской Федерации, Правительством Российской Федерации в соответствии с их полномочиями.

2.3.4. Особенности оценки соответствия продукции (работ, услуг), указанной в пункте 1 настоящей статьи, а также соответственно процессов ее проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации, утилизации, захоронения устанавливаются Правительством Российской Федерации или уполномоченными им федеральными органами исполнительной власти.

3. ст.6. Цели принятия технических регламентов

3.2.1. Технические регламенты принимаются в целях:

3.2.1. защиты жизни или здоровья граждан, имущества физических или юридических лиц, государственного или муниципального имущества;

3.2.2. охраны окружающей среды, жизни или здоровья животных и растений;

3.2.3. предупреждения действий, вводящих в заблуждение приобретателей, в том числе потребителей;

3.2.4. обеспечения энергетической эффективности и ресурсосбережения.

3.3.2. Принятие технических регламентов в иных целях не допускается.

4. ст.11. Цели стандартизации. Целями стандартизации являются:

4.2. повышение уровня безопасности жизни и здоровья граждан, имущества физических и юридических лиц, государственного и муниципального имущества, объектов с учетом риска возникновения чрезвычайных ситуаций природного и техногенного характера, повышение уровня экологической безопасности, безопасности жизни и здоровья животных и растений;

4.3. обеспечение конкурентоспособности и качества продукции (работ, услуг), единства измерений, рационального использования ресурсов, взаимозаменяемости технических средств (машин и оборудования, их составных частей, комплектующих изделий и материалов), технической и информационной совместимости, сопоставимости результатов исследований (испытаний) и измерений, технических и экономико-статистических данных, проведения анализа характеристик продукции (работ,

услуг), планирования и осуществления закупок товаров, работ, услуг для обеспечения государственных и муниципальных нужд, добровольного подтверждения соответствия продукции (работ, услуг);

4.4. содействие соблюдению требований технических регламентов;

4.5. создание систем классификации и кодирования технико-экономической и социальной информации, систем каталогизации продукции (работ, услуг), систем обеспечения качества продукции (работ, услуг), систем поиска и передачи данных, содействие проведению работ по унификации.

5. Статья 18. Цели подтверждения соответствия. Подтверждение соответствия осуществляется в целях:

5.2. удостоверения соответствия продукции, процессов проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации и утилизации, работ, услуг или иных объектов техническим регламентам, стандартам, сводам правил, условиям договоров;

5.3. содействия приобретателям, в том числе потребителям, в компетентном выборе продукции, работ, услуг;

5.4. повышения конкурентоспособности продукции, работ, услуг на российском и международном рынках;

5.5. создания условий для обеспечения свободного перемещения товаров по территории Российской Федерации, а также для осуществления международного экономического, научно-технического сотрудничества и международной торговли.

6. Статья 19. Принципы подтверждения соответствия

6.2.1. Подтверждение соответствия осуществляется на основе принципов:

6.2.1. доступности информации о порядке осуществления подтверждения соответствия заинтересованным лицам;

6.2.2. недопустимости применения обязательного подтверждения соответствия к объектам, в отношении которых не установлены требования технических регламентов;

6.2.3. установления перечня форм и схем обязательного подтверждения соответствия в отношении определенных видов продукции в соответствующем техническом регламенте;

6.2.4. уменьшения сроков осуществления обязательного подтверждения соответствия и затрат заявителя;

6.2.5. недопустимости принуждения к осуществлению добровольного подтверждения соответствия, в том числе в определенной системе добровольной сертификации;

6.2.6. защиты имущественных интересов заявителей, соблюдения коммерческой тайны в отношении сведений, полученных при осуществлении подтверждения соответствия;

6.2.7. недопустимости подмены обязательного подтверждения соответствия добровольной сертификацией.

Для подтверждение соответствия необходимы процедуры проверки существующих мер защиты информации.

Задачи проверки информационной безопасности:

1) выбор и обоснование базисов (ориентиры) – проверка адекватности политики;

2) анализ активов, ресурсов и функций – требования стандартов:

а) классификация/категорирование,

б) определение приоритетов,

в) определение уровней критичности;

3) анализ рисков;

4) проверка состояния.

В проверках принимают участие:

1) владельцы активов;

2) специалисты;

3) служащие безопасности;

4) эксперты по рискам;

5) аналитики по безопасности;

6) аналитики данных;

7. аудиторы.

Способы проверки ИБ:

1) Испытание – проверка возможностей ИБ. Выделяют:

а) *сканирование* – проверка работоспособности защитных мер и наличие уязвимостей;

б) *проникновение* – использование проверочных атак.

2) Сравнение – сверка с нормами и требованиями. Выделяют:

а) *аудит* – проверка отчетности на соответствие требованиям;

б) *оценка рисков* – сравнение средств защиты с опасностями.

3) Анализ – изучение реальных случаев. Выделяют:

а) *статистический анализ* – анализ различных инцидентов;

б) *заманивание* – анализ действий злоумышленника.

В результате проверки дается оценка состояния защиты, необходимая для принятия соответствующих мер защиты.

Рассмотрим подробнее такие способы проверки, как аудит и тестирование на проникновение.

Аудит

Аудит – систематический, независимый и документированный процесс установления степени соответствия установленным критериям.

Цели проведения аудита:

1) оценка соответствия стандартам и нормам (выдача сертификата);
2) оценка уровня качества;
3) выработка рекомендации, локализация узких мест.

Принципы аудита:

1) *этичность* – информация об уязвимостях и проблемах не должна выйти за пределы организации;
2. *беспристрастность* – непредвзятость оценки;
3. *независимость* – отсутствие заинтересованности в результатах;
4. *компетентность* – должна проводиться специалистом, имеющим необходимые знания, умения;
5. *документальность* – результаты проверки фиксируются в отчётах.

Выделяют внешний и внутренний аудит.

Внешний – проводится сторонней организацией.

Достоинства внешнего аудита:

6. независимость,
7. квалификация специалистов,
8. опыт.

Недостатки внешнего аудита:

9. стоимость,
10. возможность утечки,
11. необходимость искать, приглашать специалистов.

Внутренний (самоаудит) – проводится сотрудниками организации.

Достоинства:

12. осведомлённость о внутреннем устройстве организации,
13. возможность регулярного проведения.

Недостатки:

14. субъективность,
15. загруженность проверяющих другими обязанностями.

В сфере информационной безопасности дополнительно выделяют аудит:

1) *активный* – исследование защищенности с точки зрения злоумышленника;
2) *экспертный* – сравнение с «идеальной» системой безопасности, с точки зрения опыта экспертов;
3) *нормативный* – на соответствие требованиям стандартов по информационной безопасности.

Результаты аудита зависят от аудитора и его компетенции. Компетентность аудитора:

1) *Образование*: высшее техническое образование в области, в которой проводится аудит.
2) *Знания*:

оцениваемые процессы и критерии оценки;

2) принципы аудита;
3) методология аудита;
4) инструментарий аудита;
5) методология анализа и управления рисками;
е) законодательная и нормативная базу.

3) *Навыки и умения*:

а) технический писатель – оформление отчетов;

б) консультант – составление заключений и рекомендаций;

в) переговорщик – опрос сотрудников;

г) посредник – между организацией и проверяющими органами;

д) исследователь – проведение испытаний.

4) *Личные качества* (этика аудитора):

а) Институт SANS, основанный в 1989 году и готовящий профессионалов в компьютерной безопасности, имеет свой собственный «кодекс этики SANS»:

1) я буду стремиться к познанию себя и к тому, чтобы быть честным в своих возможностях;

2) я буду вести свой бизнес так, чтобы ИТ-профессия считалась честной и профессиональной;

3) я уважаю частную жизнь и конфиденциальность.

б) Кодекс этики аудитора и правила ассоциации аудита и контроля информационных систем ISACA.

в) Кодекс международного консорциума по сертификации в области безопасности информационных систем $(ISC)^2$[17]:

1) Поступай честно, справедливо, ответственно, в рамках закона. Защищай всеобщее благополучие.

2) Усердно трудись, предоставляй качественные услуги и развивай сферу безопасности.

3) Поощряй увеличение количества исследований: обучай, направляй и отдавай должное сертификации.

4) Избегай небезопасных действий, оберегай и усиливай целостность общественных инфраструктур.

5) Придерживайся соглашений, гласных и негласных. Давай разумные советы.

6) Избегай любого конфликта интересов, уважай веру других людей в себя, берись только за ту работу, выполнить которую тебе под силу.

7) Сохраняй и обновляй навыки, не участвуй в мероприятиях, которые могут навредить репутации других профессионалов.;

г) Кодекс института внутренних аудиторов IIA4.

д) Совет по аудиторской деятельности при Минфине РФ – Кодекс этики аудитора России (протокол № 16 от 28.08.2003).

е) этические принципы сообщества профессионалов в области информационной безопасности RISSPA5 (Россия).

5) *Опыт* (сертификация аудитора):

а) certified information systems auditor (ISACA);

б) certified information security manager (ISACA2);

в) certified internal auditor (IIA);

г) certified public accountant (AICPA);

д) certified information systems security professional (ISCP);

е) certified systems security practitioner (ISC2);

ж) chartered accountant (CICA);

з) GIAC certified security engieener (SANS);

и) certified protection professional (ASIS).

Процесс проведения аудита:

1) Определение входных данных для проведения аудита:

а) цели;

б) сфера, особенности;

в) ограничения;

г) подходы: нисходящий (исходя из требований), восходящий (в зависимости от имеющихся проблем);

д) критерии.

2) Определение ролей и обязанностей:

а) заказчик – определяет цели;

б) руководитель – определяет критерии и ограничения;

в) организатор – планирует;

г) специалист – проводит аудит.

3) Выбор модели проведения аудита. В зависимостей от целей выбирают:

а) модель оценки по показателям ИБ – оценка соответствия;

б) модель зрелости – оценка качества.

4) Проведение оценивания. Типичные шаги:

а) планирование;

2) сбор данных;

3) проверка достоверности.

5. Оформление результатов. Могут быть получены:

а) аналитический отчёт;

б) заключение аудита;

3) свидетельство соответствия степени;

4) сертификаты, аттестаты соответствия.

Тестирование на проникновение

Тестирование на проникновение (тесты на преодоление защиты, penetration testing, pentest, пентест) – метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника.

Цель тестирования на проникновение – оценить возможность осуществления атаки злоумышленника и спрогнозировать экономические потери в результате ее успешного осуществления.

Виды:

6. Проверка закрытых систем – атакующий не имеет первоначальных сведений об устройстве атакуемой цели. Первоначальная задача такого вида проверки – сбор необходимой информации о расположении целевой системы, её инфраструктуры.
7. Проверка открытых систем – доступна полная информация о целевой системе.
8. Проверка полузакрытых систем – имеется лишь частичная информация.

Результат пентеста – отчёт, содержащий в себе все найденные уязвимости системы безопасности, а также рекомендации по их устранению.

Стандарт исполнения тестирования на проникновение состоит из семи основных этапов:

9. Предварительные взаимодействия. Заключение договора на пентест (статья 272 УК РФ).
10. Сбор разведывательных данных.
11. Моделирование угроз.
12) Анализ уязвимости.
13. Эксплуатация.
14. Послеэксплуатационный.
15. Составление отчётов.

Классификация инструментов пентестера:

4. инструменты для сбора информации – поиск уязвимостей;
5) инструменты для реверс-инжиниринга – восстановление;
6) эксплойты – использование найденных уязвимостей;

7) инструменты для взлома – проведение атак;

8) инструменты для стрес-тестинга – проверка надежности.

Наиболее популярным инструментом для пентеста является бесплатный сборник эксплойтов и вспомогательных скриптов – Metasploit. Для обучения специалистов по пентесту используется виртуальная машина Metasploitable – намеренно уязвимая версия Ubuntu Linux, предназначенная для тестирования средств безопасности и демонстрации распространённых уязвимостей. В ней открыты все порты и присутствуют все известные уязвимости,

Контрольные вопросы и задания

Назовите наиболее достоверный и наиболее используемый способы оценки ИБ.

Что нужно для проведения аудита системы управления информационной безопасностью?

Опишите особенности аудита в сфере ИБ.

Какие курсы и сертификаты есть по пентесту?