МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

А.В. Замятин

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Учебное пособие

Томск
Издательский Дом Томского государственного университета
2020

УДК 519.254
ББК 32.81
З26

Замятин А.В.
З26 Интеллектуальный анализ данных : учебное пособие. – Томск : Издательский Дом Томского государственного университета, 2020. – 196 с.

ISBN 978-5-94621-898-6

В учебном пособии рассматриваются вопросы, связанные с популярной сегодня областью машинного обучения и интеллектуального анализа данных. Исследуются основные технологические тренды, наиболее активно использующие алгоритмы интеллектуальной обработки данных, – бизнес, медицина, управление, индустрия. Обсуждаются вопросы терминологии, основные методы анализа и интерпретации данных, методы и инструменты машинного обучения.

Приведены вопросы для самопроверки.

Для студентов университетов и втузов.

УДК 519.254
ББК 32.81

Рецензенты:
доктор технических наук, профессор *Л.Г. Гагарина*
доктор технических наук, профессор *С.П. Сущенко*

ISBN 978-5-94621-898-6

ОГЛАВЛЕНИЕ

Введение 6
1. Актуальность 7
2. Терминология 11
2.1. Data Mining / Data Science 15
2.2. Big Data 18
2.2.1. Основные понятия 20
2.2.2. Свойства Big Data 21
2.3. Data Mining и Big Data 22
2.4. Дедукция и индукция 22
3. Примеры применения 23
3.1. Интеллектуальный анализ данных в бизнесе 23
3.1.1. Розничная торговля 24
3.1.2. Сфера развлечений 25
3.1.3. Маркетинг, страхование, работа с персоналом 26
3.1.4. Примеры применения классификации, кластеризации и прогнозирования 27
3.2. Интеллектуальный анализ данных в решении сложных прикладных задач 29
3.2.1. Медицина 30
3.2.2. Государственное управление 31
3.3. Интеллектуальный анализ данных в ранней диагностике опасных заболеваний 32
3.4. Интеллектуальный анализ данных в индустриальной предиктивной аналитике 33
4. Основные задачи и классификация методов анализа данных 37
4.1. Этапы интеллектуального анализа данных 37
4.2. Общие типы закономерностей при анализе данных 37
4.3. Группы задач анализа данных 38
4.4. Классификация методов 42
4.5. Сравнительные характеристики основных методов 44
5. Принципиальные основы машинного обучения 46
6. Основные методы анализа и интерпретации данных 50
6.1. Предварительная обработка данных 50

6.2. Оптимизация признакового пространства 56
6.2.1. С трансформацией пространства признаков 57
6.2.2. Без трансформации пространства признаков 59
6.3. Классификация 61
6.3.1. Постановка задачи классификации 61
6.3.2. Контролируемая непараметрическая классификация 65
6.3.3. Контролируемая непараметрическая нейросетевая классификация 66
6.3.4. Классификация по методу машины опорных векторов 70
6.3.5. Деревья решений 72
6.4. Неконтролируемая классификация (кластеризация) 84
6.5. Регрессия 89
6.5.1. Понятие регрессии 89
6.5.2. Основные этапы регрессионного анализа 90
6.5.3. Методы восстановления регрессии 91
6.6. Ассоциация 92
6.6.1. Описание алгоритма 95
6.6.2. Пример исполнения алгоритма 96
6.7. Последовательная ассоциация 98
6.7.1. Алгоритмы семейства «Априори» 99
6.7.2. Алгоритм GSP 102
6.8. Многоуровневое машинное обучение 107
6.8.1. Бутстрэппинг 108
6.8.2. Бэггинг 109
6.8.3. Стекинг 111
6.8.4. Бустинг 112
6.9. Обнаружение аномалий 115
7. Визуализация 118
8. Нейросетевые подходы и глубокое обучение 120
8.1. Функции активации 120
8.2. Основные типы искусственных нейронных сетей 123
8.3. Сверточные нейронные сети (Convolutional Neural Networks) 131
8.4. Популярные архитектуры CNN 134
8.5. Среды и фреймворки глубинного обучения 138

9. Обработка естественного языка 140
9.1. Основные задачи обработки текста 140
9.2. Этапы предварительной обработки текста 143
10. Критерии точности 145
10.1. Метрики качества классификации 145
10.2. Гипотеза A/B 150
10.3. Каппа-индекс согласия 150
10.4. ROC-кривая 152
10.5. Метрика качества прогноза временного ряда 153
10.6. Метрики качества кластеризации 155
11. Высокопроизводительная обработка данных 157
11.1. Принципы высокопроизводительных вычислений 157
11.2. Особенности построения вычислительного кластера 161
11.3. Среды и инструменты высокопроизводительных вычислений 171
12. Инструменты Data Mining 176
12.1. Программные инструменты для высокопроизводительной обработки данных 177
12.1.1. Программная среда 177
12.1.2. Базы данных 178
12.1.3. Языки программирования 179
12.2. Примеры программных систем 179
12.2.1. Примеры самостоятельных систем 179
12.2.2. Примеры облачных систем 180
Вопросы и темы для самопроверки 182
Литература 184

ВВЕДЕНИЕ

Стремительная технологическая эволюция последних лет в сфере информационно-коммуникационных технологий позволила сформировать существенный задел в части развитой программно-аппаратной инфраструктуры, поддерживающей накопление и постоянное пополнение архивов данных различной природы и назначения.

Обостряющаяся конкурентная борьба в различных областях человеческой деятельности – бизнесе, медицине, корпоративном управлении и др. – и сложность внешней среды делают крайне востребованными подходы к экспертному использованию имеющихся данных для повышения обоснованности и оперативности принятия управленческих решений.

При этом не всегда сегодня возможно непосредственное эффективное применение хорошо проработанного и известного аппарата теории вероятностей или математической статистики без учета особенностей конкретной предметной области, компьютерных наук, вычислительной сложности известных и распространенных алгоритмов (включая детали хранения, передачи и обработки данных, алгоритмов машинного обучения и т.п.), современного и перспективного состояния информационных систем и технологий.

Именно поэтому относительно недавно стала привлекать особое внимание область, связанная с высокопроизводительной интеллектуальной аналитической обработкой данных, направленная на то, чтобы оперативно извлекать из значительных массивов накопленных и поступающих данных ценные экспертные знания, поддерживая эффективную управленческую деятельность.

Учитывая междисциплинарный характер этой предметной области, ее глубину и ярко выраженную прикладную направленность, до сих пор существует определенный дефицит систематизированных представлений о ней, на устранение которых в некоторой степени направлено данное пособие.

1. АКТУАЛЬНОСТЬ

С 1960-х гг. информационно-коммуникационные технологии (ИКТ) последовательно эволюционировали от простых систем обработки файлов до сложных, мощных систем управления базами данных (БД). Исследования в области БД с 1970-х гг. смещались от ранних иерархических и сетевых баз данных к реляционным системам управления базами данных (СУБД), инструментам моделирования данных, а также к вопросам индексирования и организации данных. Пользователи получили гибкий и удобный интерфейс доступа к данным с помощью языков запросов (типа SQL), пользовательские интерфейсы, управление транзакциями и т.п. При этом создаваемые и поддерживаемые БД преимущественно имели ограниченный регистрирующий характер, поддерживая рутинные операции линейного персонала. Основным требованием к таким системам было обеспечение транзакционности и оперативности выполнения всех изменений.

Технология баз данных начиная с середины 1980-х гг. характеризовалась популяризацией, широким внедрением и концентрацией исследовательских усилий на новых, все более мощных СУБД. Появились новые модели данных, такие как объектно-ориентированные, объектно-реляционные, дедуктивные модели. Возникали различные предметно-ориентированные базы данных и СУБД (пространственные, временные, мультимедийные, научные и пр.). Эффективные методы онлайновой обработки транзакций (*On-Line Transaction Processing*; *OLTP*[1]) внесли большой вклад в эволюцию и широкое внедрение реляционной технологии в качестве одного из главных универсальных инструментов эффективного хранения, извлечения и управления большими объемами структурированных данных реляционных СУБД.

С развитием сети Интернет получили развитие и вопросы построения распределенных баз данных, создания распределенных

[1] Способ организации БД, при котором система работает большим потоком с небольшими по размерам транзакциями при минимальном времени отклика системы.

глобальных информационных систем. Многократно возросла интенсивность формирования и архивирования различных данных, следствием чего стало развитие масштабируемых программно-аппаратных комплексов, дорогостоящих мощных и недорогих пользовательских компьютеров и накопителей данных.

Все это способствовало всплеску развития индустрии ИКТ и сделало огромное количество баз данных доступными для хранения разнородной информации в значительных объемах и управления транзакциями в них. При этом все больше актуализировалась потребность анализа имеющихся данных в разновременном аспекте с возможностью построения произвольных запросов при условии обработки сверхбольших объемов данных, полученных в том числе из различных регистрирующих БД. Использование для реализации таких задач традиционных регистрирующих систем и БД крайне затруднительно. Например, в регистрирующей системе информация актуальна исключительно на момент обращения к БД, а в следующий момент времени по тому же запросу можно ожидать другой результат. Интерфейс подобных систем рассчитан на проведение определенных стандартизованных операций, и возможности получения результатов на нерегламентированный произвольный запрос ограничены. Возможности обработки больших массивов данных также могут быть ограничены вследствие ориентации СУБД на нормализованные данные, характерные для стандартных реляционных регистрирующих БД.

Ответом на возникшую потребность стало появление новой технологии организации баз данных – технологии *хранилищ данных* (*Data Warehouse*[1]), предполагающей некоторую предварительную обработку данных и их интеграцию, а также онлайновую аналитическую обработку (*On-Line Analytical Processing*; *OLAP*[2]).

[1]Предметно-ориентированная информационная база данных, предназначенная главным образом для поддержки принятия решений с помощью отчетов.

[2] Технология анализа данных, предполагающая подготовку агрегированной структурированной многомерной информации на основе больших массивов данных (*OLAP*-куба), используемой в реляционной БД при построении сложных многотабличных запросов.

Несмотря на очевидную пользу такого инструмента анализа данных, он ориентирован на хорошо нормализованные табличные данные и не предполагает использования целого ряда дополнительного аналитического инструментария типа классификации, кластеризации, регрессионного анализа, моделирования, прогнозирования и интерпретации многомерных данных и т.п.

Таким образом, сегодня наблюдается высокий уровень развития масштабируемой аппаратно-программной ИКТ-инфраструктуры, позволяющей увеличивать и без того значительные архивы данных. Имеется достаточно существенный задел в области компьютерных наук и информационных технологий, разработаны теория и прикладные аспекты теории вероятностей и математической статистики. Вместе с тем следует признать, что присутствует заметный *избыток данных*[1] при *дефиците информации*[2] *и знаний*[3]. Быстро растущие объемы накопленных и пополняемых (автоматически, а не людьми – как это было когда-то) архивов данных пока существенно превышают способности человека в их практически полезной обработке. Для обострения этого тезиса иногда говорят, что «*…большие базы данных стали могилами, которые редко посещаются…*» Как следствие, важные решения порой принимаются не на основе аналитических выводов из информативных БД, а на основе интуиции человека, не имеющего подходящих инструментов для извлечения полезных знаний из имеющихся огромных объемов данных.

Поэтому в последние годы стремительное развитие получила область *Data Science / Data Mining*[4] (в отечественной литературе наиболее используемая аналогия – *интеллектуальный анализ*

[1] Под *данными* будем понимать представление некоторых фактов в формализованном виде, пригодном для хранения, обработки и передачи.

[2] Под *информацией* будем понимать сведения в любой форме; в отличие от данных информация имеет некоторый *контекст*.

[3] Под знаниями будем понимать совокупность информации о мире, свойствах объектов, закономерностях процессов и явлений, а также правилах их использования *для принятия решений*.

[4] Вопросам терминологии посвящена глава 2.

данных, ИАД), направленная на поиск и разработку методов извлечения из имеющихся данных *знаний*, позволяющих принимать на их основе конкретные, в высокой степени обоснованные, практически полезные управленческие решения.

Рис. 1. Пример обобщенного иерархического представления методологий обработки данных при принятии управленческих решений

На рис. 1 приведен пример обобщенного иерархического представления методологий обработки данных, начиная от интеграции разнородных источников данных и завершая использованием методов *Data Mining* для принятия управленческих решений.

2. ТЕРМИНОЛОГИЯ

Рассматривая вопросы терминологии, описывающей обсуждаемую предметную область интеллектуального анализа данных, логично изучить существующую и наиболее устоявшуюся в мире *англоязычную терминологию*, и уже ориентируясь на нее обсудить удачные *терминологические аналогии*, используемые в русскоязычных публикациях по данной тематике.

Выше, в главе 1, отмечено, что с развитием ИКТ-индустрии стремительно развиваются возможности генерирования значительных массивов данных, при умелом анализе которых могут быть найдены полезные знания, позволяющие повысить эффективность принятия управленческих решений в бизнесе, медицине или государственном управлении. Область, изучающую эти вопросы, принято называть *Data Mining* (сегодня за рубежом даже чаще встречается понятие *Data Science*), а специалиста этой области – *Data Scientist*.

На рис. 2 изображен график роста востребованности данных специалистов за последние несколько лет. На рис. 3 приведена диаграмма, отражающая число вакансий специалистов *Data Scientist* в последнее время на портале для поиска работы. Интересно отметить, что на сегодняшний день около 85% всех вакантных позиций такого типа открыты в США, а 15% – в странах Западной Европы [108].

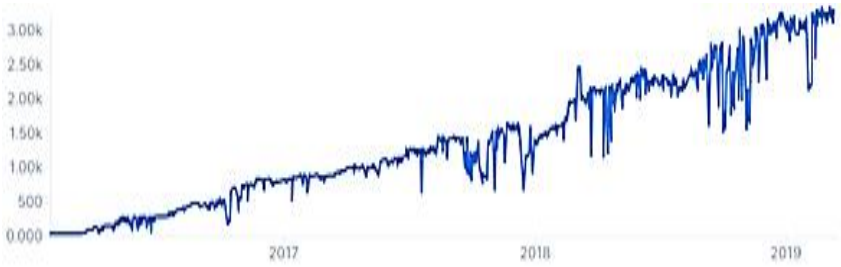


Рис. 2. График уровня востребованности специалистов Data Scientist

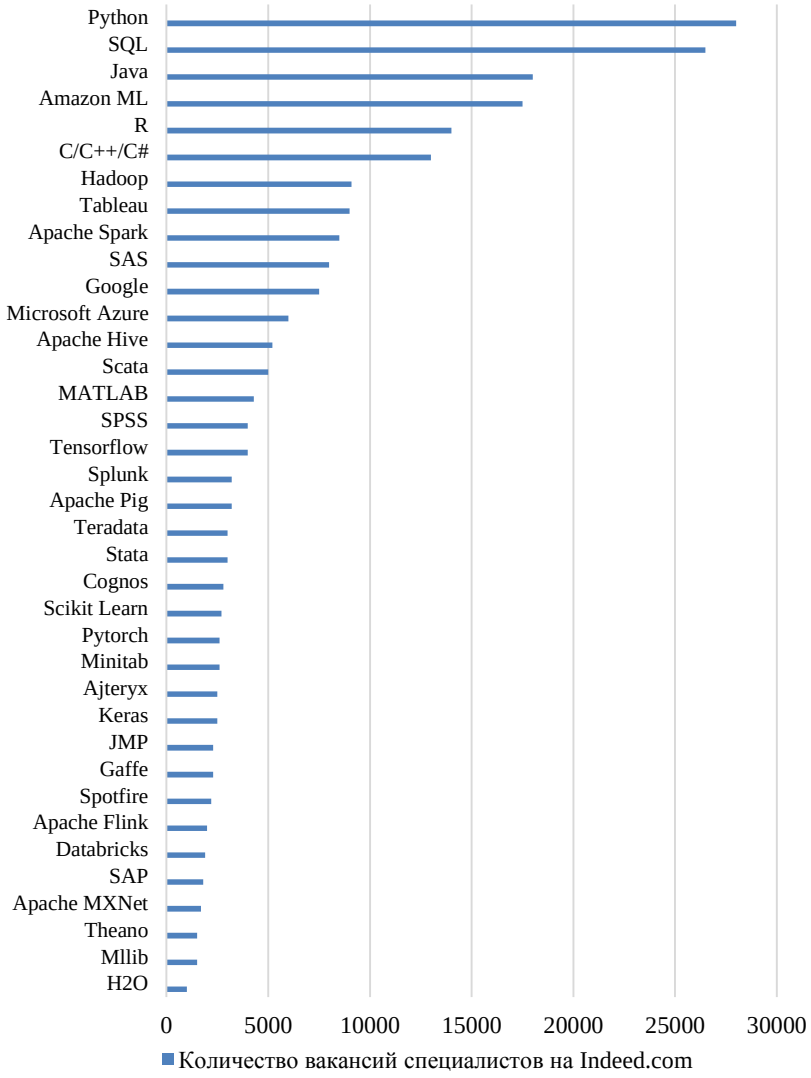


Рис. 3. Иллюстрация востребованности специалистов Data Scientist.
Источник: http//www.indeed.com/jobtrends June 2016

Интересной выглядит визуализация частоты запросов к поисковой системе, наглядно демонстрирующая широту использования основной и вспомогательной терминологии в данной предметной области (рис. 4).

а *б*

Рис. 4. Пример профиля частотности запросов:
а – о предметной области анализа данных; *б* – о специалистах

При этом, если посмотреть на желаемый профиль специалиста *Data Science*, то видно, насколько разносторонней (междисциплинарной) квалификацией, с точки зрения современного работодателя, он должен сегодня обладать (рис. 5):

- SQL – 54%;
- Python – 46%;
- R – 44%;
- SAS – 36%;
- Hadoop – 35%;
- Java – 32%;
- optimization – 23%;
- C++ – 21%4
- visualization – 20%;
- MATLAB – 18%;
- Business Intelligence – 17%;
- distributed – 16%;
- regression – 16%;
- unstructured – 16%;
- Hive – 16%;
- mobile – 15%.

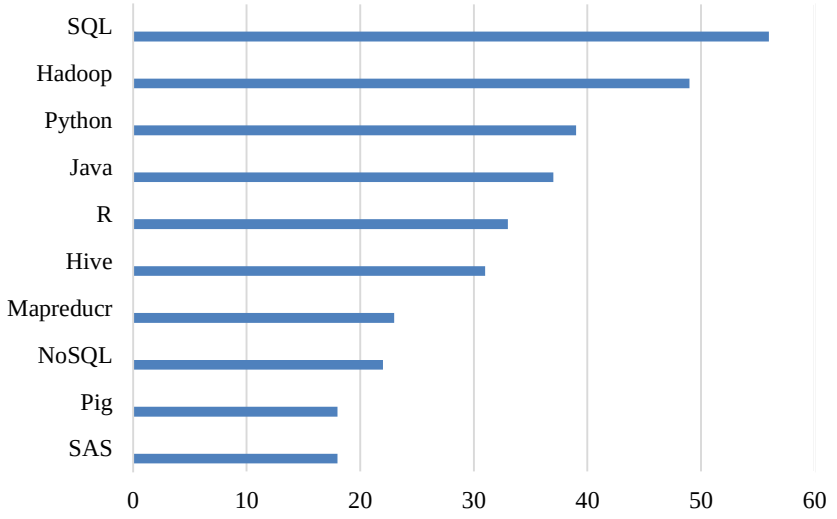


Рис. 5. Ожидаемая квалификация специалиста *Data Science*:
наиболее важные умения специалистов Data Science на Linkedln

Учитывая продолжающееся интенсивное развитие области анализа данных, встречается отличающаяся терминология, описывающая одно и тоже явление или сферу, или один термин, который может быть трактуем по-разному. Например, в англоязычной литературе можно встретить различные термины и их сочетания, описывающие область интеллектуального анализа данных и являющиеся достаточно близкими по значению:

- Data Science;
- Data Mining;
- Big Data;
- Machine Learning;
- Deep Learning;
- Statistical Analysis and Data Mining;
- Predictive Analytics and Data Mining;
- Data Science and Data Mining;
- Discovery Driven Data Mining;
- Knowledge Discovery in Databases и др.

Обзору большинства этих терминов посвящено данное пособие, однако одно из ключевых мест занимает сегодня понятие «глубинного / глубокого обучения» (англ. *Deep Learning*). Такое обучение предполагает применение преимущественно нейросетевых моделей, причем отличающихся именно сложной архитектурой и высокой обучающейся емкостью (например, к таким нейросетевым моделям относят сверточные нейронные сети, рассматриваемые в разд. 8.3).

2.1. Data Mining / Data Science

Существующие массивы данных не только характеризуются значительным объемом и регулярной пополняемостью, но и содержат порой преимущественно тривиальные (неактуальные, ошибочные и т.п.) элементы. Процесс поиска в этих данных чего-то ценного стал сравним с работой на горнорудных предприятиях, где в многотонных завалах руды осуществляется поиск (добыча) драгоценных металлов или камней, полезный выход которых может исчисляться граммами. Учитывая сходную трудоемкость процесса «добычи» (англ. *mining*) знаний из «завалов» данных термин закрепился и для области *Data Mining.*

Для *Data Mining* могут быть даны различные определения, не претендующие на исключительную полноту. *Data Mining* – это*:*

1) процесс обнаружения в базах данных нетривиальных и практически полезных закономерностей [5];

2) процесс выделения, исследования и моделирования больших объемов данных для обнаружения неизвестных до этого структур (паттернов) с целью достижения преимуществ в бизнесе [34];

3) процесс, цель которого – обнаружить новые значимые корреляции, образцы и тенденции в результате просеивания большого объема хранимых данных с использованием методик распознавания образцов и других статистических и математических методов [19];

4) исследование и обнаружение «машиной» (алгоритмами, средствами искусственного интеллекта) в «сырых» данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны, доступны для интерпретации человеком [53];

5) процесс обнаружения полезных знаний о бизнесе [45].

Поэтому для более глубокого понимания сути явления *Data Mining* только какого-либо одного определения не вполне достаточно. Под *Data Mining* (наиболее устоявшаяся аналогия в русском языке, и все-таки не прямой перевод, – *интеллектуальный анализ данных*; по-видимому, это более широкий термин) понимают совокупность методов обнаружения в данных таких знаний, которые обязательно обладают следующими свойствами:

– ранее неизвестные, неожиданные;
– практически полезные;
– доступные для интерпретации;
– необходимые для принятия решений в различных сферах человеческой деятельности.

В более широком смысле под *Data Mining* понимают концепцию анализа данных, предполагающую, что:

– данные могут быть неточными, неполными (содержать пропуски), противоречивыми, разнородными, косвенными и при этом иметь гигантские объемы; поэтому понимание данных в конкретных приложениях требует значительных интеллектуальных усилий;
– сами алгоритмы анализа данных могут обладать «элементами интеллекта», в частности способностью обучаться по прецедентам, т.е. делать общие выводы на основе частных наблюдений; разработка таких алгоритмов также требует значительных интеллектуальных усилий;
– процессы переработки сырых данных в информацию, а информации в знания уже не могут быть выполнены по старинке «вручную» и требуют порой нетривиальной автоматизации.

Data Mining – мультидисциплинарная область, возникшая и развивающаяся на базе достижений прикладной математической статистики, распознавания образов, методов искусственного интеллекта, теории баз данных и др. (рис. 6). Иногда отмечают иной характер мультидисциплинарности *Data Mining* – это объединение *компьютерных наук* (англ. *Computer Science*), *математики* (англ. *Mathematics*) и представлений о *предметной области* (англ. *Domain Expertise*) (рис. 7). В этом случае компьютерные науки описывают среду

создания информационных продуктов (англ. *data products*), математика выстраивает теоретическую основу для решения поставленных проблем, а представление о предметной области позволяет понять реальность, в которой существует проблемная ситуация.

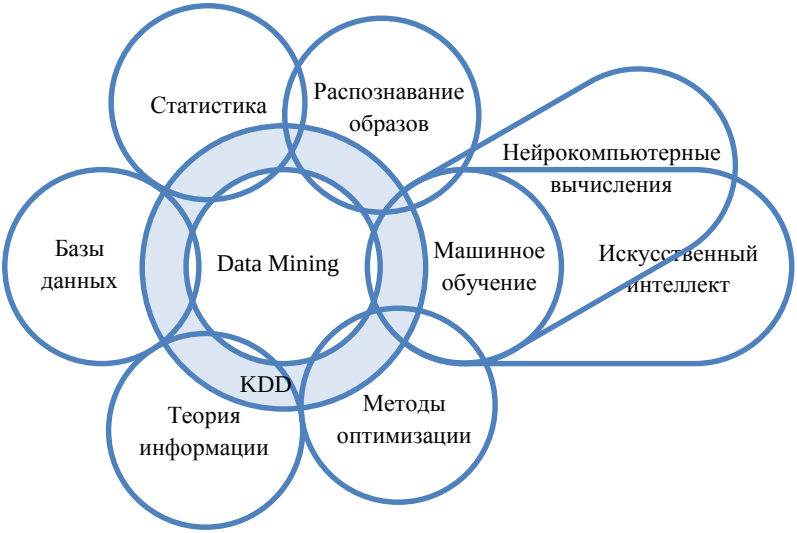


Рис. 6. Иллюстрация Data Mining как междисциплинарной области

Рис. 7. Иллюстрация Data Mining как междисциплинарной области

Именно включение *предметной области* как междисциплинарной компоненты *Data Mining* существенно осложняет практическую интеллектуальную работу в этой сфере, требуя от специалиста (*Data Scientist*) при решении каждой конкретной задачи анализа данных достаточно глубокого погружения в новую, незнакомую ему область человеческой деятельности. Однако очевидно, что без такого погружения сложно найти эффективные решения существующих проблем.

Следует отметить еще одну важную отличительную особенность *Data Mining*. Вычислительная сложность многих традиционных методов математической статистики или технологий БД определяется как $O(n^2)$, или (n^3), где n – объем исходных данных. Однако при сверхбольших объемах данных (превышающих порой несколько миллионов записей), характерных для современного этапа развития отрасли ИКТ, непосредственное применение таких традиционных методов обработки данных крайне затруднительно даже на самой современной мощной вычислительной технике. Учитывая эту особенность, *Data Mining* предполагает создание и развитие *специализированных алгоритмов*, характеризующихся значительно более высокой вычислительной эффективностью ($O(n)$, $O(\log n)$ и т.п.), достигаемой, например, за счет поиска приближенного результата (эвристики) без существенной потери точности.

2.2. Big Data

Анализ терминологии, достаточно полно описывающей область *Data Mining*, позволяет отметить, что в ней часто в схожем контексте применяется и такой термин, как *Big Data* (рус. *большие данные*). При этом широта его употребления специалистами и неспециалистами порой затрудняет однозначное толкование этого термина, в особенности учитывая обсуждение деталей понятия *Data Mining* выше. Как же следует понимать *Big Data?*

Ежедневно в мире создается более 5 эксабайтов[1] информации. В 2012 г. в мире было сгенерировано около 2,43 Зеттабайт (1 ЗБ –

[1] Единица измерения, равная 10^{18}, или 2^{60}, байт.

около 1 млрд Гб), что более чем в 2 раза превосходит объем информации в цифровом виде в 2010 г. (1,2 ЗБ). К 2020 г. информационные системы имеют дело с количеством данных, равным 40 ЗБ (примерно в 57 раз большим, чем количество песчинок на пляжах всей поверхности Земли). Очевидно, появление и активное тиражирование термина *Big Data* во многом вызвано сопровождением этого объективного процесса накопления сверхбольших объемов данных.

Действительно, обеспечивать соответствие возможностей ИКТ-инфраструктуры, предназначенной лишь для надежного хранения стремительно растущих объемов накапливаемых данных, не говоря уже о возможностях их интеллектуальной обработки, – непростая и дорогостоящая задача. Это означает перспективы формирования и развития рынка *Big Data* со значительной финансовой емкостью. Аналитики отмечают, что мировой рынок *Big Data* (технологий и сервисов для обработки данных) в ближайшие годы будет расти в среднем на 13,2% в год и к 2022 г. вырастет до 274,3 млрд долл.

Целесообразно отметить, что основным драйвером продвижения термина *Big Data* во многом являются *рыночные законы маркетинга*[1]. Они позволяют привлечь к проблеме внимание не только ученых, но и государства и бизнеса, и предусмотреть в бюджетах компаний средства на развитие этих технологий (в первую очередь – именно аппаратной ИКТ-инфраструктуры, что объясняет не столь высокую популярность термина *Data Mining*, связанного больше со специализированным алгоритмическим и программным обеспечением интеллектуальной обработки данных). Именно поэтому порой дискуссия о границах того, что является «действительно» «большими данными», а что уже не является, сводится к тому, насколько дорогостоящая инфраструктура требуется для их поддержки[2].

[1] Именно они, главным образом, «реанимировали» традиционные технологии web-хостинга в виде популярных сегодня *облачных сервисов* и способствуют развитию суперкомпьютерной тематики, регулярно продвигая различные рейтинги мощности суперкомпьютеров в мире.

[2] Например, по этой логике компания *Google* или исследовательский коллайдер *CERN*, конечно, работают в концепции *Big Data*, а вот все менее масштабные предприятия – сомнительно…

2.2.1. Основные понятия

Рассмотрим некоторые наиболее типичные определения и толкования термина *Big Data* [56]:

– данные очень большого объема;

– область управления и анализа больших объемов данных;

– область управления и анализа больших объемов данных, представленных (в отличие от реляционных БД) в слабоструктурированных форматах (веб-журналы, видеозаписи, текстовые документы, машинный код или, например, геопространственные данные и т.п.);

– область работы с информацией огромного объема и разнообразного состава, весьма часто обновляемой и находящейся в разных источниках, в целях увеличения эффективности деятельности, создания новых продуктов и повышения конкурентоспособности;

– область, объединяющая техники и технологии, которые извлекают смысл из данных на экстремальном пределе практичности;

– постоянно растущий объем информации, поступающей в оперативном режиме из социальных медиа, от сетей датчиков и других источников, а также растущий диапазон инструментов, используемых для обработки данных и выявления на их основе важных бизнес-тенденций;

– наборы данных, превосходящие возможности традиционных программно-аппаратных инструментов оперирования данными (например, в случае, когда параметры набора данных превосходят возможности обработки стандартными средствами *MS Excel*).

Очевидно, и данный набор понятий не позволит совершенно четко определить, где построить границу между «действительно» «большими данными» и «просто данными». Сравнительно большой объем данных не всегда можно по умолчанию отнести к сфере *Big Data,* так как возможности их анализа зависят не только от объема данных, но и от вычислительной сложности задачи (очевидно,

трудно сравнивать по сложности задачу расчета стандартных статистик и задачу построения комплексной оптимизационной модели). Поэтому более интересным выглядит предложение под '*Big*' в *Big Data* понимать не какой-то конкретный физический объем данных или другие количественные показатели, а рассматривать это как «важные», «ключевые» данные [6].

2.2.2. Свойства Big Data

Дополнительное понимание термину *Big Data* придают 4 свойства, кратко сформулированные по четырем английским словам[1], начинающимся на букву '*V*' латинского алфавита:

– *Volume* – отражает значительный физический объем данных;

– *Variety* – показывает существенное разнообразие типов данных (например, структурированные, частично структурированные, неструктурированные, как текст, web-контент, мультимедиа, данные), источников данных (внутренние, внешние, общественные) и их детальности;

– *Velocity* – демонстрирует скорость, с которой данные создаются и обрабатываются;

– *Veracity* – определяет варьируемый уровень помех и ошибок в данных.

Как отмечено выше, свойство *Volume* часто наименее важное, и нет какого-либо обязательного требования к минимальному объему обрабатываемых данных в концепции *Big Data*. Существенно более высокой важностью обладают свойства *Variety* и *Velocity*. Так, свойство *Variety* может привносить особенно высокую ценность в данные, скомбинированные из различных источников (например, корпоративные данные, данные социальных сетей и публичная информация), даже на небольших объемах. Наиболее важным является свойство *Veracity*, определяющее качество и корректность данных.

[1] *Four Vs* (четыре буквы «*V*», по первым буквам использованных слов): объем, многообразие, скорость, достоверность.

2.3. Data Mining и Big Data

В заключение терминологических пояснений, характеристик и свойств, описывающих области *Data Mining* и *Big Data*, сформулируем более четко отличия одного термина от другого. Итак, наиболее корректным, во избежание путаницы между терминами, представляется целесообразным под *Big Data* здесь и далее понимать некоторый *актив*[1], который при умелом применении *Data Mining* (технологий, методов, способов) позволяет получить (извлечь) практически полезный результат (экономический эффект).

2.4. Дедукция и индукция

В интеллектуальном анализе данных обсуждают два основных способа извлечения практически полезных знаний – *дедуктивный* (на основе некоторой априори сформулированной гипотезы, от общего – к частному) и *индуктивный* (на основе известных *паттернов*[2], от частного – к общему).

Дедуктивный подход к исследованию данных предполагает наличие некой сформулированной гипотезы, подтверждение или опровержение которой после анализа данных позволяет получить некоторые частные сведения.

Индуктивный подход к исследованию данных позволяет сформулировать (скорректировать существующую) гипотезу и найти с ее помощью новые пути аналитических решений.

Для поиска значимых закономерностей порой требуется совместное попеременное использование индуктивного и дедуктивного подходов, при этом формируется такая среда, в которой модели не нужно быть исключительно статической или эмпирической. Вместо этого модель непрерывно тестируется, модифицируется и улучшается до тех пор, пока не будет достаточно усовершенствована.

---

[1] В международных стандартах бухгалтерской отчетности – ресурс компании, от которого компания в будущем ожидает экономической выгоды.

[2] Например, некоторое нетривиальное утверждение о структуре данных, имеющихся закономерностях, зависимостях между атрибутами и т.п.

3. ПРИМЕРЫ ПРИМЕНЕНИЯ

3.1. Интеллектуальный анализ данных в бизнесе

Наибольший интерес к технологиям интеллектуальной обработки данных в первую очередь проявляют компании, работающие в условиях *высокой конкуренции* и имеющие *четкую группу потребителей* (розничная торговля, финансы, связь, маркетинг). Они используют любую возможность для повышения эффективности собственного бизнеса через принятие более эффективных управленческих решений. Такие компании пытаются найти связь между «*внутренними*» (цена, востребованность продукта, компетентность персонала и т.п.) и «*внешними*» (экономические показатели, конкуренция, демография клиентов и т.п.) факторами. Это позволяет им оценивать (прогнозировать) уровень продаж и удовлетворенности клиентов, размер доходов, а также формулировать на основе совокупности всей имеющейся информации практически полезные выводы и рекомендации. Иногда отдача от применения этих инструментов может составлять сотни процентов при сравнительно невысокой стоимости внедрения.

При этом результатом обработки данных должен быть такой *информационный продукт*, который позволяет предпринять конкретное управленческое действие без избыточного «погружения» лица, принимающего решение (ЛПР), в детали базовых данных или промежуточной аналитики (например, дать рекомендации по покупке / продаже на финансовом рынке, сформировать перечень мероприятий по увеличению производительности или маркетингу продукта и т.п.). Причем на практике возможна ситуация, при которой какое-либо решение в той или иной части необходимо принимать *обязательно* – вопрос только в том, принимается оно на основе объективной информации или интуитивно.

Извлечение своевременной и готовой непосредственно для принятия управленческих решений информации из различных

источников предполагает создание некоторых информационных продуктов. Примерами таких информационных продуктов в бизнесе могут быть ответы на вопросы типа:

– Какой из продуктов следует рекламировать больше для увеличения прибыли?

– Как следует усовершенствовать программу модернизации для уменьшения расходов?

– Какой процесс производства изменить, чтобы сделать продукт лучше?

Ключ к ответу на эти вопросы требует глубокого понимания имеющихся данных и их *индуктивного*[1] анализа.

Рассмотрим некоторые примеры применения методов интеллектуального анализа данных, используемых в бизнес-среде, которые подтверждают на практике возрастающую актуальность этой интеллектуальной сферы человеческой деятельности.

3.1.1. Розничная торговля

Используя методы интеллектуального анализа данных, пункт розничной торговли (магазин) может фиксировать информацию обо всех покупках клиента и таргетированно[2] рассылать рекламные предложения своим клиентам на основе истории их покупок. Анализируя демографическую информацию о клиентах, магазин может рекомендовать определенные товары и рекламные предложения для конкретного клиентского сегмента.

Всемирно известная торговая сеть США *WalMart* – пионер интеллектуального анализ данных, примененного для модернизации взаимодействия с поставщиками. Компания *WalMart* проанализировала транзакции 2 900 магазинов из шести стран, сформировав хранилище данных объемом 7,5 ТБ. При этом потребовалось выполнить более 1 млн сложных запросов к данным. Данные

[1] То есть от частного к общему.

[2] Таргетинг (англ. *target* – цель) – рекламный механизм, позволяющий выделить целевую аудиторию для демонстрации ей рекламы.

использованы для определения паттернов покупателей при совершенствовании *мерчендайзинговых*[1] стратегий для 3 500 поставщиков.

Типовыми вопросами, на которые ищутся ответы при анализе данных в розничной торговле, являются:

- Кто ваш покупатель?
- Как сегментировать клиентов?
- На какую целевую аудиторию сделать акцент?
- Какие факторы влияют на решение о покупке?
- Какова значимость каждого из факторов?
- Какие товары предлагать в совместных акциях?
- Какие существуют зависимости в поведении клиентов?
- На какой объем спроса в будущем ориентироваться?

3.1.2. Сфера развлечений

Интересными могут быть примеры анализа данных в сфере развлечений. Например, компания по продаже контента для видеопросмотра может анализировать историю пользовательских запросов и предлагать в соответствии с ними индивидуальные рекомендации.

В Национальной баскетбольной ассоциации США традиционно используются инструменты анализа данных для оценки перемещений игроков на площадке, помогая тренерам команд в тактической борьбе и выработке стратегий на игру. Еще в далеком 1995 г. такой анализ игры между *New York Knicks* и *Cleveland Cavaliers* позволил выявить, что защитник *Mark Price* позволил забить нападающему *John Williams* из команды соперника лишь один бросок из четырех, в то время как общая статистика за игру по этому показателю для *Cavaliers* была зафиксирована на уровне 49,30%. Учитывая, что видео всех игр сохраняется фрагментарно, тренер с легкостью может отыскать в большом видеомассиве данных любые интересующие моменты и проанализировать причину успеха или неудачи в конкретном игровом эпизоде без необходимости часами просматривать все видео в поисках нужного фрагмента.

[1] Мерчендайзинг (англ. *merchandising*) – искусство сбыта.

Хороший кейс применения методов интеллектуального анализа данных на заре развития этих подходов к спортивной индустрии показан в популярном фильме «Человек, который изменил все» («*Moneyball*») на примере игры в бейсбол.

3.1.3. Маркетинг, страхование, работа с персоналом

Наиболее распространенными сегодня примерами использования инструментов *Data Mining* являются различные интернет-магазины и поисковые системы. Они на основе ретроспективных запросов пользователя определяют профиль его интересов, руководствуясь которым возможно предложить товары и услуги, которые с высокой степенью вероятности также могут заинтересовать пользователя. Многие сталкиваются сегодня с такими примерами в *Google, Amazon, eBay* и т.п.

С расширением страхового рынка все большую актуальность инструменты анализа данных приобретают и для него. Эта сфера человеческой деятельности всегда требовала использования математических моделей оценки рисков. С развитием инструментов *Data Mining* появляются новые перспективные возможности оценки рисков на основе большей совокупности факторов, моделирования страховых убытков, исследования связи среднего уровня доходов территории и числа застрахованных, кластерного анализа в автостраховании (например, для избегания случаев мошенничества), оценки распределения размеров убытков, прогнозирования доходов от продажи страховых полисов и др.

Очевидно, инструменты интеллектуального анализа данных могли бы помочь специалистам кадровой службы любой компании. При этом примерами вопросов, на которые ищут ответы такие специалисты, могут быть:

- Каким требованиям должен удовлетворять соискатель?
- Кто и как часто совершает ошибки?
- Сколько компания теряет из-за ошибок сотрудников?
- Кто является мошенником?
- Как оптимизировать штат и нагрузку?

– Как оптимизировать режимы работы оборудования?
– Как сократить число сбоев и простоев по технологическим причинам?

Примерами тем, которые могут искать специалисты в области маркетинга методами *Data Mining*, могут быть:
– целевая аудитория;
– наиболее выгодные типы клиентов;
– маркетинговые каналы;
– взаимодействие с дилерами;
– политика скидок, специальные предложения.

3.1.4. Примеры применения классификации, кластеризации и прогнозирования

Характерной особенностью *Data Mining* является активное использование методов *классификации*, *кластеризации* и *прогнозирования*, используемых для выявления неявных закономерностей и свойств, присутствующих в данных. Рассмотрим без математической детализации некоторые прикладные примеры применения этих методов при решении практических задач.

Классификация. В общем виде задача *классификации* заключается в том, чтобы определить, к какому классу (типу, категории) относятся те или иные данные в соответствии с некоторым известным набором атрибутов и массивом соответствующих этим атрибутам данных. При этом множество классов, к одному из которых впоследствии можно отнести исследуемый объект, известно. Каждый класс обладает определенными свойствами, которые характеризуют его объекты.

Например, задачу классификации следует решать в банковском секторе при определении степени *достаточной кредитоспособности* клиента. В этом случае банковский служащий оперирует двумя известными ему классами – *кредитоспособный* и *некредитоспособный*. Характеристиками (признаками) исследуемого (классифицируемого) объекта являются возраст, место работы, уровень доходов, семейное положение. Фактически задача сводится к тому, чтобы

определить значение одного из параметров объекта анализа (класс «кредитоспособный» или класс «некредитоспособный») по значениям всех прочих его параметров (признаков). Говорят, что необходимо определить значение *зависимой переменной* «кредитоспособность» (которая может принимать значения «да» или «нет») при известных значениях *независимых переменных* «возраст», «место работы», «уровень дохода», «семейное положение».

Кластеризация. *Кластеризация* – задача, аналогичная предыдущей, но реализуется в случае, когда набор классов неизвестен заранее. Например, задачу кластеризации решает маркетолог, разделяя всех клиентов некоторого бизнеса на неопределенное количество групп по характеристикам условного сходства – социальному и географическому положению, основным мотивам покупки, базовым товарам персональной потребительской корзины, размеру чека и т.п. Более четкое определение целевых групп позволяет дифференцировать для них предложения, повысив результативность промоакций и снизив неэффективные расходы на их проведение. Например, более детальное социально-демографическое представление об имеющихся сегментах потребителей в розничной торговле позволяет выделить более доходные сегменты и активизировать для них свою рекламную деятельность, а также выделить менее доходные сегменты и существенно скорректировать для них используемые маркетинговые инструменты. В совокупности такой подход позволит более фокусно, целевым образом, расходовать имеющиеся ресурсы, увеличивая объемы продаж.

Прогнозирование. Метод *прогнозирования* – один из наиболее востребованных в бизнесе. Анализируя данные прошлых периодов, можно построить с некоторой точностью прогноз на будущее (которое каждый бизнесмен хотел бы знать). Причем чем более подробные и точные ретроспективные данные имеются и чем больше анализируемый отрезок времени, тем, как правило, точнее получатся результаты прогнозирования.

Данный метод нередко применяется для оценки спроса на услуги и товары, структуры сбыта, характеризующиеся сезонными колебаниями, или позволяет оценить ожидаемую потребность в кадрах.

Примером применения такого инструментария, конечно, являются фондовые рынки, на которых трейдеры пытаются угадать направления движения тех или иных графиков и определить верные моменты покупки / продажи.

Высокую актуальность и широкое распространение инструменты прогнозирования приобретают в розничной торговле продовольственными товарами. В условиях скоротечности бизнес-процессов заказа, поставки, хранения и продажи скоропортящихся продуктовых товаров крайне важно выдержать баланс между *удовлетворенностью клиента* (имеющего достаточный выбор на полках магазина и возможность найти желаемое) и *остатками товара* (приобретенного, занимающего место на складе и в торговом зале, но в итоге – не проданного), которые будут утилизированы по истечении срока годности. Именно в этом обширном сегменте рынка сегодня идет серьезная борьба между производителями соответствующих программных инструментов интеллектуального анализа данных, которые при умелом использовании быстро дают существенный экономический эффект.

Однако насколько эта задача представляется актуальной, настолько непросто идет поиск ее наиболее перспективных решений, удачное воплощение которых зависит от множества факторов и достаточно глубокого знания специалиста по интеллектуальному анализу данных деталей предметной области.

3.2. Интеллектуальный анализ данных в решении сложных прикладных задач

Область применения инструментов ИАД не ограничивается исключительно бизнес-сферами, основным показателем эффективности которых является прибыль. Очевидно, такой инструментарий может найти и находит применение в других областях человеческой деятельности, функционирование которых сопровождается генерированием и анализом различных данных. Важнейшими сферами, в которых активно адаптируются методы ИАД, являются медицина и государственное управление.

3.2.1. Медицина

Описанные методы ИАД применялись для создания алгоритмов диагностики и прогнозирования в онкологии, неврологии, педиатрии, психиатрии, гинекологии и других областях [27, 44, 94]. На основе полученных результатов построены экспертные системы для постановки диагнозов с использованием правил, описывающих сочетания симптомов разных заболеваний. Правила помогают выбирать показания (противопоказания), предсказывать исходы назначенного курса лечения. В молекулярной генетике и генной инженерии – это определение маркеров, под которыми понимаются генетические коды, контролирующие те или иные фенотипические признаки живого организма. Известно несколько крупных фирм, специализирующихся на применении ИАД для расшифровки генома человека и растений. В прикладной химии такие методы используют для выяснения особенностей строения химических соединений.

В медицине консолидируют информацию из различных источников – данные из медицинских карт, результаты анализов и проб, выходные показатели диагностирующих тест-систем. Создают и применяют *советующие системы для диагностики заболеваний*, которые выявляют значимые признаки и моделируют сложные зависимости между симптомами и заболеваниями с использованием разнообразных регрессионных моделей, деревьев решений, нейронных сетей и т.д. Также выполняют *оценку диагностических тестов* в сравнении с традиционными методиками, осуществляя подбор оптимальных порогов диагностических показателей и определение чувствительности и пределов применимости конкретной модели. Кроме того, инструменты ИАД (ассоциативные правила и последовательные шаблоны) могут применяться при выявлении связей между *приемом препаратов и побочными эффектами*.

Вместе с тем создание алгоритмического и программного обеспечения систем поддержки медицинской деятельности до сих пор не является тривиальной задачей. Потенциал ИАД может быть раскрыт в эффективном диалоге квалифицированного медицинского

специалиста и разработчика интеллектуальных систем. Однако в такой сложной сфере, как медицина, квалифицированный специалист не всегда способен доступно для разработчика объяснить свои рассуждения[1]. Поэтому потенциал применения технологий и методов ИАД в медицине все еще остается в значительной степени нереализованным.

3.2.2. Государственное управление

Большие объемы информации, концентрирующиеся в органах государственного и муниципального управления, обычно хранятся в разрозненном и не всегда готовом для непосредственной автоматизированной обработки виде, содержат значительное количество неточностей и пробелов, что является препятствием для ее эффективного использования в процессе принятия решений. Современные инструменты ИАД используют для *поиска существующих в данных противоречий, дублирования, опечаток и их корректировки*. Также полезной является *реализация различных бюджетных моделей* с возможностями сравнения различных вариантов, условного моделирования, прогнозирования развития ситуации, расчета показателей эффективности. Кроме того, возможно автоматическое *обнаружение отклонений*, их *ранжирование* и *оповещение заинтересованных лиц*. Все это успешно применяется при решении задач поиска лиц, уклоняющихся от налогов, или при поиске вероятных источников террористических угроз.

Актуальность использования инструментов ИАД для решения задач государственного и муниципального управления подтверждает набирающая обороты программа повышения квалификации «Управление, основанное на данных», организованная при содействии Агентства стратегических инициатив и направленная развитие компетенций, известных у таких специалистов как *Chief Data Officer*. В 2019 г. несколько сотен человек на базе различных организаций (Томский государственный университет, Высшая школа

[1] Для других областей человеческой деятельности этот тезис часто также справедлив.

экономики и др.) прошли повышение квалификации именно по этому профилю.

3.3. Интеллектуальный анализ данных в ранней диагностике опасных заболеваний

В условиях нарастающей потребности в персонализации медицины и необходимости ее комплексного цифрового развития сегодня в мире ощущается острый дефицит решений, позволяющих обеспечить не только разностороннюю диагностику состояния человека, но и автоматизированное сохранение персональных данных медицинской диагностики с формированием непрерывного цифрового следа, позволяя осуществлять не только интеллектуальную обработку отдельных диагностических результатов, но и анализ их динамики с формированием соответствующих медицинских заключений.

В настоящее время активно проводятся исследования в области анализа отдельных типов компьютерных изображений для задач диагностики различных заболеваний, а последние исследования все активнее используют для этого методы машинного обучения. Наибольшее распространение получила обработка данных невысокой размерности – 2D-компьютерной томографии (КТ) с получением пошаговых рентгенологических цельных изображений, маммографии или иных 2D-изображений.

Более сложным и трудоемким является обработка 3D-изображений, характеризующихся не только более высокой размерностью данных, но и более значительным информационным объемом. Примерами таких данных сегодня являются данные магнитно-резонансной томографии (МРТ), которая позволяет получить трехмерное изображение сканируемых зон посредством измерения соответствующего электромагнитного поля исследуемых тканей.

Несмотря на перспективность изложенных результатов применения методов машинного обучения к медицинским 2D- и 3D-изображениям, практически отсутствуют подходы к выбору наиболее подходящих к конкретным типам данных классификационных

архитектур глубокого обучения, включая способы предварительной обработки данных, собственно архитектуры сверточной нейросети, а также способов визуализации и интерпретации результатов.

Отдельного изучения заслуживают результаты исследований другого типа данных, который также широко применяется в задачах медицинской диагностики, – временных рядов: электрокардиографии, эхоэнцефалографии и т.п.

3.4. Интеллектуальный анализ данных в индустриальной предиктивной аналитике

В настоящее время в нашей стране и в мире большая часть промышленных процессов находится под автоматизированным или автоматическим контролем. При этом происходит генерация огромного объема технологических и нетехнологических данных, которые собираются и архивируются на базе соответствующей ИТ-инфраструктуры. Накопленные данные о протекании технологических процессов с учетом действий диспетчерского персонала, параметров среды, состояний агрегатов и характеристик исходного сырья могут содержать полезную информацию не только о текущем состоянии того или иного агрегата, но и о начавшихся критических изменениях в технических характеристиках агрегата и его потребительских свойствах.

Большинство предприятий для отслеживания негативных изменений в работе оборудования используют базовые средства контроля, предоставляемые производителями. В их основе лежат принципы агрегации и визуализации данных из различных источников в доступном формате для проведения анализа. Основные функции таких систем генерации отчетов ограничиваются, как правило, лишь построением диаграмм и графиков, а собственно анализ представленной информации остается за человеком. Именно такого рода решения сегодня предлагаются ведущими производителями систем диспетчерского контроля и управления (Wonderware, Emerson и т.д.).

Однако подобный экспертный анализ имеет существенные ограничения:

– не позволяет анализировать многочисленные косвенные факторы, влияющие на состояние оборудования, и прогнозировать критическую ситуацию априори;

– анализ ситуации проводится с существенной задержкой относительно режима реального времени;

– эксперт дополнительно использует лишь несложные методы традиционной математической статистики, область применения которых существенно ограничена известными законами распределения данных и их небольшим набором;

– требуется высокая квалификация эксперта, позволяющая анализировать неочевидные тренды, прогнозировать развитие ситуации исключительно с учетом своего практического опыта.

Поэтому все более актуальными становятся методы анализа данных, максимально исключающие человеческий фактор, но базирующиеся на использовании экспертного опыта и в значительной мере заменяющие человека-эксперта. Несмотря на успешные попытки применения этих методов в различных областях анализа аналогичных биомедицинских или речевых сигналов, в области индустриальной аналитики такие решения практически отсутствуют. Существующие сегодня отдельные примеры (решения Tibco Statistica или Fujitsu) имеют существенные технические ограничения:

– слабая интеграция непосредственно с источниками технологических данных, которая препятствует применению результатов анализа в режиме реального времени;

– большинство систем управления на промышленных предприятиях являются локальными, закрытыми системами, изолированными от сети Интернет (данный фактор требует дополнительных мер безопасности и учета специфики промышленных протоколов передачи данных);

– высокие вычислительные затраты, которые невозможно реализовать в условиях существующей инфраструктуры.

Отсутствие решений в области анализа технологических данных главным образом связано со следующими обстоятельствами:

– нестационарность и динамичность технологических процессов, обусловливающие сложность математического описания;

– неоднородная структура данных (вещественные значения, события, сообщения);

– уникальность каждого объекта управления (в отличие от биомедицинских и речевых данных, которые имеют обобщенные характерные признаки), которая требует адаптивного похода для каждого случая;

– наличие аномальных ситуаций, которые могут являться как следствием изменения интересующего тренда технологического сигнала, так и следствием ремонтных или иных регламентных процедур.

Разрешение указанных сложностей позволит извлекать полезную информацию о технических характеристиках агрегатов на производстве, локализовать с высокой точностью область и причину неисправности оборудования, прогнозировать состояние объекта с учетом его ретроспективных характеристик и характеристик других смежных агрегатов. В итоге это позволит оптимизировать управление технологическим процессом и повысить технико-экономические показатели производства.

Наиболее существенная экономическая выгода от применения интеллектуального анализа данных ожидается в области сопровождения технического обслуживания и ремонта (ТОиР) сложного технологического оборудования, которая характеризуется существенными затратами на любых производственных предприятиях. Применение методов интеллектуального анализа технологических данных позволит решить следующие проблемы:

– существенно сократить время простоя оборудования;

– оптимизировать план мероприятий по техническому обслуживанию, а также уменьшить время внепланового техобслуживания;

– проводить углубленный анализ причин отказов оборудования;

– получать более полную информацию о технологическом процессе;

– повысить срок службы сложного технологического оборудования.

При этом особое место в обеспечении надежности производства занимает задача обнаружения аномалий в данных, так как именно они зачастую являются предвестниками аварий, сбоев оборудования и нештатных ситуаций. Применительно к системам мониторинга техногенных объектов под аномалиями могут пониматься любые события, нарушающие регламентное протекание контролируемого технологического процесса. Своевременное обнаружение аномалий позволяет оперативно вносить коррективы в управление технологическим процессом, тем самым избегая возникновения негативных ситуаций на производстве.

4. ОСНОВНЫЕ ЗАДАЧИ И КЛАССИФИКАЦИЯ МЕТОДОВ АНАЛИЗА ДАННЫХ

4.1. Этапы интеллектуального анализа данных

Выделяют следующие типовые этапы, сопровождающие решение задач интеллектуального анализа данных:

1. Анализ предметной области, формулировка целей и задач исследования.
2. Извлечение и сохранение данных.
3. Предварительная обработка данных:

– очистка (англ. *cleaning*): исключение противоречий, случайных выбросов и помех[1], пропусков;

– интеграция (англ. *integration*): объединение данных из нескольких возможных источников в одном хранилище;

– преобразование (англ. *transformation*): может включать агрегирование и сжатие данных, дискретизацию атрибутов и сокращение размерности и т.п.;

4. Содержательный анализ данных методами Data Mining (установление общих закономерностей или решение более конкретных, частных задач).
5. Интерпретация полученных результатов с помощью их представления в удобном формате (визуализация и отбор полезных паттернов, формирование информативных графиков и / или таблиц).
6. Использование новых знаний для принятия решений.

4.2. Общие типы закономерностей при анализе данных

Как правило, выделяют пять стандартных типов закономерностей, которые позволяют относить используемые методы к методам *Data Mining*:

1. Ассоциация.
2. Последовательность.

[1] Если они сами не являются предметом анализа в данном случае.

3. Классы.
4. Кластеры.
5. Временные ряды.

Ассоциация (англ. *Association*) имеет место в случае, если несколько событий связаны друг с другом. Например, исследование показывает, что 75% покупателей, приобретавших кукурузные чипсы, приобретают и «колу». Эта ассоциация позволяет предложить скидку за такой тип продуктового «комплекта» и, возможно, увеличить тем самым объемы продаж.

В случае, если несколько событий связаны друг с другом во времени, имеет место тип зависимости, именуемый *последовательность* (англ. *Sequential Patterns*). Например, после покупки дома в 45% случаев в течение месяца приобретается и новая кухонная плита, а в пределах двух недель 60% новоселов обзаводятся холодильником.

Закономерность *классы* (англ. *Classes*) появляется в случае, если имеется несколько заранее сформированных классов (групп, типов) объектов. Отнесение нового объекта к какому-либо из существующих классов выполняется путем *классификации*.

Закономерность *кластеры* (англ. *Clusters*) отличается тем, что классы (группы, типы) заранее не заданы, а их количество и состав определяется автоматически в результате процедуры *кластеризации*.

Хранимая ретроспективная информация позволяет определить еще одну закономерность, заключающуюся в поиске существующих *временных рядов* (англ. *Time Series*) и прогнозировании динамики значений в них на будущие периоды времени.

4.3. Группы задач анализа данных

Наряду с поиском самых общих типов закономерностей, которые могут присутствовать в данных (см. разд. 4.2), также выделяют группы более конкретных, частных задач анализа данных. Несмотря на обширную сферу применения *Data Mining* в бизнесе, медицине или государственном управлении (см. разд. 3.1, 3.2), подавляющее большинство этих задач может быть объединено в сравнительно небольшое число групп (табл. 1).

Таблица 1

Основные группы задач анализа данных

Группа задач (англ.)	Аналог в отечественной литературе (рус.)	Пояснение	Пример задачи
Classification and Prediction	Классификация и прогнозирование	Индуктивно разрабатывается обобщенная модель или формулируется некоторая гипотеза, описывающая принадлежность объектов к соответствующим классам	Предсказание роста объемов продаж на основе текущих значений, отнесение претендента на кредит к известным классам кредитоспособности, выявление лояльных или нелояльных держателей кредитных карт, классификация стран по климатическим зонам и т.п.
Clustering	Кластеризация	Выделение некоторого количества групп, имеющих сходные в некотором смысле признаки. Основной принцип – максимизация межклассового и минимизация внутриклассового расстояния	Обнаружение новых сегментов рынка, совершенствование рекламных стратегий для различных групп потребителей
Associations, Link Analysis	Ассоциации, анализ взаимозависимостей	Поиск интересных ассоциаций и / или корреляционных связей	95% покупателей автомобильных шин и автоаксессуаров также приобретали пакет сервисного обслуживания автомобиля, 80% покупателей газировки приобретают и «воздушную» кукурузу

Продолжение табл. 1

Группа задач (англ.)	Аналог в отечественной литературе (рус.)	Пояснение	Пример задачи
Visualization	Визуализация	С использованием графических методов визуализации информации создается графический образ анализируемых данных, отражающий имеющиеся в данных интересные закономерности	Визуализация некоторых зависимостей с использованием 2D- и 3D- измерений
Summarization	Подведение итогов	Интегральное (генерализованное) описание конкретных групп объектов из анализируемого набора данных	Суммирование данных сетевого трафика при оценке эффективности каналов связи [11], подготовка краткого реферата по тексту значительного объема, визуализация многомерных данных большого объема
Deviation (Anomaly) Detection, Outlier Analysis	Определение и анализ отклонений и / или выбросов в данных	Обнаружение фрагментов данных, существенно отличающихся от общего множества данных, выявление нехарактерных патернов (шаблонов)	Анализ наличия шума / ошибок, а также выявление мошеннических действий

Окончание табл. 1

Группа задач (англ.)	Аналог в отечественной литературе (рус.)	Пояснение	Пример задачи
Estimation	Оценивание	Предсказание непрерывных значений признака	Оценка производительности процессора на определенных задачах по ряду параметров процессора, оценка числа детей в семье по уровню образования матери, оценка дохода семьи по количеству в ней автомобилей, оценка стоимости недвижимости в зависимости от ее удаленности от бизнес-центра
Feature Selection, Feature Engineering	Отбор значимых признаков	Применяется при анализе признаковых пространств большой размерности путем сокращения размерности и / или выбора значимых признаков с трансформацией признакового пространства или без трансформации	Как правило, применяется как вспомогательный метод на этапе предварительной обработки данных, а также для повышения эффективности методов визуализации в многомерных признаковых пространствах

4.4. Классификация методов

Существует большое количество различных оснований для стратификации, категоризации, классификации значительного количества существующих и вновь разрабатываемых методов *Data Mining*. Например, можно встретить классификации по принципу работы с исходными обучающими данными (подвергаются они или нет в результате обработки изменениям), по типу получаемого результата (предсказательные и описательные; рис. 8), по видам применяемого математического аппарата (статистические и кибернетические) и др.

Рис. 8. Иллюстрация примера классификации методов Data Mining

Например, по типу используемого математического аппарата, как правило, выделяют следующие основные группы методов *Data Mining*:

1. Дескриптивный анализ и описание исходных данных, предварительный анализ природы статистических данных (проверка гипотез стационарности, нормальности, независимости, однородности, оценка вида функции распределения, ее параметров и т.п.).

2. Многомерный статистический анализ (линейный и нелинейный дискриминантный анализ, кластерный анализ, компонентный анализ, факторный анализ и т.п.).

3. Поиск связей и закономерностей (линейный и нелинейный регрессионный анализ, корреляционный анализ и т.п.).

4. Анализ временных рядов (динамические модели и прогнозирование).

Таблица 2

Пример классификации методов *Data Mining* по математическому аппарату

№ п/п	Раздел	Методы, способы
1	Метрические методы классификации	Метод ближайших соседей и его обобщения, отбор эталонов и оптимизация метрики
2	Логические методы классификации	Понятия закономерности и информативности, решающие списки и деревья
3	Линейные методы классификации	Градиентные методы, метод опорных векторов
4	Байесовские методы классификации	Оптимальный байесовский классификатор, параметрическое и непараметрическое оценивание плотности, разделение смеси распределений, логистическая регрессия
5	Методы регрессионного анализа	Многомерная линейная регрессия, нелинейная параметрическая регрессия, непараметрическая регрессия, неквадратичные функции потерь, прогнозирование временных рядов
6	Нейросетевые методы классификации и регрессии	Многослойные нейронные сети
7	Композиционные методы классификации и регрессии	Линейные композиции, бустинг, эвристические и стохастические методы, нелинейные алгоритмические композиции
8	Критерии выбора моделей и методы отбора признаков	Задачи оценивания и выбора моделей, теория обобщающей способности, методы отбора признаков
9	Ранжирование	
10	Обучение без учителя	Кластеризация, сети Кохонена, таксономия, поиск ассоциативных правил, задачи с частичным обучением, коллаборативная фильтрация, тематическое моделирование, обучение с подкреплением

Детализируя используемый математический аппарат, являющийся важнейшим компонентом практически любых современных методов *Data Mining*, можно получить существенно более глубокую классификацию существующих методов (табл. 2), многие из которых более подробно изложены в главе 5.

4.5. Сравнительные характеристики основных методов

В завершение различных подходов к классификации методов *Data Mining* приведем пример сравнительного анализа наиболее широко используемых методов между собой, используя в качестве характеристики каждого из атрибутов следующую шкалу оценок: чрезвычайно низкая, очень низкая, низкая / нейтральная, нейтральная / низкая, нейтральная, нейтральная / высокая, высокая, очень высокая (табл. 3).

Таблица 3

Пример сравнительного анализа методов *Data Mining*

Метод / Характеристика	Линейная регрессия	Нейронные сети	Методы визуализации	Деревья решений	K-ближайшего соседа
Точность	Нейтральная	Высокая	Низкая	Низкая	Низкая
Масштабируемость	Высокая	Низкая	Очень низкая	Высокая	Очень низкая
Интерпретируемость	Высокая / нейтральная	Низкая	Высокая	Высокая	Высокая / нейтральная
Пригодность к использованию	Высокая	Низкая	Высокая	Высокая / нейтральная	Нейтральная
Трудоемкость	Нейтральная	Нейтральная	Очень высокая	Высокая	Низкая / нейтральная
Разносторонность	Нейтральная	Низкая	Низкая	Высокая	Низкая
Быстрота	Высокая	Очень низкая	Чрезвычайно низкая	Высокая / нейтральная	Высокая
Популярность	Низкая	Низкая	Высокая / нейтральная	Высокая/ нейтральная	Низкая

Видно, что ни один из методов нельзя признать единственно эффективным, имеющим очевидное превосходство над другими методами.

Это подтверждает тезис о том, что залогом успешного решения задач *Data Mining* является необходимость погружения не только в особенности предметной области, но и в математические основы различных методов обработки и анализа данных.

5. ПРИНЦИПИАЛЬНЫЕ ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

Машинное обучение (англ. *Machine Learning*) изучает способы построения особого класса алгоритмов из области искусственного интеллекта, отличающихся способностью к обучению.

Такое обучение в некоторой степени аналогично обучению, которое доступно человеку. Например, когда родители показывают ребенку автомобиль, он позднее способен его отличать от других объектов (дерева, дома, человека и др.), причем даже если это автомобиль другого цвета, модели, размера и др. Алгоритмы машинного обучения предполагают аналогичный подход, при котором некоторая модель (статистическая, нейросетевая, комбинированная и т.п.) в результате обучения настраивает собственные параметры таким образом, чтобы отличать предъявленные ей образы. При этом обучение осуществляется с использованием специально подготовленных обучающих данных, позволяющих предусмотреть достаточно однозначное соответствие между предъявляемыми признаками и ожидаемым на них ответом модели (рис. 9).

В общем случае для построения алгоритмов машинного обучения требуется подготовить три типа выборок – обучающие (англ. *train set*), валидационные (англ. *validation set*), тестовые (англ. *test set*). Сначала алгоритм обучается на обучающей выборке, с использованием которой происходит начальная настройка «внутренних» параметров модели (собственно обучение). Это размеченные данные (каждому набору признаков в выборке уже сопоставлено значение целевой переменной – метки с «правильным ответом»), предварительно обработанные и выбранные с учетом самого главного критерия – *репрезентативности*. Предполагается, что обучающая выборка должна включать в себя наиболее характерные прецеденты, описывая статистические свойства генеральной совокупности.

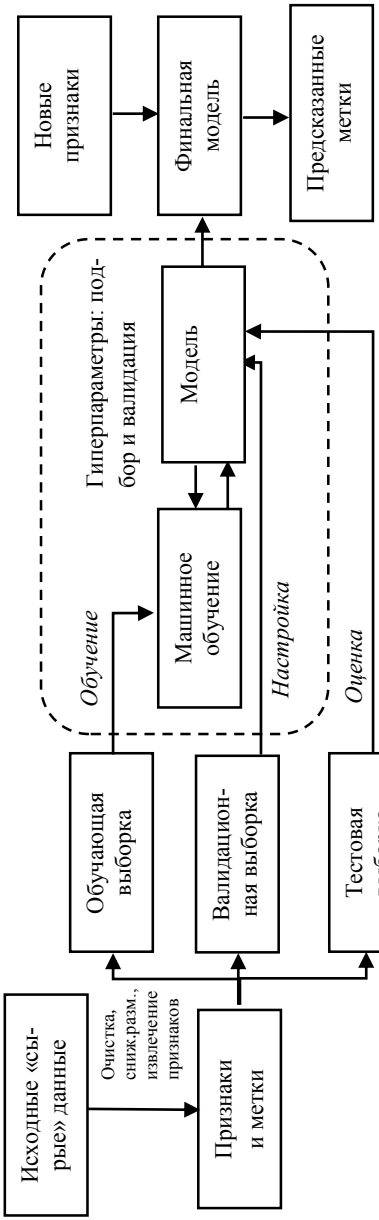


Рис. 9. Принципиальная схема машинного обучения

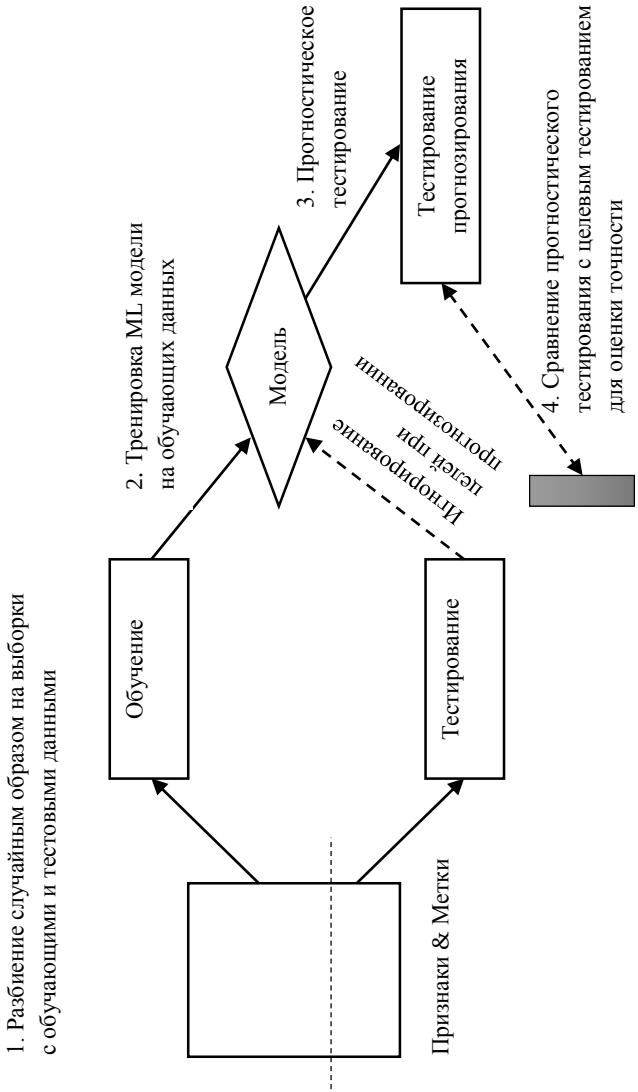


Рис. 10. Оценка качества модели

Далее обученная модель применяется на валидационной выборке (англ. *validation set, development set*), в процессе чего производятся настройка гиперпараметров / метапараметров модели, настройка признакового пространства, осуществляется промежуточный контроль переобучения. Например, в регрессионной модели таким гиперпараметром будет вид регрессионной зависимости, а собственно коэффициенты регрессионной модели, значение которых будет «подстраиваться» в процессе обучения, такими гиперпараметрами не будут. Аналогично, например, архитектура нейросети в нейросетевой модели будет гиперпараметром, а собственно весовые коэффициенты формальных нейронов, значения которых будут «подстраиваться» в процессе обучения, такими гиперпараметрами не будут.

Финальное качество обученной модели оценивают по тестовой выборке. Основное ее отличительное свойство заключается в следующем: тестовая выборка не должна участвовать в обучении модели, ее настройке и оптимизации. Очевидно, тестовая выборка также должна быть размеченной (т.е. каждому набору признаков в выборке должен соответствовать правильный ответ, который мы ожидаем от модели) для проведения оценки качества обучения. Несмотря на очевидность и простоту данного правила, на практике нередки случаи, когда в процессе обучения в обучающую или валидационную выборку попадают элементы тестовой выборки, разрушая корректность обучения модели и проверки ее адекватности.

В общем случае оценка качества модели производится путем сравнения результатов применения обученной модели на тестовой выборке, сопоставляя ответы модели и размеченные ответы тестовой выборки (рис. 10).

На сегодняшний день существует множество рекомендаций и уже сформированных практик оптимального разбиения данных на выборки, так же как и существует множество критериев оценки качества моделей, о которых пойдет речь ниже.

6. ОСНОВНЫЕ МЕТОДЫ АНАЛИЗА И ИНТЕРПРЕТАЦИИ ДАННЫХ

6.1. Предварительная обработка данных

Практическое применение методов *Data Mining* предполагает многоэтапную процедуру, основные этапы которой изложены в разделе 3.1. Одним из ключевых этапов этой процедуры, предваряющей, собственно, применение методов *Data Mining*, является этап предварительной обработки данных, включающий различные типы преобразований. Рассмотрим их более подробно.

Одним из ключевых преобразований этапа предварительной обработки данных является «*очистка*» *данных* (англ. *Data Cleaning, Data Cleansing, Data Scrubbing*), предполагающая обнаружение и корректировку / удаление поврежденных элементов данных. Данные, имеющие такие повреждения (неточные, неполные, дублированные, противоречивые, зашумленные), называют «*грязными*». Источниками «грязных» данных могут быть поврежденные инструменты сбора данных, проблемы во введении исходных данных, «человеческий фактор» в случае неавтоматического варианта формирования данных, проблемы в каналах передачи данных, ограничения технологий передачи данных, использование разных наименований в пределах одной номенклатуры и т.п.

Особую актуальность очистки грязных данных подтверждает известное в информатике выражение: «*Мусор на входе – мусор на выходе*» (англ. *Garbage In – Garbage Out, GIGO*[1]). Оно означает, что при неверных входных данных будут получены неверные результаты работы в принципе верного алгоритма. Действительно, практически полезными результаты применения каких бы то ни было

[1] В отличие от известной дисциплины обслуживания *FIFO* встречается нечасто, а жаль…

методов *Data Mining* будут только в случае использования ими корректных достоверных данных. Учитывая, что такие данные могут быть доставлены из разных источников и быть достаточно существенными в объеме, задача получения и обработки «чистых» данных может быть крайне непростой.

Более того, следует отметить, что наличие «грязных» данных порой более проблематично, чем их отсутствие вовсе – извлечение полезных знаний из таких данных может потребовать значительного времени, причем безрезультатно. При этом еще более проблематичным будет успешное извлечение из таких данных недостоверных знаний и дальнейшее их практическое использование с трудно предсказуемыми последствиями. Именно поэтому этапу получения «чистых», готовых к анализу данных придают большое значение, а по затратам времени этот этап может быть одним из самых длительных [36].

Сегодня проблемам получения «чистых» данных посвящены отдельные достаточно емкие исследования [31]. В них обсуждается целый спектр различных особенностей этой проблематики, начиная от концептуальных вопросов и завершая деталями современных технологических решений в базах данных и хранилищах данных. Отметим здесь некоторые наиболее принципиальные моменты.

Все проблемы очистки данных разделяют на две группы, вызванные *интеграцией различных источников данных* (англ. *Multi-Source Problems*) или обусловленные проблемами *единственного источника данных* (англ. *Single-Source Problems*). В свою очередь, каждая из групп может быть разделена на две самостоятельные группы, определяемые либо *несовершенством схем интегрируемых баз данных* (англ. *Schema Level*), либо *несовершенством на уровне собственно элементов данных* (англ. *Instance Level*): записей, объектов и т.п. Далее каждая из ветвей полученного дерева классификации детализируется конкретным перечнем возможных проблем очистки данных (рис. 11).

В табл. 4 и 5 приведены некоторые примеры «грязных» данных, порожденные на разных уровнях – на *Schema Level* и на *Instant Level.*

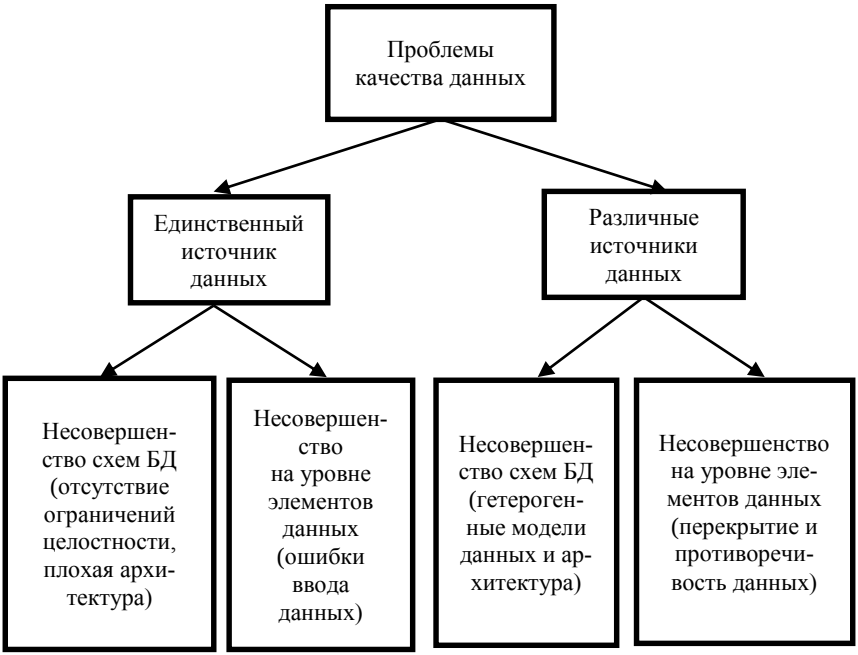


Рис. 11. Пример классификации проблем качества данных в различных источниках

Таблица 4

Примеры «грязных» данных единственного источника на уровне схемы данных

Проблема		«Грязные» данные	Причины
Атрибут	Недопустимые значения	дата рождения = 30.13.70	Значение за пределами диапазона
Запись	Нарушение зависимости атрибутов	возраст = 22 дата рождения = 12.02.70	Возраст = (текущая дата – дата рождения)
Тип записи	Нарушение уникальности	сотр. 1 = (имя = Иван, SSN = 123) сотр. 2 = (имя = Петр, SSN = 123)	SNN должен быть уникальным
Источник	Нарушение ссылочной целостности	сотр. 1 = (имя = Иван, отд. = 789)	Отдела с номером 789 не существует

Таблица 5

Примеры «грязных» данных единственного источника на уровне записей

Причина		«Грязные» данные	Причина
Атрибут	Пропущенное значение	тел. = 9999-999999	Недопустимые (некорректные, null и т.п.) значения при вводе
	Орфографические ошибки	город = Тамск город = Москваа	Орфографическая ошибка
	Сокращения и аббревиатуры	должность = А, отдел = ЛТО	
	Объединенные значения	имя = Иван 12.07.70 Томск	Несколько значений в атрибуте
Запись	Нарушение зависимости атрибутов	город = Томск, инд. = 666777	Город и индекс не соответствуют друг другу
Тип записи	Дубликаты записей	сотр. 1 = (имя = Иван, SSN = 123) сотр. 2 = (имя = Иван, SSN = 123)	
	Противоречащие записи	сотр. 1 = (имя = Иван, SSN = 123) сотр. 1 = (имя = Иван, SSN = 321)	Записи одного и того же сотрудника с разным SSN
Источник	Неверные ссылки	сотр. = (имя = Иван, отд. = 789)	Отдел с номером 789 существует, но указан неверно

Выделяют следующее этапы очистки данных:

1. **Анализ данных** (англ. *Data analysis*). Для того чтобы определить, какие виды ошибок и несоответствий должны быть удалены, требуется детальный анализ данных. В дополнение к инспекции данных или отдельных выборок данных «вручную» следует использовать и метаданные.

2. **Определение способов трансформации потоков данных и правил отображения** (англ. *Definition of transformation workflow and mapping rules*). На данном этапе выполняется оценка количества

источников данных, степени их неоднородности и «загрязненности». На основе этой информации создаются схемы потоков данных, позволяющие преобразовать множество источников данных в один, избегая создания ошибок *Multi-Source* слияния (например, появление дублирующих записей).

3. **Верификация** (англ. *Verification*). Оценка корректности и результативности выполнения предыдущего этапа (например, на небольшой выборке данных). При необходимости производится возврат к этапу 2 для его повторного выполнения.

4. **Трансформация** (англ. *Transformation*). Загрузка данных в единое хранилище с использованием правил трансформации, определенных и отлаженных на этапах 2 и 3. Очистка данных уровня *Single-Source*.

5. **Обратная загрузка очищенных данных** (англ. *Backflow of cleaned data*). Имея на этапе 4 очищенный набор данных в едином хранилище, целесообразно этими «чистыми» данными заменить аналогичные «грязные» данные в исходных источниках. Это позволит в будущем не выполнять повторно все этапы преобразований по очистке данных.

Реализовать эти этапы можно самыми различными путями с использованием существующих и созданных специально способов и технологий. Рассмотрим наиболее интересные из них.

Этап анализа данных предполагает анализ использования метаданных, которых, как правило, недостаточно для оценки качества данных из имеющихся источников. Поэтому важно анализировать реальные примеры данных, оценивая их характеристики и сигнатуры значений. Это позволяет находить взаимосвязи между атрибутами в схемах данных различных источников. Выделяют два подхода решения этой задачи – *профилирование данных* (англ. *data profiling*) и *извлечение данных* (англ. *data mining*).

Профилирование данных ориентировано на анализ индивидуальных атрибутов, характеризующихся их конкретными свойствами: тип данных, длина, диапазон значений, частота встречаемости дискретных значений, дисперсия, уникальность, встречаемость «*null*» значений, типичная сигнатура записи (например, у телефонного

номера). Именно набор подобных свойств (профиль) позволяет оценить различные аспекты качества данных.

Извлечение данных предполагает поиск взаимосвязей между несколькими атрибутами достаточно большого набора данных. Учитывая то, что этот способ получил название *data mining*, здесь используют упоминавшиеся выше (см. табл. 1) методы *кластеризации*, *подведения итогов*, *поиска ассоциаций* и *последовательностей*. Кроме того, для дополнения пропущенных значений, корректировки недопустимых значений или идентификации дубликатов могут быть использованы существующие ограничения целостности (англ. *integrity constraints*), принятые в реляционных базах данных, наложенные дополнительно на бизнес-связи между атрибутами. Например, известно, что «*Total* = *Quantity* × *Unit_Price*». Все записи, не удовлетворяющие этому условию, должны быть изучены более внимательно, исправлены или исключены из рассмотрения.

Для разрешения проблем очистки данных в одном источнике (*single-source problems*), в том числе перед его интеграцией с другими источниками данных, реализуют следующие этапы:

– **Извлечение значений из атрибутов свободной формы** (разбиение атрибутов, англ. *Extracting values from free-form attributes* (*attribute split*)). В данном случае речь может идти о строковых значениях, сохраняющих несколько слов подряд (например, адрес или полное имя человека). В этом случае требуется четкое понимание того, на какой позиции этого значения находится интересующая нас часть атрибута. Возможно, потребуется даже сортировка составных частей такого атрибута.

– **Валидация и коррекция** (англ. *Validation and correction*). Данный этап предполагает поиск ошибок ввода данных и их исправление наиболее автоматическим способом, например используя автоматическую проверку правописания во избежание орфографических ошибок и опечаток. Словарь географических названий и почтовых кодов также следует использовать для корректировки значений вводимых адресов. Зависимость атрибутов (дата рождения – возраст; *Total* = *Quantity* × *Unit_Price* и т.п.) также способствует избеганию множества ошибок в данных.

– **Стандартизация** (англ. *Standardization*). Этот этап предполагает приведение всех данных к единому универсальному формату. Примерами таких форматов являются формат написания даты и времени, размер регистра в написании строковых значений. Текстовые поля должны исключать префиксы и суффиксы, аббревиатуры в них должны быть унифицированы, исключены проблемы с различной кодировкой.

Одной из основных проблем, вызванных интеграцией различных источников (*multi-source problems*) данных, является устранение дублирования записей. Этот этап выполняется после подавляющего большинства преобразований и чисток. Он предполагает сначала идентификацию сходных в некотором смысле записей, а затем их слияние с объединением атрибутов. Очевидно, решение этой задачи при наличии у дублирующих записей первичного ключа достаточно просто. Если такого однозначно идентифицирующего признака нет, то задача устранения дубликатов значительно осложняется, требуя применения нечетких (англ. *fuzzy*) подходов сравнения (близости в некотором смысле) записей между собой.

6.2. Оптимизация признакового пространства

Современные массивы данных, к которым могут быть применены те или иные методы *Data Mining*, могут характеризоваться большим числом признаков, формирующих признаковое пространство большой размерности. Поэтому актуальной является задача снижения размерности такого пространства до размерности, позволяющей без лишних затруднений осуществлять обработку данных и / или их визуализацию. Решение такой задачи называют *оптимизацией признакового пространства* или *поиском значимых признаков* (англ. *Feature Selection*, иногда – *Feature Engineering*). Ее решение сегодня часто становится самостоятельной исследовательской задачей, которую решают с применением различных подходов.

При этом все подходы к снижению размерности исходного признакового пространства могут быть разделены на два больших класса.

Первый класс предусматривает трансформацию признакового пространства. Один из наиболее известных и применяемых на практике подходов этого класса – *метод главных компонентов* (МГК, англ. *Principal Component Analysis*) [50, 67]. Рассмотрим его кратко в разд. 6.2.1.

Другой класс методов заключается в выборе наиболее информативных, полезных признаков и исключении из рассмотрения неинформативных признаков без трансформации исходного пространства [32, 33]. Применяют различные методы и подходы, с которыми можно подробнее ознакомиться в специальной литературе [51]:

- полного или усеченного перебора;
- ветвей и границ;
- эволюционные;
- со случайным выбором.

Ознакомимся здесь с одним из подходов (разд. 6.2.2), который может быть использован при усеченном переборе признаков, в основе которого лежит *критерий попарной разделимости Джеффриса–Матуситы* (ДМ) [33].

6.2.1. С трансформацией пространства признаков

Одним из широко используемых традиционных методов решения задачи трансформации исходного пространства признаков в новое является МГК [50]. В основе МГК лежит идея нахождения для исходного набора признаков $\mathbf{x} = \{x_i, i = 1, \ldots, P\}$ размерности P такого набора скрытых (латентных) переменных $\mathbf{y} = \{y_i, i' = 1, \ldots, P'\}$ (главных компонент) размерности P', который бы максимально объяснял дисперсии многомерных переменных $\mathbf{x}$ при выполнении условия $P' < P$.

Главные компоненты представляют собой новое множество исследуемых признаков $\mathbf{y}$, каждый из которых получен в результате некоторой линейной комбинации исходных признаков $\mathbf{x}$. Причем полученные в результате преобразования новые признаки $\mathbf{y}$ некоррелированы между собой и упорядочены по степени рассеяния

(дисперсии) таким образом, что первый признак обладает наибольшей дисперсией.

В общем случае [i']-й главной компонентой исходного признакового пространства $\mathbf{\Omega}$ с ковариационной матрицей $\mathbf{\Sigma}$ и вектором средних $\boldsymbol{\mu}$ называется нормированная линейная комбинация компонент исходного P-мерного признакового вектора $\mathbf{x}$

$$y_{i'} = l_{i'1}x_1 + l_{i'2}x_2 + \ldots + l_{i'P}x_P = L_{i'}\mathbf{x}, \qquad (6.1)$$

где $L_{i'} = (l_{i'1}, l_{i'2}, \ldots, l_{i'P})$ – [i']-й собственный вектор матрицы $\mathbf{\Sigma}$ пространства Ω.

Геометрическая модель МГК для двумерного случая показана на рис. 12.

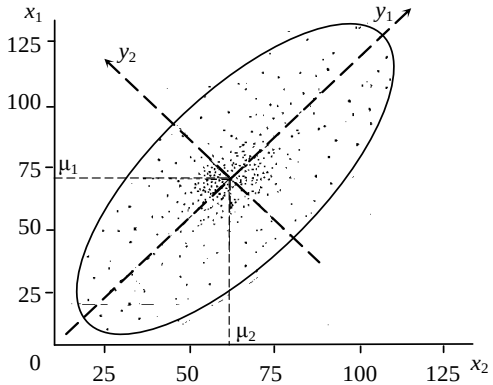


Рис. 12. Пример двумерной модели главных компонент

Множество признаковых векторов $\mathbf{x}^j = (x_1^j, x_2^j)$, $j = 1, \ldots, N$, располагается примерно в очертаниях эллипсоида рассеивания, и оси главных компонент y_1 и y_2 проходят вдоль его осей. Обобщенная дисперсия Σ_y и сумма дисперсий ($Dy_1 + Dy_2 + \ldots + Dy_{P'}$) главных компонент $\mathbf{y}$ равны обобщенной дисперсии Σ_x и сумме дисперсий ($Dx_1 + Dx_2 + \ldots + Dx_P$) исходных признаков $\mathbf{x}$. На основании этого выносится решение о том, сколько последних главных компонент P' следует практически без ущерба для информативности изъять из рассмотрения, сократив тем самым размерность исследуемого пространства Ω.

Анализируя изменение относительной доли дисперсии $\Psi(P')$

$$\Psi(P') = \frac{Dy_1+Dy_2+\cdots+Dy_{P'}}{Dx_1+Dx_2+\cdots+Dx_P}, (1 \le P' \le P), \quad (6.2)$$

вносимой первыми главными компонентами, в зависимости от числа этих компонент можно разумно определить число компонент P', которое целесообразно оставить в рассмотрении.

Выбрав предельную величину относительной доли дисперсии $\Psi(P')$, можно задать порог, определяющий количество главных компонент P', используемых в дальнейшем для классификации.

Несмотря на относительную простоту, МГК обладает двумя существенными недостатками. Во-первых, при использовании МГК предполагается, что распределение исходных многомерных данных подчинено нормальному закону и трансформация происходит относительно многомерного гиперэллипсоида рассеивания (хотя исходные измерения могут быть распределены не в рамках такого гиперэллипсоида). Во-вторых, трансформация исходного признакового пространства способна повлечь за собой значительные искажения признакового пространства, что может привести к снижению разделимости в таком новом признаковом пространстве для объектов и снизить итоговое качество классификации.

6.2.2. Без трансформации пространства признаков

Основной идеей метода попарной разделимости ДМ для снижения размерности многомерных данных метода является использование перебора исходного P-мерного набора признаков $\mathbf{x} = \{x_i, i = 1, \ldots, P\}$ с целью максимизации заданного критерия информативности и выделения подпространства $\mathbf{y} = \{y_i, i' = 1, \ldots, P'\}$ наиболее информативных полезных признаков. При этом в качестве критерия информативности признаков используется критерий интегральной разделимости известного набора классов $\{\omega_i, i = 1, \ldots, M\}$, определяемых соответствующими обучающими выборками $\{V_i, i = 1, \ldots, M\}$. Интегральная разделимость вычисляется как средневзвешенное значение попарных классовых разделимостей, определяемых расстоянием ДМ (*JM*) [33], характеризующим среднее

расстояние между функциями условных плотностей распределения $p(\mathbf{x}|\omega_i)$ и $p(\mathbf{x}|\omega_j)$ соответствующей пары классов ω_i и ω_j. Расстояние JM_{ij} между классами ω_i и ω_j определяется следующим выражением:

$$JM_{ij} = \int_{\mathbf{x}} \left\{ \sqrt{p(\mathbf{x}|\omega_i)} - \sqrt{p\left(\mathbf{x}\middle|\omega_j\right)} \right\}^2 d\mathbf{x}. \qquad (6.3)$$

Если в выражении (6.3) использовать предположение о нормальном распределении условных плотностей вероятности распределения признаков **x**, то расчет критерия ДМ может быть представлен в виде:

$$JM_{ij} = 2(1 - e^{-B}), \qquad (6.4)$$

где B – расстояние Бхаттачария [33], зависящее от параметров двух многомерных нормальных распределений $N_i(\mathbf{x}|\mu_i, \Sigma_i)$ и $N_j(\mathbf{x}|\mu_j, \Sigma_j)$, определяемых вектором средних $\boldsymbol{\mu}$ и ковариационной матрицей $\boldsymbol{\Sigma}$ как

$$B = \frac{1}{8(\mu_i-\mu_j)^{\mathrm{T}}\left\{\frac{\Sigma_i+\Sigma_j}{2}\right\}^{-1}(\mu_i-\mu_j)} + \frac{1}{2\ln\left\{\frac{\left|(\Sigma_i+\Sigma_j)/2\right|}{|\Sigma_i|^{\frac{1}{2}}|\Sigma_j|^{\frac{1}{2}}}\right\}}. \qquad (6.5)$$

Для нахождения интегральной метрики разделимости JM_{ave} всех классов признакового пространства используют вычисления с использованием элементов симметричной матрицы $\|\mathbf{JM}\|$ попарных разделимостей, расположенных выше главной диагонали

$$JM_{\text{ave}} = \sum_{i=1}^{M} \sum_{j=i+1}^{M} p(\omega_i) p\left(\omega_j\right) JM_{ij}, \qquad (6.6)$$

где $p(\omega_i)$ и $p(\omega_j)$ – весовые коэффициенты (априорные вероятности) классов ω_i и ω_j соответственно. Функция JM_{ij} – расстояния (6.3) асимптотически сходится к значению 2,0. Если элемент матрицы $JM_{ij} = 2{,}0$, то это означает, что классы ω_i и ω_j абсолютно разделимы и вероятность их перепутывания равна нулю.

Для снятия ограничения на согласованность распределения с гауссовым в распознаваемых классах вместо оценок (6.4) и (6.5) для расчета попарных классовых разделимостей JM_{ij} используется оценка (6.3). В качестве метода численного интегрирования для нахождения значений оценки (6.3) применяется метод Монте-Карло вычисления кратных определенных интегралов.

В силу высокой вычислительной сложности нахождения значений оценки (6.3) задача полного перебора комбинаций исходных

признаков (количество сочетаний из P по P' – $C_P^{P'}$) заменяется усеченным перебором, основанном на отыскании информативных поднаборов (1–2 признака) в исходном P-мерном наборе признаков **x** из условия максимума критерия JM_{ave} (6.6). Каждый найденный поднабор признаков добавляется в информативный набор **y**, и критерий JM_{ave} проверяется для всех попавших в набор **y** компонент. Усеченный перебор исходных признаков из **x** завершается по достижении приемлемого значения критерия JM_{ave} (обычно 1,9–2,0) или по завершении перебора всех исходных признаков **x**.

Несмотря на бóльшую по сравнению с МГК вычислительную сложность, данный метод не накладывает ограничения на вид распределения исходных признаков классифицируемых объектов и не вносит дополнительные искажения в признаковую структуру пространства.

6.3. Классификация

6.3.1. Постановка задачи классификации

Задачу классификации математически в общем случае можно представить следующим образом [67]. Для каждого класса ω_i из исходного алфавита $\{\omega_i, i = 1, ..., M\}$ (M – количество предопределенных типов) введем понятие вероятности появления класса ω_i в пространстве признаков $\mathbf{\Omega}(\mathbf{x})$. Данная вероятность $p(\omega_i)$ называется *априорной вероятностью* класса ω_i. Также предположим, что для каждого класса ω_i известна многомерная (P-мерная) функция $p(\mathbf{x} \mid \omega_i)$, описывающая условную плотность распределения (УПР) вектора признаков **x** в классе ω_i, для которой справедливо

$$\int_{\omega_i} p(\mathbf{x} \mid \omega_i) d\mathbf{x} = 1, \qquad (6.7)$$

Априорная вероятность $p(\omega_i)$ и УПР $p(\mathbf{x} \mid \omega_i)$ являются наиболее полными вероятностными характеристиками класса ω_i.

Таким образом, задача классификации может быть сформулирована в виде задачи статистических решений (испытание M стати-

стических гипотез) с помощью определения *дискриминантной функции* $\phi(\mathbf{x})$, принимающей значение ϕ_i в случае, когда принимается гипотеза H_i:$\mathbf{x} \in \omega_i$. Полагается, что принятие классификатором решения ϕ_i, когда в действительности входной образ принадлежит к классу ω_j, приводит к потере, определяемой *функцией потерь* $L(\phi_i | \omega_j)$. Тогда условный риск $R(\phi_i | \mathbf{x})$ принятия решения ϕ_i в случае $\mathbf{x} \in \omega_j$ находится как

$$R(\phi_i | \mathbf{x}) = \sum_{j=1}^{M} L(\phi_i | \omega_j) p(\omega_j | \mathbf{x}), \qquad (6.8)$$

где $p(\omega_j | \mathbf{x})$ носит название *апостериорной вероятности* события $\mathbf{x} \in \omega_j$ и вычисляется исходя из априорной вероятности $p(\omega_i)$ и условной плотности распределения $p(\mathbf{x} | \omega_i)$ согласно теореме Байеса [67] следующим образом:

$$p(\omega_j | \mathbf{x}) = \frac{p(\omega_j) p(\mathbf{x} | \omega_j)}{\sum_{k=1}^{M} p(\omega_k) p_k(\mathbf{x} | \omega_k)}. \qquad (6.9)$$

Задача классификации сводится к выбору наименьшего условного риска (6.8), т.е.

$$\phi(\mathbf{x}) = \phi_i: (\mathbf{x} \in \omega_i), \text{ если } R(\phi_i | \mathbf{x}) < R(\phi_j | \mathbf{x}), \forall\ i \neq j. \qquad (6.10)$$

Правило классификации (6.10) носит название *байесовского решающего правила классификации.* При использовании в (6.8) нуль-единичной функции потерь

$$L(\varphi_i | \omega_j) = \begin{cases} 0, i = j \\ 1, i \neq j \end{cases} i, j = 1, ..., M \qquad (6.11)$$

риск, соответствующий такой функции, является средней вероятностью ошибки (ложной классификации) и, исходя из (6.8) и (6.11), определяется как

$$R(\phi_i | \mathbf{x}) = \sum_{j \neq i} p(\omega_j | \mathbf{x}) = 1 - p(\omega_i | \mathbf{x}),\ i, j = 1, ..., M. \qquad (6.12)$$

Исходя из (6.10) и (6.12), дискриминантная функция $\phi_i(\mathbf{x})$ при использовании нуль-единичной функции потерь (6.11) выглядит следующим образом:

$$\phi_i(\mathbf{x}) = p(\omega_i)\, p(\mathbf{x} | \omega_i),\ i, j = 1, ..., M, \qquad (6.13)$$

и согласно (6.9) и (6.13) байесовское решающее правило (6.10) можно записать:

$$m(\mathbf{x})\colon \mathbf{x} \in \omega_i, \text{ если } p(\omega_i)\, p(\mathbf{x}\,|\,\omega_i) > p(\omega_j)\, p(\mathbf{x}\,|\,\omega_j),\ \forall\ i \neq j. \quad (6.14)$$

В *параметрическом подходе* к классификации при оценке УПР $p(\mathbf{x}\,|\,\omega_i)$ принимается гипотеза о некотором известном виде плотности распределения признаков (например, гауссовом), что позволяет использовать для нахождения $p(\mathbf{x}\,|\,\omega_i)$ ее параметрическую оценку. Следует отметить, что УПР называют *правдоподобием*, а такой подход к классификации – методом *максимального правдоподобия* (англ. *maximum likelihood, ML*). В случае гауссова распределения используется оценка вида [33]:

$$\hat{p}(\mathbf{x}\,|\,\omega_i) = (2\pi)^{-P/2}\left|\Sigma_i\right|^{-1/2} \exp\left\{-1/2(\mathbf{x}-\hat{\mu}_i)^t\, \Sigma_i^{-1}(\mathbf{x}-\hat{\mu}_i)\right\},$$
$$i = 1, ..., M, \quad (6.15)$$

где μ_i – выборочный вектор средних типа ω_i; Σ_i – выборочная ковариационная матрица типа ω_i; P – количество признаков; $|\Sigma_i|$ – детерминант выборочной ковариационной матрицы типа ω_i; Σ_i^{-1} – обратная выборочная ковариационная матрица типа ω_i [67, 112]. При этом вычисление априорной вероятности $p(\omega_i)$ в выражении (6.9) производится с помощью простых способов, в которых $p(\omega_i)$ принимается равной для всех классов:

$$p(\omega_i) = p(\omega_j),\ \forall\ i, j = 1, ..., M, \quad (6.16)$$

либо пропорциональной размеру имеющихся обучающих данных:

$$p(\omega_i) = \frac{N_i}{\sum_{k=1}^{M} N_k},\ i = 1, ..., M, \quad (6.17)$$

где N_k – размер обучающей выборки, соответствующий типу ω_k. Классификацию, в основе которой лежит байесовское решающее правило (6.9), причем вне зависимости от вида используемой оценки УПР (параметрическая или непараметрическая), называют *байесовской* (англ. *Bayes* или *Naive Bayes*[1]).

[1] «Наивным» байесовский классификатор называют из-за предположения о независимости признаков вектора **x** между собой.

Таким образом, основой большинства современных классификаторов является теорема Байеса, а их математический аппарат может быть реализован с использованием различных подходов. Среди них выделяют *непараметрические статистические классификаторы*, включающие простые, такие как классификатор по правилу параллелепипеда, и значительно более сложные, использующие в своей основе непараметрическое оценивание УПР признаков [58, 71, 97, 117]. При классификации оценка УПР в выражении (6.9) на основе непараметрического подхода характеризуется меньшей чувствительностью к статистическим характеристикам данных, эффективна с точки зрения точности классификации при произвольном (неизвестном) распределении признаков, в том числе и отличном от нормального (гауссова). Однако непараметрические классификаторы требуют перебора всех значений обучающей выборки при оценке УПР для каждого **x**, поэтому их отличают высокие вычислительные затраты. Рассмотрим более подробно этот подход в разд. 6.3.2.

Также к непараметрическим относят *нейросетевые классификаторы*, основанные на использовании аппарата *искусственных нейронных сетей* (ИНС) [100]. Однако в этом случае, как правило, не осуществляется оценка УПР. Для краткости изложения такие классификаторы называют *нейросетевыми классификаторами*, а искусственные нейронные сети – просто *нейросетями*. Рассмотрим их более подробно в разд. 6.3.3.

В последнее время набирают популярность более математически сложные классификаторы с использованием *машины опорных векторов* (англ. *support vector machine – SVM*) [7, 33]. Они характеризуются достаточно высокой устойчивостью к статистическим характеристикам исходных векторов признаков [37, 59]. При этом классификаторы *SVM* по сравнению с нейросетевыми классификаторами для задач классификации в пространстве большой размерности выглядят предпочтительнее – при практической реализации они вместо многоэкстремальной задачи (с вероятностью попадания в локальный экстремум) решают задачу квадратичного программирования, имеющую единственное решение. Кроме того, разделяющие поверхности решения классификаторов *SVM* обладают более

высокими дифференцирующими (разделяющими) возможностями за счет максимизации ширины разделяющей полосы [37, 59]. Это относит такие классификаторы к перспективным с точки зрения вычислительной эффективности и точности. Детали построения таких классификаторов изложены в разд. 6.3.4.

6.3.2. Контролируемая непараметрическая классификация

Как отмечено выше, для проведения классификации признаков, распределение которых априори неизвестно или не согласовано с нормальным законом распределения, используют различные подходы к *непараметрической оценке плотности распределения.* Среди них наиболее широкое распространение получил подход к оценке УПР по методу *k*-го ближайшего соседа (для краткости будем приводить англоязычную аббревиатуру названия метода – *k*-NN), которая определяется, исходя из выражения [117]:

$$p(\mathbf{x} \mid \omega_i) = \frac{1}{N} \frac{k_P - 1}{\mathrm{V}(k_P, N, \mathbf{x})}, \; i = 1, \ldots, M, \tag{6.18}$$

где k_P – параметр близости соседа, N – величина выборки, $\mathrm{V}(k_P, N, \mathbf{x})$ – объем множества всех элементов обучающей выборки, расстояние которых до точки $\mathbf{x}$ в P-мерном пространстве меньше или равно R_k^P. В случае использования евклидова расстояния

$$\mathrm{V}(k_P, N, \mathbf{x}) = \frac{\pi^{P/2} R_k^P}{\left|\mathrm{A}\right|^{1/2} \Gamma[(P+2)/2]}, \tag{6.19}$$

где Г – гамма-функция, А – единичная матрица.

В выражении (6.18) величина k_P является параметром, при этом существует ряд методик нахождения ее оптимального значения $k_{\text{опт}}$ [117]. К сожалению, поиск значения $k_{\text{опт}}$ ведет к увеличению вычислительной сложности алгоритмов оценки УПР, что затрудняет их использование при решении практических задач. Поэтому на практике значение k_P часто принимают фиксированным (например, 1, 3, 21, 87, $\sqrt{N}$, где N – размер выборки [20, 23, 25, 117]). При этом очевидно, что большее значение k_P требует большего количества

операций по расчету расстояния R_k^P в (6.19), что ведет к дополнительным вычислительным затратам при классификации. Поэтому, учитывая важность проведения непараметрической классификации с высокой вычислительной эффективностью, это значение принимают фиксированным (например, $k_P = 3$).

Выше отмечено, что более широкому использованию непараметрических подходов к оценке плотности распределения препятствует их низкая вычислительная эффективность, связанная с необходимостью перебора всех значений обучающей выборки для оценки УПР в точке **x** *P*-мерного пространства. Поэтому задача повышения вычислительной эффективности оценки УПР в непараметрических методах оценки плотности для решения практических задач классификации является отдельной, интенсивно разрабатываемой, областью исследований [77, 99].

6.3.3. Контролируемая непараметрическая нейросетевая классификация

На сегодняшний день нейросетевой аппарат достаточно широко применяется при решении самых различных задач классификации, прогнозирования, оценки плотности распределения, поэтому приведем здесь лишь некоторые необходимые пояснения, связанные с областью ИНС [7, 100]. Так, широко используется такое понятие, как *формальный нейрон,* представляющий собой упрощенную математическую абстракцию биологического нейрона (рис. 13).

Каждый такой нейрон обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал может поступать на синапсы других нейронов. Каждый синапс характеризуется величиной синаптической связи или ее весом w_i. Кроме того, важной характеристикой любого формального нейрона является функция активации, или *пороговая функция f* (*S*). Наиболее распространенными пороговыми функциями являются сигмоидные функции типа гиперболического тангенса и логистической функции [98].

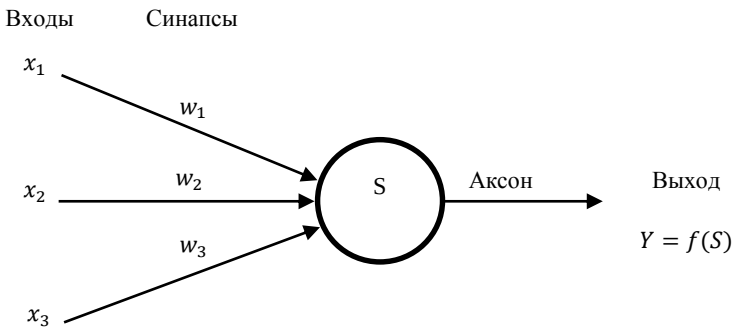


Рис. 13. Схема формального нейрона

Наиболее популярными и широко распространенными, в том числе и для решения задач контролируемой классификации, являются нейросети прямого распространения – *многослойные персептроны* [98]. Они позволяют работать с данными произвольного распределения и учитывать такие закономерности в данных, которые затруднительно учесть другими методами [7].

Потенциально нейросетевые классификаторы обладают рядом существенных преимуществ по сравнению с традиционными статистическими классификаторами. Так, в качестве входных данных для них можно использовать трудноформализуемые взаимозависимые факторы произвольного распределения. Нейросетевые классификаторы учитывают такие закономерности в данных, которые порой не могут быть учтены никаким другим классификатором [7, 100]. При этом точность классификации нейросетевых классификаторов высока и приближается к байесовской [98].

Несмотря на очевидные достоинства нейросетевого подхода при классификации, существует ряд проблем, требующих решения [7, 100]. Одной из основных проблем является необходимость экспертного обучения нейросети. При этом требует решения задача определения оптимальных параметров, задающих структуру нейросети, а также параметров ее обучения. Решение этой задачи для данных различной природы может иметь длительный итерационный характер, что для конечного пользователя может оказаться неприемлемым.

Существуют некоторые сложности практического применения нейросетей при решении практических задач классификации. В частности, применение многослойного персептрона (далее – просто нейросети) связано с решением одной из таких задач, заключающейся в определении его оптимальной топологии – количества слоев и элементов (нейронов) в них. Именно правильно выбранная топология во многом определяет перспективность использования нейросетей в том или ином случае. Минимально необходимое количество элементов нейросети позволит быстро ее обучить и получить точные результаты ее применения. Однако задача определения топологии нейросети является сложной и окончательно до сих пор не решена. В [41] предлагается неравенство для оценки числа синаптических связей N_w в виде:

$$\frac{N_y N_p}{1+\log_2\left(N_p\right)} \le N_w \le N_y\left(\frac{N_p}{N_x}+1\right)\left(N_x+N_y+1\right)+N_y, \quad (6.20)$$

где N_y – размерность выходного вектора, N_p – число примеров обучающей выборки, N_x – размерность входного вектора. Для практически значимого варианта нейросети с одним скрытым слоем число нейронов N в нем можно определить как

$$N = \frac{N_w}{N_x + N_y}, \quad (6.21)$$

При использовании выражений (6.20) и (6.21) число нейронов скрытого слоя, как правило, получается бóльшим, чем выбранное при решении практических задач эмпирически, поэтому предлагается использовать это число только в качестве рекомендации, а окончательное решение по параметрам нейросети принимать эмпирически.

Отметим два наиболее широко используемых на практике способа определения числа нейронов в выходном слое:

1. Число нейронов равно одному (рис. 14, *а*). Выход интерпретируется как вероятность принадлежности к конкретному типу (классу).

2. Число нейронов более одного. Например, оно может быть равно количеству классов (рис. 14, *б*) или представлять собой некоторый вектор, значения которого интерпретируется в более сложной процедуре использования нейросети при оценке плотности распределения [40].

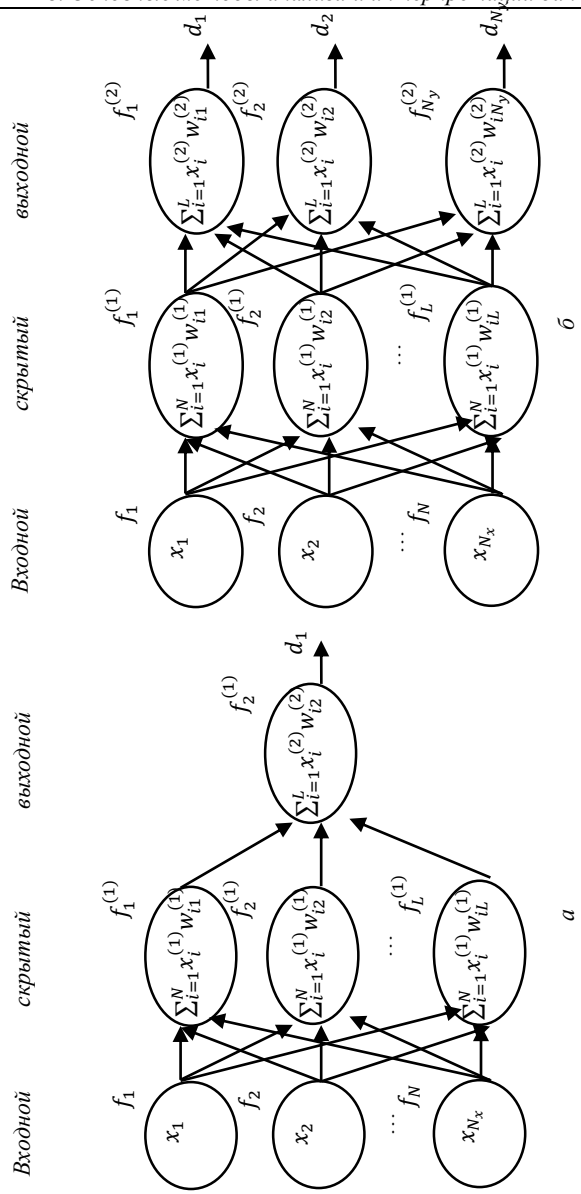


Рис. 14. Нейросетевые топологии:
а – один выход, *б* – несколько выходов

Важным и необходимым этапом практического использования любой нейросети является процесс ее обучения. Обучение нейросети в общем случае представляет собой поиск глобального минимума многомерной целевой функции путем «исследования» нейросетью многомерного пространства выборочных обучающих данных и подстройкой w_i и параметров функций активации f [7, 100].

Обучающие данные представляют собой множество входных векторов $X = \{X_i, i = 1, ..., N_p\}$ и множество известных выходных векторов $A = \{A_i, i = 1, ..., N_p\}$, где $X_i = \{x_1, x_2, ..., x_{Nx}\}$ и $A_i = \{a_1, a_2, ..., a_{Ny}\}$. В процессе обучения минимизируется значение среднеквадратической ошибки (СКО), вычисляемой согласно выражению [100]:

$$E = \frac{1}{2}\sum_{i=1}^{M}\left(A_i - Y_i\right)^2 , \qquad (6.22)$$

где $Y_i = \{y_1, y_2, ..., y_{Ny}\}$ – фактические значения, получаемые с выходов нейросети.

При программной реализации моделей нейросетей и их использовании наиболее часто в качестве алгоритма обучения применяют градиентный *алгоритм обратного распространения ошибки* или его модификации [41]. Этот алгоритм обладает устойчивой сходимостью и приемлемыми требованиями к ресурсам вычислительной техники.

6.3.4. Классификация по методу машины опорных векторов

Основная идея метода *машины опорных векторов* (*SVM* классификатора) – отображение исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве [37]. Суть работы стандартного классификатора *SVM* для случая двух классов можно представить с использованием следующего выражения:

$$f(\mathbf{x}, \mathbf{W}) = \text{sign}(g(\mathbf{x}, \mathbf{W})), \text{ где } g(\mathbf{x}, \mathbf{W}) = < \mathbf{x}, \mathbf{W} > + b, \qquad (6.23)$$

где параметры $\mathbf{W}$ (вектор весов) и b (свободный коэффициент) определяются процедурой обучения. Границы решения классификатора $g(\mathbf{x},\mathbf{W}) = 0$ представляют собой гиперплоскость порядка $L - 1$

в L-мерном пространстве (рис. 15). Для определения параметров $\mathbf{W}$ и b решается задача квадратичной оптимизации:

$$\begin{cases} -L(\lambda) = -\sum\limits_{i=1}^{\ell} \lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell} \lambda_i \lambda_j y_i y_j < x_i, x_j > \to \min\limits_{\lambda} \\ 0 \le \lambda_i \le \mathrm{C}, i = 1,\ldots,\ell \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0 \end{cases}, \quad (6.24)$$

где y – вектор меток классов пикселей обучающих выборок $y_i = \{1; -1\}$; C – управляющая константа метода решения задачи; ℓ – размер обучающих выборок. Для ее решения применяют последовательный метод активных ограничений (англ. *incremental active set method*, *INCAS*) [17]. Решение этой задачи позволяет найти вектор двойственных переменных $\lambda = (\lambda_1, \ldots, \lambda_n)$, что, в свою очередь позволяет вычислить параметры алгоритма $\mathbf{W}$ и b как

$$\mathbf{W} = \sum_{i=1}^{\ell} \lambda_i y_i x_i ; b = < \mathbf{W}, x_i > - y_i, \lambda_i > 0, i = 1 \ldots \ell, \quad (6.25)$$

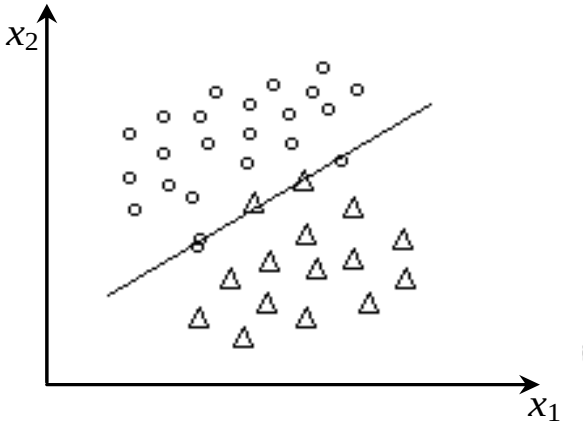


Рис. 15. Иллюстрация работы классификатора SVM для двумерного пространства

Решение задачи классификации для случая нескольких классов может быть реализовано путем множественной попарной классификации и объединения результатов (например, по мажоритарному правилу) [37]. Классификатор *SVM* отличается достаточно высокой алгоритмической сложностью, но при этом обладает высокой вычислительной эффективностью. Кроме того, его отличают высокие точность и робастность результатов при различных статистических характеристиках обучающих данных.

6.3.5. Деревья решений

Деревья принятий решений (деревья решений, англ. *decision trees*) – еще один распространенный метод контролируемой классификации, позволяющий обеспечивать поддержку принятия решений. В основе этого метода классификации лежит использование ориентированного[1] дерева как связного ациклического графа[2] (рис. 16).

Дерево решений имеет одну вершину (корень), а завершается вершинами с нулевой степенью исхода (из них не исходит ни одна дуга) – *листьями*. При этом принято, что дерево решений растет вниз (а не вверх, как настоящее дерево). Кроме того, дерево решений имеет различные метки:

– *узлы* (вершины, не являющиеся листьями) – *переменные* набора данных;

– на *дугах* (ветвях) отмечают *атрибуты* (значения переменных), от которых зависит целевая функция;

– в листьях отмечают значения целевой функции.

Если все узлы дерева имеют по две дуги, то такое дерево называют *бинарным*.

[1] Только одна вершина имеет степень захода 0 (в нее не ведут дуги – *корень дерева*), а все остальные вершины имеют степень захода 1 (в них ведет только по одной дуге).
[2] *Связность* – наличие путей между любой парой вершин, *ацикличность* – отсутствие циклов и наличие между парами вершин единственного пути.

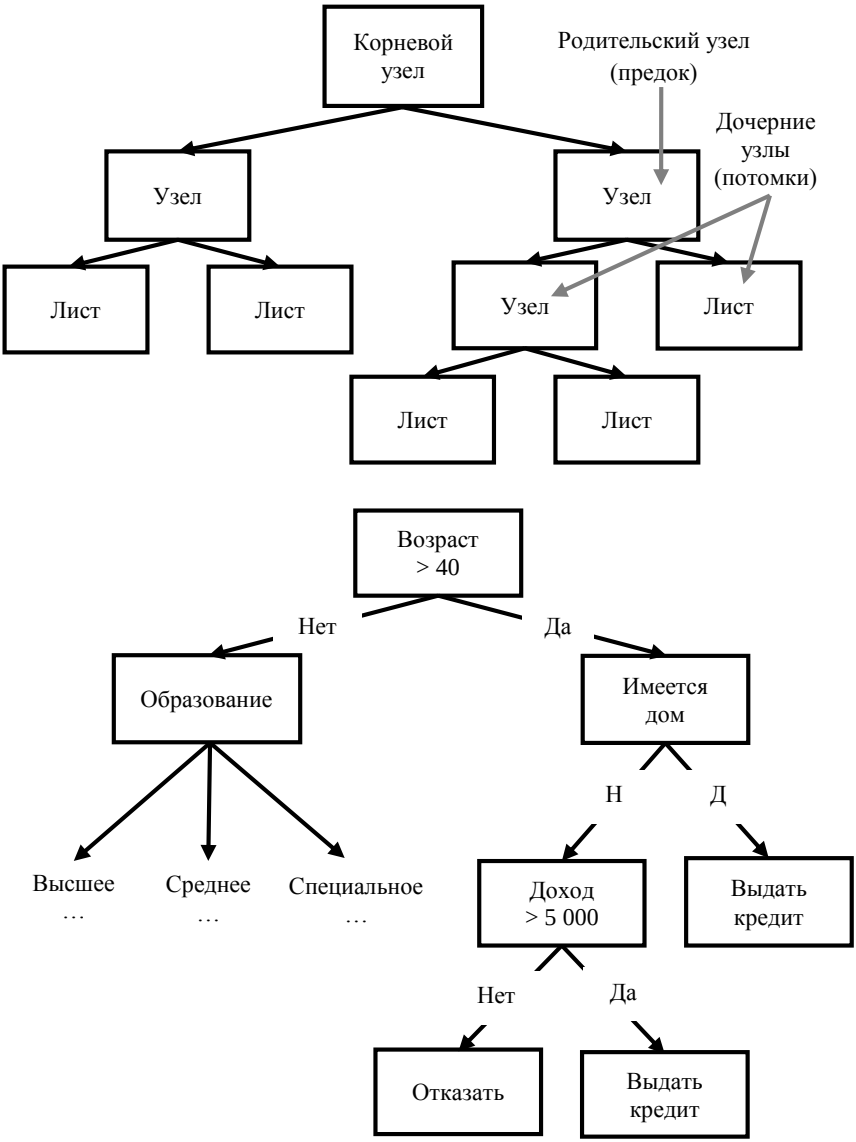


Рис. 16. Пример ориентированного дерева

Область применения деревьев решений обширна (в банковском деле –при оценке кредитоспособности клиентов, в промышленности – при контроле качества, в медицине – при диагностике заболеваний и т.п.) и может быть разделена на три класса:

– *Описание данных* (деревья не только позволяют сохранять информацию об исходных данных в компактной форме, но и могут быть логически интерпретируемыми).

– *Классификация* (возможна в случае дискретных значений целевой переменной).

– *Регрессия* (если целевая переменная имеет непрерывные значения, может быть установлена ее зависимость от независимых входных переменных, например в задаче численного прогнозирования).

Построение дерева. Пусть имеется обучающая выборка примеров $T = \{\mathbf{x}_i, f(\mathbf{x}_i) = \omega_i\}$, где $\mathbf{x}_i$ – переменные, каждой из которых соответствуют некоторый набор атрибутов (атрибут – условие перемещения по дуге) $\mathbf{Q}_i = \{\mathbf{Q}_j, j = 1, \ldots, q\}$, а ω_i – классы, которым принадлежат переменные.

Если переменная, которая проверяется в узле, принимает категориальные значения, то каждому возможному значению соответствует ветвь, выходящая из узла дерева. Если значением переменной является число, то проверяется – больше или меньше это значение некоторой константы. Иногда область числовых значений разбивают на интервалы и выполняют проверку попадания значения в один из интервалов.

Для классификации методом дерева решений следует разбить множество T на некоторые подмножества. Для этого выбирается один из признаков **x**, имеющий два и более отличных друг от друга значений $x_1, x_2, \ldots, x_n$. T разбивается на подмножества $T_1, T_2, \ldots, T_n$, где каждое подмножество T_i содержит все примеры, имеющие значение $f(\mathbf{x} = x_i)$ для выбранного признака. Эта процедура будет рекурсивно продолжаться до тех пор, пока конечное множество не будет состоять из примеров, относящихся к одному и тому же классу. Фиксируя эти преобразования в виде элементов дерева решений, будет выполнено его построение сверху вниз.

Другими словами, построение дерева решений по известным обучающим данным с использованием указанных обозначений выглядит следующим образом:

1. Выбрать переменную $\mathbf{x}_i$, поместив ее в корень дерева ($\mathbf{x}_i$ имеет n значений, что позволяет разбить T на n подмножеств).

2. Далее создается n потомков корня, каждому из которых поставлено в соответствие свое подмножество, полученное при разбиении T.

3. Повторять рекурсивно для всех i шаги 1 и 2, пока:

– в вершине не окажутся примеры из одного класса (тогда она становится листом, а класс, которому принадлежат ее примеры, будет решением листа);

– вершина оказалась ассоциированной с пустым множеством (тогда она становится листом, а в качестве решения выбирается наиболее часто встречающийся класс у непосредственного предка этой вершины).

Такой процесс построения дерева «сверху вниз» является примером наиболее распространенного поглощающего «жадного» алгоритма. Рассмотрим конкретный пример построения дерева решений для некоторого набора данных.

Предположим, нам известна некоторая статистика игры футбольной команды, а мы на ее основе хотим предсказать исход следующего матча (табл. 6).

Таблица 6

Статистика игр футбольной команды

x_1	x_2	x_3	x_4	$f(\mathbf{x})$
Соперник по турнирной таблице	*Место проведения*	*Наличие лидеров команды*	*Погодные условия*	*Результат*
Выше	Дома	На месте	Осадки	Поражение
Выше	Дома	На месте	Без осадков	Победа
Выше	Дома	Пропускают	Без осадков	Победа
Ниже	Дома	Пропускают	Без осадков	Победа
Ниже	В гостях	Пропускают	Без осадков	Поражение
Ниже	Дома	Пропускают	Осадки	Победа
Выше	В гостях	На месте	Осадки	Поражение
Ниже	В гостях	На месте	Без осадков	?

Построим дерево решений, выбрав в качестве корневого узла переменную x_1, а остальные атрибуты выберем по порядку упоминания в табл. 6 (рис. 17). Очевидно, чем меньше глубина[1] построенного дерева, чем меньше в нем количество узлов и ветвей, тем им легче оперировать на практике.

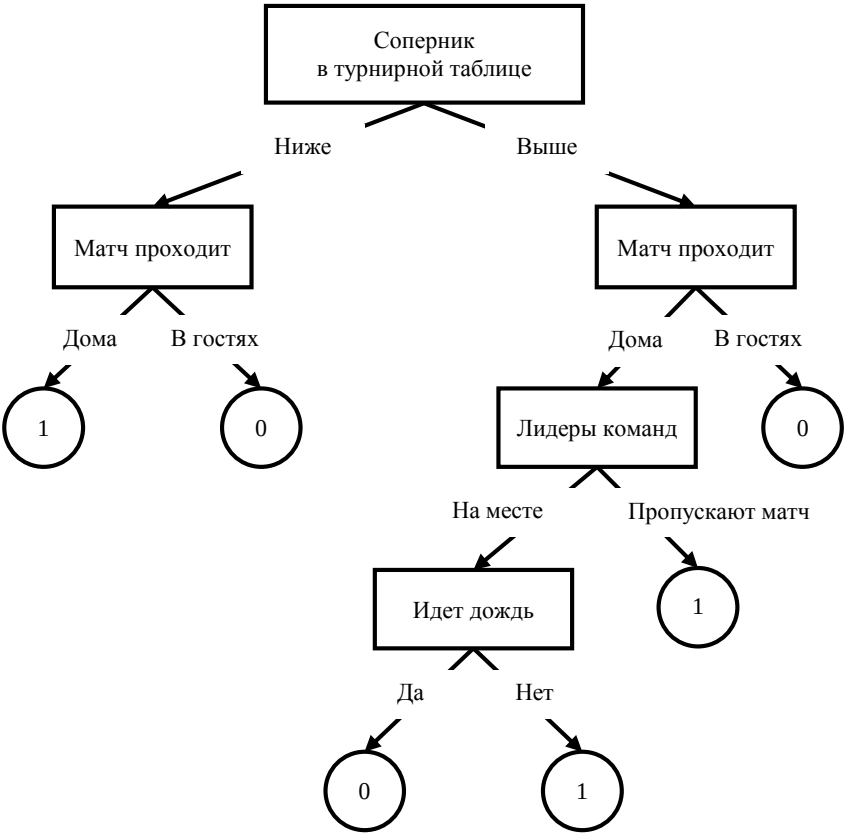


Рис. 17. Пример дерева решения с корнем x_1

[1] Максимальный уровень листа дерева – длина самого длинного пути от корня к листу.

Теперь построим аналогичное дерево решений, но порядок появления узлов зададим иначе (рис. 18).

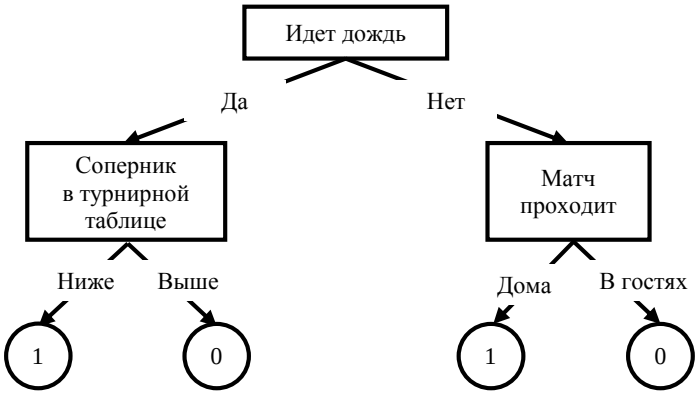


Рис. 18. Пример дерева решения с корнем x_4

Очевидно, полученное дерево (см. рис. 18) существенно проще, а его глубина в два раза меньше, чем дерева на рис. 17. При этом оно позволяет решать ту же задачу прогнозирования результатов матча по известным параметрам. Это означает, что аналогичное дерево решений может быть построено для одних и тех же исходных данных различными способами, выбирая очередность для той или иной переменной $\mathbf{x}_i$ и ее атрибутов.

Поэтому именно критерий выбора очередной переменной при построении дерева решений отличает наиболее распространенные алгоритмы построения деревьев решений:

– *Алгоритм ID3*: выбор атрибута происходит на основании *прироста* информации (англ. *Gain*) либо на основании индекса Джини (англ. *Gini*).

– *Алгоритм C4.5*: модифицированная версия алгоритма *ID3*, выбор атрибута происходит на основании *нормализованного прироста* информации (англ. *Gain Ratio*).

– *Алгоритм CART*: для бинарных деревьев, выбор атрибута зависит от отношений числа примеров в правом и левом потомке к общему числу примеров.

Следует отметить, что задача построения наилучшего, оптимального дерева не может быть решена данными методами и требует построения всех возможных вариантов деревьев решений и полного их перебора (т.е. является *NP*-полной) [22].

Достоинства и недостатки. Использование деревьев решений в сравнении с другими распространенными методами классификации, прогнозирования или регрессионного анализа имеет несколько существенных достоинств:

– *Не требуется предварительной обработки данных*. Например, не нужны нормализация, добавление фиктивных переменных, удаление пропущенных данных. Метод способен работать как с категориальными, так и с интервальными переменными.

– *Имеется возможность интуитивного понимания и интерпретации*. Правила передвижения по узлам деревьев могут иметь четкую логику («белый ящик») и могут быть формализованы даже там, где это затруднительно сделать эксперту. Отметим, что нейросеть такой возможности пользователю, как правило, не предоставляет («черный ящик»).

– *Высокие надежность и точность*. Сравнительно высокие надежность и точность модели могут быть статистически оценены. Отсутствие априорных предположений о законах распределения данных относит их к непараметрическим, устойчивым к данным различных априори неизвестных законов распределения.

– *Высокая вычислительная эффективность*. Метод отличает быстрый процесс обучения (построения дерева), а также высокая вычислительная эффективность обработки значительных объемов данных за счет простоты структуры данных дерева решений.

Вместе с важными достоинствами метода отмечают и его недостатки:

– *Проблематичность построения оптимального дерева решений*. Построение и поиск такого дерева решений является *NP*-полной задачей, сложно разрешимой на практике. Поэтому практическое построение деревьев решений связано с применением эвристических «жадных» алгоритмов, оптимальных только в каждом узле дерева, но неоптимальных для дерева в целом. При этом требуется

обеспечить непростой баланс между точностью и сложностью дерева, уделять внимание опасности переобучения, для чего применять дополнительные алгоритмы регулирования глубины или упрощения дерева (англ. *pruning, reduction*).

– *Вероятность построения избыточно сложного дерева*. Возможны случаи, при которых применение традиционных алгоритмов построения дерева решений приведет к описанию модели «сложным» путем и непомерно большому дереву (например, когда число возможных атрибутов велико, а не просто «да» / «нет») [21]. В этом случае потребуются дополнительная проработка постановки задачи и формирование иных суждений о предметной области.

– *Вероятность ошибок при построении дерева*. Ключевым элементом алгоритма построения дерева является порядок выбора очередной переменной при построении очередного узла. В случае, если набор исходных данных включает категориальные переменные, больший информационный вес априори присваивается тем переменным, которые имеют большее количество уровней [95].

Таким образом, метод с использованием деревьев решений наиболее эффективно может быть применен для задач с дискретными (категориальными) значениями с четким набором отличных атрибутов, а также в тех случаях, где важно понимать логику получения и интерпретации результатов.

Примеры алгоритмов. Выше изложены базовые принципы построения деревьев решений, на которых основано подавляющее большинство применяемых на практике алгоритмов. Основное отличие таких алгоритмов друг от друга заключается в способах определения очередности для той или иной переменной и ее атрибутов при построении очередного узла дерева.

Очевидно, наиболее удачным будет считаться атрибут, который позволит получить такие подмножества данных, которые будут принадлежать в подавляющем большинстве элементов к одному классу.

Алгоритм ID3. Алгоритм ID3 (англ. *Iterative Dichotomizer,* итеративный дихотомайзер), предложенный Д. Куинланом, определяет очередность переменной и ее атрибутов через их информационную значимость (информационную энтропию) [30]. Для этого следует

найти энтропию всех неиспользованных признаков и их атрибутов относительно тестовых экземпляров и выбрать тот, для которого энтропия минимальна (а информативность – максимальна).

Энтропию при условии неравновероятных событий p_i находят по известной формуле Шеннона:

$$I = -\sum_i p_i log_2 p_i, \qquad (6.26)$$

где I – количество информации в битах, которую можно передать, используя m элементов в сообщении при n буквах в алфавите, причем $p_i = m/n$.

В случае с деревом решений m – число значений целевой функции $f(\mathbf{x}_i)$ для $\mathbf{x}_i$ (поэтому корректнее использовать запись m_i), n – общее число элементов (записей) исходного набора (множества T). Тогда энтропия множества T по отношению к свойству $S = f(\mathbf{x}_i)$, которое может принимать s значений, может быть найдена как

$$\text{H (T, } S) = -\sum_{i=1}^{s} \frac{m_i}{n} \log_2 \frac{m_i}{n} . \qquad (6.27)$$

Если свойство S бинарное (т.е. целевая функция $f(\mathbf{x}_i)$ может принимать только два значения), то запись для энтропии будет выглядеть так:

$$\text{H (T, } S) = -\frac{m}{n} \log_2 \frac{m}{n} - \frac{n-m}{n} \log_2 \frac{n-m}{n} . \qquad (6.28)$$

В этом случае, если вероятность появления S равна 0,5 (т.е. равновероятно), то энтропия равна 1 (т.е. нужен 1 бит информации для ее кодирования). Если же появление S не будет равновероятно, то энтропия будет меньше, а значит закодировать информацию из T именно для такого $S = f(\mathbf{x}_i)$ будет более эффективно. Этот пример показывает, что выбор признака $\mathbf{x}_i$ следует осуществлять так, чтобы соответствующая ему энтропия стала минимально возможной.

В общем случае энтропия будет различной для различных потомков узла, поэтому итоговый результат нужно считать с учетом того, сколько исходов осталось в рассмотрении в каждом из потомков. Например, если множество T со свойством $S = f(\mathbf{x}_i)$ классифицировано признаком $\mathbf{x}_i$ с атрибутом Q, имеющим q возможных значений, то прирост информации определяется как

$$\text{Gain(T, } S) = \text{H (T, } S) - \sum_{k=1}^{q} \frac{|T_k|}{|T|} \text{H}(T_k, S), \qquad (6.29)$$

или, используя более обобщенную запись имеем выражение:

$$\text{Gain}(T, S) = \text{Info}(T) - \text{Info}_S(T), \qquad (6.30)$$

где T_k – множество элементов T, для которых атрибут Q имеет значение k. На каждом шаге алгоритм выбирает тот атрибут Q, для которого это прирост информации максимален.

Результат разницы в выражении (6.30) может быть найден в этом алгоритме альтернативно с использованием *индекса Джини*, упомянутого выше.

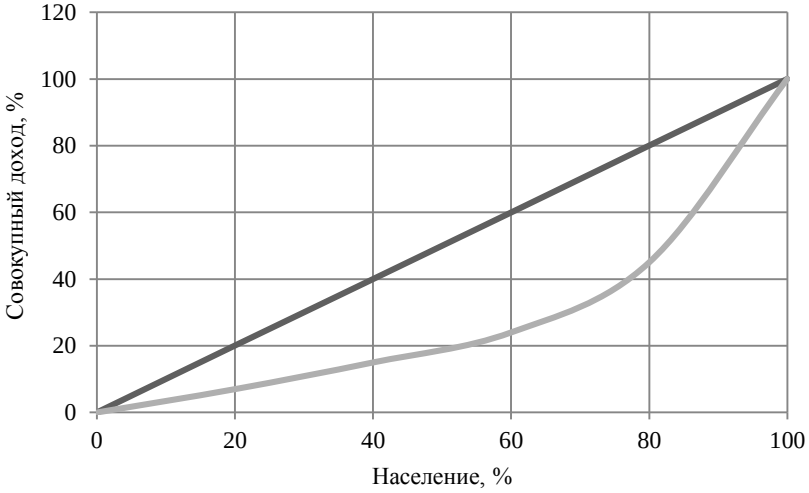


Рис. 19. Иллюстрация зависимости двух переменных Info(T) и Infos(T) и кривой Лоренца, используемых в расчете индекса Джини

Индекс Джини начали применять для оценки неравномерности распределения некоторого изучаемого признака (например, годового дохода) для различных социальных групп и широко используют в экономических, социальных и демографических исследованиях [5]. В алгоритме ID3 для его расчета применяется зависимость двух величин (например, Info(T) и Infos(T)), упорядоченных по возрастанию и обычно нормированных в процентах (рис. 19). Индекс Джини численно равен площади под кривой Лоренца, которая может принимать значения от 0 до 1.

Алгоритм C4.5. Алгоритм C 4.5 также предложен Д. Куинланом в развитие алгоритма ID3 и расширяет его возможности в части [5]:

– дополнительной функциональности по отсечению ветвей во избежание проблемы переобучения;

– построения дерева из обучающей выборки с пропусками данных (отсутствуют значения для некоторых атрибутов);

– работы не только с дискретными, но и с непрерывными числовыми атрибутами.

Основной недостаток алгоритма ID3 – склонность к переобучению. Действительно, при большом количестве возможных значений признака (возможно, несущественно отличающихся друг от друга) будет построено большое число ветвей с атрибутами этого признака. В крайнем случае число листьев такого дерева может быть эквивалентно числу имеющихся примеров обучающего множества. Очевидно, для задач классификации такое дерево решений будет бесполезным. Более того, поиск верного признака $\mathbf{x}_i$ при построении следующего узла в таком дереве также проблематичен – в (6.27) $\log_2(1) = 0$, поэтому в (6.30) прирост информации для всех $\mathbf{x}_i$ максимален.

Во избежание указанных проблем в алгоритме предусмотрено нормирование при расчете прироста информации путем расчета дополнительного показателя с оценкой потенциальной информации, созданной при разбиении множества T на n подмножеств T_i:

$$\text{Split\_Info(T)} = -\sum_{k=1}^{n}\left[\frac{|T_k|}{|T|}\log_2\left(\frac{|T_k|}{|T|}\right)\right] \qquad (6.31)$$

Критерий прироста информации (6.30) с использованием выражения (6.32) модифицирован следующим образом:

$$\text{Split\_Info}(S)=\frac{\text{Gain (T},S)}{\text{Split}_{\text{Info(T)}}}. \qquad (6.32)$$

С учетом того, что способ расчета критерия прироста информации (6.32) учитывает не только прирост, но и оценку ее потенциальной информативности («полезности»), это способствует выбору более удачного атрибута, построению менее избыточного дерева решений и улучшению классификации.

Еще одной отличительной особенностью алгоритма C4.5 по отношению к ID3 является наличие адаптации к присутствию пропусков данных в исходном обучающем множестве.

Вообще проблема пропусков в данных является для практики очень важной, для ее решения на этапе предварительной обработки данных применяют различные приемы и технологии (см. разд. 6.1), позволяющие представить данные для обработки методами *Data Mining* без столь очевидных недостатков.

Тем не менее для полноты представления алгоритма отметим суть вышеуказанной особенности работы с пропусками в данных.

В алгоритме C4.5 предполагается, что экземпляры обучающего множества с неизвестными значениями имеют статистическое распределение соответствующего признака согласно относительной частоте появления известных значений. Для фиксации этой характеристики введен параметр F, который представляет собой число наблюдений в наборе исходных данных с известным значением данного признака, отнесенное к общему числу наблюдений. Тогда модифицированный для работы с пропущенными значениями критерий прироста информации будет иметь вид:

$$\text{Gain}(S) = F \times [\text{Info}(T) - \text{Info}_S(T)]. \qquad (6.33)$$

Алгоритм CART. CART (англ. *Classification and Regression Tree*, дерево классификации и регрессии), предложенный в 1984 г., предназначен для построения бинарного дерева решений (каждый узел имеет только две ветви-потомка) [10].

Основными отличиями алгоритма *CART* от алгоритмов семейства ID3 являются:

- бинарное представление дерева решений;
- функция оценки качества разбиения;
- механизм отсечения дерева;
- алгоритм обработки пропущенных значений;
- возможность построения деревьев регрессии.

На каждом шаге построения дерева правило, формируемое в узле, делит заданное множество примеров на две части, в которых выполняется (первая часть) и не выполняется (вторая часть) решающее правило. При этом производится перебор всех признаков,

на основе которых может быть построено разбиение, и выбирается тот, который максимизирует значение некоторого показателя. Например, таким показателем может быть

$$\mathrm{H(T)} = 2\,P_L P_R \sum_{j=1}^{q}\left[P_L(\omega_j) - P_R(\omega_j)\right], \qquad (6.34)$$

где P_L и P_R – отношение числа примеров в левом и правом потомках к их общему числу в обучающем множестве T, $P_L(\omega_j)$ и $P_R(\omega_j)$ – отношение числа примеров класса ω_i в левом и правом потомках соответственно к их общему числу в каждом из них.

Также в качестве такого показателя применяют выражение, основанное на использовании индекса Джини. Если множество T содержит данные *n* классов, тогда индекс Джини определяется как

$$\mathrm{Gini(T)} = 1 - \sum_{i=1}^{n} p_i^2, \qquad (6.35)$$

где p_i – вероятность (относительная частота) класса ω_i в T. Для узла бинарного дерева с двумя ветвями-потомками после ряда преобразований и упрощений показатель «успешности» разбиения множества рассчитывается как

$$G_{\mathrm{split}} = \frac{1}{L}\sum_{i=1}^{n} l_i^2 + \frac{1}{R}\sum_{i=1}^{n} r_i^2, \qquad (6.36)$$

где L, R – число примеров соответственно в левом и правом потомке, l_i и r_i – число экземпляров ω_i-го класса в левом и правом потомке. Лучшим будет то разбиение, для которого величина G_{split} будет максимальной.

Алгоритм CART предусматривает построение не только классификационных деревьев, но и регрессионных. Для этого процесс реализуется аналогично, но вместо меток классов в листьях будут располагаться числовые значения.

6.4. Неконтролируемая классификация (кластеризация)

Любой способ кластеризации требует максимизации межклассовых расстояний и минимизации внутриклассовых расстояний. Выделяют целый ряд способов кластеризации, различающихся критерием построения кластеров. Критериями могут быть:

– связность;
– центроидность;

– распределения;
– плотность;
– прочие.

Неконтролируемая классификация (англ. *clustering*, кластеризация) позволяет разбить исходный набор данных на конечное количество однородных в некотором смысле кластеров. Неконтролируемая классификация реализуется методами кластерного анализа и позволяет выявлять свойства данных группироваться около некоторых значений (центров). В общем концепцию кластерного анализа многомерных данных можно определить как распределение всех возможных точек (объектов) *P*-мерного пространства признаков по соответствующим кластерам [70], т.е. разбиение пространства признаков на взаимно непересекающиеся области, каждая из которых соответствует некоторому кластеру C_i с центром μ_i' (i = 1, …, M'). При этом объекты одного кластера группируются в признаковом пространстве компактно, т.е. расстояние между объектами одного кластера меньше, чем расстояние между объектами различных кластеров.

Среди методов кластерного анализа можно выделить такие, как *методы итерационной оптимизации* (англ. *Iterative Self Organizing Data Analysis Technique, ISODATA* метод) [50], *методы иерархической кластеризации* [70], *анализ пиков гистограмм* [103].

Метод ISODATA осуществляет итерационную оценку структуры исходных многомерных данных. На каждой итерации определяется новое уточненное пространство кластеров C_i и их центров μ_i' (рис. 20, *б, в*), исходя из условия минимума расстояния между точками и центрами каждого кластера. Для этого на каждом шаге определяются ближайшие к центрам кластеров элементы изображения и вычисляются новые центры, также осуществляется слияние близких кластеров на основе заданных критериев. В качестве совокупного критерия точности кластеризации метод *ISODATA* использует *суммарную квадратичную ошибку S*, определяемую как

$$S = \sum_{i=1}^{M'} \sum_{x \in C_i} d\left(x, \mu_i'\right). \qquad (6.37)$$

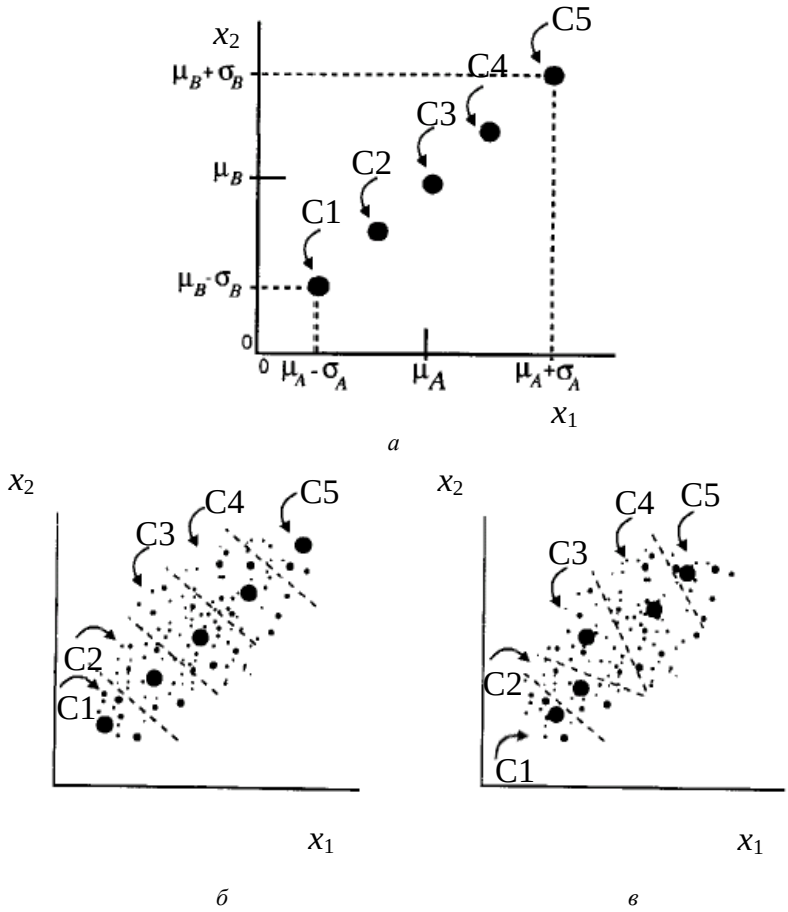


Рис. 20. Иллюстрация процедуры кластерного анализа
(итерационный метод ISODATA):
а – начальное размещение центров; *б* – первая итерация; *в* – вторая итерация

Таким образом, качество кластеризации будет наилучшим при наименьшем значении S, т.е. при минимальном значении суммы расстояний от всех точек кластеров C_i до их центров μ'_i ($i = 1, \ldots, M$). Критерием останова итерационного процесса кластеризации методом ISODATA является заданное количество итераций и порог

суммарной квадратичной ошибки S (6.37). При этом в качестве меры близости между точками кластера C_i и его центром μ'_i могут быть использованы $L1$ или Евклидова метрика расстояния. $L1$ – расстояние между P-мерными вектором $\mathbf{x}$ и центром μ'_i – определяется выражением:

$$d^{L1}(\mathbf{x}, \mu'_i) = \sum_{\nu=1}^{P} \left| x_\nu - \mu'_{i\nu} \right|, \tag{6.38}$$

а Евклидово расстояние – выражением:

$$d^{E}(\mathbf{x}, \mu'_i) = \sqrt{\sum_{\nu=1}^{P} (x_i - \mu'_{i\nu})^2}\ , \tag{6.39}$$

Отмечают, что метрика (6.39) более предпочтительна по критерию точности по сравнению с метрикой (6.38), поэтому ее использование при кластеризации более целесообразно.

Алгоритм *k-means* (алгоритм *k*-внутригрупповых средних, *k*-средних). Основан на минимизации функционала Q суммарной выборочной дисперсии, характеризующего разброс элементов относительно центров кластеров:

$$\mathrm{Q} = \sum_{\mathrm{i}} |X_i| \sum_{\mathrm{x} \in X_i} d(\mathbf{x}, C_{\mathrm{i}}) \to \min, \tag{6.40}$$

где $\mathrm{C}_i = \frac{1}{|X_i|} \sum_{\mathrm{x} \in \mathrm{X_i}} \mathrm{x}$ – центр кластера X_i .

Алгоритм выполняется итерационно (рис. 21). На каждой итерации находятся центры кластеров, а также производится разбиение выборки на кластеры. Вычисления продолжаются, пока функционал Q не перестанет уменьшаться.

Порядок выполнения алгоритма следующий:

1. Выделяются начальные центры кластеров $\mathrm{C}_1^{(0)}, \ldots, \mathrm{C}_m^{(0)}$, $k = 0$.
2. Выборка разбивается на m кластеров по принципу ближайшего соседства, и получаются некоторые кластеры $X_1^{(k)}, \ldots, X_m^{(k)}$.
3. Находим новые центры кластеров как

$$C_i^{(k+1)} = \frac{1}{|X^{(k)}|} \sum_{\mathrm{x} \in \mathrm{X_i^{(k)}}} \mathrm{x}., \tag{6.41}$$

4. Если не выполняется условие $\mathrm{c}_i^{(k+1)} = \mathrm{c}_i^{(k)}$ для всех $k = 1, \ldots, \mathrm{m}$, то переходим на шаг 2.

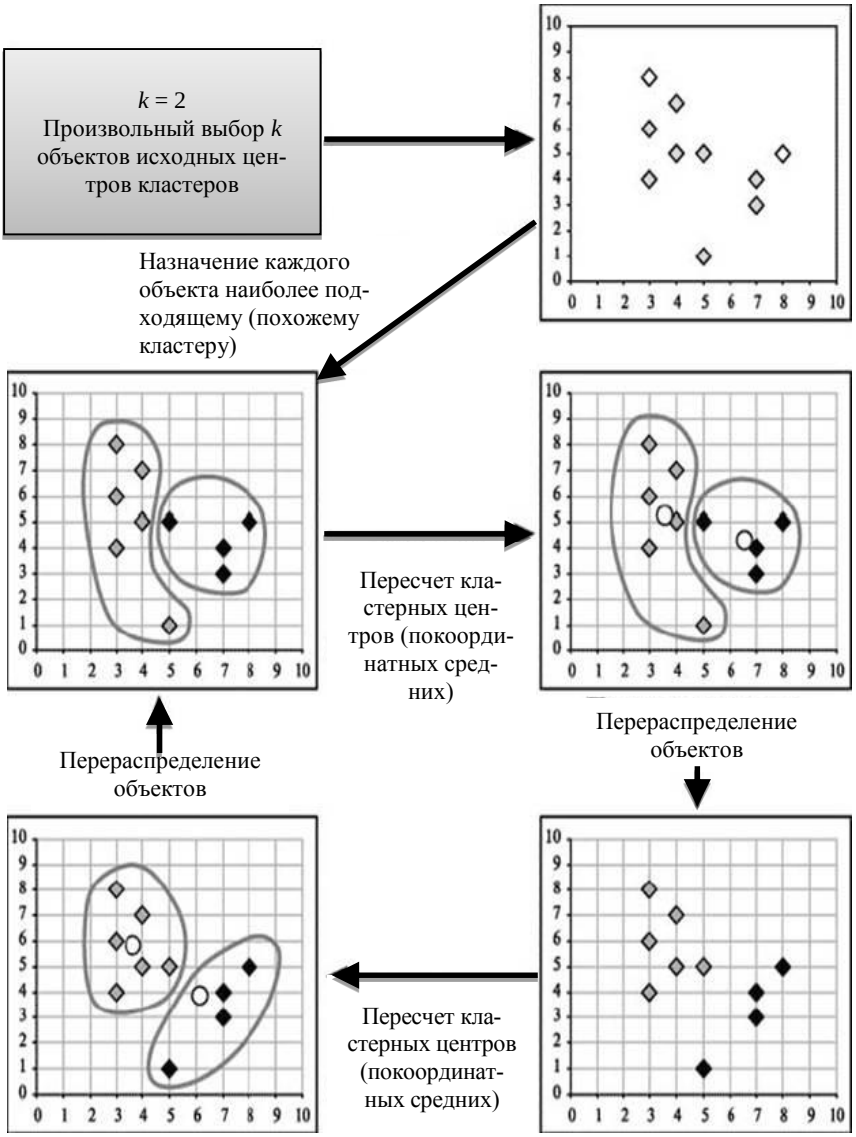


Рис. 21. Графическая иллюстрация работы алгоритма *k*-средних

Преимуществом алгоритма *k*-средних являются быстрота и простота реализации. К его недостаткам можно отнести неопределенность выбора числа и исходных центров кластеров, неверный выбор которых может вести к неудовлетворительным результатам работы итерационной процедуры или чрезмерному количеству необходимых итераций. Кроме того, алгоритм может быть чувствителен к выбросам, которые могут существенно искажать среднее. В этом случае может быть применена модификация алгоритма с использованием *k*-медианы, имеющая, однако, бóльшую вычислительную сложность, препятствующую работе с большими наборами данных.

6.5. Регрессия

6.5.1. Понятие регрессии

Термин *регрессия* (англ. *regression*, обратное движение) введен в 1886 г. антропологом Ф. Гальтоном при изучении статистических закономерностей наследственности роста, которые позволили выявить *линейную связь* между средним ростом отцов и их сыновей. Причем рост отцов в среднем оказался выше, чем средний рост сыновей, что позволило исследователю сделать вывод о «*...регрессии к посредственности*…», т.е. о снижении роста в популяции к ее среднему значению [18]. Вместе с подобным «прикладным» появлением термина «регрессия» справедливо отмечают и еще одно дополнительное принципиальное основание. Термин *регрессия* также отражал новизну в очередности этапов исследования: сначала собраны данные, а затем по ним угадана модель зависимости, в то время как традиционно данные использовались для проверки априори построенных теоретических моделей. Позднее термин «регрессия» закрепился как канонический, отражающий методы восстановления зависимостей между переменными.

Сегодня под *регрессией* принято понимать зависимость среднего значения какой-либо величины от некоторой другой величины или от нескольких величин, а под *регрессионным анализом* – методы исследования взаимосвязи переменных. Если пространство объектов

обозначить как X и множество возможных ответов как Y, то существует неизвестная целевая зависимость y^*: $X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)^\ell_{i=1}$, $y_i = y^*(x_i)$. Требуется построить алгоритм, который принято называть *функцией регрессии* f: $X \to Y$, аппроксимирующий целевую зависимость y^*. Наконец, задачу обучения по прецедентам (x_i, y_i), позволяющую найти регрессионную зависимость y^*, называют *восстановлением регрессии* [66].

Следует отметить, что регрессионный подход не только позволяет выявлять зависимости между признаками, но и решает задачи прогнозирования (например, временного ряда на основе ретроспективных данных), а также задачи классификации (например, путем использования кривой регрессии в качестве разделяющей плоскости между классами), примеры которых рассмотрены в разд. 3.1.4.

6.5.2. Основные этапы регрессионного анализа

Выделяют следующий обобщенный многоэтапный подход к регрессионному анализу:

5. Формулировка задачи. На этом этапе формируются предварительные гипотезы о зависимости исследуемых явлений.
6. Определение зависимых и независимых (объясняющих) переменных.
7. Сбор статистических данных. Данные должны быть собраны для каждой из переменных, включенных в регрессионную модель.
8. Формулировка гипотезы о форме связи (простая или множественная, линейная или нелинейная).
9. Определение функции регрессии (заключается в расчете численных значений параметров уравнения регрессии).
10. Оценка точности регрессионного анализа.
11. Интерпретация полученных результатов.
12. Полученные результаты регрессионного анализа сравниваются с предварительными гипотезами. Оцениваются корректность и правдоподобие полученных результатов.
13. Предсказание неизвестных значений зависимой переменной.

6.5.3. Методы восстановления регрессии

Существует целый ряд методов восстановления регрессии. Примерами таких методов являются непараметрическая регрессия с ядерным сглаживанием, линейная и нелинейная регрессия, опорные векторы, регрессионные деревья решений, многомерные адаптивные регрессионные сплайны (англ. *Multivariate Adaptive Regression Splines*, MARS), мультилинейная интерполяция (англ. *Multilinear Interpolation*), радиальные базисные функции (англ. *Radial Basis Functions*), робастная регрессия (англ. *Robust Regression*), каскадная корреляция (англ. *Cascade Correlation*) и многие другие способы и подходы. Детали каждого из этих подходов могут быть найдены в специальной литературе.

Вместе с тем большинство исследуемых на практике зависимостей может быть аппроксимировано стандартными нелинейными математическими функциями, для построения которых широко используют *метод наименьших квадратов* (МНК). Его суть заключается в следующем.

Пусть задана модель регрессии – параметрическое семейство функций $g(x, \alpha)$, где $\alpha \in R^p$ – вектор параметров модели. Определим функционал качества аппроксимации целевой зависимости на выборке X^ℓ как сумму квадратов ошибок:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i)^2. \qquad (6.42)$$

Обучение МНК состоит в том, чтобы найти вектор параметров α^*, при котором достигается минимум среднего квадрата ошибки на заданной обучающей выборке X^ℓ:

$$\alpha^* = \arg\min_{\alpha \in R^p} Q(\alpha, X^\ell). \qquad (6.43)$$

Стандартный способ решения этой оптимизационной задачи – воспользоваться необходимым условием минимума. Если функция $g(x, \alpha)$ достаточное число раз дифференцируема по α, то в точке минимума выполняется система p уравнений относительно p неизвестных:

$$\frac{\partial Q}{\partial \alpha}(\alpha, X^\ell) = 2 \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i) \frac{\partial g}{\partial \alpha}(x_i, \alpha) = 0. \qquad (6.44)$$

Решение системы таких уравнений (методами Гаусса или Крамера) позволяет найти необходимые коэффициенты α в уравнении регрессии.

6.6. Ассоциация

Еще одним методом, который выделяют в *Data Mining*, является *ассоциация* – выявление закономерностей между связанными событиями. Следует отметить, что в *Data Mining* методы *ассоциаций, поиска ассоциативных правил* (англ. *association rule induction*) принято выделять в отдельный класс, хотя по сути они являются частью более общих методов неконтролируемой классификации [72]. Примером ассоциативной закономерности (*ассоциативного правила*) служит правило, указывающее, что из события *X* следует событие *Y*.

Впервые эта задача апробирована в торговой отрасли при нахождении типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее называют *анализом рыночной корзины* (англ. *market basket analysis*). *Ассоциативные правила* помогают выявлять группы товаров, как правило, приобретаемые совместно. Знание этих правил позволяет соответствующим образом размещать товары на прилавках, стимулируя интенсивность их продаж. Задача поиска ассоциативных правил актуальна и в сфере обслуживания, где интерес представляет то, какими услугами клиенты предпочитают пользоваться в совокупности. В медицине интересной представляется возможность выявлять наиболее сочетаемые болезни и симптомы, требующиеся для определения диагноза.

Наиболее известным и широко используемым (в ритейле и консалтинговых компаниях) примером практической реализации поиска ассоциативных правил является алгоритм *Априори* (англ. *Apriori, A priori*) [3].

Продемонстрируем работу алгоритма *Априори* на примере задачи анализа рыночной корзины, оперируя привычными на практике табличными записями. Одним из наиболее распространенных типов данных в этой задаче являются *транзакционные данные* (англ. *transaction data*), отражающие в таблице базы данных взаимодействие клиентов и магазина (табл. 7).

В примере представлено 20 записей, полученных при продаже семи видов товаров. В большом супермаркете аналогичные реальные

данные могут быть представлены миллионами записей в месяц, а число групп товаров может достигать тысячи и более.

Таблица 7

Пример транзакционных данных клиентов и магазина

ID	Яблоки	Пиво	Сыр	Финики	Яйца	Рыба	Конфеты
Условное обозначение	*a*	*b*	*c*	*d*	*e*	*f*	*g*
1	1	1		1			1
2			1	1	1		
3		1	1			1	
4		1				1	
5					1		1
6						1	
7	1			1			
8						1	
9			1		1		
10		1					1
11					1		1
12	1						
13			1			1	
14			1			1	
15							
16				1			
17	1					1	
18	1	1	1	1			
19	1	1		1			1
20					1		

При анализе рыночной корзины оперируют утверждениями типа:

«*Если* корзина содержит *X* и *Y*,
то она также содержит и *Z*». (6.45)

Подобное правило сопровождается двумя метриками:

1. *Достоверность* или *точность* (англ. *confidence* или *accuracy*). Отражает, как часто при справедливости выражения в части «если», оно также справедливо в части «то».

2. *Покрытие* или *поддержка* (англ. *coverage* или *support*). Отражает долю выражений в части «если» в базе данных.

Для утверждения (6.45) в случае X = пиво, Y = сыр, Z = рыба в рассматриваемом примере достоверность составляет 1/2 = 50%, а поддержка 2/20 = 10%.

Какие же утверждения конструкции «если / то» следует признать интересными и достойными внимания? Логично к таким утверждениям отнести такие, которые справедливы чаще, чем случайно.

Например, статистический анализ потребительской корзины показывает, что 10% в ней занимает хлеб, а 4% – стиральный порошок. Это означает, что вероятность встретить в такой корзине хлеб P(хлеб) = 0,1, а порошок – P(порошок) = 0,04. Какова же вероятность встретить эти товары в корзине совместно? Очевидно, P(хлеб | порошок) = P(хлеб) × P(порошок) = 0,1×0,04 = 0,004. На практике, анализируя, например, тысячу таких потребительских корзин, лишь в четырех из них ожидается увидеть хлеб и порошок. Поэтому, если в действительности такую комбинацию удастся обнаружить в 20 или 30 случаях из тысячи, это будет сюрпризом, достойным внимательного анализа (между товарами в магазине или торговой сети есть какая-то неочевидная связь).

Каким же образом в базе данных можно найти наиболее интересные для анализа правила (ассоциации)? Очевидно, такие правила не только будут отличаться более высокой *точностью*, но и должны иметь значительную *поддержку* (обеспечивая воспроизводимость найденного правила на данном наборе данных). Однако, задача поиска таких ассоциативных правил на практике осложняется высокой вычислительной сложностью. Можно оперировать 500 000 000 возможными правилами типа (6.45), и если база данных состоит из 20 000 000 записей, то число возможных вычислительных операций может доходить до 10^{16}. Проверка уровней *точности* и *поддержки* в таком массиве – нетривиальная задача.

Поэтому для решения подобных задач был предложен несложный, вычислительно эффективный алгоритм *Априори*, позволяющий находить интересные ассоциативные связи в данных. Алгоритм оперирует не собственно ассоциативными правилами типа (6.45), а *множествами* (англ. *itemsets*), элементы которых определенным образом ассоциированы между собой.

6.6.1. Описание алгоритма

Рассмотрим описание алгоритма *Априори*, основываясь на утверждении, что любое множество, содержащееся в некотором часто встречающемся множестве (ЧВМ), является ЧВМ. Другими словами,

$$\text{если } Y \subseteq X \text{ и } \mathrm{P}(X) \geq c, \text{ то } \mathrm{P}(Y) \geq c. \qquad (6.46)$$

При этом k-множеством будем называть множество, состоящее из k элементов.

Обозначим через L_k множество всех часто встречающихся k-множеств. Объединение L_k по всем k дает все искомое множество ЧВМ. Построение L_k выполняется по шагам.

Сначала находится L_1 (множество одноэлементных ЧВМ). Затем для каждого фиксированного $k \geq 2$, используя найденное множество L_{k-1}, определяется L_k. Процесс завершается, как только k станет больше максимального количества элементов.

Определение L_k при известном L_{k-1} выполняется в два шага:

1) генерируются множества-кандидаты C_k;

2) затем из этого множества исключаются лишние элементы.

Полученное таким образом множество и будет равно L_k.

Генерация множества кандидатов. Множество кандидатов C_k составляется путем *слияний* всех *допустимых пар* $l_1, l_2 \in L_{k-1}$.

Сокращение. Множество кандидатов C_k содержит все множества из L_k, но содержит дополнительно и лишние множества, не являющиеся ЧВМ. Чтобы получить L_k, необходимо лишь исключить такие множества. Для этого необходимо для каждого набора из C_k посчитать количество его повторений в базе данных и исключить это множество, если число повторений меньше заданного порога (уровня поддержки). Но такой подсчет – довольно трудоемкая процедура, так как C_k может иметь очень большой размер. Поэтому рекомендуется сначала произвести его предварительную очистку следующим образом.

Пусть l – некоторое множество из C_k (следовательно, он состоит из k элементов). Если l – ЧВМ, то в соответствии с утверждением (6.46) все подмножества l, состоящие из $k - 1$ элементов, должны

быть также ЧВМ, т.е. принадлежать множеству L_{k-1}. Поэтому если хотя бы одно множество, полученное из l удалением одного элемента, не принадлежит L_{k-1}, то l не может являться ЧВМ и должно быть исключено из C_k.

Допустимая пара и их слияние. Пусть $l_1, l_2 \in L_{k-1}$ – два множества из множества L_{k-1}. Обозначим через $l_i[j]$ j-й элемент в множестве l_i. Например, $l_1[k-2]$ – это предпоследний элемент в l_1. Предполагается, что на исходном множестве элементов задано некоторое отношение порядка ‘<’ (например, по номерам элементов), и в наборе l_i элементы отсортированы в соответствии с данным отношением порядка.

Пара l_1, l_2 – допустимая для слияния, если:

$$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1]). \quad (6.47)$$

Условие $(l_1[k-1] < l_2[k-1])$ гарантирует, что дубликатов в множестве C_k не будет. Слиянием $u(l_1, l_2)$ допустимых наборов l_1, l_2 будет множество, состоящее из элементов $l_1[1], l_1[2], \ldots, l_1[k-1], l_2[k-1]$.

6.6.2. Пример исполнения алгоритма

Для демонстрации работы алгоритма *Априори* приведем его краткое пошаговое описание.

Шаг 1. Найти все одноэлементные ЧВМ множества.

Шаг 2. For (k = 2; while L_{k-1} <> ‘ ‘; k++)

{

Шаг 3. C_k = *apriori_gen*(L_{k-1})

Шаг 4. Для каждого c в C_k, $c.count$ = 0

Шаг 5. Для всех записей r в БД

{

Шаг 6. C_r = *subset*(C_k, r); For each c in C_r, $c.count$++

}

Шаг 7. Set L_k := all c in C_k whose *count* >= *minsup*

Шаг 8. Вывод $\bigcup_k L_k$ } // возвращаем все множества L_k.

Обозначим с помощью *minsup* – параметр минимального уровня *поддержки*. Пусть *minsup* = 4 (т.е. 20%).

Шаг 1. Находим все одноэлементные ЧВМ с заданным *minsup* (т.е. записи, представленные в БД минимум 4 раза):

Результат шага: L_1 = {{a}, {b}, {c}, {d}, {e}, {f}, {g}}.

Шаг 2, 3. Задаем k = 2 и запускаем процедуру *apriori_gen* для формирования из L_1 множества кандидатов C_2, отсортированных по алфавиту.

Результат: C_2 = {{a, b}, {a, c}, {a, d}, {a, e}, {a, f}, {a, g},
{b, c}, {b, d}, {b, e}, {b, f}, {b, g},
{c, d}, {c, e}, {c, f}, {c, g},
{d, e}, {d, f}, {d, g},
{e, f}, {e, g},
{f, g}}.

Шаг 4. Инициализируем счетчик *c.count* нулевым значением.

Шаг 5, 6. Перебираем все записи БД и находим те, которые содержатся в C_2.

Первая запись $r1$ = {*a, b, d, g*}. Элементами C_2, содержащими элементы из $r1$ будут: C_{r1} = {{*a, b*}, {*a, d*}, {*a, g*}, {*a, d*}, {*a, g*}, {*b, d*}, {*b, g*}, {*d, g*}}. Также для каждого из C_r для этих множеств признаков инкрементируем счетчик *c.count*++ .

Вторая запись $r2$ = {*c, d, e*}, а C_{r2} = {{*c, d*}, {*c, e*}, {*d, e*}}.

И так далее, для всех записей из БД. После анализа 20 записей БД проверяем, для каких множеств-кандидатов значения счетчика *c.count* не ниже заданного уровня *поддержки* – т.е. ≥ *minsup* (4): {*a, b*}{*a, c*}{*a, d*}{*c, d*}{*c, e*}{*c, f*}.

Результат: L_2 = {{*a, b*}{*a, c*}{*a, d*}{*c, d*}{*c, e*}{*c, f*}}

Затем осуществляется переход к шагу 2 алгоритма, k = 3 и выполняется процедура *apriori_gen* по данным L_2. Формируются следующие пары значений: {*a, b*}:{*a, c*} {*a, c*}:{*a, d*} {*c, d*}:{*c, e*} {*c, d*}:{*c, f*} {*c, e*}:{*c, f*}. Множества из 3 элементов будут выглядеть следующим образом: {*a, b, c*}, {*a, c, d*}, {*c, d, e*}, {*c, d, f*}, {*c, e, f*}.

- {*a, b, c*} исключается из рассмотрения, так как {*b, c*} не в L_2;
- {*c, d, e*} исключается из рассмотрения, так как {*d, e*} не в L_2;
- {*c, d, f*} исключается из рассмотрения, так как {*d, f*} не в L_2;
- {*c, e, f*} исключается из рассмотрения, так как {*e, f*} не в L_2.

Таким образом, остается $C_3 = \{a, c, d\}$. Выполняются шаги 5–7 с подсчетом количества появлений C_3 в исходной БД. Их число составляет 4, поэтому $L_3 = \{a, c, d\}$.

Затем осуществляется очередной переход к шагу 2, $k = 4$ и выполняется процедура *apriori_gen* по данным L_3. В данном случае $L_4 = \{\}$, выполнение алгоритма завершается, возвращая все полученные непустые множества.

Результат: *Ls* = {{a}, {b}, {c}, {d}, {e}, {f}, {g}, {a, c}, {a, d}, {c, d}, {c, e}, {c, f}, {a, c, d}}.

Каждый из 13 элементов множества данных *L*s может представлять бизнес-интерес, интерпретируя эти элементы можно составить несложные, но практически полезные ассоциативные правила.

6.7. Последовательная ассоциация

При анализе ассоциаций, которым посвящен разд. 6.6, может быть полезной не только найденная в совокупности транзакций базы данных связь между переменными, но и последовательность появления таких транзакций в базе данных во времени. В таком случае появляются анализ *последовательности* (англ. *sequence*) и поиск *последовательных ассоциаций* (англ. *sequential association*), которые могут содержать интересную информацию. Фактически *ассоциация* является частным случаем *последовательной ассоциации* с временным лагом, равным нулю.

При наличии закономерностей в таких последовательностях возможно с некоторой долей вероятности предсказывать появление событий в будущем и принимать на основе этого более обоснованные решения. Такую разновидность поиска ассоциативных правил называют *сиквенциальным анализом* (англ. *Sequential Pattern Mining, SPM*), отличающимся учетом отношения порядка между исследуемыми наборами. Данное отношение может быть определено разными способами. При анализе последовательности событий, происходящих во времени, объектами таких наборов являются события, а отношение порядка соответствует хронологии их появления. В этом случае примером закономерности (шаблоном, паттерном) служит

правило, указывающее, что из события X спустя время t последует событие Y.

Сиквенциальный анализ широко используется, например, в телекоммуникационных компаниях для анализа данных об авариях на различных узлах сети [118]. Информация о последовательности совершения аварий может помочь в обнаружении неполадок и предупреждении новых аварий. Например, если известна типичная последовательность сбоев некоторой телекоммуникационной среды: $\langle e_5, e_2, e_7, e_{13}, e_6, e_1, ... \rangle$, где e_i – сбой с кодом i, то на основании факта появления сбоя e_2 можно сделать вывод о скором появлении сбоя e_7. Это позволяет априори предпринять профилактические меры, устраняющие причины возникновения сбоя. Кроме того, если дополнительно обладать и знаниями о времени между сбоями, то можно предсказать не только факт его появления, но и время.

При анализе поведения пользователей интернет-сайтов полезным может быть определение профиля пользователя по закономерностям в последовательности навигации по тем или иным разделам (web-страницам) сайта. В биоинформатике такие ассоциации могут быть информативны при поиске каркасов белковых последовательностей.

В маркетинге и менеджменте при управлении циклом работы с клиентом (англ. *Customer Lifecycle Management*) эти подходы крайне востребованы, они позволяют предугадать связь приобретения одного товара / услуги с вероятностью приобретения других товаров / услуг позднее. Например, после покупки квартиры большинство людей в течение двух недель приобретают холодильник, а в течение двух месяцев – телевизор. Или может существовать последовательная связь между покупкой кровати и постельных принадлежностей для нее.

6.7.1. Алгоритмы семейства «Априори»

Первыми алгоритмами, разработанными для решения задач сиквенциального анализа, были алгоритмы, построенные на базе алгоритма *Априори* (см. разд. 6.6), но отличающиеся учетом допол-

нительного параметра – времени совершения транзакции (например, *AprioriAll*, *AprioriSome*, *DynamicSome*) [1, 118]. Эти алгоритмы используют подход генерации и отбора кандидатов часто встречающихся последовательностей, а также свойство, заключающееся в том, что каждая подпоследовательность ЧВП должна также быть ЧВП.

Опишем кратко суть этих алгоритмов сиквенциального анализа.

Пусть имеется база данных, в которой каждая запись представляет собой клиентскую транзакцию – *идентификатор клиента*, *дата / время транзакции*, *набор приобретенных товаров*. При этом есть условие – клиент не может иметь две и более транзакций, совершенных в один момент времени.

Введем несколько основных понятий, требуемых для описания сути алгоритмов.

Предметное множество (англ. *itemset*) – непустой набор предметов (товаров), появившихся в одной транзакции, т.е. приобретенных одновременно. Для обозначения используем фигурные скобки: $I = \{i1, i2, ..., im\}$, где ij – предмет.

Последовательность – упорядоченное предметное множество. Для обозначения используем треугольные скобки: $S = \langle I1, I2, ..., Im\rangle$, где Ii – предметное множество. *Длиной* последовательности будем называть количество предметов в этой последовательности, а *k-последовательностью* – последовательность длины k.

Последовательность $S1$ *содержится* в последовательности $S2$, если все предметные множества $S1$ содержатся в предметных множествах $S2$, при этом порядок надмножеств из $S2$ соответствует порядку предметных множеств $S1$. Например, последовательность $\langle\{3\}, \{4,5\}, \{8\}\rangle$ содержится в последовательности $\langle\{7\}, \{3,8\}, \{9\}, \{4,5,6\}, \{8\}\rangle$, поскольку $\{3\}$ содержится в $\{3,8\}$, $\{4,5\}$ содержится в $\{4,5,6\}$ и $\{8\}$ – в $\{8\}$.

Однако $\langle\{3\}, \{5\}\rangle$ не содержится в $\langle\{3,5\}\rangle$ (и наоборот), поскольку в первой последовательности предметы 3 и 5 приобретены один за другим, а во второй – совместно.

Все транзакции одного клиента могут быть показаны в виде последовательности, в которой они упорядочены по дате, времени или номеру визита. Такие последовательности будем называть

клиентскими. Формально это записывается следующим образом. Пусть клиент совершил несколько упорядоченных во времени транзакций $T1$, $T2$, ..., Tk. Тогда каждый предметный набор в транзакции Ti обозначим $I(Ti)$, а каждую клиентскую последовательность для данного клиента запишем как $\langle I(T1), I(T2), ..., I(Tk)\rangle$.

Последовательность S называется *поддерживаемой* клиентом, если она содержится в клиентской последовательности данного клиента. Тогда *поддержка* последовательности определяется как число клиентов, поддерживающих данную последовательность (аналогично *поддержке* при ассоциативном анализе в разд. 6.6).

Для базы данных клиентских транзакций задача поиска шаблонов заключается в обнаружении последовательностей, имеющих *поддержку* выше заданного порогового значения. Каждая такая последовательность является *шаблоном* (*паттерном*) последовательных событий. Последовательность, удовлетворяющую ограничению минимальной поддержки, будем называть ЧВП.

Последовательность S называется *максимальной*, если она не содержится в какой-либо другой последовательности.

Упомянутые выше алгоритмы сиквенционального анализа решают задачу поиска последовательных шаблонов и состоят в общем виде из следующих этапов.

Этап 1. Сортировка. Заключается в перегруппировке записей в таблице транзакций. Сначала записи сортируются по уникальному ключу покупателя, а затем по времени внутри каждой группы.

Этап 2. Отбор кандидатов. В исходном наборе данных производится поиск всех ЧВП. В частности, на этом этапе происходит поиск всех одноэлементных шаблонов.

Этап 3. Трансформация. Производится для ускорения процесса проверки присутствия последовательностей в наборе транзакций покупателей. Трансформация заключается в замене каждой транзакции списком ЧВП, которые в ней содержатся. При этом если в транзакции отсутствуют частые предметы, то данная транзакция не учитывается. Аналогичным образом не учитываются предметы, не являющиеся частыми, а также последовательности, транзакции которых не содержат ЧВП.

Этап 4. Генерация последовательностей. Из полученных на предыдущих этапах последовательностей строятся более длинные шаблоны последовательностей.

Этап 5. Максимизация (опционально). Среди имеющихся последовательностей происходит поиск тех, которые не входят в более длинные последовательности.

Все упомянутые алгоритмы реализованы аналогично (различаются в деталях реализации этапа 4 и обладают невысокой вычислительной эффективностью).

6.7.2. Алгоритм GSP

Для существенного повышения вычислительной эффективности (до 20 раз) сиквенционального анализа предложена модификация алгоритма *AprioriAll*, названная *GSP* (англ. *Generalized Sequential Pattern*, обобщенный сиквенциональный паттерн), учитывающая ограничения по времени между соседними транзакциями [1, 35].

В случае с алгоритмом *GSP* требуется учитывать дополнительные условия, чтобы определить, содержит ли последовательность указанную последовательность (подпоследовательность).

Введем такие параметры, как минимальное и максимальное допустимое время между транзакциями (*min_gap* и *max_gap*), а также понятие скользящего окна размера *win_size*. Допускается, что элемент последовательности может состоять не из одной, а из нескольких транзакций, если разница во времени между ними меньше, чем размер окна.

Последовательность $d = \langle d1 \ldots dm \rangle$ содержит последовательность $s = \langle s_1 \ldots s_n \rangle$, если существуют такие целые числа $l1 \le u1 < l2 \le u2 < \ldots < ln \le un$, что:

1) *si* содержится в объединении *dk*, где $k = l_i..u_i$, $1 \le i \le n$;

2) $t_{\text{транзакции}}(du[i]) - t_{\text{транзакции}}(dl[i]) \le win\_size$, $1 \le i \le n$;

3) $min\_gap < t_{\text{транзакции}}(dl[i]) - t_{\text{транзакции}}(du[i-1])$, $2 \le i \le n$;

4) $t_{\text{транзакции}}(du[i]) - t_{\text{транзакции}}(dl[i-1]) \le max\_gap$, $2 \le i \le n$.

Выполнение алгоритма *GSP* предусматривает несколько проходов по исходному набору данных. При первом проходе вычисляется

поддержка для каждого предмета и из них выделяются частые. Каждый подобный предмет представляет собой одноэлементную последовательность. В начале каждого последующего прохода имеется некоторое число ЧВП, выявленных на предыдущем шаге алгоритма. Из них будут формироваться более длинные последовательности-кандидаты.

Каждый кандидат представляет собой последовательность, длина которой *на один больше* чем у последовательностей, из которых кандидат был сформирован. Таким образом, число элементов всех кандидатов одинаково. После формирования кандидатов происходит вычисление их поддержки. В конце шага определяется, какие кандидаты являются ЧВП. Найденные ЧВП послужат исходными данными для следующего шага алгоритма. Работа алгоритма завершается тогда, когда не найдено ни одной новой ЧВП в конце очередного шага или когда невозможно сформировать новых кандидатов.

Таким образом, в работе алгоритма можно выделить следующие основные этапы:

1. Генерация кандидатов.
   1.1. Объединение.
   1.2. Упрощение.
2. Подсчет поддержки кандидатов.

Рассмотрим эти операции более подробно.

Этап 1. Генерация кандидатов. Пусть Lk содержит все частые k-последовательности, а Ck – множество кандидатов из k-последовательностей. В начале каждого шага имеем $Lk - 1$ – набор из $(k – 1)$ ЧВП. На их основе необходимо построить набор всех k ЧВП.

Введем понятие *смежной подпоследовательности*.

При наличии последовательности $s = \langle s1\ s2\ ...\ sn\rangle$ и подпоследовательности c, c будет являться *смежной последовательностью* s, если соблюдается одно из условий:

– c получается из s при удалении предмета из первого $\{s1\}$ или последнего $\{sn\}$ предметного множества;

– c получается из s при удалении одного предмета из предметного множества si, если в его составе не менее двух предметов;

– c – смежная подпоследовательность c', где c' – смежная подпоследовательность s.

Например, дана последовательность $s = \langle\{1,2\}, \{3,4\}, \{5\}, \{6\}\rangle$. Последовательности $\langle\{2\}, \{3,4\}, \{5\}\rangle$, $\langle\{1,2\}, \{3\}, \{5\}, \{6\}\rangle$ и $\langle\{3\}, \{5\}\rangle$ являются смежными подпоследовательностями s, а последовательности $\langle\{1,2\}, \{3,4\}, \{6\}\rangle$ и $\langle\{1\}, \{5\}, \{6\}\rangle$ таковыми не являются.

Если некоторая последовательность содержит последовательность s, то она также содержит и все смежные подпоследовательности s.

Генерация кандидатов происходит в два этапа (условно назовем эти этапы функцией *candidate_gen_SPM*()).

Этап 1.1. Объединение (англ. *join*). Создаем последовательности-кандидаты путем объединения двух последовательностей $Lk - 1$ и $Lk - 1$. Последовательность $s1$ объединяется с $s2$, если подпоследовательность, образующаяся путем удаления первого предмета из $s1$ будет та же, что и в случае удаления последнего предмета из $s2$. Объединение последовательностей происходит путем добавления к $s1$ соответствующего предмета из последнего предметного множества $s2$. Причем возможно два варианта:

– если последний предмет из $s2$ составлял одноэлементное предметное множество, то при объединении он будет добавлен к $s1$ как новое предметное множество;

– в противном случае он будет включен в последнее предметное множество $s1$ как его элемент.

При объединении $L1$ с $L1$ нужно добавить предмет к $s2$ как отдельное предметное множество, а также в качестве дополнительного предмета в предметное множество последовательности $s1$. Так, объединение $\langle\{x\}\rangle$ с $\langle\{y\}\rangle$ дадут как $\langle(x, y)\rangle$, так и $\langle(x), (y)\rangle$. Причем x и y упорядочены.

Этап 1.2. Упрощение (англ. *prune*). Удаляем последовательности-кандидаты, которые содержат смежные $(k - 1)$-последовательности, чья поддержка меньше минимально допустимой.

Этап 2. Подсчет поддержки кандидатов. Сканируя набор последовательностей, обрабатываем их по очереди. Для тех кандидатов,

которые содержатся в обрабатываемой последовательности, увеличиваем значение поддержки на единицу. Для уменьшения количества кандидатов, требующих проверки вхождения в обрабатываемые последовательности, а также для увеличения скорости проверки используется хэш-дерево [4].

Проиллюстрируем основные этапы работы алгоритма *GSP* на примере данных в табл. 8–10.

Таблица 8

Пример таблицы транзакций магазина

ID клиента	Время транзакции	Транзакция
1	20 июля 2015	30
1	25 июля 2015	90
2	9 июля 2015	10, 20
2	14 июля 2015	30
2	20 июля 2015	40, 60, 70
3	25 июля 2015	30, 50, 70
4	25 июля 2015	30
4	29 июля 2015	40, 70
4	2 августа 2015	90
5	12 июля 2015	90

Таблица 9

Пример таблицы последовательностей на основе БД транзакций

ID клиента	Последовательность данных
1	⟨{30} {90}⟩
2	⟨{10, 20} {30} {40, 60, 70}⟩
3	⟨{30, 50, 70}⟩
4	⟨{30} {40, 70} {90}⟩
5	⟨{90}⟩

Таблица 10

Пример итогового набора последовательностей различной длины с заданным уровнем поддержки

Виды последовательностей	Последовательные паттерны с поддержкой $\geq 25\%$
1-последовательности	⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩,
2-последовательности	⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40} {70}⟩
3-последовательности	⟨{30} {40, 70}⟩

Формальная запись алгоритма *GSP* может быть представлена следующим образом:

Алгоритм GSP(*S*):

Шаг 1. $C_1 \leftarrow$ *init_pass*(*S*);
// первый проход через *S*

Шаг 2. $F_1 \leftarrow \{(\{ f \}) \mid f \in C_1, f.count/n \geq minsup\}$;
//n – число последовательностей в *S*

Шаг 3. **for** (k = 2; $F_{k-1} \neq \varnothing$; k++) **do**
// формирование подпоследовательностей из *S*

Шаг 4. $C_k \leftarrow$ *candidate_gen_SPM*(F_{k-1});
// однократное сканирование набора данных

Шаг 5. **for** (для каждой последовательности данных $s \in S$) **do**

Шаг 6. **for** (для каждого кандидата $c \in C_k$) **do**

Шаг 7. **if** *c* содержится в *s* **then**

Шаг 8. *c.count*++;
// инкремент счетчика поддержки

Шаг 9. **end**

Шаг 10. **end**

Шаг 11. $F_k \leftarrow \{c \in C \mid c.count / n \geq minsup\}$

Шаг 12. **end**

Шаг 13. return $\cup_k F_k$;

Детали реализации функции генерирования кандидата-последовательности *candidate_gen_SPM*() изложены выше.

Пример генерирования кандидатов-последовательностей приведен в табл. 11.

Таблица 11

Пример работы этапа генерирования кандидатов-последовательностей

3-последовательности	4-последовательности	
	После присоединения	После упрощения
⟨{1, 2} {4}⟩	⟨{1, 2} {4, 5}⟩	⟨{1, 2} {4, 5}⟩
⟨{1, 2} {5}⟩	⟨{1, 2} {4} {6}⟩	
⟨{1} {4, 5}⟩		
⟨{1, 4} {6}⟩		
⟨{2} {4, 5}⟩		
⟨{2} {4} {6}⟩		

Недостатками подобных алгоритмов, существенно снижающими вычислительную эффективность обработки данных, являются:

– большое количество обращений к базе данных, соответствующее длине максимального кандидата-последовательности;

– большое число генерируемых кандидатов-последовательностей.

6.8. Многоуровневое машинное обучение

Для обеспечения устойчивости результатов машинного обучения используются различные методы ансамблирования моделей, основная суть которых заключается в объединении по тем или иным правилам результатов работы семейства моделей («слабых» классификаторов), позволяя в результате получить «сильный» классификатор. Такое ансамблирование должно позволить усовершенствовать итоговый результат работы модели, уменьшая дисперсию или смещение ее результатов (рис. 22).

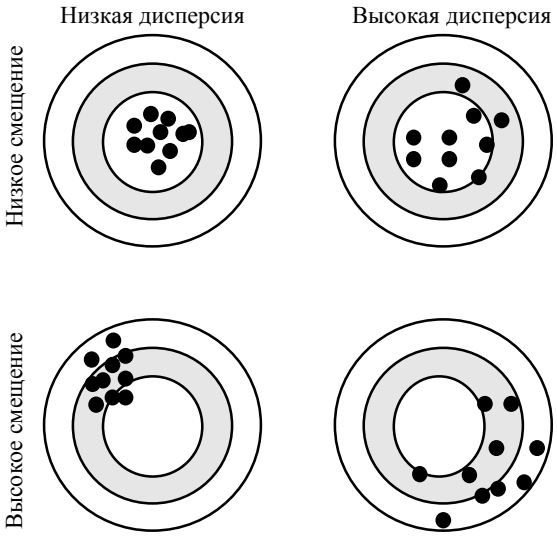


Рис. 22. Иллюстрация различной точности результатов модели

Этот раздел посвящен основным подходам к ассамблевым построениям моделей машинного обучения, получившим наименования на основе их англоязычных аналогов – бэггинг (англ. *bagging*), стекинг (англ. *stacking*) и бустинг (англ. *boosting*). При этом начать изложение следует с подхода к формированию обучающих данных для ансамблевого обучения, получившего название бутстрэпппинг (англ. *bootstrapping*).

6.8.1. Бутстрэппинг

Применение любого подхода к использованию семейства моделей предполагает разделение обучающей выборки так, чтобы каждой из моделей досталась достаточная и репрезентативная подвыборка обучающих данных.

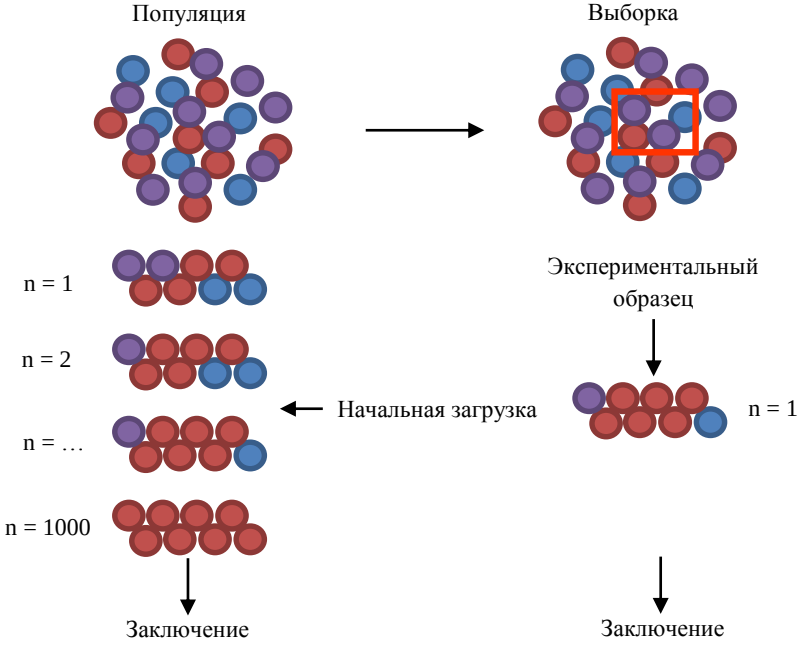


Рис. 23. Иллюстрация создания семейства подвыборок методом бутстрэп

На практике такое формирование подвыборок для каждой из порой многочисленных моделей семейства может быть затруднительным вследствие дефицита обучающих данных (иногда даже для одной единственной модели). В этом случае получил распространение метод **бутстрэп** – практический метод формирования семейства (как правила значительного числа) подвыборок на базе элементов имеющейся выборки с возвратом в нее элементов, получая возможность построения статистик, предельно близких к статистикам исходной выборки (рис. 23).

Суть метода бутстрэп заключается в следующем:

– для формирования новой «бутстрэп выборки» из исходной выборки **X** с количеством экземпляров N статистически случайно по методу Монте-Карло (с одинаковой вероятностью $1/N$) возьмем N объектов с возвращением объекта обратно в исходную выборку. Причем каждый раз выбор осуществляем из всех исходных N объектов. Из-за возвращения среди них могут оказаться и «повторы»;

– обозначим новую «бутстрэп выборку» через $\mathbf{X}_1$. Повторяя процедуру M раз, сгенерируем M подвыборок $\mathbf{X}_1, \ldots, \mathbf{X}_M$;

– при достаточно большом M получим возможность оценивать различные статистики исходного распределения с высокой точностью.

6.8.2. Бэггинг

Bagging (от *Bootstrap aggregation*) – один из самых простых видов ансамблей, в котором обучение базовых моделей производится параллельно. Изначально был придуман для улучшения результатов моделей на основе деревьев решений, однако может использоваться на любых моделях.

Главным преимуществом бэггинга является значительное увеличение точности предсказания ансамбля относительно их базовых моделей (увеличение может достигать 10–40%), что реализуется за счет уменьшения разброса ответов базовых моделей при усреднении.

Отмечают следующие недостатки этого метода:

– слабая математическая обоснованность улучшения точности предсказаний;

– недетерминированность результата из-за случайного формирования выборок;

– относительная сложность интерпретации результатов.

Схематически описать принцип бэггинга можно следующим образом (рис. 24). Из обучающей выборки методом бутстреппинга формируется m тренировочных выборок T_1, …, T_m. На каждой выборке обучаются модели с использованием одинаковых алгоритмов. Итоговая модель будет усреднять предсказания всех этих алгоритмов (в случае классификации это будет голосование): P_f.

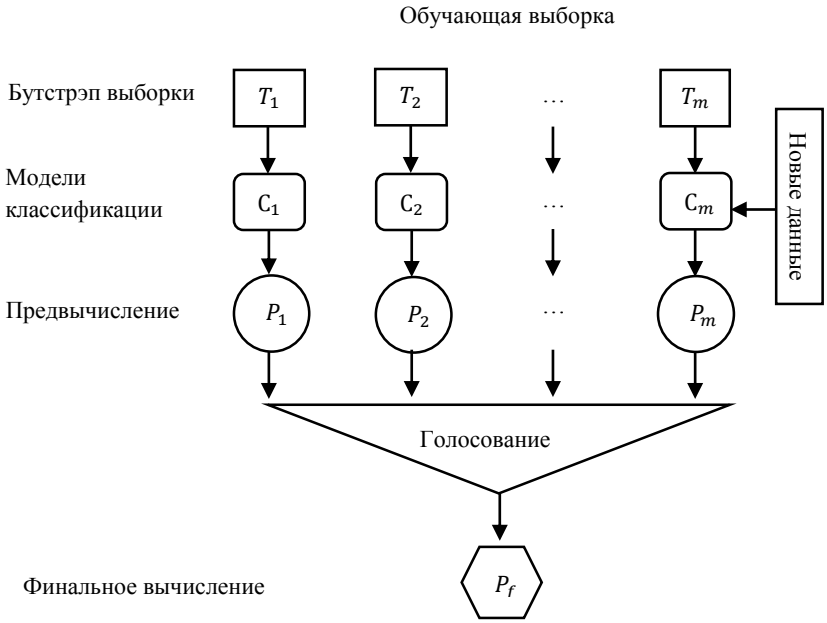


Рис. 24. Бэггинг

В качестве примера широко используемой реализации алгоритма бэггинга следует рассмотреть случайные леса (random forest, или forest of randomized trees). Алгоритм представляет собой усовершенствованный метод бэггинга Лео Бреймана путем использования метода случайных подпространств, предложенного Tin Kam Ho.

Основная идея заключается в использовании большого ансамбля решающих деревьев, отдельные результаты которых не дают высокого качества классификации (предсказания). Хороший итоговый результат получается за счет большого количества деревьев в ансамбле. Метод случайных подпространств позволяет снизить коррелированность между деревьями и избежать переобучения. Базовые алгоритмы обучаются на различных подмножествах признакового описания, которые также выделяются случайным образом.

Данные для ансамбля генерируются методом бутстрэппинга. Далее по каждой из подвыборок X_n строим решающее дерево b_n. Дерево строится до полного исчерпания подвыборки. Для задачи классификации выбирается решение голосованием по большинству (каждое дерево относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев), а в задаче регрессии – средним.

Таким образом, случайный лес – это бэггинг над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков.

6.8.3. Стекинг

Стекинг (*Stacked Generalization*, *Stacking*) – метод ансамблирования моделей, который объединяет несколько моделей классификации или регрессии соответствующим метаалгоритмом. Базовые модели обучаются на тренировочной выборке, а метаалгоритм обучается на ответах базовых моделей. Простейшая схема стекинга – блендинг (blending) (рис. 25), когда обучающую выборку делят на две части. На первой части обучаются базовые алгоритмы, а на второй части и на тестовой выборке получаются ответы обученных моделей. Ответ каждого базового алгоритма рассматривается как «метапризнак», на таких «метапризнаках» обучается «метаалгоритм». Полученный алгоритм запускается на «метапризнаках» теста и получается финальный ответ.

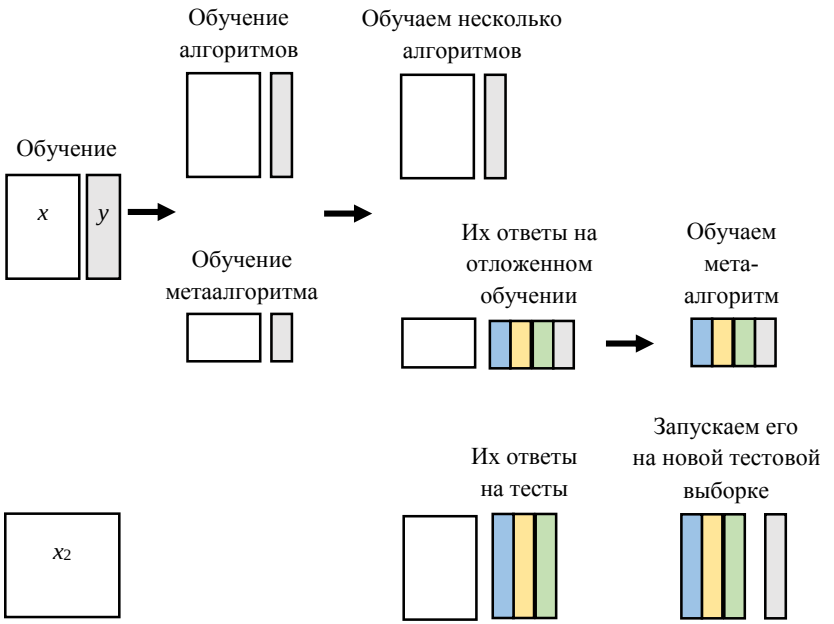


Рис. 25. Блендинг

В общем алгоритм стеккинга выглядит следующим образом. На входе имеем обучающую выборку **D** = {x_i, y_i} размерностью m. Сначала обучаем базовые модели на одной обучающей выборке **D** (h_1, …, h_T). Новая обучающая выборка будет состоять из ответов базовых моделей $D_h = \{x_i', y_i\}$, где $x_i' = \{h_1(x_i), \dots, h_T(x_i)\}$. На полученной выборке обучаем метамодель H.

6.8.4. Бустинг

Boosting (усиление) – итеративный композиционный метаалгоритм обучения с учителем, корректирующий веса наблюдений по результатам каждой из итераций (если наблюдение классифицировано ошибочно, то его вес увеличивается, и наоборот), преобразуя слабые обучающие алгоритмы к сильным. Алгоритм бустинга

лучше всего можно описать на примере одного из наиболее широко используемых алгоритмов этого типа AdaBoost (Adaptive Boosting).

1. Начальное распределение весов $w_n = \frac{1}{N}, n = 1, \dots, N$.

2. Для каждого $m = 1, \dots, M$:

– обучается классификатор $y_m(x)$, который минимизирует взвешенную ошибку классификации

$$\epsilon_m = \sum_{n=1}^{N} w_n^{(m)} 1[y_m(x_n) \neq t_n] \Big/ \sum_{n=1}^{N} w_n^{(m)}.$$

– вычисляется взвешенная ошибка классификатора $y_m(x)$ $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$.

– обновляются распределение весов

$$w_n^{(m+1)} = w_n^{(m)} exp\{\alpha_m 1[y_m(x_n) \neq t_n]\}.$$

3. Производится предсказание используя финальную модель: $Y_M(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$.

Другим популярным вариантом бустинга является градиентный бустинг и его реализация на деревьях – Extreme Gradient Boosting (XGBoost). Градиентный бустинг – техника машинного обучения для задач классификации и регрессии, где строится модель предсказания в виде ансамбля слабых предсказывающих моделей. Библиотека создавалась как исследовательский проект Tianqi Chen в 2014 г. и описана в статье «XGBoost: a Scalable Tree Boosting System». Первоначально это было только терминальное приложение, теперь у него есть пакеты на Python, R, Java, C++ и Julia. XGBoost является лидером среди соревнований по машинному обучению в сообществе Kaggle. В отличие от бэггинга обучение ансамбля проводится последовательно.

Иллюстрация работы XGBoost изображена на примере на рис. 26.

Box 1: Первый классификатор создает вертикальную линию, которая верно определяет две точки «+», но ничего не знает о других точках «+» и «–».

Box 2: Следующий классификатор будет пытаться исправить ошибки первого, поэтому он дает больший вес трем «+» точкам и

создает вертикальную линию на D2. Тем не менее он ошибается, неправильно классифицируя три точки «–».

Box 3: Третий классификатор продолжает улучшать результат. Снова он дает большие веса трем «–» точкам, неверно определенным на прошлом классификаторе, и создает линию D3. Однако классификатор ошибочно определил точки в окружности.

Box 1, Box 2 и Box 3 – слабые классификаторы. Эти классификаторы будут использованы для создания сильного классификатора Box 4.

Box 4: Это взвешенная комбинация слабых классификаторов. Как можно увидеть, он хорошо справляется с правильной классификацией всех точек.

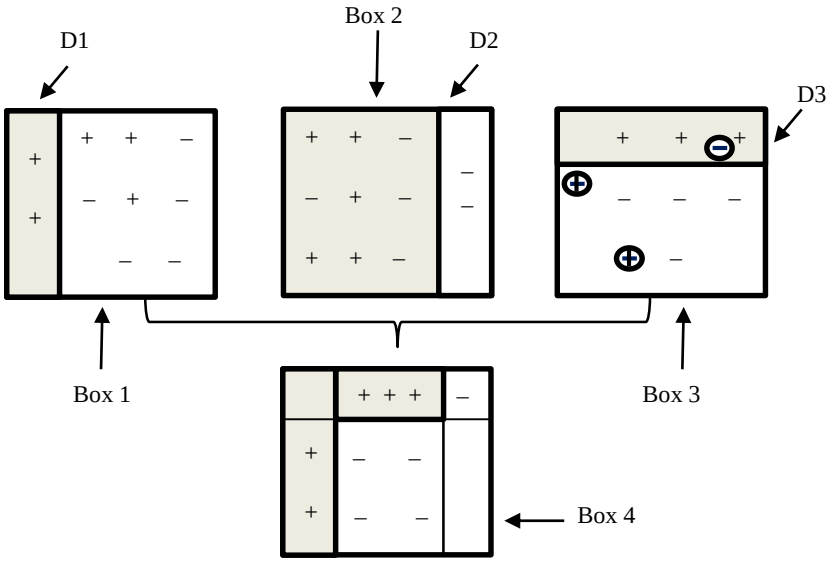


Рис. 26. Иллюстрация работы алгоритма XGBoost

XGBoost успешно используется в прогнозировании продаж, классификации текстовых документов Интернета, прогнозировании поведения клиента, классификации вредоносных программ, категоризации продуктов и пр.

Из других популярных реализаций алгоритмов бустинга можно отметить такие библиотеки и пакеты, как LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost.

6.9. Обнаружение аномалий

Еще одним крайне полезным направлением научно-практической деятельности в *Data Mining* принято считать обнаружение аномалий (англ. *Anomaly Detection*). Это направление, по сути, можно рассматривать как противоположное направлению кластеризации (см. разд. 6.4), так как в данном случае необходимо решить обратную задачу – найти такие экземпляры данных, которые являются нетипичными, редкими для некоторого обычного (типичного, традиционного) профиля исследуемого набора данных. Очевидно, такие аномалии (явления с низкой частотой возникновения) могут появиться и быть выявленными только в случае значительного объема накопленных данных с «типичным профилем».

Учитывая высокую практическую ценность, порой заложенную в такие аномалии, сегодня присутствует целый ряд областей приложения методов обнаружения аномалий:

– обнаружение фактов мошенничества со счетами кредитных карт, подозрительных сделок в страховании и т.п.;

– обнаружение фактов несанкционированного вторжения в информационно-коммуникационные сети и среды;

– обнаружение ошибок.

На рис. 27 представлен пример двумерной диаграммы с данными признаков *X* и *Y*, позволяющей визуально оценить выборки без аномалий (N_1 и N_2), а также выборку (O_3) и отдельные экземпляры с аномалиями (o_1, o_2).

Для решения задач обнаружения аномалий применяют огромное количество подходов, методов, алгоритмов и их модификаций, основанных на анализе разновременных данных различным математическим аппаратом. Приводят следующую классификацию методов обнаружения аномалий, позволяющую в некоторой степени систематизировать массив имеющихся подходов и способов:

1. Основанные на графическом представлении (англ. *Graphical-based*). В этом случае используют различные диаграммы и скаттерграммы 1D–3D-мерные. Основные недостатки: трудоемкость построения и субъективность.

2. Основанные на статистических методах (англ. *Statistical-based*). Требуют предположения о наличии смеси распределений «типичных» данных и данных отклонения, а также гипотез о законах их распределения. К недостаткам относят то, что обычно распределение данных не известно, а в случае многомерной среды еще и затруднительно достаточно точно оценить статистическую плотность распределения выборки.

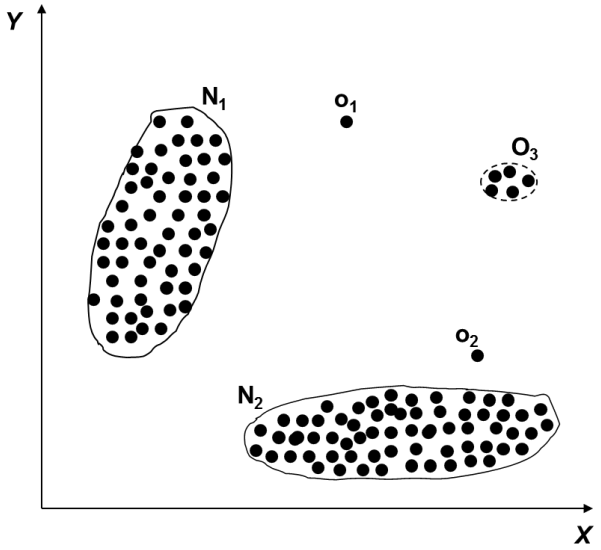


Рис. 27. Пример двумерной диаграммы данных признаков X и Y, отражающей наличие и отсутствие аномалий

3. Основанные на использовании различных метрик расстояния (англ. *Distance-based*). В этом случае данные представлены в виде набора векторов признаков, а для их анализа используют три основных подхода – ближайшего соседа, оценки плотности и кластеризации.

4. Основанные на использовании моделей (англ. *Model-based*). В этом случае общий алгоритм обнаружения аномалий может быть представлен в виде трех основных этапов:

– прогнозирование значений наблюдаемых временных рядов $x_i(t)$ на один шаг в соответствии с построенной моделью (x_i – интересующий параметр системы или среды, t – время);

– измерение отклонений между прогнозными значениями модели и фактическими значениями среды;

– механизм (набор решающих правил и процедур), определяющий, является ли значение (или последовательность значений) «слишком» отклоняющимся от прогнозного (соответствующего профилю «нормальной» работы сети).

Примерами конкретных методов для решения задач обнаружения отклонений могут быть авторегрессионная модель, модель скользящего среднего, комбинированная модель авторегрессии и скользящего среднего, фильтр Калмана, *SVM*, нейросети, байесовские сети и др., некоторые из которых подробно изложены выше. Активно развиваются новые научно-исследовательские методы и появляются примеры коммерческого применения, требующие отдельного внимательного рассмотрения и изучения [49, 57, 118].

7. ВИЗУАЛИЗАЦИЯ

Еще один метод *Data Mining*, который упоминают как имеющий самостоятельную ценность – *визуализация* (англ. *Visualization*). Его суть заключается в том, чтобы обеспечить для экспертного рассмотрения данные в таком визуальном представлении, которое позволит более контрастно показать имеющиеся в данных закономерности, связи и исключения (паттерны), неочевидные при ином рассмотрении. Визуализация знакомит зрителя с конечным результатом анализа, помогая выбрать нужное направление исследования данных.

При этом стоит отметить чрезвычайную важность визуализации – ведь имея одни и те же исходные данные, их можно представить различными способами, провоцируя различные (в общем случае – даже противоположные) варианты принятия решений!

Для решения этой задачи используют имеющийся инструментарий графического представления данных (варьирование размерности данных, типа диаграмм и графиков, используемых цветов, форм и других характеристик отображаемых элементов), который может (а порой и должен) быть применен совместно с другими методами *Data Mining*. Таких вариантов визуализации насчитывают десятки (рис. 28) [1]. При выборе метода визуализации необходимо учитывать все параметры. В первую очередь нужно понимать, как человек воспринимает зрительные образы. Классификации и рекомендации, составленные современными исследователями, позволяют оптимально подобрать метод визуализации под конкретную задачу.

Методы визуализации могут находить следующее применение:

– представлять пользователю информацию в наглядном виде;
– снижать размерность или сжимать информацию;
– компактно описывать закономерности, присущие исходному набору данных;
– восстанавливать пробелы в наборе данных;
– находить шумы и выбросы в наборе данных.

Рис. 28. Варианты визуализации данных

8. НЕЙРОСЕТЕВЫЕ ПОДХОДЫ И ГЛУБОКОЕ ОБУЧЕНИЕ

Нейросетевой аппарат достаточно широко применяется при решении самых различных задач классификации, прогнозирования, оценки плотности распределения и др. Базовые понятия искусственных нейронных сетей со схемой формального нейрона, архитектурой многослойного персептрона и алгоритмом обучения на основе обратного распространения ошибки, рассмотрены выше в разд. 6.3.3.

Однако, в последнее время именно нейросетевой подход (в очередной раз!) получает дополнительную популярность и практическое распространение, позволяя на практически значимом уровне решать сложные задачи.

8.1. Функции активации

Нейронная сеть, как и любой другой обучаемый алгоритм, принимает на входе некоторые данные, а на выходе отдает ответ. Входные данные представляют собой числовой вектор, который на рис. 14 представлен множеством $\{x_1 \dots x_n\}$.

По аналогии с принципами работы биологического нейрона сигналы, проходящие через связи между нейронами (синапсами), меняются, становясь слабее или сильнее. Этот фактор в концепции нейрона можно выразить числом, называемым весом. В схеме искусственного нейрона веса обозначены $\{w_1 \dots w_n\}$. Таким образом, поступившие на вход сигналы (входные данные) умножаются на свои веса: сигнал первого входа x_1 умножается на соответствующий этому входу вес w_1 и так для каждого из n входов.

Сумматор, как понятно из названия, представляет собой формулу следующего вида:

$$x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = \sum_{i=1}^{n} x_i w_i. \qquad (8.1)$$

Результатом работы сумматора является так называемая взвешенная сумма, обозначенная на схеме как net. Полученная взвешенная сумма подается на вход активационной функции $\varphi(net)$ (рис. 29), которая преобразует входной сигнал в итоговый ответ сети. Для разных типов нейронной сети используют самые разные функции активации.

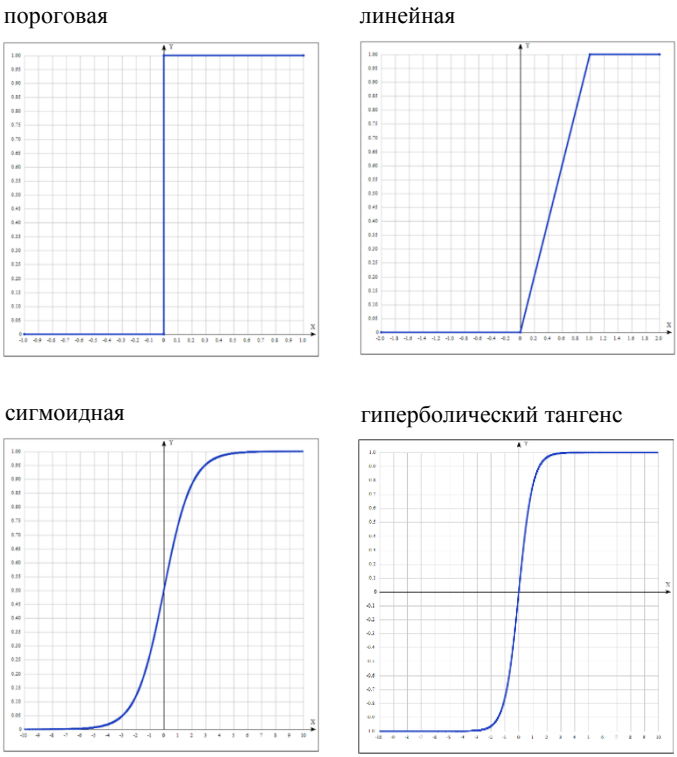


Рис. 29. Графики активационных функций

Основные виды функций активации:

1. Пороговая.
2. Линейная.
3. Логистическая.

4. Сигмоида:
   - гиперболический тангенс;
   - радиально-базисная.
5. ReLu.

Пороговая активационная функция:

$$f(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}. \quad (8.2)$$

Линейная активационная функция:

$$f(x) = \begin{cases} U, & x < U \\ x, & 0 \leq x < 1 \\ 1, & x \geq 0 \end{cases}. \quad (8.3)$$

Сигмоидальная активационная функция:

$$f(x) = \frac{1}{1+e^{-\alpha x}}, \quad (8.4)$$

где α – коэффициент наклона сигмоиды.

Гиперболический тангенс:

$$f(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} - e^{-\alpha x}}, \quad (8.5)$$

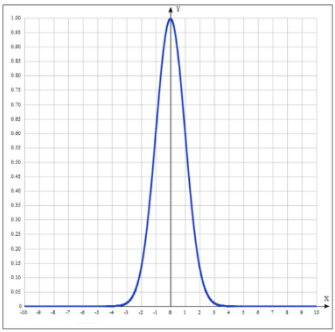
где α – коэффициент наклона сигмоиды.

Радиально-базисная активационная функция (рис. 30):

$$f(x) = e^{-\frac{x^2}{\sigma^2}}, \quad (8.6)$$

где σ – ширина окна.

радиально-базисная



ReLu

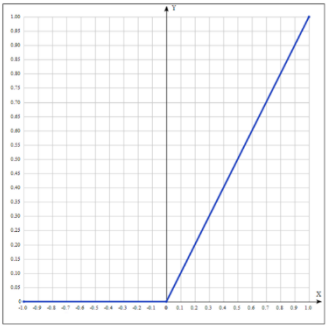


Рис. 30. Графики радиально-базисных активационных функций

ReLu:

$$f(x) = \begin{cases} 0, & x<0 \\ x, & x\geq 0 \end{cases}. \quad (8.7)$$

8.2. Основные типы искусственных нейронных сетей

Помимо использования различных функций активации, преобразующих совокупность входных сигналов в ответ, сети отличаются и способами связи нейронов между собой (топологией). На схеме представлены основные типы нейронных сетей (рис. 31).

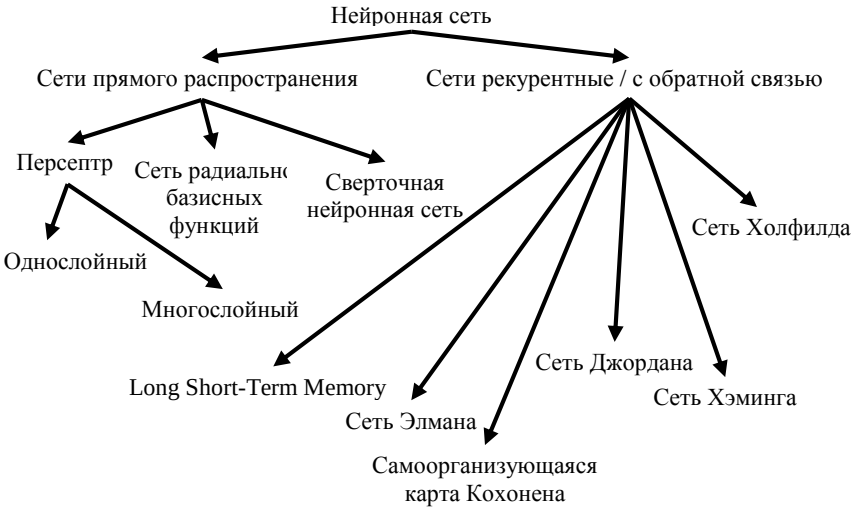


Рис. 31. Основные типы искусственных нейронных сетей

Сети прямого распространения. В сетях прямого распространения сигналы идут строго от входного слоя к выходному. В обратно направлении сигнал не распространяется.

В случае однослойной сети сигналы с входного слоя сразу подаются на выходной, производящий все вычисления, результаты которых сразу отдаются на выход в качестве ответа сети. Такие сети широко используются для решения определенного класса задач:

прогнозирование, кластеризация и распознавание. На рис. 31 приведен пример именно такой архитектуры ИНС.

Сеть радиально-базисных функций (рис. 32) использует радиально-базисную активационную функцию. Общий вид радиально-базисной функции:

$$f(x) = \varphi\left(\frac{x^2}{\sigma^2}\right), \tag{8.8}$$

где x – вектор входных сигналов нейрона, σ – ширина окна функции, φ – убывающая функция, например:

$$f(x) = e^{-\frac{x^2}{\sigma^2}}. \tag{8.9}$$

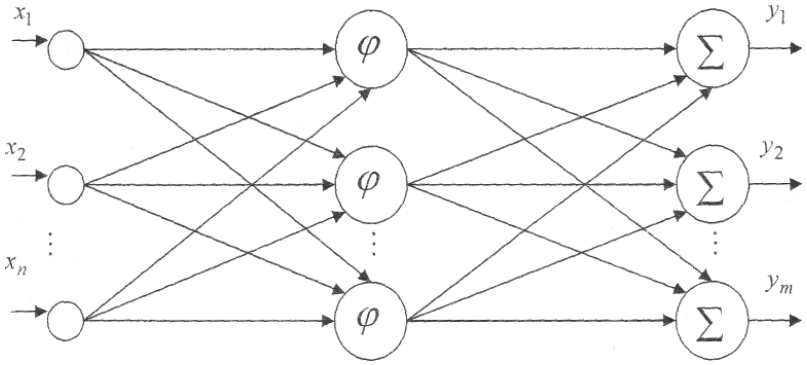


Рис. 32. Схема сети радиально-базисных функций

Радиально-базисные сети характеризуются следующими особенностями:

– моделирует произвольную нелинейную функцию с помощью всего одного промежуточного слоя, тем самым избавляя разработчика от необходимости решать вопрос о числе слоев;

– параметры линейной комбинации в выходном слое можно полностью оптимизировать с помощью известных методов линейной оптимизации, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении с использованием алгоритма обратного распространения ошибки. Поэтому сеть РБФ обучается очень быстро – на порядок быстрее, чем с использованием алгоритма с обратной связью.

Рекуррентные сети (с обратным распространением). Сигнал между нейронами может ходить не только в одном направлении от входа к выходу, но также и в обратную сторону. Это особенности сетей с обратной связью, где сигнал с выходных нейронов или нейронов скрытого слоя передается (возможно частично) обратно на входы нейронов входного слоя (обратная связь).

Наличие обратных связей позволяет запоминать и воспроизводить целые последовательности реакций на один стимул, что дает возможность решать задачи компрессии данных и построения ассоциативной памяти. На рис. 33 изображена схема сети с обратной связью.

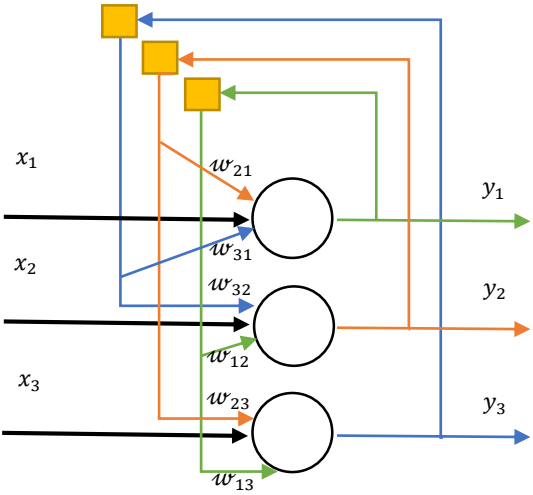


Рис. 33. Схема сети с обратной связью

В общем случае алгоритм обучения нейронной сети с обратной связью можно описать следующим образом:

Шаг 0. Начальные значения весов всех нейронов полагаются случайными:

$$W(t = 0)$$

Шаг 1. Сети предъявляется входной образ x^{α}, в результате формируется выходной образ:

$$y'^{\alpha} \neq y^{\alpha}.$$

Шаг 2. Вычисляется вектор ошибки, делаемой сетью на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе и равно нулю, если ошибка равна нулю.

$$\delta^{\alpha} = (y'^{\alpha} - y^{\alpha}).$$

Шаг 3. Вектор весов модифицируется по следующей формуле:

$$\mathrm{W}(t + \Delta t) = W(t) + \eta \times x^{\alpha} \times (\delta^{\alpha})^{T} \times \text{Градиент}(),$$

где η – темп обучения (learning rate), $0 < \eta < 1$, а ошибка δ^{α} передается на предыдущий слой.

Шаг 4. Шаги 1–3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется эпохой. Обучение завершается по истечении нескольких эпох:

а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или

б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

На рис. 34 изображена схема сети Хопфилда.

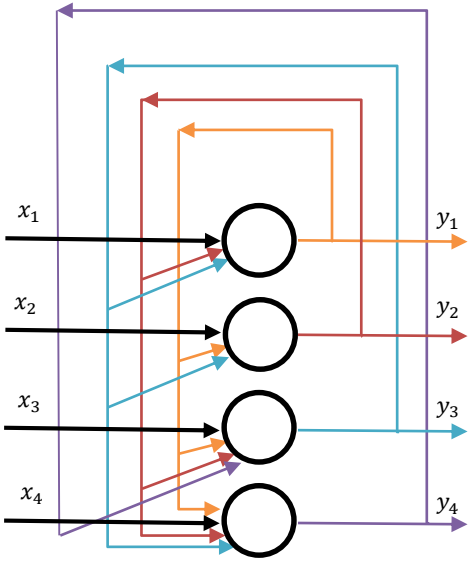


Рис. 34. Схема сети Хопфилда

Особенности:

– каждый нейрон имеет связь с выходом остальных нейронов, но не имеет связи со своим выходом;

– каждый вектор будет принимать бинарные значения (например, –1; +1);

– размерность векторов X и Y идентичная (n);

– «фильтрует» входные данные, возвращаясь к устойчивому состоянию;

– вместо последовательного приближения к нужному состоянию с вычислением ошибок все коэффициенты матрицы рассчитываются по одной формуле, за один цикл, после чего сеть сразу готова к работе;

– число запоминаемых образов $m \leq 0{,}15 \times n$ (приближенно).

Сеть Хэмминга. Трехслойная сеть с обратной связью. Используется для классификации бинарных векторов; основным критерием в ней является расстояние Хэмминга (рис. 35).

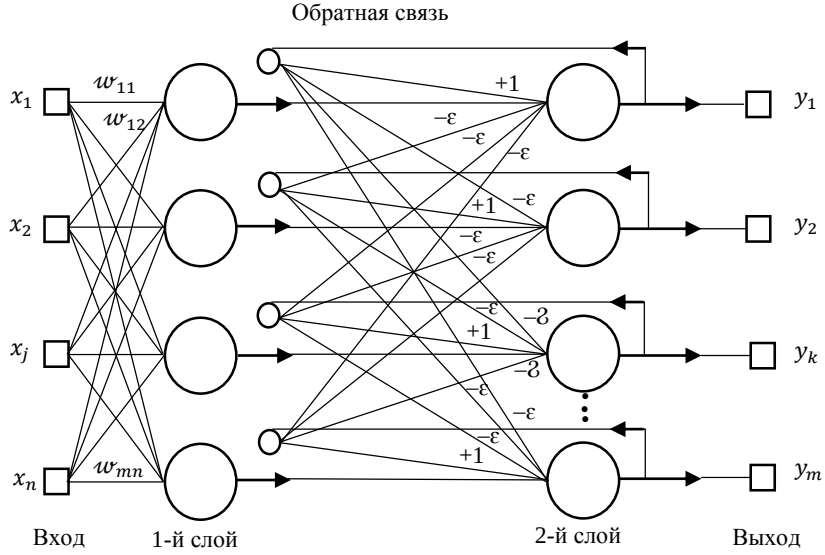


Рис. 35. Схема сети Хэмминга

При такой топологии каждый нейрон имеет связь с выходом остальных нейронов, в том числе и с самим собой. Каждый вектор будет принимать только бинарные значения. Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Размерность векторов слоя X и слоя Y идентичная, т.е. количество нейронов на втором и третьем слоях равно количеству классов.

По сравнению с сетью Хопфилда сеть Хэмминга характеризуется меньшими затратами на память и объемом вычислений, в том числе потому, что обучение происходит за один цикл.

Самоорганизующаяся сеть Кохонена. Широко используется для задач визуализации и кластеризации и относится к классу алгоритмов с обучением без учителя (рис. 36).

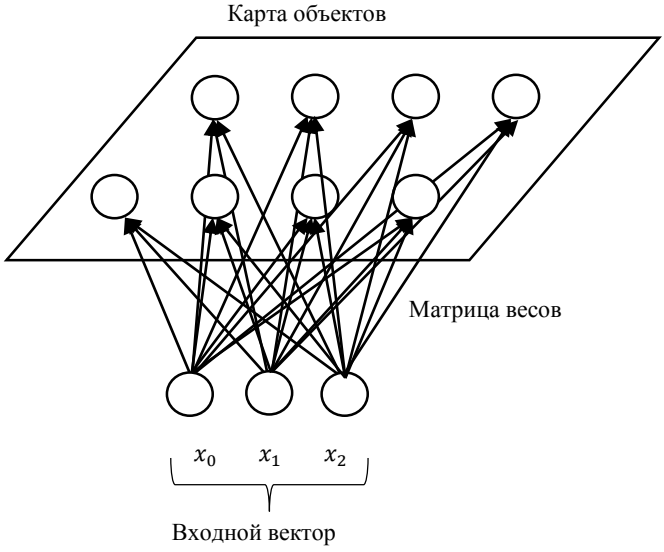


Рис. 36. Схема самоорганизующейся сети Кохонена

Алгоритм обучения такой сети можно описать следующим образом:

Шаг 1. Инициализация весовых коэффициентов.

Шаг 2. Подача входного вектора.

Шаг 3. Изменение весовых коэффициентов нейронов по формуле

$$w_i^{new} = w_i^{old} + \eta \times h_{ci} \times \left(x - w_i^{old}\right),$$

где h_{ci} – функция соседства нейронов.

Шаги 2, 3. Повторять *n* раз

Фактически самоорганизующаяся сеть Кохонена является методом преобразования *n*-мерного пространства в пространство более низкой размерности, в частности двумерное.

Сеть Джордана / Элмана. Получается из многослойного персептрона введением обратных связей, только связи идут, например, не от выхода сети, а от выходов внутренних нейронов с задержкой на один или несколько тактов.

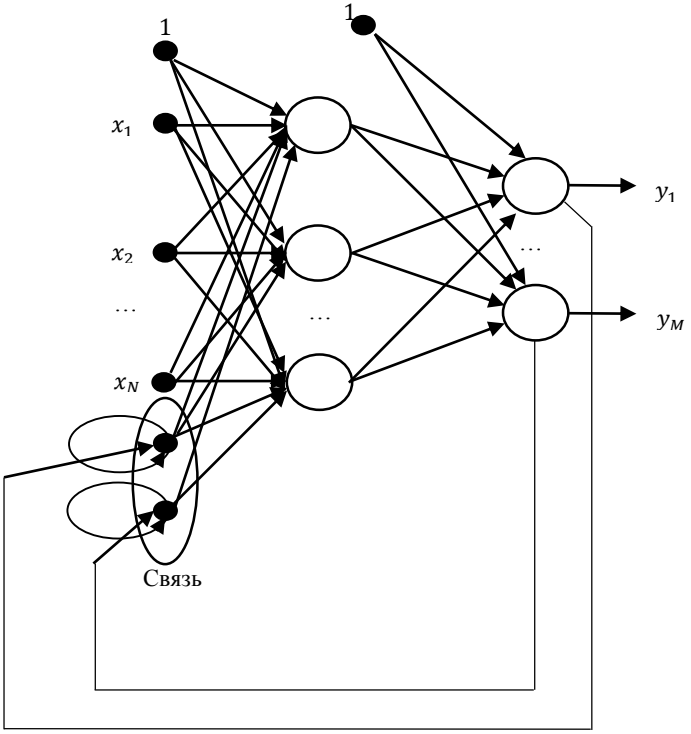


Рис. 37. Топология сети Джордана

Это позволяет учесть предысторию наблюдаемых процессов и накопить информацию (краткосрочно) для выработки правильной стратегии управления (например, движущимися объектами), так как их главной особенностью является запоминание последовательностей (рис. 37, 38)

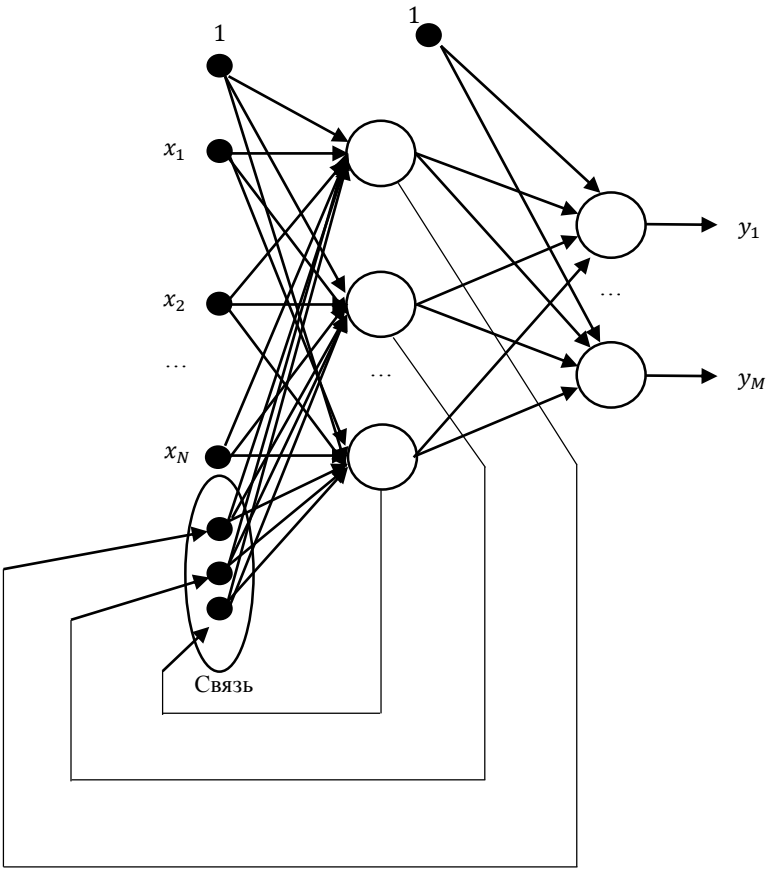


Рис. 38. Топология сети Элмана

У сети Джордана слой контекста обрабатывает данные с выходного слоя, а у сети Элмана – со скрытого слоя.

8.3. Сверточные нейронные сети (Convolutional Neural Networks)

Сверточная нейронная сеть – специальная архитектура искусственной нейронной сети, разработанная для эффективного распознавания изображений (Deep Learning). Название архитектура сети получила из-за наличия операции свертки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свертки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения. Состоит из нескольких слоев (5–1 000).

Можно выделить следующие основные области применения сверточных нейронных сетей:

- Обработка изображений:
  - определение расположения и распознавание объектов;
  - сегментация;.
  - оптическое распознавание символов (OCR);
  - определение и распознавание лиц;
  - восстановление изображений.
- Классификация текстов.
- Обработка аудио:
  - классификация музыка;
  - распознавание речи.

На рис. 39 представлена классическая упрощенная схема CNN.

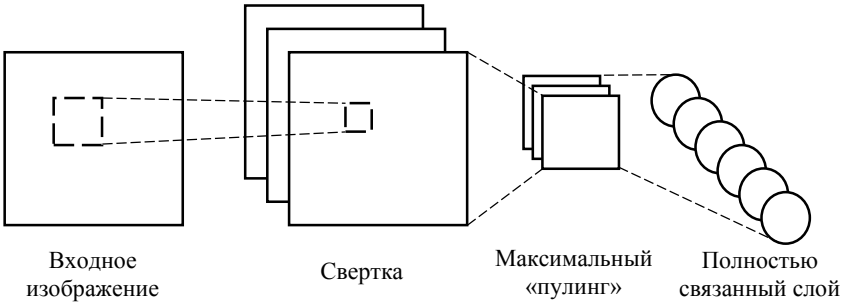


Рис. 39. Архитектура простейшей сверточной нейронной сети

Сверточный слой (convolution layer) – основной блок сверточной нейронной сети. В операции свертки используется ограниченная матрица весов небольшого размера (ядро свертки), которая применяется для всех нейронов выходного слоя:

$$l_{a,b} = f\left(\sum_i \sum_j x_{i,j} \times k_{i,j}\right). \quad (8.10)$$

Слой, получающийся в результате операции свертки такой матрицей, формирует так называемую карту признаков (feature map). При этом ядра свертки формируются в процессе обучения сети методом обратного распространения ошибки.

Далее выполняется уменьшение размерности сформированных карт признаков. Эта операция называется субдискретизацией (англ. *pooling*, операция подвыборки).

Поскольку карта признаков представляет собой набор признаков и их координат, то после операции субдискретизации на выходе получается уплотненная карта признаков, когда группа пикселей уплотняется до одного пикселя. При этом выбирается пиксель, имеющий максимальное значение (когда используется для преобразования функция максимума). Также могут использоваться функции усредненного значения (рис. 40).

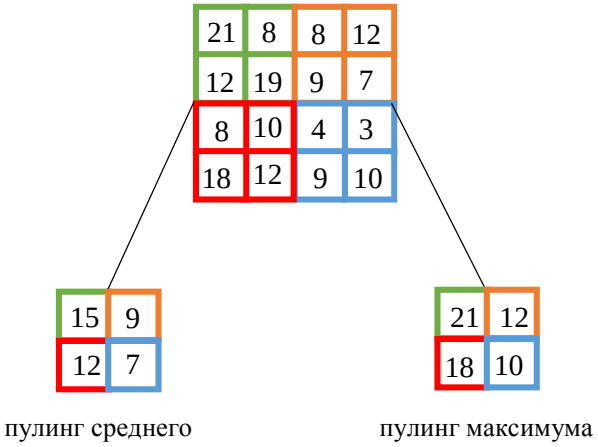


Рис. 40. Уменьшение размера карты признаков

Слои свертки и пулинга могут чередоваться любое количество раз. После нескольких слоев свертки и пулинга входное изображение превращается из сетки пикселей в более абстрактную карту признаков. В итоге остается большое число каналов входных сигналов, хранящих небольшое число данных. Эти данные передаются на обычную полносвязную нейронную сеть, при этом слои такой сети уже обладают сравнительно небольшой размерностью.

Как и в любой нейронной сети, в CNN в основе преобразований лежит использование функций активации. Традиционно в качестве функции активации использовались функции типа гиперболического тангенса или сигмоиды. С 2000-х гг. наиболее широко используемой стала функция на ReLU (англ. *rectified linear unit*), по сути, представляющая собой функцию отсечения отрицательной части скалярной величины (рис. 41):

$$f(x) = \max(0, x). \tag{8.11}$$

Softplus на рис. 41.– функция приближения для функции ReLu, называемая также SmoothReLU:

$$f(x) = \ln[1 + \exp(x)]. \tag{8.12}$$

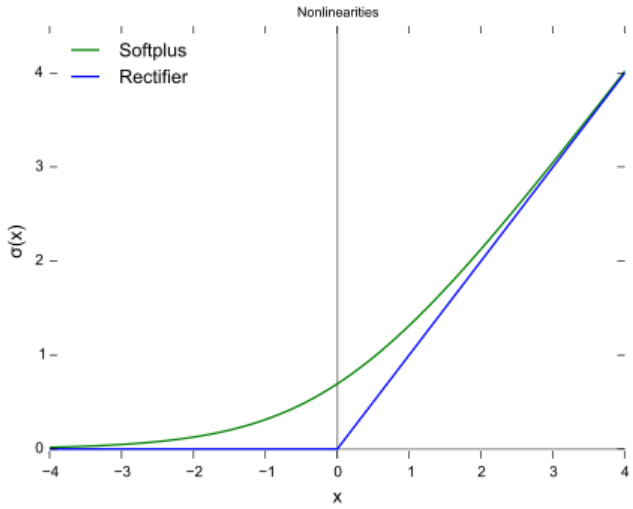


Рис. 41. Функция активации ReLU

Основные преимущества использования такой функции активации:

1) биологическая правдоподобность (0 при отрицательных значениях);

2) легкость для вычисления;

3) повышение скорости сходимости стохастического градиентного спуска.

Недостатки:

1) значение не ограничено;

2) может приводить к «смерти» нейронов (при обучении веса станут такими, что нейрон никогда больше не активируется, с данного момента градиент для этого нейрона всегда будет равен нулю).

8.4. Популярные архитектуры CNN

Рассмотрим примеры наиболее популярных архитектур сверточных нейронных сетей.

AlexNet. CNN, оказавшая большое влияние на развитие алгоритмов компьютерного зрения. Сеть с большим отрывом выиграла конкурс по распознаванию изображений ImageNet LSVRC-2012 в 2012 г. (с количеством ошибок 15,3% против 26,2% у второго места) (рис. 42).

GoogleNet. Победитель ILSVC 2014 г. с ошибкой всего 6,67%. Состоит из Inception модулей, основанных на нескольких очень маленьких свертках для резкого уменьшения количества признаков. Такая архитектура, например, позволила сократить количество признаков до 4 млн по сравнению с 60 млн в AlexNet (рис. 43).

VGG16-19. Одна из самых знаменитых моделей, отправленных на соревнование ILSVRC 2014 г., достигшая точности в 92,7% при тестировании на ImageNet в задаче распознавания объектов на изображении (рис. 44). Она является улучшенной версией AlexNet, в которой заменены большие фильтры (размера 11 и 5 в первом и втором сверточных слоях соответственно) на несколько фильтров размера 3 × 3, следующих один за другим.

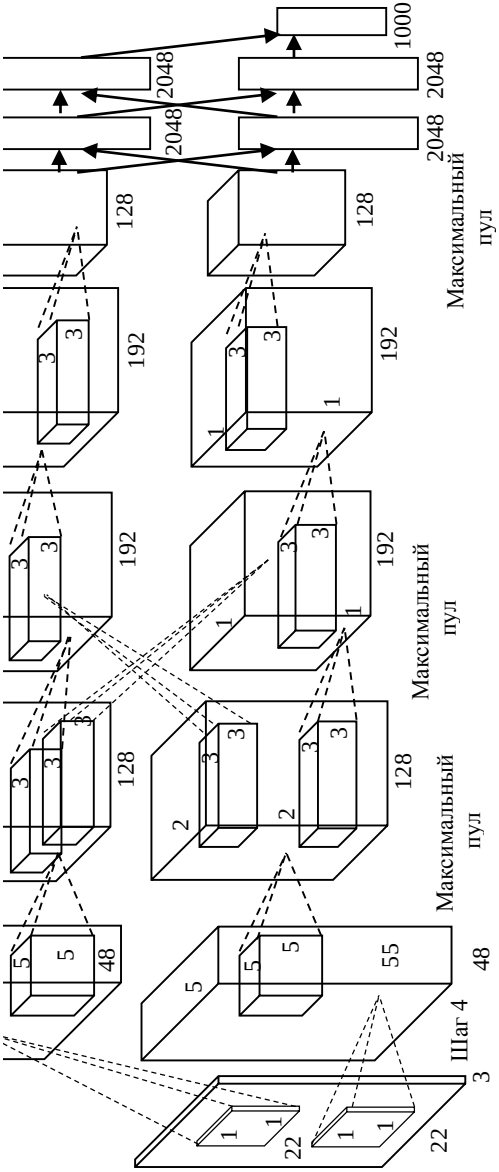


Рис. 42. Архитектура AlexNet

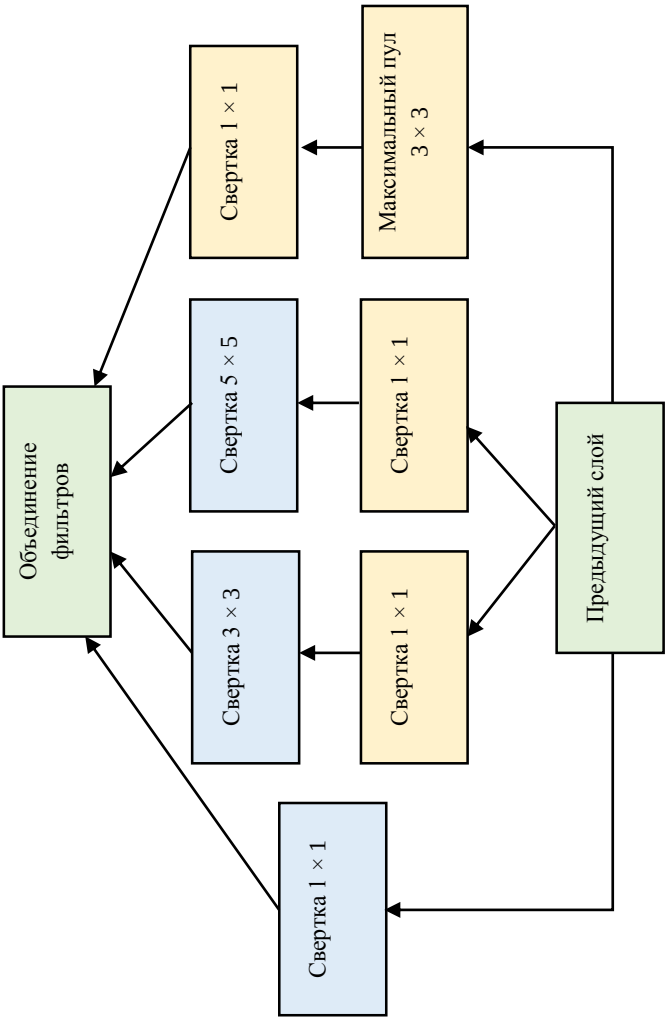


Рис. 43. Inception модуль GoogleNet

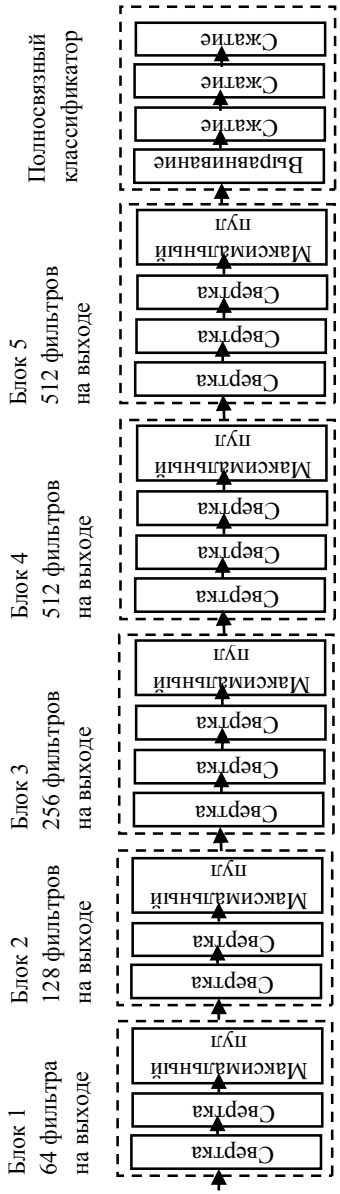


Рис. 44. Архитектура сверточной нейросети VGG 16

ResNets (Residual Networks). Глубокие сверточные нейронные сети превзошли человеческий уровень классификации изображений в 2015 г. Когда более глубокая сеть начинает сворачиваться, возникает проблема: с увеличением глубины сети точность сначала увеличивается, а затем быстро ухудшается. Это означает, что если просто добавлять слои в обычную CNN, она будет тренироваться хуже. Одно из решений этой проблемы – давать возможность пропускать сигнал без изменений от определенных слоев.

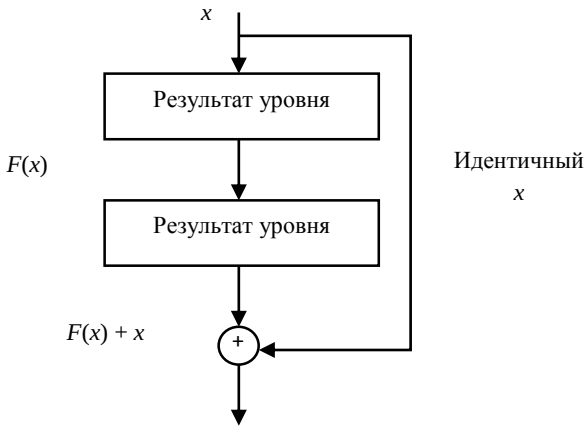


Рис. 45. Быстрое соединение в ResNets

Особенность ResNets основана на понятии быстрого соединения, которое превращает сеть в ее остаточную (residual) версию (рис. 45).

8.5. Среды и фреймворки глубинного обучения

Torch – библиотека для научных вычислений с широкой поддержкой алгоритмов машинного обучения, реализована на языке Lua с использованием C и CUDA.

Theano – это расширение языка Python, позволяющее эффективно вычислять математические выражения, содержащие многомерные массивы. В состав Theano входит компилятор, который

переводит математические выражения, написанные на языке Python, в эффективный код на C или CUDA.

Tensorflow – библиотека для машинного обучения от Google.

Caffe разработан Berkeley AI Research. Один из основных фреймворков индустрии. Есть интерфейсы под множество языков, в том числе Python, C++, Matlab. Поддерживает вычисления с помощью CUDA.

Keras – удобный python-интерфейс для построения архитектур и обучения нейронных сетей на популярных фреймворках.

DIGITS – разработка NVIDIA, web-интерфейс с инструментами для обучения сетей на выборках изображений. Работает на основе одной из популярных библиотек для машинного обучения.

Таблица 12

Использование различных типов сетей при решении задач машинного обучения

Задача	Используемые типы сетей
Классификация	Персептрон, сверточная нейронная сеть, сеть радиально-базисных функций, сеть Хопфилда, сеть Хэмминга, сеть Джордана, сеть Элмана, Long Short-Term Memory
Кластеризация	Самоорганизующаяся карта Кохонена
Аппроксимация	Персептрон, сеть радиально-базисных функций
Предсказание	Персептрон, сеть радиально-базисных функций, сеть Джордана, сеть Элмана, Long Short-Term Memory
Управление	Персептрон, сеть радиально-базисных функций, сеть Джордана, сеть Элмана, Long Short-Term Memory
Сжатие данных и ассоциативная память	Сеть Хопфилда, сеть Хэмминга
Оптимизация	Самоорганизующаяся карта Кохонена

Сравнение использования различных типов сетей при решении задач машинного обучения приведено в табл. 12.

9. ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

Обработка естественного языка (англ. *Natural Language Processing*) – общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез – генерацию грамотного текста. Решение этих проблем будет означать создание более удобной формы взаимодействия компьютера и человека [102]. Интеллектуальная обработка текстов – это целый комплекс задач, направленный на извлечение из текста интересующей информации (знаний) или генерация осмысленного текста на естественном для понимания человеком языке.

9.1. Основные задачи обработки текста

Перечислим основные задачи обработки текста в порядке увеличения их сложности [55]. При этом выделим три основных класса таких задач.

Задачи первого класса можно условно назвать синтаксическими; эти задачи, как правило, очень хорошо определены и представляют собой задачи классификации или задачи порождения дискретных объектов, и решаются многие из них сейчас уже довольно неплохо. Например:

– частеречная разметка (англ. part-of-speech tagging) – разметка в заданном тексте слова по частям речи (существительное, глагол, прилагательное и т.п.) и, возможно, по морфологическим признакам (род, падеж и т.п.);

– морфологическая сегментация (англ. morphological segmentation) – разделение слова в заданном тексте на морфемы, т.е. синтаксические единицы вроде приставок, суффиксов и окончаний; для некоторых языков (например, английского) это не очень актуально, но в русском языке морфологии очень много;

– стемминг (англ. stemming) – выделение основы слов;

– лемматизация (англ. lemmatization) – приведение слов к базовой форме (например, форме единственного числа мужского рода);

– пословная сегментация (англ. word segmentation) – нетривиальная задача, например, для языков с иероглифами без пробелов, где деление на слова происходит по-разному;

– выделение границ предложения (англ. sentence boundary disambiguation) – разбиение заданного текста на предложения (не всегда тривиальная задача, так как внутри предложения могут быть различные знаки препинания, включая «.»);

– распознавание именованных сущностей (англ. named entity recognition) – поиск в тексте собственных имен людей, географических и прочих объектов, разметка их по типам сущностей (имена, топонимы и т.п.);

– разрешение смысла слов (англ. word sense disambiguation) – выбор из омонимов по смыслу одного и того же слова в контексте;

– синтаксический парсинг (англ. syntactic parsing) – построение синтаксического дерева по заданному предложению (и, возможно, его контексту);

– разрешение кореференций (англ. coreference resolution) – определение объектов или частей текста, к которым относятся те или иные слова и обороты.

Второй класс включает задачи, которые требуют понимания текста, но по форме все еще представляют собой хорошо определенные задачи с правильными ответами (например, задачи классификации), для которых легко придумать не вызывающие сомнений метрики качества. К таким задачам, в частности, относят:

– языковые модели (англ. language models) – по заданному отрывку текста предсказать следующее слово или символ. Задача важна, например, для распознавания речи;

– информационный поиск (англ. information retrieval) – главная задача, решаемая поисковыми системами: по заданному запросу найти среди огромного множества документов наиболее релевантные данному запросу;

– анализ тональности (англ. sentiment analysis) – определение по тексту его тональности (позитив / негатив). Анализ тональности используется в онлайн-торговле для анализа отзывов пользователей, в финансах и трейдинге для анализа статей в прессе, отчетов компаний и тому подобных текстов и т.п.;

– выделение отношений или фактов (англ. relationship extraction, fact extraction) – выделение из текста хорошо определенных отношений или фактов об упоминающихся сущностях; например, кто с кем находится в родственных отношениях, в каком году основана упоминающаяся в тексте компания и т.д.;

– ответы на вопросы (англ. question answering) – в зависимости от постановки это может быть классификация из ограниченного небольшого набора вариантов ответов, или классификация с очень большим числом классов (ответы на фактологические вопросы вроде «кто?» или «в каком году?»), или даже порождение текста (если отвечать на вопросы нужно в рамках естественного диалога).

Наконец, к третьему классу отнесем задачи, в которых требуется не только понять уже написанный текст, но и породить (генерировать) новый. Здесь метрики качества уже не всегда очевидны. К таким задачам относят, например:

– собственно порождение текста (англ. text generation);

– автоматическое реферирование (англ. automatic summarization) – породить краткое содержание текста, сохранив основную суть. Это можно рассмотреть как задачу классификации, если просить модель выбрать из текста готовые предложения, лучше всего отражающие общий смысл, а можно как задачу порождения, если краткое содержание нужно написать с нуля;

– машинный перевод (англ. machine translation) – по тексту на одном языке породить соответствующий текст на другом языке;

– диалоговые модели (англ. dialog and conversational models) – поддержать разговор с человеком. Уже сегодня есть удачные примеры таких чат-ботов (например, «онлайн-консультанты» на разных торговых сайтах).

9.2. Этапы предварительной обработки текста

Перед тем как приступить к решению задач собственно обработки текста на естественном языке, необходимо провести предварительную обработку текстовых данных, применив ряд методов, среди которых:

– Графематический анализ (сегментация), т.е. выделение в тексте предложений и словоформ, точнее токенов (так как в тексте могут быть не только слова). Эти токены – простейшие единицы информации для модели. Обычно используются униграммы (токены, состоящие из одного слова). Часто бывает полезно включить в анализ биграммы (два слова) или триграммы (три слова), чтобы извлечь дополнительное смысловое содержание.

– Лемматизация и стемминг, о которых упомянуто выше. При решении практических задач рекомендуют лемматизаторы для русского языка на базе библиотек mystem и pymorphy. Следует отметить, что стемминг (отбрасывание окончаний и других изменяемых частей слов) больше подходит для английского языка, а для русского языка более предпочтительна лемматизация.

– Фильтрация стоп-слов. Этот этап предполагает исключение наиболее частых слов, встречающихся в текстах любой тематики (предлоги, союзы, числительные, местоимения, некоторые глаголы, прилагательные и наречия). Они бесполезны для тематического анализа и могут быть отброшены. Их отбрасывание почти не влияет на объем словаря, но может приводить к заметному сокращению длины текстов. Редкие слова также рекомендуется отбрасывать, поскольку они не могут повлиять на тематику коллекции. Отбрасывание редких слов, а также строк, не являющихся словами естественного языка (например, чисел), помогает во много раз сокращать объем словаря, снижая затраты времени и памяти на построение моделей [54].

– Модель мешка слов (bag-of-words). После предварительной обработки текстовые данные необходимо представить в понятном для алгоритмов машинного обучения виде. Простейшим способом является модель мешка слов, которая позволяет представить текст

как числовые векторы признаков. В основе модели мешка слов лежит довольно простая идея, которую можно резюмировать следующим образом:

1. Создать из всего набора документов словарь уникальных лексем (tokens), например слов.

2. Построить из каждого документа вектор признаков, содержащий частотности вхождений каждого слова в определенный документ.

– Масштабирование данных с помощью tf-idf. Следующий подход – масштабировать признаки в зависимости от степени их информативности. Одним из наиболее распространенных способов такого масштабирования является метод «частота термина – обратная частота документа» (англ. term frequency – inverse document frequency, tf-idf). Идея этого метода заключается в том, чтобы присвоить большой вес термину, который часто встречается в конкретном документе, но при этом редко встречается в остальных документах корпуса. Если слово часто появляется в конкретном документе, но при этом редко встречается в остальных документах, оно, вероятно, будет описывать содержимое именно этого документа лучше других слов. Коэффициент tf-idf определяется как произведение частоты термина и обратной частоты документа:

$$tf - idf(t,d) = tf(t,d) \times idf(t,d), \qquad (9.1)$$

где $tf(t, d)$ – количество раз, которое термин t появляется в документе d, при этом обратную частоту документа $idf(t, d)$ можно вычислить как

$$idf(t,d) = \log \frac{n_d}{1+df(d,t)}, \qquad (9.2)$$

где n_d – это общее число документов и $df(d, t)$ – число документов d, которые содержат термин t. Отметим, что добавление константы 1 в знаменатель является факультативным и служит для назначения ненулевого значения терминам, которые встречаются во всех тренировочных образцах. Логарифм используется для того, чтобы низкие частоты документов гарантированно не получали слишком большого веса [105]. После предварительной обработки и представления текста в виде числовых векторов он становится доступен для анализа широкому спектру алгоритмов машинного обучения.

10. КРИТЕРИИ ТОЧНОСТИ

Различные классы задач машинного обучения требуют собственного набора критериев оценки, наилучшим образом учитывающего специфику оцениваемых данных. Причем сегодня можно встретить десятки метрик такой оценки, но при решении некоторых практических задач, учитывая сложность их решения и интерпретации результатов, приходится задавать собственные метрики оценки или использовать их некоторую оригинальную модификацию. При этом, очевидно, не существует какой-то единственной метрики, наилучшим образом позволяющей во всех случаях оценить точность и адекватность построенной модели, а каждая отдельная метрика позволяет оценить точность модели только в какой-то, относительно небольшой, ее части. Рассмотрим наиболее известные метрики точности, используемые в различных классах задач машинного обучения.

10.1. Метрики качества классификации

Метрики, которые считаются по матрице ошибок (confusion matrix):

- Accuracy – доля правильных ответов алгоритма;
- Precision (точность);
- Recall (полнота);
- F-score (F-мера);
- ROC/AUC (относительная операционная характеристика);
- Lift (прирост концентрации).

Метрики, которые считаются по матрице сопряженности:

- Kappa (каппа-индекс согласия).

Метрики, которые считаются с использованием вероятности принадлежности к классу:

- Logistic loss (логистическая потеря);
- Multiclass logistic loss.

Рассмотрим критерии качества в задачах классификации. Задачи классификации могут быть двух видов: бинарная классификация (всего 2 класса; например, определение спам / не спам, болен / здоров, кредитоспособен / некредитоспособен) и мультиклассовая классификация (например, определение типа заболевания, типа ландшафтного покрова).

Задачу бинарной классификации в общем случае можно сформулировать следующим образом. Дано множество классов $Y \in \{y_i\}$, где $y_i = \{0,1\}$, и множество объектов, которые необходимо отнести к одному из этих классов: $X \in (x_i), i \leq l$, где l – размер выборки. Алгоритм $a(x_i)$ возвращает произвольное вещественное число $p(x_i)$, которое с помощью порога t переводится в бинарный ответ (0 или 1): $a(x_i) = [p(x_i) > t]$. Например, при пороге $t = 0{,}5$, если классификатор возвращает $a(x_i) = 0{,}6$, то это класс 1, а если возвращает $a(x_i) = 0{,}4$, то это класс 0. Порог выбирается в зависимости от выборки.

Однако перед переходом к описанию самих метрик необходимо ввести понятие матрицы ошибок. Это способ разделения всех объектов классификации на четыре категории в зависимости от соотношения реального класса объекта и предсказанного классификатором. В терминах матрицы ошибок классы, которые определяет алгоритм, отмечены как «положительные» (класс 1) и «отрицательные» (класс 0). Тогда эти четыре категории объектов будут обозначаться следующим образом:

– true positive (TP), истинно-положительная, когда реальная метка класса 1 и предсказанный класс 1;

– false positive (FP), ложно-положительная, ошибка I рода, когда реальная метка класса 1, а предсказанный класс 0;

– false negative (FN), ложно-отрицательная, ошибка II рода, когда реальная метка класса 0, а предсказанный класс 1;

– true negative (TN), истинно-отрицательная, когда реальная метка класса 0 и предсказанный класс тоже 0.

Наглядно матрица ошибок представлена на рис. 46.

Ошибки в классификации бывают двух видов: False Positive и False Negative. В статистике первый вид ошибок называется ошибкой

I рода, а второй – ошибкой II рода. Примеры ошибок I и II рода в компьютерной безопасности:

– ошибка I рода: авторизованные пользователи классифицируются как нарушители;

– ошибка II рода: нарушители классифицируются как авторизованные пользователи.

	$y = 1$	$y = 0$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = 0$	True Positive (FP)	True Negative (TP)

Рис. 46. Матрица ошибок

Из матрицы ошибок можно получить простейшие метрики:

– Sensitivity (true positive rate, recall, probability of detection, чувствительность) – доля верно идентифицированных положительных результатов (из «positive» подмножества).

– Specificity (true negative rate, специфичность) – доля верно идентифицированных отрицательных результатов (из «negative» подмножества).

– Positive predictive values (precision) – доля верно идентифицированных положительных результатов (из всех полученных положительных результатов, в том числе ошибочно).

– Negative predictive values – доля верно идентифицированных отрицательных результатов (из всех отрицательных результатов, в том числе ошибочно).

В качестве примера расчета метрик рассмотрим выборку из 2 030 человек, из которых 30 больных (positive, положительный класс) и 2 000 здоровых (negative, отрицательный класс). На рис. 47 визуально представлен расчет метрик по матрице ошибок для данной выборки.

		Пациенты с раком кишечника (подтверждено эндоскопией)		
		Положительное состояние	Негативное состояние	
Результаты теста на скрытую кровь в экскрементах	Положительный результат	True positive (TR) = 20	False positive (FR) = 180	Положительное прогностическое значение TP/(TP + FP) = 10%
	Отрицательный результат	False negative (FR) = 180	True negative (TN) = 1 820	Отрицательное прогностическое значение TN/(FN + TN) ~ 99,5%
		Чувстви-тельность TP/(TP + FN) ~ 67%	Специфичность TN/(FP + TN) = 91%	

Рис. 47. Пример расчета метрик по матрице ошибок

Accuracy. Accuracy – доля правильных ответов алгоритма, самый очевидный и простой способ оценки качества:

$$accuracy = \frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i]. \quad (10.1)$$

В терминах матрицы ошибок долю правильных ответов можно выразить так:

$$accuracy = \frac{TP+TN}{TP+FN+FP+TN}. \quad (10.2)$$

Однако в большинстве случаев этот способ наиболее бесполезный, как можно заметить на любом примере с несбалансированными классами. Например, в задаче оценки спам-фильтра, допустим, из 100 не-спам писем классификатор определил 90 верно (т.е. TN = 90, FP = 10), а из 10 спам-писем 5 он также определил верно (т.е. TP = 5, FN = 5). В таком случае accuracy = 86,4, но если все письма классификатор предскажет как не-спам, то точность получится более высокой: accuracy = 90,9. Однако такая модель практической предсказательной ценностью обладать не будет.

Precision, Recall, F1 score. Более информативными являются показатели качества на каждом из классов. Precision (точность) – доля верных идентифицированных положительных результатов (из всех полученных положительных результатов, в том числе ошибочно):

$$precision = \frac{TP}{TP+FP}. \quad (10.3)$$

Recall (полнота) – доля верно идентифицированных положительных результатов (из 'positive' подмножества):

$$recall = \frac{TP}{TP+FN}. \quad (10.4)$$

Если $a(x_i) = [p(x_i) > 1]$, то обычно при увеличении t точность будет расти, а полнота падать. Recall отражает способность алгоритма определять нужный класс вообще, а recall – способность отличать один класс от других. При этом точность и полнота не зависят от соотношения размеров классов. Даже если объектов положительного класса на порядки меньше, чем объектов отрицательного класса, данные показатели будут корректно отражать качество работы и поэтому применимы на несбалансированных выборках.

F-score (F-мера) – агрегированный критерий качества, объединяющий две предыдущие метрики:

$$F = 2\frac{precision \times recall}{precision+recall}. \quad (10.5)$$

F-мера достигает максимума (1) при полноте и точности, равных единице, и близка к нулю, если один из аргументов близок к нулю.

На практике часто возникают задачи, связанные с выбором подмножества. Заказчика может интересовать вопрос, насколько выгоднее работать с этим подмножеством по сравнению со всем множеством. Например, если при рассылке предложений о кредите клиентам из подмножества и всем клиентам будет получаться одна и та же доля откликнувшихся, то подмножество не будет представлять особой ценности. Формально это измеряется с помощью прироста концентрации (*lift*), который равен отношению точности к доле положительных объектов в выборке размера l:

$$lift = \frac{precision}{(TP+FN)/l} \quad (10.6)$$

Эту величину можно интерпретировать как улучшение доли положительных объектов в данном подмножестве относительно доли в случайно выбранном подмножестве такого же размера.

10.2. Гипотеза A/B

A/B-тестирование – метод исследования, при котором эксперимент проводится с использованием контрольной группы признаков (A) и набора тестовых групп признаков (B), в которых модифицированы показатели (как правило, один), позволяя выяснить, какие именно показатели влияют на изменение целевого фактора.

Наиболее частым примером использования такого тестирования является проверка эффективности корректировки пользовательского интерфейса и сценариев использования web-приложений, которая позволяет определить маркетинговую эффективность вносимых изменений на некоторой тестовой подгруппе пользователей. Целевыми показателями при этом являются посещаемость сайта (отдельных страниц), конверсия (доход), длительность сеанса пользователя на ресурсе и т.п. Анализ таких показателей может быть выполнен, например, инструментом Google Analytics. Основным требованием к применению такого подхода является обеспечение статистической достоверности результатов, что достигается достаточным размером выборки и заданным уровнем статистической значимости.

10.3. Каппа-индекс согласия

Для случаев мультиклассовой классификации вместо матрицы ошибок строится матрица сопряженности (contingency table). Такая матрица отражает соответствие двух переменных по каждому из классов: реального класса объекта и предсказанного алгоритмом. Например, на рис. 48 представлен пример матрицы сопряженности для классификатора, определяющего тип поверхности.

Эта матрица нужна также для построения такой метрики, как Каппа-индекс согласия. В общем случае каппа-индекс используется для оценки степени согласия оценки объектов двумя экспертами.

В случае же оценки качества алгоритма эта метрика отражает степень согласованности ответов алгоритма с реальными метками классов объектов.

Прогноз \ Истина	Асфальт	Бетон	Трава	Дерево	Здание	Сумма
Асфальт	2 385	4	0	1	4	2 394
Бетон	0	332	0	0	1	333
Трава	0	1	908	8	0	917
Дерево	0	0	0	1 084	9	1 093
Здание	12	0	0	6	2 053	2 071
Сумма	2 397	337	908	1 099	2 067	6 808

Рис. 48. Матрица сопряженности

Рассчитывается каппа-индекс следующим образом:

$$\widehat{K} = \frac{N \times \sum_{i=1}^{k} x_{ii} - \sum_{i=1}^{k}(x_{i+} \times x_{+i})}{N^2 - \sum_{i=1}^{k}(x_{i+} \times x_{+i})}, \quad (10.7)$$

где x_{i+} – сумма элементов в i-й строке, x_{+i} – сумма элементов в i-м столбце, N – общее количество элементов.

В отношении интерпретации значения каппа-индекса часто идут споры: какой диапазон значений считать признаком хорошего качества классификатора. Однако в общем случае можно ориентироваться на следующие показатели:

- $\widehat{K} < 0{,}2$ – плохо;
- $0{,}2 < \widehat{K} > 0{,}4$ – слабо;
- $0{,}4 < \widehat{K} < 0{,}6$ – средне;
- $0{,}6 < \widehat{K} < 0{,}8$ – хорошо;
- $0{,}8 < \widehat{K} < 1{,}0$ – отлично.

10.4. ROC-кривая

Такие метрики, как точность, полнота или F-мера, применимы к классификаторам при конкретном выборе порога *t*, относительно которого все вещественные ответы алгоритма сводятся к бинарному значению. Что если мы будем менять порог в некотором диапазоне значений, варьируя тем самым и значения точности и полноты? В таком случае применима такая интегральная характеристика, как площадь под ROC-кривой, или AUC ROC (*Area Under Curve ROC – Receiver / Relative Operator Characteristic*). Строится эта кривая следующим образом.

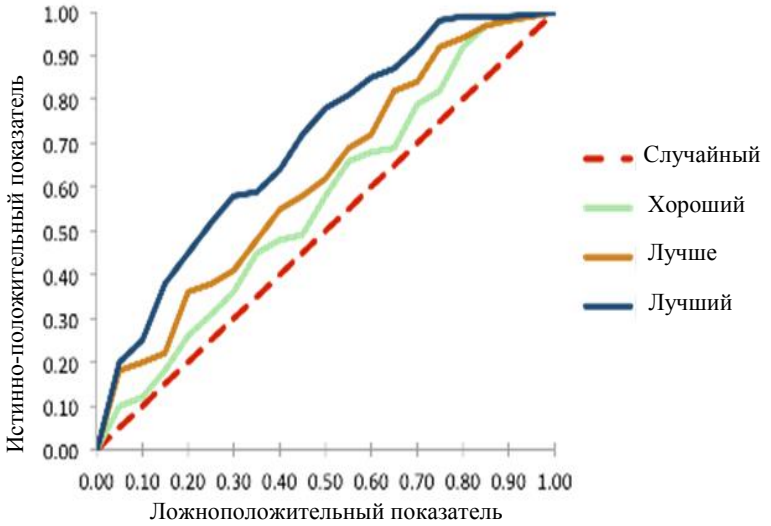


Рис. 49. ROC-кривая

Рассмотрим набор возможных порогов $t \in \{0, l+1\}$ и для каждого из них посчитаем два значения: долю ложно-положительных ответов (false positive rate, *FPR*) и долю истинно-положительных ответов (true positive rate, *TPR*):

$$FPR = \frac{FP}{FP+TN}, \qquad (10.8)$$

$$TPR = \frac{TP}{TP+FN}. \quad (10.9)$$

По этим значениям строится ROC-кривая: по одной оси наносятся значения TPR, а по другой – FPR (рис. 49).

Максимальный порог t даст классификатор с $TPR = 0, FPR = 0$, минимальный порог даст $TPR = 1, FPR = 1$. Таким образом, ROC-кривая – это кривая с концами в точках $(0, 0)$ и $(1, 1)$, которая последовательно соединяет точки, соответствующие всем возможным порогам t в заданном диапазоне. Площадь под данной кривой принимает значения от 0 до 1. Если алгоритм не будет давать ошибок, то AUC-ROC для него будет равен 1. Для бесполезного «случайного» алгоритма AUC-ROC будет равен 0,5.

10.5. Метрика качества прогноза временного ряда

Выше отмечено, что одной из наиболее распространенных задач, в которой применяется машинное обучение, является задача регрессионного анализа и прогнозирования значений некоторого временного ряда при его экстраполяции. Как и большинство классов иных задач, оценка точности подобного вида модели требует собственного набора критериев оценки, наилучшим образом учитывающего специфику оцениваемых данных. Среди таких критериев выделяют следующие:

- MAE – средняя абсолютная ошибка;
- MAPE – средняя относительная ошибка;
- ME – средняя ошибка;
- MSE – среднеквадратическая ошибка;
- Max абсолютной ошибки;
- Max относительной ошибки;
- SMAPE – симметричная MAPE.

Если за Y_i принять фактическое значение наблюдаемого фактора в момент времени i, а за Yp_i – прогнозное значение наблюдаемого фактора в момент времени i, то критерии могут быть представлены математическими формулами (табл. 13).

Таблица 13

Метрика качества прогноза временного ряда

Обозначение / название	Формула	Единица измерения	Описание
MAE – средняя абсолютная ошибка	$\frac{1}{N}\sum_{i=1}^{N}\lvert Y_i - Yp_i\rvert$	Единица измерения самого фактора	Всегда неотрицательная, показывает среднюю арифметическую ошибку по наблюдениям. Применяется при малых значениях наблюдаемого фактора
MAPE – средняя относительная ошибка	$\frac{1}{N}\sum_{i=1}^{N}\frac{\lvert Y_i - Yp_i\rvert}{\lvert Y_i\rvert}$	Доли или проценты	Самая применяемая и простая метрика для ошибки прогнозирования. Всегда неотрицательная. Показывает среднюю долю (%) отклонения прогноза от факта. Чувствительна к завышенным прогнозам
ME – средняя ошибка	$\frac{1}{N}\sum_{i=1}^{N}(Y_i - Yp_i)$	Единица измерения самого фактора	Применяется чаще в финансах и логистически-складских задачах. По знаку показателя определяют «завышенность» или «заниженность» построенного прогноза
MSE – средняя абсолютная ошибка		Квадратная единица измерения самого фактора	Всегда неотрицательная. Часто используют корень MSE для сопоставимости единиц измерения. За счет квадрата ошибка более чувствительна к большим разовым отклонениям
Max – средняя относительная ошибка		Единица измерения самого фактора	Выдает отличие факта от прогноза. Применяется, когда критично отклонение выше какой-то величины
Max – средняя ошибка		Доли или проценты	Аналогично, только относительная величина. Применяется, когда критично отклонение на какой-либо %
SMAPE		Доли или проценты	Доработанная MAPE, более симметрична относительно завышенных и заниженных прогнозов

10.6. Метрики качества кластеризации

В отличие от задач классификации в задачах кластеризации отсутствуют размеченные данные, с которыми можно было бы использовать интуитивно понятные метрики качества.

Для начала необходимо ввести несколько понятий, используемых при расчетах метрик качества кластеризации. Пусть X – кластеризуемое множество, N – количество элементов в множестве X, c – число кластеров, n_{c_i} – число элементов в кластере C_i, $v_i = \frac{\sum_{x \in c_i} x}{n_{c_i}}$ – центр кластера c_i, $\bar{X} = \frac{1}{N}\sum_{j=1}^{N} x_j$ – центральный элемент множества, $\bar{v} = \frac{1}{c}\sum_{i=1}^{c} v_i$ – центр центров, $\dim$ – размерность множества X.

Hubert Γ statistic. Данная метрика отражает среднее расстояние между объектами разных кластеров:

$$\Gamma = \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j)Q(i,j), \qquad (10.10)$$

где $M = \frac{n \times (n-1)}{2}$, $P(i,j)$ – матрица близости, $Q(i,j)$ – расстояние между центрами кластеров v_{c_i} и v_{c_j}, к которым принадлежат элементы x_i и x_j соответственно. Чем больше значение метрики – тем лучше.

CS-индекс (отделимость кластеров). CS-индекс измеряет отношение максимального расстояния между точками в кластере к минимальному межкластерному расстоянию. Оптимальная структура кластеров характеризуется меньшим показателем CS:

$$\mathrm{CS} = \frac{\sum_{i=1}^{c}\left\{\frac{1}{n_{c_i}}\sum_{x_j, x_k \in c_i}\max(\|x_j - x_k\|)\right\}}{\sum_{i=1}^{c}\min_{j \neq i}(\|v_i - v_j\|)}. \qquad (10.11)$$

VNND-индекс (индекс ближайших соседей). $\mathbf{d}_{\min}(x_j) = \min_{y \in c_i}(\|x_i - y\|)$ – расстояние от элемента x_j до его ближайшего соседа.

$\overline{d_{\min}(c_i)} = \frac{1}{n_{c_i}}\left(\sum_{x_j \in c_i} \mathrm{d}_{\min}(x_i)\right)$ – среднее расстояние между ближайшими соседями в кластере c_i.

Отклонения от расстояния между ближайшими соседями:

$$V(c_i) = \frac{1}{n_{c_i}-1}\sum_{x_j \in c_i}\left(d_{\min}(x_j) - \overline{d_{\min}(c_i)}\right)^2. \qquad (10.12)$$

Итоговое значение индекса определяется так:

$$\mathrm{VNND} = \sum_{i=1}^{c} V(c_i). \qquad (10.13)$$

11. ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

11.1. Принципы высокопроизводительных вычислений

Применение сложных, в том числе интеллектуальных, подходов к обработке данных, представленных мега-, гига- и терабайтами, требует особого внимания к обеспечению их высокопроизводительной обработки. Выше отмечено, что в условиях стремительного наращивания объемов данных многие стандартные вычислительные средства и традиционные алгоритмические подходы для такой сложной и затратной обработки в значительной степени непригодны. Сформулируем несколько основных принципов организации высокопроизводительных вычислений, которые следует учитывать в практическом применении методов Data Mining.

Первый принцип организации высокопроизводительных вычислений заключается в увеличении производительности обработки данных за счет совершенствования вычислительной эффективности алгоритмов, позволяя применять достаточно сложную обработку данных, но в некоторых случаях доступную даже для ПЭВМ со стандартными характеристиками.

Сегодня доступны различные вычислительные мощности. Вычислительные кластеры организуют как на базе стандартных недорогих ПЭВМ, объединенных в локальную вычислительную сеть, так и в специализированных центрах, оснащенных суперкомпьютерной многопроцессорной техникой [60]. Поэтому *второй принцип* определяет целесообразность использования таких подходов к обработке данных, которые применимы как на дорогостоящей суперкомпьютерной технике, так и на недорогих ПЭВМ, объединенных в сети. При этом подразумевается, что высокопроизводительная обработка достигается двумя способами, которые могут быть использованы совместно.

Первый способ предполагает обработку исключительно за счет применения более высокопроизводительной вычислительной техники. Второй – адаптацию существующих подходов для возможности не только канонического *параллельного*, но и так называемого *распределенно-параллельного* исполнения. Обширный класс *параллельных вычислений* характеризуется возможностью одновременного решения одной вычислительной задачи путем ее декомпозиции [60]. Очевидно, параллельные вычисления могут быть реализованы на одной ПЭВМ в многопроцессном режиме под управлением многозадачной операционной системы [75]. Параллельные вычисления на нескольких вычислительных узлах (например, нескольких ПЭВМ, объединенных в локальную вычислительную сеть, или нескольких процессорах суперкомпьютера) терминологически относят к *распределенным вычислениям.* Именно поэтому здесь и далее, применяя термин *распределенно-параллельные вычисления*, будем иметь в виду вычислительный процесс, реализуемый не только как параллельный, но и как распределенный.

Практическая организация высокопроизводительных распределенно-параллельных вычислений с использованием указанных выше принципов требует ряда важных пояснений.

Организация распределенных вычислений возможна с использованием множества различных подходов и архитектур [63]. Однако при всем многообразии возможных вариантов принципиально отличают два различных класса построения вычислительных систем – *системы с общей* (*разделяемой*) *памятью* и *системы с распределенной памятью* [60].

К первому варианту построения вычислительных систем относят системы с симметричной архитектурой и симметричной организацией вычислительного процесса – *SMP-системы* (англ. *Symmetric MultiProcessing*). Поддержка *SMP*-обработки данных присутствует в большинстве современных ОС при организации вычислительных процессов на многопроцессных (многоядерных) ПЭВМ [75]. Однако разделяемая память требует решения вопросов синхронизации и исключительного доступа к разделяемым данным, а построение высокопроизводительных систем в этой архитектуре

затруднено технологически сложностями объединения большого числа процессоров с единой оперативной памятью [60, 75].

Второй вариант предполагает объединение нескольких вычислительных узлов (ВУ) с собственной памятью в единой коммуникационной среде, взаимодействие между узлами в которой осуществляется путем пересылки сообщений. Причем такими ВУ могут быть как стандартные недорогие ПЭВМ, так и процессоры дорогостоящего суперкомпьютера. Такая архитектура построения вычислительной системы характеризуется бóльшими возможностями построения высокопроизводительных вычислительных систем и значительного масштабирования доступных вычислительных мощностей. Однако, в отличие от *SMP*-систем, здесь более высокая *латентность* (существенные накладные коммуникационные расходы), негативно влияющая на производительность системы и требующая более внимательной организации распределенно-параллельной обработки [60]. Причем в случае построения вычислительного кластера на базе ПЭВМ такая латентность, как правило, существенно выше, чем при построении кластера на базе суперкомпьютера, за счет значительно более развитой коммуникационной среды.

В соответствии с классификацией архитектур вычислительных систем М. Флинна наибольшее распространение на практике получили две модели параллельных вычислений – *Multiple Process* / *Program – Multiple Data* (*MPMD*, множество процессов / программ, множество данных) и *Single Process* / *Program – Multiple Data* (*SPMD*, один процесс / программа, множество данных) [60]. Модель *MPMD* предполагает, что параллельно выполняющиеся процессы исполняют различные программы (процессы, потоки) на различных процессорах. Модель *SPMD* обусловливает работу параллельно выполняющихся программ (процессов, потоков), но исполняющих идентичный код при обработке отличных (в общем случае) массивов данных.

Очевидно, организацию распределенных вычислений целесообразно реализовать с использованием программ с параллельной обработкой данных на основе модели *SPMD*. Во-первых, это более практично из-за необходимости разрабатывать и отлаживать лишь

одну программу, а во-вторых, такие программы, как правило, применимы и при традиционном последовательном исполнении, что расширяет области их практической применимости.

Кроме вышеизложенных особенностей организации высокопроизводительных вычислений в различных архитектурах и на основе различных моделей, при разработке алгоритмического обеспечения параллельной обработки необходимо:

– определить фрагменты вычислений, которые могут быть исполнены параллельно;

– распределить данные по модулям локальной памяти отдельных ВУ;

– согласовать распределение данных с параллелизмом вычислений.

Важным является выполнение всех перечисленных условий, так как иначе значительные фрагменты вычислений не удастся представить как параллельно исполняемые и реализация алгоритма в распределенно-параллельной архитектуре не позволит добиться роста производительности. Задачи распределения данных по модулям памяти и задача согласования распределения данных важны для обеспечения низкой латентности между ВУ и обеспечения возможностей масштабирования системы.

Помимо этого, при построении такого рода вычислительных системы важно иметь четко измеримые показатели их эффективности. Первым таким критерием можно назвать параллельное ускорение, которое показывает ускорение выполнения по сравнению с последовательным вариантом. Вычисляется оно для количества параллельных процессов p следующим образом:

$$S_p(n) = {}^{T_1(n)}\Big/_{T_p(n)}, \qquad (11.1)$$

где n – параметр вычислительной сложности задачи (например, количество входных данных), T – время решения задачи. Фактически это отношение времени выполнения на скалярной ПЭВМ к времени выполнения параллельного алгоритма. В общем случае $S_p(n) < p$, так как параллельный алгоритм не может обеспечить идеальные

балансировку ВУ и латентность. Максимальное значение $S_p(n) = n$ (линейное ускорение).

Вторым критерием является эффективность использования параллельным алгоритмом ВУ. Этот критерий определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально используются для решения задачи, и вычисляется следующим образом:

$$E_p(n) = {T_1(n)}/{pT_p(n)} = {S_p(n)}/{p}. \qquad (11.2)$$

Как видно, в лучшем случае $S_p(n) = p$ и $E_p(n) = 1$.

Таким образом, повышение производительности обработки данных может быть реализовано путем:

– наращивания мощности аппаратной инфраструктуры;

– применения распределенно-параллельных подходов к организации вычислений на кластерах различной стоимости и конфигурации, различных моделей параллельных вычислений и архитектур организации вычислительного процесса;

– увеличения вычислительной эффективности алгоритмов за счет оптимизации, использования приближенных эвристик вместо трудоемких расчетов и т.п.

11.2. Особенности построения вычислительного кластера

Понятия «параллельное вычисление» и «распределенное вычисление» похожи тем, что в обоих случаях имеется в виду, что существует несколько потоков или процессов, работающих сообща, чаще всего над одной задачей. Обычно считается, что при параллельном вычислении процессы или потоки работают в разделяемой памяти без пересылки данных по сети, в распределенной системе существует несколько узлов, между которыми идет обмен данными по сети.

Характер решаемой задачи определяет необходимые параметры вычислительной системы как на уровне оборудования, так и на уровне программного обеспечения. К параметрам оборудования относятся такие характеристики, как объем общего или локального хра

нилища данных, объем оперативной памяти, число и параметры процессоров, устройство сети, ее пропускная способность и задержки.

На уровне программного обеспечения определяются конкретная логика хранения и передачи данных, а также перераспределения задач, и, что немаловажно, контроль нагрузки и отказоустойчивости вычислительной системы.

Определение параметров проектируемой вычислительной системы и логики ее работы невозможно без определения параметров решаемой задачи. К таким параметрам относятся:

– тип требуемого анализа: пакетная обработка (есть некий накопленный объем данных, который нужно обработать), обработка в реальном времени (данные будут обработаны сразу после поступления), микропакетная обработка (занимает промежуточное положение между двумя предыдущими типами);

– объем данных: объем поступающих данных, скорость поступления и объем имеющихся данных (имеет особое значение, поскольку критичным является его соотношение с объемом имеющегося хранилища);

– тип данных: структурированные данные (таблицы баз данных), неструктурированные (текст, изображения и видео) и полуструктурированные (JSON и XML);

– скорость выдачи результата – рамки допустимого времени выдачи результата.

Далее при рассмотрении вариантов построения вычислительных систем будем пользоваться понятием роли ВУ. Каждая из приведенных ниже ролей представляет собой набор некоторых функций, связанных общей задачей. Роль может быть реализована как в виде отдельного приложения, так и как часть приложения.

1. *Менеджер задач* – принятие решений о передаче конкретной задачи конкретному исполнителю, мониторинг выполнения.

2. *Исполнитель задач* – выполнение вычислений в рамках задачи и выделенных ресурсов, возврат результата менеджеру задач.

3. *Менеджер ресурсов* – принятие решения о выделении процессорной мощности и оперативной памяти под конкретную задачу, мониторинг доступных ресурсов, ведение перечня доступных ресурсов.

4. *Менеджер данных* – принятие решения о месте хранения данных или фрагментов данных, выдача данных по запросу путем сбора из конкретных мест хранения, если требуется.

5. *Хранилище данных* – место хранения данных.

Варианты построения вычислительной системы. Выделяют три основных класса задач, требующих учета при построении вычислительного кластера:

1. Небольшой объем данных, пакетная обработка, отсутствие требований ко времени выдачи результата.

2. Большой объем данных, пакетная обработка, отсутствие требований к времени выдачи результата.

3. Большой объем данных, пакетная обработка, результат должен быть получен как можно быстрее.

Рассмотрим каждый из этих вариантов.

Небольшой объем данных, пакетная обработка, отсутствие требований ко времени выдачи результата. Под небольшим объемом будем иметь в виду возможность хранить данные на одном компьютере. При таких требованиях вполне достаточно применения традиционных подходов, предполагающих простое перераспределение задач по узлам. В зависимости от требуемого объема дискового пространства и требований к производительности может быть использован как один вычислительный узел (ВУ), так и несколько. В обоих случаях схема взаимодействия процессов будет примерно такой, как показано на рис. 50.

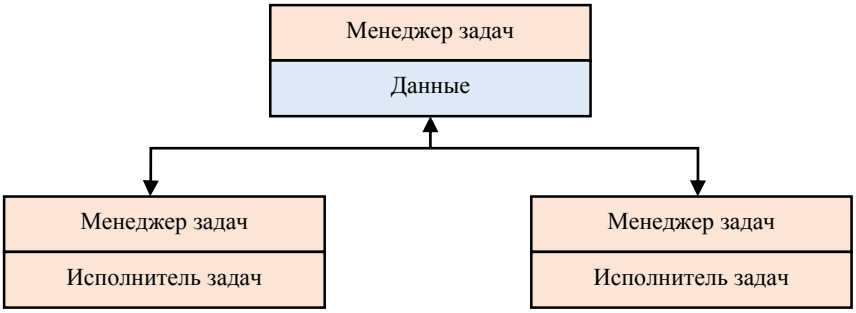


Рис. 50. Схема варианта построения вычислительной системы

В рамках данного подхода работают такие известные средства, как OpenMP и MPI. OpenMP используется для параллельных вычислений на одном вычислительном узле с помощью многопоточности, а MPI может применяться и на одном ВУ, и на нескольких и порождает процессы. В такой схеме исполняемый программный код содержит инструкции по разделению и обмену данными. В MPI данные могут как пересылаться между ВУ по сети, так и браться из разделяемых сетевых папок.

Большой объем данных, пакетная обработка, отсутствие требований к времени выдачи результата. В данной ситуации возникает потребность в распределенном хранении данных, а значит, требуется включить в систему менеджера данных. Также необходимо помнить, что значительный объем данных (например, огромный текст с необходимостью оценки частотности слов в нем) не получится загрузить в оперативную память, следовательно, нужно позаботиться о взаимодействии ОЗУ и ПЗУ (рис. 51).

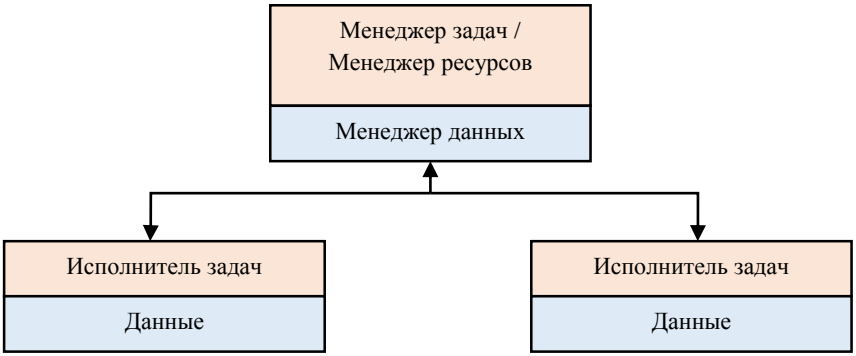


Рис. 51. Схема варианта построения вычислительной системы

Также в распределенной системе полезно иметь менеджер ресурсов, который собирает информацию о том, насколько загружен каждый из узлов; такая информация важна для балансировки загрузки.

Приведенной на рис. 52 структуре соответствует Hadoop с MapReduce версии 1. Применяемая в Hadoop HDFS (распределенная файловая система Hadoop, англ. *Hadoop Distributed File System*)

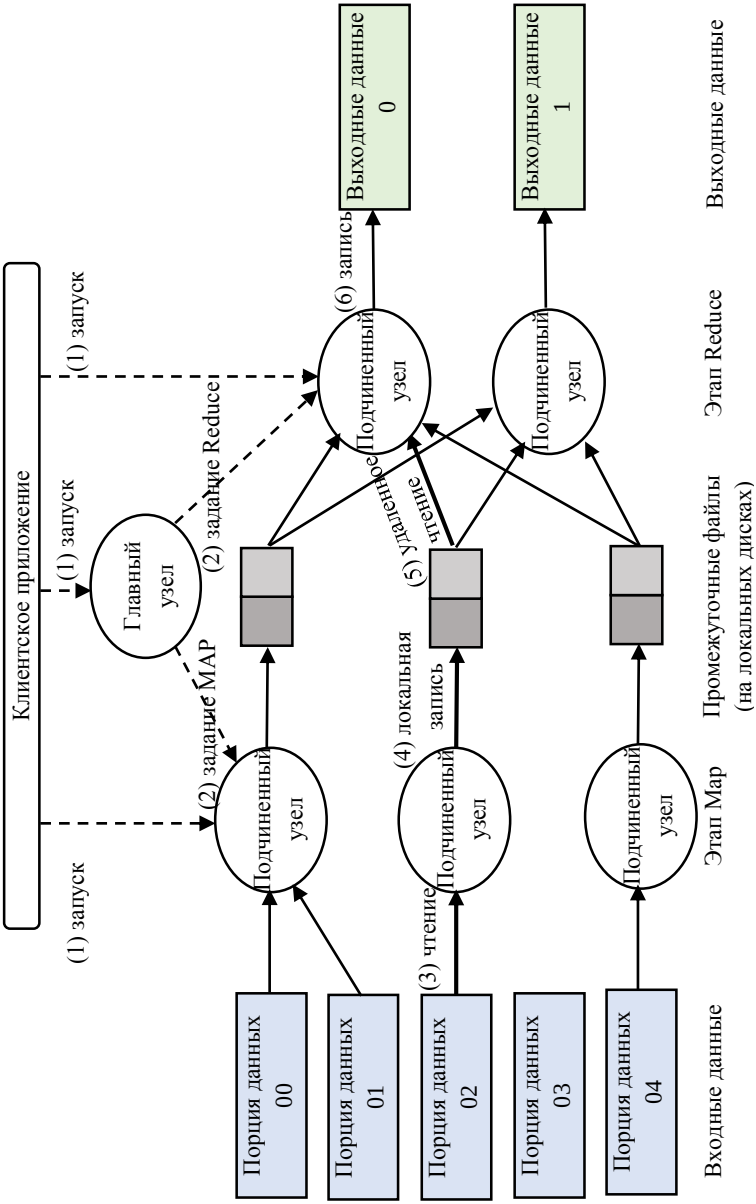


Рис. 52. Фреймворк MapReduce

разделяет файлы на части и следит за тем, где находятся части того или иного запрашиваемого файла. За запись и выдачу файла отвечает менеджер данных.

В качестве системы, выполняющей задачи, используется фреймворк MapReduce, который обеспечивает автоматическое распараллеливание и распределение задач по узлам (см. рис. 52). В рамках данного фреймворка задача разбивается на три стадии, которые могут итеративно повторяться много раз: map (выборка), shuffle (перегруппировка) и reduce (объединение по заданному принципу). После каждой операции идет сохранение данных на диск, что снимает нагрузку с оперативной памяти. Ресурсом в данном случае является число так называемых слотов – map или reduce.

Важными показателями вычислительной системы являются также масштабируемость и отказоустойчивость. Недостаток приведенной схемы – наличие единой точки выхода из строя – менеджера задач. Также менеджер задач здесь выполняет распределение всех поступающих задач, что также накладывает ограничения на их число.

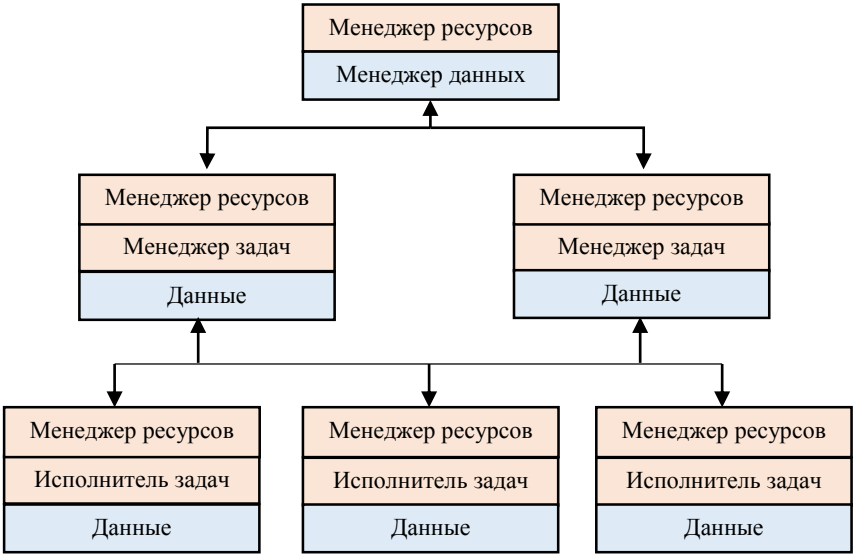


Рис. 53. Вторая версия фреймворка MapReduce

Выходом из данной ситуации может быть разделение функций управления ресурсами и управления задачами. Эта идея реализована во второй версии MapReduce, где функции Job Tracker по управлению ресурсами переданы Resource Manager и NodeManager (на локальных узлах), а управление задачами – Application Master (рис. 53).

Также в новой версии изменилось понятие ресурса, теперь под ним подразумевается так называемый контейнер, который определяет ресурс процессора и объем оперативной памяти, выделяемой на задачу.

Большой объем данных, пакетная обработка, результат должен быть получен как можно быстрее. В случае необходимости получения результата как можно быстрее, необходимо пересмотреть организацию вычислений и, возможно, отказаться от некоторых архитектурных решений, обеспечивающих большую отказоустойчивость в пользу решений, обеспечивающих быстродействие. Одним из таких решений может быть отказ от сохранения промежуточных результатов и перенос вычислений в оперативную память (рис. 54).

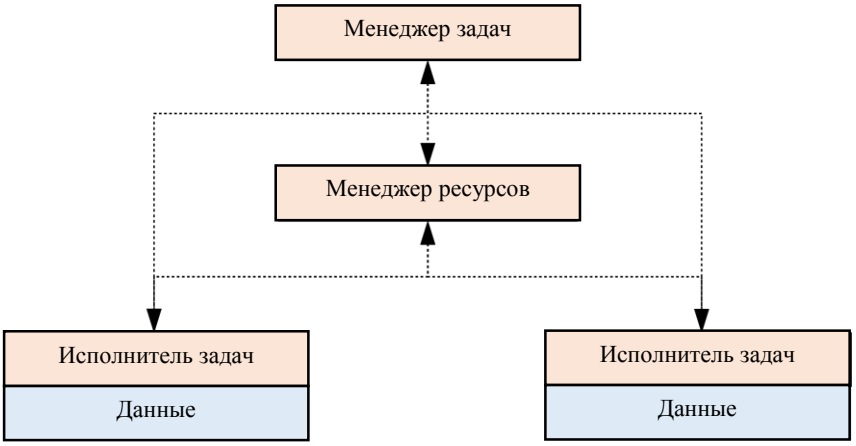


Рис. 54. Схема варианта построения вычислительной системы без сохранения промежуточных результатов

Тогда схема вычислительного кластера упрощается, как показано на рис. 55. В частности, данной схеме соответствует система Spark. В данной системе используется сторонний менеджер ресурсов, который может быть размещен как на одной машине с менеджером задач, так и на выделенной.

В случае структурированных данных также часто применяется база данных на так называемой лямбда-архитектуре. Лямбда-архитектура имеет структуру, состоящую из трех уровней:

– пакетный уровень;
– сервисный уровень;
– уровень ускорения.

Пакетный уровень представляет собой архив сырых данных, чаще всего на базе Hadoop, хотя встречается и в форме OLAP-хранилища (например, Vertica). Этот уровень, как понятно из названия, поддерживает и оперирует пакетной передачей данных. Важно заметить, что старые данные здесь остаются неизменными, происходит лишь добавление новых.

Сервисный уровень индексирует пакеты и обрабатывает результаты вычислений, происходящих на пакетном уровне. За счет индексации и обработки поступающей информации результаты немного отстают во времени.

Уровень ускорения отвечает за обработку данных, поступающих в систему в реальном времени. Представляет собой совокупность складов данных, в которых данные находятся в режиме очереди, в потоковом или в рабочем режиме. На этом уровне компенсируется разница в актуальности данных, а в отдельные представления реального времени добавляется информация с коротким жизненным циклом (чтобы исключить дублирование данных). Эти представления обрабатывают свои запросы параллельно с сервисным уровнем.

Как можно заметить, в лямбда-архитектуре наряду с пакетной обработкой используются способы повышения скорости получения информации, но за счет потери их точности из-за неполного набора использующихся данных.

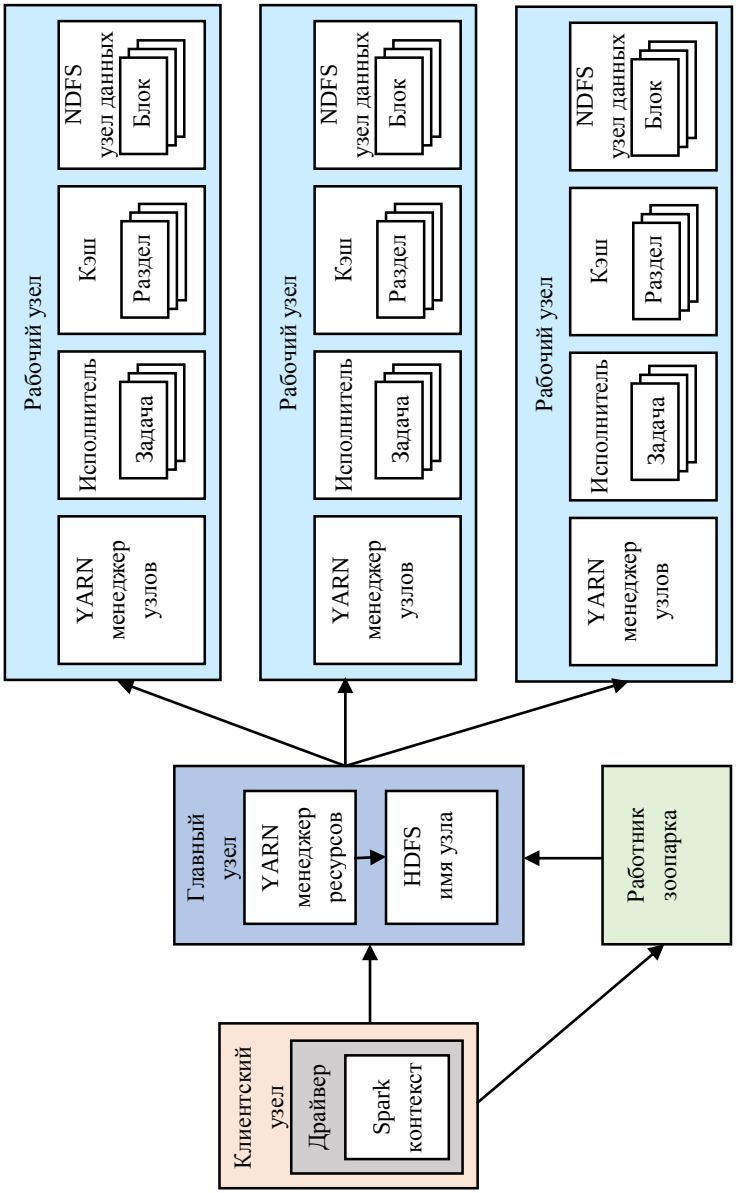


Рис. 55. Схема вычислительного кластера

Небольшой объем поступающих данных, обработка результата в реальном времени (streaming). Иногда возникает потребность обрабатывать данные оперативно, как только они поступают в систему. В этом случае главным становится скорость прохождения новых данных в системе, а также отказоустойчивость (рис. 56).

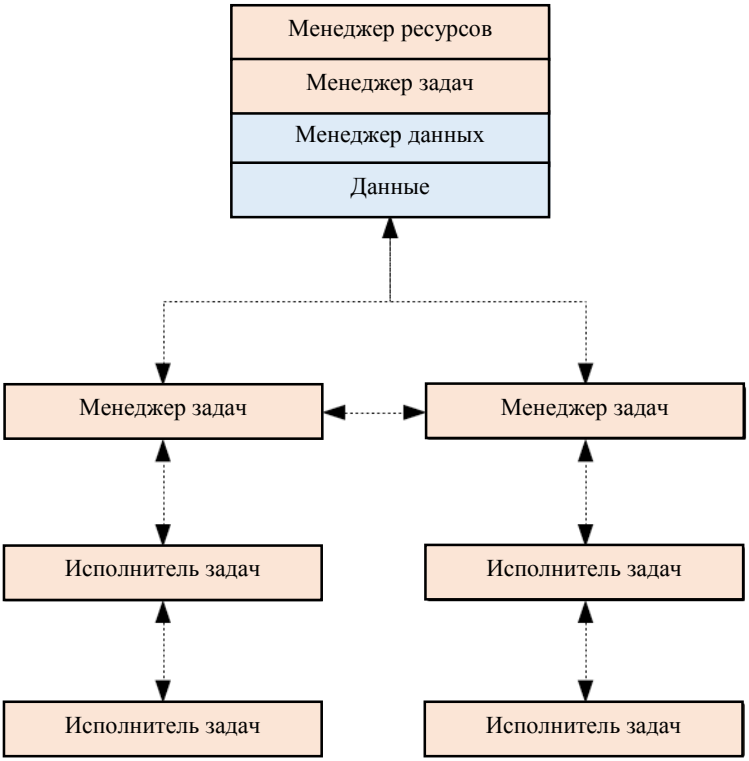


Рис. 56. Схема системы потоковой обработки данных Apache Storm.

Приведенной схеме соответствует система потоковой обработки данных Apache Storm. В данной системе определяется топология, которая задает последовательность обработки данных. Топология состоит из двух основных элементов – источников данных (spout) и обработчиков (bolt), объединенных определенной схемой. Данные

в системе пересылаются в виде кортежей (tuple). В системе обработчики распределяются по исполнителям менеджером задач. Система Storm масштабируема.

Защита от сбоев обеспечивается сохранением состояния в кластере Zookeeper, что позволяет восстановить состояние после сбоя. Так же как и в других системах обработки, если какой-либо из обработчиков отказывает, задачи переадресуются на другие обработчики.

11.3. Среды и инструменты высокопроизводительных вычислений

Различные технологии предполагают использование различных аппаратных платформ с точки зрения использования общей или разделенной памяти, гомогенной или гетерогенной среды, а также использования CPU или GPU. Выбор конкретной технологии зависит от решаемой задачи и доступной аппаратной платформы. Также стоит отметить, что различные технологии предполагают различную степень изменения исходного кода, что важно учитывать и соотносить со сроками разработки.

Одной из лидирующих технологий, относящихся к классу Big Data, является открытая платформа Hadoop, позволяющая обрабатывать огромные объемы данных в распределенной среде и обеспечивающая, таким образом, существенное снижение стоимости такого решения.

Apache Hadoop состоит из двух ключевых компонентов: распределенная файловая система Hadoop (Hadoop Distributed File System, HDFS), которая отвечает за хранение данных, и программный фреймворк MapReduce, выполняющий роль системы распределения и распараллеливания задач по узлам.

HDFS многократно копирует блоки данных и распределяет эти копии по вычислительным узлам кластера, тем самым обеспечивая высокую надежность и скорость вычислений. Архитектурно HDFS-кластер состоит из NameNode-сервера и DataNode-серверов, которые хранят непосредственно данные. NameNode-сервер управляет

пространством имен файловой системы и доступом клиентов к данным. Чтобы разгрузить NameNode-сервер, передача данных осуществляется только между клиентом и DataNode-сервером.

MapReduce – это фреймворк и модель распределенных вычислений, предназначенных для высокоскоростной обработки больших объемов данных на больших параллельных кластерах вычислительных узлов. В рамках данного фреймворка задача разбивается на три стадии, которые могут итеративно повторяться много раз: map (выборка данных), shuffle (перегруппировка) и reduce (объединение по заданному принципу).

Современный стандарт (вторая версия) этого фреймворка обеспечивает автоматическое распараллеливание и распределение задач, имеет встроенные механизмы сохранения устойчивости и работоспособности при сбое отдельных элементов, а также обеспечивает чистый уровень абстракции для программистов. Преимущество MapReduce заключается в том, что он позволяет распределенно производить операции предварительной обработки и свертки.

Следующее поколение вычислительной платформы Hadoop – **Apache YARN** (Yet Another Resource Negotiator). YARN был представлен в Hadoop 2.x. Он предоставляет различные механизмы обработки данных, такие как обработка графа, интерактивная обработка, обработка потока, а также пакетная обработка для запуска и обработки данных, хранящихся в HDFS. Помимо управления ресурсами, YARN также используется для планирования работ. YARN расширяет возможности Hadoop для других развивающихся технологий, поэтому они могут воспользоваться преимуществами HDFS.

YARN Apache также рассматривается как операционная система данных для Hadoop 2.x. Архитектура Hadoop 2.x на основе YARN обеспечивает платформу обработки данных общего назначения, которая не ограничивается только MapReduce. Это дает возможность Hadoop обрабатывать другую целевую систему обработки данных, отличную от MapReduce. YARN позволяет запускать несколько разных фреймворков на том же оборудовании, где развернут Hadoop.

Apache YARN Framework состоит из мастер-демона, известного как Resource Manager, вспомогательного демона, называемого

диспетчером узлов (по одному на подчиненный узел) и Application Master (по одному на приложение).

Основная идея YARN состоит в том, чтобы разделить функциональные возможности управления ресурсами и планирования работы (мониторинга) на отдельные демоны, т.е. в том, чтобы иметь глобальный ResourceManager (RM) и ApplicationMaster (AM) для каждого приложения. Приложение представляет собой либо одно задание, либо DAG заданий.

С появлением YARN подход MapReduce превратился просто в распределенное приложение (хотя по-прежнему полезное и очень популярное) и теперь называется MRv2. MRv2 – это просто реализация классического механизма MapReduce (который теперь называется MRv1), выполняющегося поверх YARN.

Не менее популярным инструментом обработки больших данных сегодня является **Apache Spark** – фреймворк с открытым исходным кодом для параллельной обработки и анализа слабоструктурированных данных в оперативной памяти, входящий в экосистему проектов Hadoop. Главные преимущества Spark – производительность, удобный программный интерфейс с неявной параллелизацией и отказоустойчивостью. Spark поддерживает четыре языка: Scala, Java, Python и R.

Фреймворк состоит из пяти компонентов: ядра и четырех библиотек, каждая из которых решает определенную задачу:

– *Spark Core* – основа фреймворка. Обеспечивает распределенную диспетчеризацию, планирование и базовые функции ввода-вывода.

– *Spark SQL* – библиотека структурированной обработки данных. Spark Streaming – простой в использовании инструмент для обработки потоковых данных.

– *MLlib* – это распределенная система машинного обучения с высокой скоростью. MLlib включает в себя такие популярные алгоритмы, как классификация, регрессия, деревья решений, кластеризация.

– *GraphX* – библиотека для масштабируемой обработки графовых данных.

Spark может работать в среде кластеров Hadoop на YARN, под управлением Mesos, в облаке на AWS или других облачных сервисах и полностью автономно. Spark поддерживает несколько распределенных систем хранения, таких как HDFS, OpenStack Swift, Cassandra, Amazon S3, Kudu, MapR-FS и NoSQL базы данных.

Кроме всего прочего, следует упомянуть о примерах систем хранения данных. В частности, часто используемой является система **Apache Cassandra**. Это распределенная система управления базами данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надежных хранилищ огромных массивов данных, представленных в виде хэша. Известно, что данная СУБД используется такими компаниями, как Cisco, IBM, Cloudkick, Reddit, Digg, Rackspace, Apple и Twitter. Написана на языке Java, реализует распределенную hash-систему, что обеспечивает практически линейную масштабируемость при увеличении объема данных. Использует модель хранения данных на базе семейства столбцов, чем отличается от систем, которые хранят данные только в связке «ключ – значение», имеет возможность организовать хранение хэшей с несколькими уровнями вложенности. Относится к категории отказоустойчивых.

Помещенные в базу данные автоматически реплицируются на несколько узлов распределенной сети или даже равномерно распределяются в нескольких дата-центрах, при сбое узла его функции на лету подхватываются другими узлами; добавление новых узлов в кластер и обновление версии Cassandra также производится на лету, без дополнительного ручного вмешательства и переконфигурации других узлов.

Для упрощения взаимодействия с базой данных поддерживается язык формирования структурированных запросов CQL (Cassandra Query Language), который в какой-то степени сходен с SQL, но существенно урезан по функциональным возможностям. Драйверы с поддержкой CQL реализованы для языков Python (DBAPI2), Java (JDBC), Ruby (gem cassandra-cql), PHP (Thrift, cassandra-pdo, Cassandra-PHP-Client-Library) и JavaScript (Node.js). Среди главных преимуществ Cassandra:

– высокая масштабируемость и надежность без элементов, отказ которых приводит к выходу из строя всей системы;

– очень высокая пропускная способность для операций записи и хорошая пропускная способность для операций считывания;

– настраиваемая согласованность и поддержка репликации;

– гибкая схема.

12. ИНСТРУМЕНТЫ DATA MINING

Стремительное развитие информационных технологий, включая нарастающую интенсивность процессов сбора, хранения и обработки данных, которые рассмотрены подробно в гл. 1, порождает новый рынок специализированного программного обеспечения для решения задач *Data Mining*. Причем здесь справедливо выделить два различных класса такого программного обеспечения.

Один класс представляет собой прикладные пользовательские программные системы (как коммерческие, так и свободно распространяемые), как правило, реализованные в виде самостоятельных приложений или модулей (англ. *standalone application*) и предназначенные для решения конкретных предметных задач в той или иной области человеческой деятельности. Учитывая достаточно высокую сложность и междисциплинарность *Data Mining*, широкий набор потенциально доступных средств и методов (лишь основные из которых рассмотрены в данном пособии – см. гл. 1–4), такие системы позволяют только в некоторой степени закрыть потребность в квалифицированном анализе данных.

Учитывая эти обстоятельства, значительную популярность приобретают программно-аналитические системы, реализованные с использованием «облачной» архитектуры. В этом случае пользователю предоставляется лишь «тонкий» клиент (как правило, web-клиент), а мощная интеллектуальная вычислительная поддержка реализуется сервером поставщика услуг. Очевидно, это требует адаптации такой программной системы для задач пользователя со стороны экспертов. Такое привлечение профессиональной экспертизы при постановке задачи и ее адаптации к программно-вычислительной среде, а также потенциально более мощное методическое и вычислительное наполнение «облака» позволяют существенно увеличить эффективность ее решения. Рассмотрим кратко некоторые примеры таких систем в разд. 12.2.

Другой класс программных систем охватывает специализированные инструменты, которые при квалифицированном применении позволяют решать задачи *Data Mining*. Рассмотрим их кратко в разд. 12.1.

12.1. Программные инструменты для высокопроизводительной обработки данных

В гл. 2 приведен перечень компетенций, которыми должен обладать специалист в области *Data Mining* с точки зрения современного работодателя. Заметно, что набор компетенций, навыков и инструментария состоит не только из традиционных позиций для современного разработчика программного обеспечения (англ. *Software Engineer*), он дополнен некоторыми отличительными элементами, требующимися при распределенно-параллельной высокопроизводительной работе с *Big Data* методами *Data Mining*. Рассмотрим кратко некоторые наиболее распространенные из них, как правило, относящиеся к свободно распространяемому программному обеспечению.

12.1.1. Программная среда

Для повышения эффективности и упрощения процессов распределенно-параллельной обработки данных фондом *Apache Software Foundation* реализуется проект по разработке программной системы *Apache Hadoop* (часто просто *Hadoop*) разработки высокопроизводительных приложений. Данная среда характеризуется возможностью горизонтальной масштабируемости кластера путем добавления недорогих вычислительных узлов, без использования дорогостоящих суперкомпьютерных мощностей.

По состоянию на 2014 г. этот программный сервис представлен четырьмя модулями:

– *Hadoop MapReduce* (платформа программирования и выполнения распределенных *MapReduce*-вычислений);

– *HDFS* (*Hadoop Distributed File System*, распределенная файловая система);

– *YARN* (система для планирования заданий и управления ресурсами кластера);

– *Hadoop Common* (набор инфраструктурных программных библиотек и утилит, используемых для других модулей и родственных проектов типа *Mahout, Cassandra, Spark* и др.).

12.1.2. Базы данных

Важным элементом вычислительной среды обработки данных являются БД и СУБД. Необходимость обработки «больших данных», особенности которых рассмотрены в разд. 2.1–2.3, диктует и новые требования к «распределенным» принципам построения и функционирования БД и систем управления ими, отличных от принятых для них реляционных аналогов. Отличают три основных свойства данных (согласованность, доступность и устойчивость к разделению), которые являются противоречивыми, и добиться выполнения в одинаковой степени одновременно можно только двух из них. Современные нереляционные БД и СУБД направлены на попытку решения именно этой задачи.

Одним из примеров нереляционной распределенной БД стала БД с открытым исходным кодом *HBase*, реализуемая в развитие проекта *Apache Hadoop*. Эта БД функционирует совместно с распределенной файловой системой *HDFS* и обеспечивает отказоустойчивый способ хранения больших объемов разреженных данных.

Еще одним аналогичным и распространенным на практике примером нереляционной БД является *MongoDb* – документо-ориентированная БД, не имеющая строгой схемы данных, позволяющая добиваться высокой скорости записи и чтения, масштабируемости, но уступающя в сохранности и целостность данных.

Отметим, что в подобных БД присутствует примат масштабируемости и доступности данных над их согласованностью. В таких БД оперировать данными приходится не только с использованием стандартного структурированного языка запросов SQL, принятого

в реляционных БД, но и с помощью так называемого *NoSQL* (англ. *not only SQL*, не только *SQL*), который обеспечивает доступность и масштабируемость, но невысокую степень согласованности данных.

12.1.3. Языки программирования

Анализ многочисленных языков программирования, сопровождающих разработку и развитие программных систем анализа данных, вряд ли позволяет выявить очевидного лидера в этой сфере. Как и в любой другой области, выбор конкретного языка программирования, интегрированной среды разработки или компилятора зависит от множества специфических факторов. Вместе с тем нельзя не отметить возрастающую популярность языка программирования *Python*, приобретающего все большую востребованность вместе с традиционно распространенными в среде *Data Scientist* инструментами типа языков *R*, *Mathlab* или *Ruby*. Подробнее об этих и других языках программирования см.: [13, 107].

12.2. Примеры программных систем

12.2.1. Примеры самостоятельных систем

В качестве примеров универсальных, полнофункциональных, распространенных статистических пакетов называют *SAS Enterprise Miner* (компания *SAS Institute*), *SPSS* (*SPSS Modeler Professional* и *SPSS Modeler Premium*), *Statistica* (*StatSoft*) и др.

Большинство современных СУБД также включает поддержку функциональности *Data Mining:*

- Microsoft SQL Server Analysis Services (Microsoft Corp.);
- Oracle Business Intelligence (Oracle Corp.);
- IBM DB2 Intelligent Miner (IBM).

Кроме того, существует целый ряд систем, основанных главным образом на какой-то одной группе методов *Data Mining*:

- нейронные сети (BrainMaker (CSS), NeuroShell (Ward Systems Group), OWL (HyperLogic));

– деревья решений (See5/C5.0 (RuleQuest), Clementine (Integral Solutions), SIPINA (University of Lyon), IDIS (Information Discovery), KnowledgeSeeker (ANGOSS));

– генетические алгоритмы (*GeneHunter, Ward Systems Group*);

– алгоритмы ограниченного перебора (*WizWhy* от компании *WizSoft*);

– системы рассуждений на основе аналогичных случаев (англ. Case Based Reasoning; KATE tools (Acknosoft), Pattern Recognition Workbench (Unica));

– визуализация многомерных данных (*DataMiner 3D* компании *Dimension5*).

Очевидно, данными программными системами рынок современного программного обеспечения *Data Mining* не исчерпывается. Более того, в каждой из упомянутых групп регулярно появляются новинки, направленные на снижение требований к квалификации пользователя таких систем, повышение адекватности в результатах решения задач и т.п.

12.2.2. Примеры облачных систем

Примерами «не коробочных» программных систем *Data Mining*, набирающих популярность, являются системы, реализованные в «облачной» архитектуре. Число таких примеров сегодня наиболее велико на высоко конкурентных рынках, где значима ценность предсказательной аналитики, – рынках США и Западной Европы.

Интересным примером здесь является компания *Blue Yonder*, развивающая линейку продуктов в парадигме *SaaS* (англ. *Software as a Service*, ПО как сервис) [8]. Основным ядром создаваемого ПО является нейросетевой алгоритм оценки условной плотности распределения вероятности, разработанный в результате научных исследований на адронном коллайдере в рамках проекта ЦЕРН (фр. CERN – (*Conseil Européen pour la Recherche Nucléaire*, Европейский совет по ядерным исследованиям) [16]. Сегодня на основе этой разработки компания реализует целый ряд решений, предварительно адаптированных для сфер ритейла, производства,

телекоммуникаций, энергетики, финансов, медиа. Заявляется, что предлагаемые решения отличаются более высокой точностью, достигаемой глубокой научной проработкой применяемых алгоритмов.

ВОПРОСЫ И ТЕМЫ ДЛЯ САМОПРОВЕРКИ

1. Какие тренды информационно-коммуникационных технологий способствовали развитию Data Mining?
2. Приведите примеры применения методов Data Mining для решения практических задач.
3. Какие области человеческой деятельности наиболее и наименее подходят для их анализа методами Data Mining?
4. Что понимается под Data Mining и Big Data? Почему возникла такая терминология?
5. В чем состоит суть индуктивных и дедуктивных подходов в Data Mining?
6. Каковы основные этапы интеллектуального анализа данных?
7. Какие классификации методов Data Mining существуют? Приведите примеры.
8. В чем заключается предварительная обработка данных и какова ее цель? Какие подходы при этом применяются?
9. В чем заключается оптимизация признакового пространства? Какие методы с трансформацией и без трансформации пространства применяют, в чем их отличия?
10. В чем заключается метод классификации? Какие подходы для его реализации могут быть использованы и в чем их суть?
11. Что такое неконтролируемая классификация? Какие методы применяют для ее реализации?
12. В чем заключается суть метода машины опорных векторов и в чем его преимущество перед аналогами?
13. Как работают деревья принятия решений? Какие их разновидности существуют? Каковы пределы применимости этого метода?
14. Что такое регрессия? Какие подходы применяют для ее реализации?
15. Как работают ассоциативные алгоритмы?
16. Как работают алгоритмы последовательной ассоциации?

17. Что такое обнаружение аномалий? Приведите примеры применения этого подхода и укажите методы его реализации.

18. Что такое визуализация и какие инструменты ее реализации существуют?

19. Какие инструменты, модели и технологии существуют сегодня для реализации высокопроизводительных вычислений? Какие критерии эффективности при этом используют?

20. Примеры коммерческих многофункциональных систем и свободно распространяемых решений, реализующих инструментарий Data Mining. Их сравнительные характеристики.

21. Архитектуры и особенности функционирования информационных систем, реализующих методы Data Mining как сервис.

ЛИТЕРАТУРА

1. Abela A. Advanced presentations by design. Creating communication that drives action. John Wiley & Sons Limited, 2013. 224 p.

2. Agrawal R., Srikant R. Mining Sequential Patterns // Proc. of the 11th Int'l Conference on Data Engineering, 1995. P. 3–14.

3. Agrawal R., Srikant R. Fast algorithms for mining association rules in large databases // Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, 1994. P. 487–499.

4. Ayres J., Flannick J., Gehrke J., Yiu T. Sequential Pattern Mining using a Bitmap Representation // ACM SIGKDD Conference, 2002. P. 429–435

5. BaseGroup Labs. Технологии анализа данных. URL: http://www.basegroup.ru/ (дата обращения: 03.03.2020).

6. Big Data Analytics Methodological Training in Statistical Data Science. URL: http://www.statoo.com/dm/ (accessed: 03.03.2020).

7. Bishop C.M. Neural Networks for Pattern Recognition. Oxford University Press, 1995. 508 p.

8. Blue Yonder. URL: http://www.blue-yonder.com/ (accessed: 03.03.2020).

9. Boulding K.E. General Systems Theory – The Skeleton of Science // Management Science. 1956. № 2. P. 197–208.

10. Breiman L., Friedman J.H., Olshen R.A., Stone C.T. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984. 358 p

11. Chandola V., Kumar V. Summarization – compressing data into an informative representation // Knowledge and Information Systems. 2007. Vol. 12, is. 3. P. 355–378.

12. Davenport T.H. Analytics 3.0 // Harvard Business Review. 2013. Vol. 91, № 4. P. 64–72.

13. Data Mining Community Top Resource. URL: http://www.kdnuggets.com/ (accessed: 02.03.2020).

14. Deng H., Runger G., Tuv E. Bias of importance measures for multi-valued attributes and solutions // Proceedings of the 21st Inter-

national Conference on Artificial Neural Networks (ICANN). 2011. P. 293–300.

15. Emerson Process Management. Automation Solutions URL: http://www2.emersonprocess.com/en-IN/industries/Power/Coal/Optimize-Operator-and-Engineer-Effectiveness/Pages/Remote-Monitoring-Data-Analytics.aspx. (accessed: 26.02.2020).

16. Feindt M. A Neural Bayesian Estimator for Conditional Probability Densities // Cornell University. 2004. URL: http://arxiv.org/abs/physics/0402093 (accessed: 03.03.2020).

17. Fine S., Scheinberg K. INCAS: An incremental active set method for SVM : Technical Report / IBM Research Labs. Haifa, 2002.

18. Galton F. Regression Towards Mediocrity in Hereditary Stature // Journal of the Anthropological Institute. 1886. № 15. P. 246–263.

19. GartnerGroup. URL: www.gartner.com (accessed: 03.03.2020).

20. Giacinto G., Roli F. Dynamic Classifier Selection Based on Multiple Classifier Behaviour // Pattern Recognition. 2001. Vol. 34 (9). P. 179–181.

21. Horvath T., Yamamoto A. (eds.). Inductive Logic Programming : 13th International Conference, ILP 2003, Szeged, Hungary, September 29 – October 1, 2003 : Proceedings. Springer, 2003. P. 215–232. (Lecture Notes in Computer Science, vol. 2835).

22. Hyafil L., Rivest R. Constructing Optimal Binary Decision Trees is NP-complete // Information Processing Letters. 1976. Vol. 5 (1). P. 15–17.

23. Jain A., Zongker D. Feature Selection: Evaluation, Application, and Small Sample Performance // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997. Vol. 19, № 2. P. 153–158.

24. Jurafsky B., Ng A.Y. et al. Building DNN acoustic models for large vocabulary speech // IEEE Transactions on Audio, Speech, and Language Processing. 2017. Т. 34. № 9.

25. Kaneko I.S., Igarashi S. Combining Multiple k-Nearest Neighbour Classifiers Using Feature Combinations // J. IECI. 2000. Vol. 2, № 3. P. 23–31.

26. Knowledge Discovery Through Data Mining: What Is Knowledge Discovery? Tandem Computers Inc., 1996.

27. Kuznetsova A.V., Sen'ko O.V., Matchak G.N., Vakhotsky V.V., Zabotina T.N., Korotkova O.V. The Prognosis of Survivance in Solid Tumor Patients Based on Optimal Partitions of Immunological Parameters Ranges //Journal Theoretical Medicine. 2000. Vol. 2. P. 317–327.

28. Petrushin V.A., Khan L. Multimedia Data Mining and Knowledge Discovery. New York : Springer-Verlag, 2006.

29. Piatetsky-Shapiro G. Knowledge Discovery in Real Databases : a Report on the IJCAI-89 Workshop // AI Magazine. 1991. № 11 (5). P. 68–70.

30. Quinlan J.R. Induction of Decision Trees // Machine Learning. 1986. Vol. 1, № 1. P. 81–106.

31. Rahm E., Do H.H. Data Cleaning: Problems and Current Approaches // IEEE Bulletin on Data Engineering. 2000. Vol. 23 (4).

32. Raymer M.L., Punch W.F., Goodman E.D., Kuhn L.A., Jain L.C. Dimensionality reduction using genetic algorithms. // IEEE Trans. on Evolutionary Computation. 2000. Vol. 4 (2). P. 164–171.

33. Richards J.A., Xiuping Jia. Remote Sensing Digital Image Analysis : an Introduction. Berlin : Springer, 1999. – 363 p.

34. SAS Institute. URL: www.sas.com/ (accessed: 03.03.2020).

35. Srikant R., Agrawal R. Mining Sequential Patterns: Generalizations and Performance Improvements // Advances in Database Technology – EDBT '96 / P. Apers, M. Bouzeghoub, G. Gardarin (eds) Berlin, Heidelberg : Springer, 1996. (Lecture Notes in Computer Science, vol. 1057).

36. Stanton J.M. Introduction to Data Science. 3rd ed. 2012 // iTunes Open Source eBook. URL: https://itunes.apple.com/us/book/introduction-to-data-science/id529088127?mt=11 (accessed: 03.03.2020).

37. Support vector machines: Theory and applications / ed. by L. Wang. Springer, 2005. 434 p.

38. Tadviser. URL: https://goo.gl/2xLSFy (accessed: 26.03.2020).

39. Tibco Statistica Inc. URL: http://statistica.io/wordpress/wp-content/uploads/Striim-Partner-Solution-Brief.pdf (accessed: 24.03.2020).

40. Weigend A.S., Srivastava A.N. Predicting Conditional Probability Distributions: a Connectionist Approach // International Journal of Neural Systems. 1995. Vol. 6, № 2. P. 109–118.

41. Widrow B., Lehr M.A. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation // Proceedings of the IEEE. 1990. Vol. 78, № 9. P. 1415–1442.

42. Witten I.H., Frank E., Hall M.A., Kaufmann M. Data Mining: Practical Machine Learning Tools and Techniques. 3rd ed. Elsevier, 2011. 629 p.

43. Wonderware Software – Powering the Industrial World. URL: https://www.wonderware.com/industrial-information-management/historian-client (accessed: 26.03.2020).

44. Zhirnova I.G., Kuznetsova A.B., Rebrova O.Yu., Labunsky D.A., Komelkova L.V., Poleshchuk V.V., Sen'ko O.V. Logical and Statistical Approach for the Analysis of Immunological Parameters in Patients with Wilson's Disease // Russian Journal of Immunology. 1998. Vol. 3, № 2. P. 174–184.

45. Абдикеев Н.М. Когнитивная бизнес-аналитика. М. : ИНФРА-М, 2011. 510 с.

46. Абдикеев Н.М., Данько Т.П., Ильдеменов С.В., Киселев А.Д. Реинжиниринг бизнес-процессов. М. : Эксмо, 2005. 592 с. (Курс MBA).

47. Абдикеев Н.М., Киселев А.Д. Управление знаниями в корпорации и реинжиниринг бизнеса. М. : ИНФРА-М, 2011. 382 с.

48. Агентство стратегических инициатив. URL: https://asi.ru/ (дата обращения: 01.03.2020).

49. Аграновский А.В., Репалов С.А., Хади Р.А., Якубец М.Б. О недостатках современных систем обнаружения вторжений // Информационные технологии. 2005. № 5. С. 39–43.

50. Айвазян С.А. Классификация многомерных наблюдений. М. : Статистика, 1978. – 232 с.

51. Айвазян СА., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: классификация и снижение размерности. М. : Финансы и статистика, 1989. 607 с.

52. Асеев М.Г., Баллюзек М.Ф., Дюк В.А. Разработка медицинских экспертных систем средствами технологий Data Mining. URL: http://www.datadiver.nw.ru (дата обращения: 03.03.2020).

53. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining, СПб. : БХВ-Петербург, 2007. 336 с.

54. Большакова Е.И., Воронцов К.В., Ефремова Н.Э. и др. Автоматическая обработка текстов на естественном языке и анализ данных : учеб. пособие. М. : Изд-во НИУ ВШЭ, 2017. 269 с.

55. Большакова Е.И., Воронцов К.В., Ефремова Н.Э. и др. Глубокое обучение. Погружение в мир нейронных сетей. СПб. : Питер, 2018. 480 с.

56. Большие данные (Big Data). URL: http://www.tadviser.ru/ (дата обращения: 03.03.2020).

57. Еремеев В.Б. Разработка математического и программного обеспечения активного мониторинга вычислительной сети // Вести высших учебных заведений Черноземья. Автоматизация и информатика. 2008. № 4 (14). URL: http://www.stu.lipetsk.ru/files/materials/2408/2008_04_013.pdf (дата обращения: 03.03.2020).

58. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. М. : Наука, 1979. 488 с.

59. Вапник В.Н., Червоненкис А.Я. Теория распознавания образов. Статистические проблемы обучения. М. : Наука, 1974. 416 с.

60. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб. : БХВ-Петербург, 2004. 608 с.

61. Гвозденко С.В. Интеллектуальный анализ сложных нестационарных сигналов на примере электрокардиографических сигналов // Фундаментальные исследования. 2016. № 3. С. 537–542.

62. Гик Дж. ван. Прикладная общая теория систем. М. : Мир, 1981. 731 с.

63. Головкин Б.А. Параллельные вычислительные системы. М. : Наука, 1980. 520 с.

64. Давыдов А.А. Системная социология: анализ мультимедийной информации в Интернете URL: http://www.isras.ru/files/File/Publication/Multimedia_Information_DavydovA.pdf (дата обращения: 03.03.2020).

65. Доровских И.В., Кузнецова А.В., Сенько О.В., Реброва О.Ю. Прогноз динамики депрессивных синдромов в остром периоде сотрясения головного мозга по показателям первичного обследования

(с использованием логико-статистических методов) // Социальная и клиническая психиатрия. 2003. № 4. С. 18–24.

66. Дрейпер Н., Смит Г. Прикладной регрессионный анализ. Множественная регрессия. 3-е изд. М. : Диалектика, 2007. 912 с.

67. Дуда Р., Харт П. Распознавание образов : пер. с англ. М. : Наука, 1981. 450 с.

68. Дюк В.А. Обработка данных на ПК в примерах. СПб. : Питер, 1997. 240 с.

69. Дюк В., Самойленко А. Data Mining : учеб. курс. СПб. : Питер, 2001. 386 с.

70. Дюран Б., Оделл П. Кластерный анализ : пер. с англ. М. : Статистика, 1977. 128 с.

71. Епанечников В.А. Непараметрическая оценка многомерной плотности вероятности // Теория вероятностей и ее применения. 1969. Т. 14, вып. 1. С. 156–161.

72. Загоруйко Н.Г. Прикладные методы анализа данных и знаний. Новосибирск : Изд-во Ин-та математики СО РАН, 1999. 268 с.

73. Загоруйко Н.Г., Лбов Г.С. Проблема выбора в задачах анализа данных и управления // Сибирский журнал индустриальной математики. 2000. Vol. 3 (1). P. 101–109.

74. Замятин А.В. Введение в интеллектуальный анализ данных : учеб. пособие. Томск : Изд. Дом Том. гос. ун-та, 2016. 120 с.

75. Замятин А.В. Операционные системы : учеб. пособие. Томск : Изд-во Том. политехн. ун-та, 2010. 167 с.

76. Замятин А.В., Марков Н.Г. Анализ динамики земной поверхности с использованием данных дистанционного зондирования Земли. М. : Физматлит, 2007. 176 с.

77. Замятин А.В., Марков Н.Г., Напрюшкин А.А. Адаптивный алгоритм классификации с использованием текстурного анализа для автоматизированной интерпретации аэрокосмических изображений // Исследование Земли из космоса. 2004. № 2. С. 32–40.

78. Замятин А.В., Аксенов С.В., Костин К.А., Иванова А.В., Лианг Дж. Диагностика патологий по данным видео эндоскопии с использованием ансамбля сверточных нейронных сетей // Современные технологии в медицине. 2018. № 2.

79. Киселев М., Соломатин Е. Средства добычи знаний в бизнесе и финансах // Открытые системы. 1997. № 4. С. 41–44.

80. Кислова О.Н. Интеллектуализация информационных технологий как фактор развития интеллектуального анализа социологических данных // Методологія, теорія та практика соціологічного аналізу сучасного суспільства : збірник наукових праць. Харків : вид. центр ХНУ ім. В.Н. Каразіна, 2009. С. 318–324.

81. Консалтинговая компания IDC. URL: http://idc-group.ru (дата обращения: 03.03.2020).

82. Костюкова Н.И. Применение технологии Data Mining для решения задач оптимизации проектирования сложных технических систем // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 5 (36). С. 60–61.

83. Костюкова Н.И. Принятие решений в условиях риска // Приложение к журналу «Открытое образование». М., 2010. С. 90–93.

84. Костюкова Н.И. Создание новой технологии в среде С++, JAVA на базе вычисления группы, допускаемой дифференциальными уравнениями // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 7 (38). С. 59–61.

85. Костюкова Н.И. Технология Data Mining в задачах исследования сетевого трафика // Приложение к журналу «Открытое образование». М., 2010. С. 148–149.

86. Костюкова Н.И., Залевский А.А., Москвин Н.В. Разработка системы поддержки принятия решений // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 5 (36). С. 59–60.

87. Костюкова Н.И., Кудинов А.Е. Математические модели лечения с учетом эффективности // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 3 (34). С. 17–21.

88. Костюкова Н.И. Система принятия решений в области медицинской диагностики и выбора оптимальных решений по технологии Data Mining // Приложение к журналу «Открытое образование». М., 2010. С. 145–146.

89. Костюкова Н.И. Создание автоматизированной системы анализа технологии добычи данных для обнаружения сетевого

вторжения // Приложение к журналу «Открытое образование». М., 2010. С. 149–151.

90. Костюкова Н.И., Кудинов А.Е. Автоматизация научных исследований в области медицины с применением технологии Data Mining // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 3 (34), ч. 1. С. 22–24.

91. Костюкова Н.И., Родин Е.В. Система поддержки принятия решений для отраслей, связанных с риском // Альманах современной науки и образования. Тамбов : Грамота, 2010. № 7 (38). С. 41–44.

92. Костюкова Н.И., Кудинов А.Е. Статистические методы в медицине // Альманах современной науки и образования. Тамбов : Грамота, 2011. № 4 (47). С. 100–107.

93. Кречетов Н. Продукты для интеллектуального анализа данных // Рынок программных средств. 1997. № 14-15. С. 32–39.

94. Кузнецов В.А., Сенько О.В., Кузнецова А.В. и др. Распознавание нечетких систем по методу статистически взвешенных синдромов и его применение для иммуногематологической нормы и хронической патологии // Химическая физика. 1996. Т. 15, № 1. С. 81–100.

95. Кузнецова А.В., Сенько О.В. Возможности использования методов Data Mining при медико-лабораторных исследованиях для выявления закономерностей в массивах данных // Врач и информационные технологии. 2005. № 2. С. 49–56.

96. Кузнецова А.В. Диагностика и прогнозирование опухолевого роста по иммунологическим данным с помощью методов синдромного распознавания : автореф. дис. … канд. биол. наук. М., 1995. 23 с.

97. Лапко А.В., Ченцов С.В. Непараметрические системы обработки информации : учеб. пособие. М. : Наука, 2000. 350 с.

98. Назаров Л.Е. Применение многослойных нейронных сетей для классификации земных объектов на основе анализа многозональных сканерных изображений // Исследование Земли из космоса. 2000. № 6. С. 41–50.

99. Напрюшкин А.А. Алгоритмическое и программное обеспечение системы интерпретации аэрокосмических изображений

для решения задач картирования ландшафтных объектов : дис. … канд. техн. наук. Томск, 2002. 168 с.

100. Нейроинформатика / А.Н. Горбань, В.Л. Дунин-Барковский, А.Н. Кирдин и др. Новосибирск : Наука, Сиб. отд-ние, 1998. 296 с.

101. Нейронные сети. Statistica Neural Networks : пер. с англ. М. : Горячая линия-Телеком, 2000. 182 с.

102. Обработка естественного языка. URL: https://ru.wikipedia.org/wiki/ (дата обращения: 10.03.2020).

103. Прэтт У. Цифровая обработка изображений : пер. с англ. М. : Мир, 1982. Кн. 2. 480 с.

104. Рангайян Р.М. Анализ биометрических сигналов : практический подход / под ред. А.П. Немирко. М. : Физматлит, 2007. 222 с.

105. Рашка С. Python и машинное обучение : пер. с англ. М. : ДМК-Пресс, 2017. 418 с.

106. Реброва О.Ю. Статистический анализ медицинских данных. Применение пакета прикладных программ STATISTICA. М. : Медиа Сфера, 2002. 305 с.

107. Рейтинг языков программирования для data mining. URL: http://computerscinece.ru/posts/48 (дата обращения: 03.03.2020).

108. Рекрутинговая компания. URL: www.indeed.com (дата обращения: 03.03.2020).

109. Селевцов Л.И. Автоматизация технологических процессов. М. : Академия, 2014. 345 с.

110. Татарова Г.Г. Методология анализа данных в социологии (введение) : учебник для вузов. М. : Nota Bene, 1999. 224 с.

111. Толстова Ю.Н. Анализ социологических данных. Методология, дескриптивная статистика, изучение связей между номинальными признаками. М. : Научный мир, 2000. 352 с.

112. Ту Д., Гонсалес Р. Принципы распознавания образов : пер. с англ. М. : Мир, 1978. 412 с.

113. Финн В.К. Об интеллектуальном анализе данных // Новости искусственного интеллекта. 2004. № 3. С. 3–18.

114. Хуссейн Х.Ш., Якунин А.Г. Методы выявления аномалий при контроле динамических процессов природных и техногенных

объектов // Вестник ИжГТУ им. М.Т. Калашникова. 2015. № 1 (65). С. 79–83.

115. Что такое Data Mining. URL: http://www.iso.ru/ (дата обращения: 03.03.2020).

116. Чубукова И.А. Курс Data Mining. URL: http://www.intuit.ru/department/database/datamining/ (дата обращения: 03.03.2020).

117. Шапиро Е.И. Непараметрические оценки плотности вероятности в задачах обработки результатов наблюдений // Зарубежная радиоэлектроника. 2000. № 2. С. 3–22.

118. Якубец М.Б. Обнаружение сетевых атак методом поиска аномалий на основе вероятностного и верификационного моделирования // Искусственный интеллект. 2006. № 3. С. 816–823.

Учебное издание

Замятин Александр Владимирович

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Учебное пособие

Редактор Е.Г. Шумская
Оригинал-макет Е.Г. Шумской
Дизайн обложки Л.Д. Кривцовой

Подписано к печати 07.05.2020 г. Формат 60×84 $^1/_{16}$
Бумага для офисной техники. Гарнитура Times.
Усл. печ. л. 11,4.
Тираж 500 экз. Заказ № 4308.

Отпечатано на оборудовании
Издательского Дома
Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел. 8+(382-2)–52-98-49
Сайт: http://publish.tsu.ru
E-mail: rio.tsu@mail.ru

ISBN 978-5-94621-898-6