# Анализ данных и процессов

## 3-е издание

- Хранилища данных
- OLAP — оперативный анализ
- Data Mining — интеллектуальный анализ
- Visual Mining — визуальный анализ
- Text Mining — анализ текстовой информации
- Real-Time Mining — анализ в реальном времени
- Process Mining — анализ процессов
- Web Mining — анализ Web-ресурсов

+CD

bhv

# Анализ данных и процессов

## 3-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73
Б26

**Барсегян, А. А.**

Б26 Анализ данных и процессов: учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 512 с.: ил. + CD-ROM — (Учебная литература для вузов)

ISBN 978-5-9775-0368-6

Излагаются основные направления в области разработки корпоративных систем: организация хранилищ данных, оперативный (OLAP) и интеллектуальный (Data Mining) анализ данных. В третьем издании по сравнению со вторым, выходившем под названием "Технологии анализа данных: Data Mining, Text Mining, Visual Mining, OLAP", добавлены визуальный (Visual Mining) и текстовый (Text Mining) анализ данных, анализ процессов (Process Mining), анализ Web-ресурсов (Web mining) и анализ в режиме реального времени (Real-Time Data Mining). Приведено описание методов и алгоритмов решения основных задач анализа: классификации, кластеризации и др. Описание идеи каждого метода дополняется конкретным примером его использования.

Прилагаемый компакт-диск содержит стандарты Data Mining, библиотеку алгоритмов Xelopes и графический интерфейс к ней; JDK 1.6 и драйверы, необходимые для работы графического интерфейса, свободно распространяемую среду разработки Eclipse и лабораторный практикум по интеллектуальному анализу данных.

*Для студентов, инженеров*
*и специалистов в области анализа данных и процессов*

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Татьяна Лапина* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Екатерина Капалыгина* |
| Компьютерная верстка | *Ольги Сергиенко* |
| Корректор | *Зинаида Дмитриева* |
| Дизайн серии | *Инны Тачиной* |
| Оформление обложки | *Елены Беляевой* |
| Фото | *Кирилла Сергеева* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 08.05.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 41,28.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0368-6

# Оглавление

**Предисловие авторов** ..... **11**

**Data Mining и перегрузка информацией** ..... **13**

**Глава 1. Системы поддержки принятия решений** ..... **15**

1.1. Задачи систем поддержки принятия решений ..... 15
1.2. Базы данных — основа СППР ..... 18
1.3. Неэффективность использования OLTP-систем для анализа данных ..... 23
Выводы ..... 28

**Глава 2. Хранилище данных** ..... **29**

2.1. Концепция хранилища данных ..... 29
2.2. Организация ХД ..... 36
2.3. Очистка данных ..... 41
2.4. Концепция хранилища данных и анализ ..... 47
Выводы ..... 47

**Глава 3. OLAP-системы** ..... **50**

3.1. Многомерная модель данных ..... 50
3.2. Определение OLAP-систем ..... 54
3.3. Концептуальное многомерное представление ..... 55
    3.3.1. Двенадцать правил Кодда ..... 55
    3.3.2. Дополнительные правила Кодда ..... 56
    3.3.3. Тест FASMI ..... 58
3.4. Архитектура OLAP-систем ..... 59
    3.4.1. MOLAP ..... 60
    3.4.2. ROLAP ..... 63
    3.4.3. HOLAP ..... 66
Выводы ..... 67

**Глава 4. Интеллектуальный анализ данных....68**

4.1. Добыча данных — Data Mining....68
4.2. Задачи Data Mining....69
4.2.1. Классификация задач Data Mining....69
4.2.2. Задача классификации и регрессии....71
4.2.3. Задача поиска ассоциативных правил....73
4.2.4. Задача кластеризации....75
4.3. Практическое применение Data Mining....77
4.3.1. Интернет-технологии....77
4.3.2. Торговля....77
4.3.3. Телекоммуникации....78
4.3.4. Промышленное производство....78
4.3.5. Медицина....79
4.3.6. Банковское дело....80
4.3.7. Страховой бизнес....81
4.3.8. Другие области применения....81
4.4. Модели Data Mining....81
4.4.1. Предсказательные модели....81
4.4.2. Описательные модели....82
4.5. Методы Data Mining....84
4.5.1. Базовые методы....84
4.5.2. Нечеткая логика....84
4.5.3. Генетические алгоритмы....87
4.5.4. Нейронные сети....89
4.6. Процесс обнаружения знаний....90
4.6.1. Основные этапы анализа....90
4.6.2. Подготовка исходных данных....92
4.7. Управление знаниями (Knowledge Management)....94
4.8. Средства Data Mining....95
Выводы....100

**Глава 5. Классификация и регрессия....102**

5.1. Постановка задачи....102
5.2. Представление результатов....103
5.2.1. Правила классификации....103
5.2.2. Деревья решений....104
5.2.3. Математические функции....105
5.3. Методы построения правил классификации....106
5.3.1. Алгоритм построения 1-правил....106
5.3.2. Метод Naive Bayes....108
5.4. Методы построения деревьев решений....111
5.4.1. Методика "разделяй и властвуй"....111
5.4.2. Алгоритм покрытия....119
5.5. Методы построения математических функций....124
5.5.1. Общий вид....124

5.5.2. Линейные методы. Метод наименьших квадратов ... 126
5.5.3. Нелинейные методы ... 127
5.5.4. Support Vector Machines (SVM) ... 128
5.5.5. Регуляризационные сети (Regularization Networks) ... 131
5.5.6. Дискретизации и редкие сетки ... 133
5.6. Прогнозирование временных рядов ... 136
5.6.1. Постановка задачи ... 136
5.6.2. Методы прогнозирования временных рядов ... 136
Выводы ... 138

**Глава 6. Поиск ассоциативных правил ... 140**

6.1. Постановка задачи ... 140
6.1.1. Формальная постановка задачи ... 140
6.1.2. Секвенциальный анализ ... 143
6.1.3. Разновидности задачи поиска ассоциативных правил ... 146
6.2. Представление результатов ... 148
6.3. Алгоритмы ... 152
6.3.1. Алгоритм Apriori ... 152
6.3.2. Разновидности алгоритма Apriori ... 157
Выводы ... 158

**Глава 7. Кластеризация ... 159**

7.1. Постановка задачи кластеризации ... 159
7.1.1. Формальная постановка задачи ... 161
7.1.2. Меры близости, основанные на расстояниях, используемые в алгоритмах кластеризации ... 163
7.2. Представление результатов ... 165
7.3. Базовые алгоритмы кластеризации ... 167
7.3.1. Классификация алгоритмов ... 167
7.3.2. Иерархические алгоритмы ... 168
7.3.3. Неиерархические алгоритмы ... 171
7.4. Адаптивные методы кластеризации ... 184
7.4.1. Выбор наилучшего решения и качество кластеризации ... 184
7.4.2. Использование формальных критериев качества в адаптивной кластеризации ... 184
7.4.3. Пример адаптивной кластеризации ... 187
Выводы ... 190

**Глава 8. Визуальный анализ данных — Visual Mining ... 192**

8.1. Выполнение визуального анализа данных ... 192
8.2. Характеристики средств визуализации данных ... 194
8.3. Методы визуализации ... 199
8.3.1. Методы геометрических преобразований ... 199
8.3.2. Отображение иконок ... 203

8.3.3. Методы, ориентированные на пикселы ... 205
8.3.4. Иерархические образы ... 207
Выводы ... 209

**Глава 9. Анализ текстовой информации — Text Mining ... 211**

9.1. Задача анализа текстов ... 211
9.1.1. Этапы анализа текстов ... 211
9.1.2. Предварительная обработка текста ... 213
9.1.3. Задачи Text Mining ... 214
9.2. Извлечение ключевых понятий из текста ... 215
9.2.1. Общее описание процесса извлечения понятий из текста ... 215
9.2.2. Стадия локального анализа ... 218
9.2.3. Стадия интеграции и вывода понятий ... 221
9.3. Классификация текстовых документов ... 223
9.3.1. Описание задачи классификации текстов ... 223
9.3.2. Методы классификации текстовых документов ... 225
9.4. Методы кластеризации текстовых документов ... 226
9.4.1. Представление текстовых документов ... 226
9.4.2. Иерархические методы кластеризации текстов ... 228
9.4.3. Бинарные методы кластеризации текстов ... 230
9.5. Задача аннотирования текстов ... 230
9.5.1. Выполнение аннотирования текстов ... 230
9.5.2. Методы извлечения фрагментов для аннотации ... 233
9.6. Средства анализа текстовой информации ... 236
9.6.1. Средства Oracle — Oracle Text ... 236
9.6.2. Средства от IBM — Intelligent Miner for Text ... 237
9.6.3. Средства SAS Institute — Text Miner ... 238
9.6.4. Средства Мегапьютер Интеллидженс — TextAnalyst ... 239
Выводы ... 240

**Глава 10. Стандарты Data Mining ... 242**

10.1. Кратко о стандартах ... 242
10.2. Стандарт CWM ... 242
10.2.1. Назначение стандарта CWM ... 242
10.2.2. Структура и состав CWM ... 244
10.2.3. Пакет Data Mining ... 247
10.3. Стандарт CRISP ... 251
10.3.1. Появление стандарта CRISP ... 251
10.3.2. Структура стандарта CRISP ... 251
10.3.3. Фазы и задачи стандарта CRISP ... 253
10.4. Стандарт PMML ... 258
10.5. Другие стандарты Data Mining ... 264
10.5.1. Стандарт SQL/MM ... 264
10.5.2. Стандарт Microsoft Data Mining eXtensions (DMX) ... 266
10.5.3. Стандарт Java Data Mining ... 267
Выводы ... 269

**Глава 11. Библиотека Xelopes ..... 271**

11.1. Архитектура библиотеки ..... 271
11.2. Диаграмма Model ..... 274
11.2.1. Классы модели для Xelopes ..... 274
11.2.2. Методы пакета Model ..... 276
11.2.3. Преобразование моделей ..... 277
11.3. Диаграмма Settings ..... 278
11.3.1. Классы пакета Settings ..... 278
11.3.2. Методы пакета Settings ..... 280
11.4. Диаграмма Attribute ..... 280
11.4.1. Классы пакета Attribute ..... 280
11.4.2. Иерархические атрибуты ..... 281
11.5. Диаграмма Algorithms ..... 282
11.5.1. Общая концепция ..... 282
11.5.2. Класс *MiningAlgorithm* ..... 283
11.5.3. Расширение класса *MiningAlgorithm* ..... 284
11.5.4. Дополнительные классы ..... 286
11.5.5. Слушатели ..... 286
11.6. Диаграмма DataAccess ..... 286
11.6.1. Общая концепция ..... 287
11.6.2. Класс *MiningInputStream* ..... 288
11.6.3. Классы Mining-векторов ..... 288
11.6.4. Классы, расширяющие класс *MiningInputStream* ..... 288
11.7. Диаграмма Transformation ..... 289
11.8. Примеры использования библиотеки Xelopes ..... 291
11.8.1. Общая концепция ..... 291
11.8.2. Решение задачи поиска ассоциативных правил ..... 294
11.8.3. Решение задачи кластеризации ..... 296
11.8.4. Решение задачи классификации ..... 298
Выводы ..... 301

**Глава 12. Распределенный анализ данных ..... 303**

12.1. Системы мобильных агентов ..... 303
12.1.1. Основные понятия ..... 303
12.1.2. Стандарты многоагентных систем ..... 304
12.1.3. Системы мобильных агентов ..... 307
12.1.4. Система мобильных агентов JADE ..... 307
12.2. Использование мобильных агентов для анализа данных ..... 309
12.2.1. Проблемы распределенного анализа данных ..... 309
12.2.2. Агенты-аналитики ..... 309
12.2.3. Варианты анализа распределенных данных ..... 311
12.3. Система анализа распределенных данных ..... 313
12.3.1. Общий подход к реализации системы ..... 313
12.3.2. Агент для сбора информации о базе данных ..... 314

12.3.3. Агент для сбора статистической информации о данных ... 317
12.3.4. Агент для решения одной задачи интеллектуального анализа данных ... 320
12.3.5. Агент для решения интегрированной задачи интеллектуального анализа данных ... 323
Выводы ... 324

**Глава 13. Data Mining в реальном времени (Real-Time Data Mining) ... 325**

13.1. Идея Data Mining в реальном времени ... 325
13.1.1. Адаптация системы к общей концепции ... 325
13.1.2. Адаптивная добыча данных ... 326
13.1.3. Статический Data Mining и Data Mining в реальном времени ... 328
13.1.4. Применение Data Mining в реальном времени ... 329
13.2. Рекомендательные машины ... 330
13.2.1. Классификация рекомендательных машин ... 330
13.2.2. Подход на основе содержания ... 331
13.2.3. Совместное фильтрование ... 331
13.2.4. Анализ рыночной корзины и секвенциальный анализ ... 335
13.2.5. Усиление обучения и агенты ... 335
13.3. Инструменты Data Mining в реальном времени ... 345
13.3.1. Инструмент Amazon.com — механизм рекомендаций ... 345
13.3.2. Инструмент Prudsys — рекомендательная машина Prudsys ... 345
13.3.3. Приложение с открытым кодом — SpamAssassin ... 348
Выводы ... 349

**Глава 14. Извлечение знаний из Web — Web Mining ... 350**

14.1. Web Mining ... 350
14.1.1. Проблемы анализа информации из Web ... 350
14.1.2. Этапы Web Mining ... 351
14.1.3. Web Mining и другие интернет-технологии ... 352
14.1.4. Категории Web Mining ... 353
14.2. Методы извлечения Web-контента ... 356
14.2.1. Извлечение Web-контента в процессе информационного поиска ... 356
14.2.2. Извлечение Web-контента для формирования баз данных ... 363
14.3. Извлечение Web-структур ... 365
14.3.1. Представление Web-структур ... 365
14.3.2. Оценка важности Web-структур ... 366
14.3.3. Поиск Web-документов с учетом гиперссылок ... 370
14.3.3. Кластеризация Web-структур ... 371
14.4. Исследование использования Web-ресурсов ... 372
14.4.1. Исследуемая информация ... 372
14.4.2. Этап препроцессинга ... 375
14.4.3. Этап извлечения шаблонов ... 377
14.4.4. Этап анализа шаблонов и их применение ... 379
Выводы ... 381

**Глава 15. Средства анализа процессов — Process Mining.....382**

15.1. Автоматизация выполнения бизнес-процессов .....382
15.1.1. Бизнес-процессы .....382
15.1.2. Формализация бизнес-процессов.....384
15.1.3. Workflow-системы.....386
15.1.4. Сервисно-ориентированная архитектура .....387
15.1.5. Проектирование бизнес-процессов.....389
15.2. Анализ процессов.....389
15.2.1. Технология Process Mining .....389
15.2.2. Анализ протоколов.....391
15.2.3. Стандарт записи протоколов MXML.....393
15.2.4. Задачи Process Mining .....395
15.2.5. Проблемы анализа протоколов .....396
15.3. Методы Process Mining.....398
15.3.1. Первые вероятностные методы Process Mining.....398
15.3.2. Метод построения дизъюнктивной Workflow-схемы .....404
15.3.3. α-алгоритм .....415
15.3.4. Методы на основе генетических алгоритмов .....428
15.4. Библиотека алгоритмов Process Mining — ProM .....432
15.4.1. Архитектура ProM.....432
15.4.2. ProM Import Framework .....434
Выводы .....436

**ПРИЛОЖЕНИЯ .....439**

**Приложение 1. Нейронечеткие системы .....441**

П1.1. Способы интеграции нечетких и нейронных систем .....441
П1.2. Нечеткие нейроны.....445
П1.3. Обучение методами спуска .....447
П1.4. Нечеткие схемы рассуждений.....448
П1.5. Настройка нечетких параметров управления с помощью нейронных сетей ....454
П1.6. Нейронечеткие классификаторы.....461

**Приложение 2. Особенности и эффективность генетических алгоритмов.....467**

П2.1. Методы оптимизации комбинаторных задач различной степени сложности...467
П2.2. Сущность и классификация эволюционных алгоритмов.....472
П2.2.1. Базовый генетический алгоритм .....472
П2.2.2. Последовательные модификации базового генетического алгоритма ...473
П2.2.3. Параллельные модификации базового генетического алгоритма.....475
П2.3. Классификация генетических алгоритмов .....478
П2.4. Особенности генетических алгоритмов, предпосылки для адаптации.....479

П2.5. Классификация адаптивных ГА ..... 482
П2.5.1. Основа адаптации ..... 482
П2.5.2. Область адаптации ..... 484
П2.5.3. Основа управления адаптацией ..... 486
П2.6. Двунаправленная интеграция ГА и нечетких алгоритмов продукционного типа ..... 487

**Приложение 3. Описание прилагаемого компакт-диска ..... 494**

**Список литературы ..... 497**

**Предметный указатель ..... 509**

# Предисловие авторов

Повсеместное использование компьютеров привело к пониманию важности задач, связанных с анализом накопленной информации для извлечения новых знаний. Возникла потребность в создании хранилищ данных и систем поддержки принятия решений, основанных, в том числе, и на методах теории искусственного интеллекта.

Действительно, управление предприятием, банком, различными сферами бизнеса, в том числе электронного, немыслимо без процессов накопления, анализа, выявления определенных закономерностей и зависимостей, прогнозирования тенденций и рисков.

Именно давний интерес авторов к методам, алгоритмическим моделям и средствам их реализации, используемым на этапе анализа данных, явился причиной подготовки данной книги.

В книге представлены наиболее перспективные направления анализа данных: хранение информации, оперативный и интеллектуальный анализ. Подробно рассмотрены методы и алгоритмы интеллектуального анализа. Кроме описания популярных и известных методов анализа приводятся оригинальные результаты.

Книга ориентирована на студентов и специалистов, интересующихся современными методами анализа данных. Наличие в приложении материала, посвященного нейронным сетям и генетическим алгоритмам, делает книгу самодостаточной. Как пособие, книга в первую очередь предназначена для бакалавров и магистров, обучающихся по направлению "Информационные системы". Кроме того, книга будет полезна специалистам, занимающимся разработкой корпоративных информационных систем. Подробное описание методов и алгоритмов интеллектуального анализа позволит использовать книгу не только для ознакомления с данной областью вычислительной техники, но и для разработки конкретных систем.

Первые четыре главы книги, содержащие общую информацию о современных направлениях анализа данных, пригодятся руководителям предприятий, планирующим внедрение и использование методов анализа данных.

Благодарности:

- Григорию Пятецкому-Шапиро — основателю направления Data Mining за поддержку и полезные замечания;
- Валентину Степаненко — за неоценимую помощь при подготовке первых изданий книги.

# Data Mining и перегрузка информацией

В 2002 году, согласно оценке профессоров из калифорнийского университета Berkeley, объем информации в мире увеличился на пять миллиардов (5 000 000 000 000 000 000) байт. Согласно другим оценкам, информация удваивается каждые 2—3 года. Этот потоп, цунами данных приходит из науки, бизнеса, Интернета и других источников. Среди самых больших баз данных в 2003 году France Telecom имела СППР (DSS system) размером 30 000 миллиардов байт, а Alexa Internet Archive содержал 500 000 миллиардов байт.

На первом семинаре, посвященном поиску знаний в данных (Knowledge Discovery in Data workshop), который я организовал в 1989 году, один мегабайт (1 000 000) считался размером большой базы данных. На последней конференции KDD-2003 один докладчик обсуждал базу данных для астрономии размером во много терабайт и предсказывал необходимость иметь дело с петабайтами (1 терабайт = 1 000 миллиардов байт, а 1 петабайт = 1 000 терабайт).

Из-за огромного количества информации очень малая ее часть будет когда-либо увидена человеческим глазом. Наша единственная надежда понять и найти что-то полезное в этом океане информации — широкое применение методов Data Mining.

Технология Data Mining (также называемая Knowledge Discovery In Data — обнаружение знаний в данных) изучает процесс нахождения новых, действительных и потенциально полезных знаний в базах данных. Data Mining лежит на пересечении нескольких наук, главные из которых — это системы баз данных, статистика и искусственный интеллект.

Область Data Mining выросла из одного семинара в 1989 году до десятков международных конференций в 2003 году с тысячами исследователей во многих странах мира. Data Mining широко используется во многих областях с большим объемом данных: в науке — астрономии, биологии, биоинформатике, медицине, физике и других областях; в бизнесе — торговле, теле-

коммуникациях, банковском деле, промышленном производстве и т. д. Благодаря сети Интернет Data Mining используется каждый день тысячи раз в секунду — каждый раз, когда кто-то использует Google или другие поисковые системы (search engines) на просторах Интернета.

Виды информации, с которыми работают исследователи, включают в себя не только цифровые данные, но и все более текст, изображение, видео, звук и т. д. Одна новая и быстро растущая часть Data Mining — это анализ связей между данными (link analysis), который имеет приложения в таких разных областях, как биоинформатика, цифровые библиотеки и защита от терроризма.

Математический и статистический подходы являются основой для Data Mining. Как уроженцу Москвы и ученику известной в 1970-е годы 2-й математической школы, мне особенно приятно писать предисловие к первой книге на русском языке, покрывающей эту важную и интересную область.

Эта книга дает читателю обзор технологий и алгоритмов для хранения и организации данных, включая ХД и OLAP, а затем переходит к методам и алгоритмам реализации Data Mining.

Авторы приводят обзор наиболее распространенных областей применения Data Mining и объясняют процесс обнаружения знаний. В ряде глав рассматриваются основные методы Data Mining, включая классификацию и регрессию, поиск ассоциативных правил и кластеризацию. Книга также обсуждает главные стандарты в области Data Mining.

Важная часть книги — это обзор библиотеки Xelopes компании Prudsys, содержащей многие важные алгоритмы для Data Mining. В заключение дается более детальный анализ продвинутых на сегодняшний день методов — самоорганизующихся, нейронечетких систем и генетических алгоритмов.

Я надеюсь, что эта книга найдет много читателей и заинтересует их важной и актуальной областью Data Mining и поиска знаний.

Григорий Пятецкий-Шапиро, KDnuggets
Закоровье, Нью Гемпшир, США

# ГЛАВА 1

# Системы поддержки принятия решений

## 1.1. Задачи систем поддержки принятия решений

С появлением первых ЭВМ наступил этап информатизации разных сторон человеческой деятельности. Если раньше человек основное внимание уделял веществу, затем энергии (рис. 1.1), то сегодня можно без преувеличения сказать, что наступил этап осознания процессов, связанных с информацией. Вычислительная техника создавалась прежде всего для обработки данных. В настоящее время современные вычислительные системы и компьютерные сети позволяют накапливать большие массивы данных для решения задач обработки и анализа. К сожалению, сама по себе машинная форма представления данных содержит информацию, необходимую человеку, в скрытом виде, и для ее извлечения нужно использовать специальные методы анализа данных.

Большой объем информации, с одной стороны, позволяет получить более точные расчеты и анализ, с другой — превращает поиск решений в сложную задачу. Неудивительно, что первичный анализ данных был переложен на компьютер. В результате появился целый класс программных систем, призванных облегчить работу людей, выполняющих анализ (аналитиков). Такие системы принято называть *системами поддержки принятия решений* — СППР (DSS, Decision Support System).

Для выполнения анализа СППР должна накапливать информацию, обладая средствами ее ввода и хранения. Можно выделить три основные задачи, решаемые в СППР:

- ввод данных;
- хранение данных;
- анализ данных.

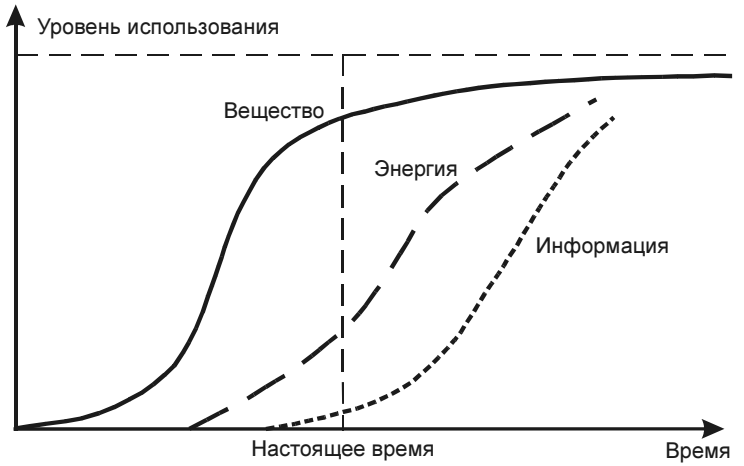



**Рис. 1.1.** Уровень использования человеком различных объектов материального мира

Таким образом, СППР — это системы, обладающие средствами ввода, хранения и анализа данных, относящихся к определенной предметной области, с целью поиска решений.

Ввод данных в СППР осуществляется либо автоматически от датчиков, характеризующих состояние среды или процесса, либо человеком-оператором. В первом случае данные накапливаются путем циклического опроса или по сигналу готовности, возникающему при появлении информации. Во втором случае СППР должны предоставлять пользователям удобные средства ввода данных, контролирующие корректность вводимых данных и выполняющие сопутствующие вычисления. Если ввод осуществляется одновременно несколькими операторами, то система должна решать проблемы параллельного доступа и модификации одних и тех же данных.

Постоянное накопление данных приводит к непрерывному росту их объема. В связи с этим на СППР ложится задача обеспечить надежное хранение больших объемов данных. На СППР также могут быть возложены задачи предотвращения несанкционированного доступа, резервного хранения данных, архивирования и т. п.

Основная задача СППР — предоставить аналитикам инструмент для выполнения анализа данных. Необходимо отметить, что для эффективного использования СППР ее пользователь-аналитик должен обладать соответствующей квалификацией. Система не генерирует правильные решения, а только предоставляет аналитику данные в соответствующем виде (отчеты, таблицы, графики и т. п.) для изучения и анализа, именно поэтому такие системы обес-

печивают выполнение функции поддержки принятия решений. Очевидно, что, с одной стороны, качество принятых решений зависит от квалификации аналитика. С другой стороны, рост объемов анализируемых данных, высокая скорость обработки и анализа, а также сложность использования машинной формы представления данных стимулируют исследования и разработку интеллектуальных СППР. Для таких СППР характерно наличие функций, реализующих отдельные умственные возможности человека.

По степени "интеллектуальности" обработки данных при анализе выделяют три класса задач анализа:

- *информационно-поисковый* — СППР осуществляет поиск необходимых данных. Характерной чертой такого анализа является выполнение заранее определенных запросов;
- *оперативно-аналитический* — СППР производит группирование и обобщение данных в любом виде, необходимом аналитику. В отличие от информационно-поискового анализа в данном случае невозможно заранее предсказать необходимые аналитику запросы;
- *интеллектуальный* — СППР осуществляет поиск функциональных и логических закономерностей в накопленных данных, построение моделей и правил, которые объясняют найденные закономерности и/или прогнозируют развитие некоторых процессов (с определенной вероятностью).

Таким образом, обобщенная архитектура СППР может быть представлена следующим образом (рис. 1.2).

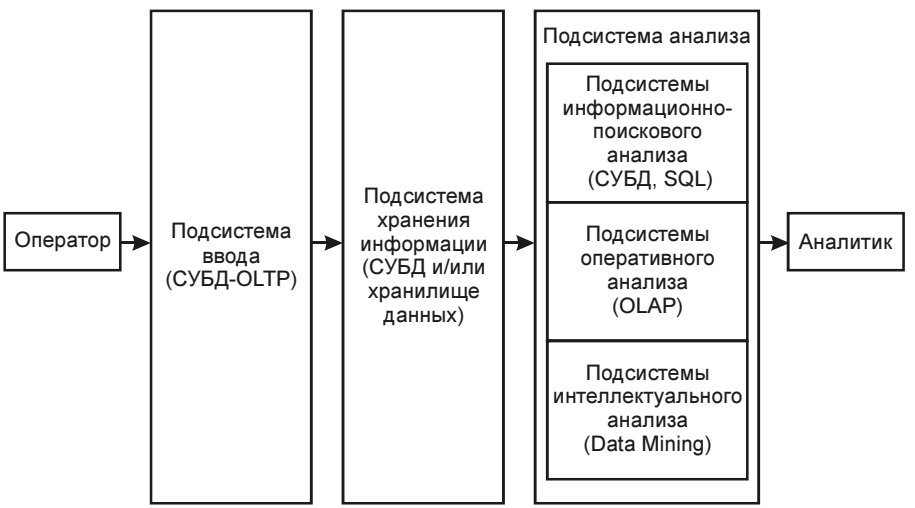



**Рис. 1.2.** Обобщенная архитектура системы поддержки принятия решений

Рассмотрим отдельные подсистемы более подробно.

- **Подсистема ввода данных.** В таких подсистемах, называемых OLTP (On-line transaction processing), выполняется операционная (транзакционная) обработка данных. Для реализации этих подсистем используют обычные системы управления базами данных (СУБД).
- **Подсистема хранения.** Для реализации данной подсистемы используют современные СУБД и концепцию хранилищ данных.
- **Подсистема анализа.** Данная подсистема может быть построена на основе:
  - подсистемы информационно-поискового анализа на базе реляционных СУБД и статических запросов с использованием языка структурных запросов SQL (Structured Query Language);
  - подсистемы оперативного анализа. Для реализации таких подсистем применяется технология оперативной аналитической обработки данных OLAP (On-line analytical processing), использующая концепцию многомерного представления данных;
  - подсистемы интеллектуального анализа. Данная подсистема реализует методы и алгоритмы Data Mining *("добыча данных")*.

## 1.2. Базы данных — основа СППР

Ранее было отмечено, что для решения задач анализа данных и поиска решений необходимо накопление и хранение достаточно больших объемов данных. Этим целям служат базы данных (БД).

> ***Внимание!***
> База данных является моделью некоторой предметной области, состоящей из связанных между собой данных об объектах, их свойствах и характеристиках.

Средства для работы с БД представляют СУБД. Не решая непосредственно никаких прикладных задач, СУБД является инструментом для разработки прикладных программ, использующих БД.

Чтобы сохранять данные согласно какой-либо модели предметной области, структура БД должна максимально соответствовать этой модели. Первой такой структурой, используемой в СУБД, была иерархическая структура, появившаяся в начале 60-х годов прошлого века.

Иерархическая структура предполагала хранение данных в виде дерева. Это значительно упрощало создание и поддержку таких БД. Однако невозможность представить многие объекты реального мира в виде иерархии привела к использованию таких БД в сильно специализированных областях. Типичным

представителем (наиболее известным и распространенным) иерархической СУБД является Information Management System (IMS) фирмы IBM. Первая версия этого продукта появилась в 1968 году.

Попыткой улучшить иерархическую структуру была сетевая структура БД, которая предполагает представление данных в виде сети. Она основана на предложениях группы Data Base Task Group (DBTG) Комитета по языкам программирования Conference on Data Systems Languages (CODASYL). Отчет DBTG был опубликован в 1971 году.

Работа с сетевыми БД представляет гораздо более сложный процесс, чем работа с иерархической БД, поэтому данная структура не нашла широкого применения на практике. Типичным представителем сетевых СУБД является Integrated Database Management System (IDMS) компании Cullinet Software, Inc.

Наиболее распространены в настоящее время реляционные БД. Термин *"реляционный"* произошел от латинского слова *"relatio"* — отношение. Такая структура хранения данных построена на взаимоотношении составляющих ее частей. Реляционный подход стал широко известен благодаря работам Е. Кодда, которые впервые были опубликованы в 1970 году. В них Кодд сформулировал следующие 12 правил для реляционной БД:

1. **Данные представляются в виде таблиц.** БД представляет собой набор таблиц. Таблицы хранят данные, сгруппированные в виде рядов и колонок. Ряд представляет собой набор значений, относящихся только к одному объекту, хранящемуся в таблице, и называется *записью*. Колонка представляет собой одну характеристику для всех объектов, хранящихся в таблице, и называется *полем*. Ячейка на пересечении ряда и колонки представляет собой значение характеристики, соответствующей колонке для объекта соответствующего ряда.
2. **Данные доступны логически.** Реляционная модель не позволяет обращаться к данным физически, адресуя ячейки по номерам колонки и ряда (нет возможности получить значение в ячейке (колонка 2, ряд 3)). Доступ к данным возможен только через идентификаторы таблицы, колонки и ряда. Идентификаторами таблицы и колонки являются их имена. Они должны быть уникальны в пределах, соответственно, БД и таблицы. Идентификатором ряда является первичный ключ — значения одной или нескольких колонок, однозначно идентифицирующих ряды. Каждое значение первичного ключа в пределах таблицы должно быть уникальным. Если идентификация ряда осуществляется на основании значений нескольких колонок, то ключ называется составным.
3. **NULL трактуется как неизвестное значение.** Если в ячейку таблицы значение не введено, то записывается значение NULL. Его нельзя путать с пустой строкой или со значением 0.

4. **БД должна включать в себя метаданные.** БД хранит два вида таблиц: пользовательские и системные. В пользовательских таблицах хранятся данные, введенные пользователем. В системных таблицах хранятся метаданные: описание таблиц (название, типы и размеры колонок), индексы, хранимые процедуры и др. Системные таблицы тоже доступны, т. е. пользователь может получить информацию о метаданных БД.

5. **Должен использоваться единый язык для взаимодействия с СУБД.** Для управления реляционной БД должен использоваться единый язык. В настоящее время таким инструментом стал язык SQL.

6. **СУБД должна обеспечивать альтернативный вид отображения данных.** СУБД не должна ограничивать пользователя только отображением таблиц, которые существуют. Пользователь должен иметь возможность строить виртуальные таблицы — представления (View). Представления являются динамическим объединением нескольких таблиц. Изменения данных в представлении должны автоматически переноситься на исходные таблицы (за исключением нередактируемых полей в представлении, например вычисляемых полей).

7. **Должны поддерживаться операции реляционной алгебры.** Записи реляционной БД трактуются как элементы множества, на котором определены операции реляционной алгебры. СУБД должна обеспечивать выполнение этих операций. В настоящее время выполнение этого правила обеспечивает язык SQL.

8. **Должна обеспечиваться независимость от физической организации данных.** Приложения, оперирующие с данными реляционных БД, не должны зависеть от физического хранения данных (от способа хранения, формата хранения и др.).

9. **Должна обеспечиваться независимость от логической организации данных.** Приложения, оперирующие с данными реляционных БД, не должны зависеть от организации связей между таблицами (логической организации). При изменении связей между таблицами не должны меняться ни сами таблицы, ни запросы к ним.

10. **За целостность данных отвечает СУБД.** Под целостностью данных в общем случае понимается готовность БД к работе. Различают следующие типы целостности:

    - *физическая целостность* — сохранность информации на носителях и корректность форматов хранения данных;
    - *логическая целостность* — непротиворечивость и актуальность данных, хранящихся в БД.

Потеря целостности базы данных может произойти из-за сбоев аппаратуры ЭВМ, ошибок в программном обеспечении, неверной технологии ввода и корректировки данных, низкой достоверности самих данных и т. д.

За сохранение целостности данных должна отвечать СУБД, а не приложение, оперирующее ими. Различают два способа обеспечения целостности: *декларативный* и *процедурный*. При декларативном способе целостность достигается наложением ограничений на таблицы, при процедурном — обеспечивается с помощью хранимых в БД процедур.

11. **Целостность данных не может быть нарушена.** СУБД должна обеспечивать целостность данных при любых манипуляциях, производимых с ними.
12. **Должны поддерживаться распределенные операции.** Реляционная БД может размещаться как на одном компьютере, так и на нескольких — распределенно. Пользователь должен иметь возможность связывать данные, находящиеся в разных таблицах и на разных узлах компьютерной сети. Целостность БД должна обеспечиваться независимо от мест хранения данных.

На практике в дополнение к перечисленным правилам существует также требование минимизации объемов памяти, занимаемых БД. Это достигается проектированием такой структуры БД, при которой дублирование (избыточность) информации было бы минимальным. Для выполнения этого требования была разработана *теория нормализации*. Она предполагает несколько уровней нормализации БД, каждый из которых базируется на предыдущем. Каждому уровню нормализации соответствует определенная нормальная форма (НФ). В зависимости от условий, которым удовлетворяет БД, говорят, что она имеет соответствующую нормальную форму. Например:

- БД имеет 1-ю НФ, если каждое значение, хранящееся в ней, неразделимо на более примитивные (неразложимость значений);
- БД имеет 2-ю НФ, если она имеет 1-ю НФ, и при этом каждое значение целиком и полностью зависит от ключа (функционально независимые значения);
- БД имеет 3-ю НФ, если она имеет 2-ю НФ, и при этом ни одно из значений не предоставляет никаких сведений о другом значении (взаимно независимые значения) и т. д.

В заключение описания реляционной модели необходимо заметить, что она имеет существенный недостаток. Дело в том, что не каждый тип информации можно представить в табличной форме, например изображение, музыку и др. Правда, в настоящее время для хранения такой информации в реляционных СУБД сделана попытка использовать специальные типы полей — BLOB

(Binary Large OBjects). В них хранятся ссылки на соответствующую информацию, которая не включается в БД. Однако такой подход не позволяет оперировать информацией, не помещенной в базу данных, что ограничивает возможности по ее использованию.

Для хранения такого вида информации предлагается использовать постреляционные модели в виде объектно-ориентированных структур хранения данных. Общий подход заключается в хранении любой информации в виде объектов. При этом сами объекты могут быть организованы в рамках иерархической модели. К сожалению, такой подход, в отличие от реляционной структуры, которая опирается на реляционную алгебру, недостаточно формализован, что не позволяет широко использовать его на практике.

В соответствии с правилами Кодда СУБД должна обеспечивать выполнение операций над БД, предоставляя при этом возможность одновременной работы нескольким пользователям (с нескольких компьютеров) и гарантируя целостность данных. Для выполнения этих правил в СУБД используется механизм управления транзакциями.

> ***Внимание!***
> *Транзакция* — это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция переводит БД из одного целостного состояния в другое.

Как правило, транзакцию составляют операции, манипулирующие с данными, принадлежащими разным таблицам и логически связанными друг с другом. Если при выполнении транзакции будут выполнены операции, модифицирующие только часть данных, а остальные данные не будут изменены, то будет нарушена целостность. Следовательно, либо все операции, включенные в транзакцию, должны быть выполненными, либо не выполнена ни одна из них. Процесс отмены выполнения транзакции называется откатом транзакции (ROLLBACK). Сохранение изменений, производимых в результате выполнения операций транзакции, называется фиксацией транзакции (COMMIT).

Свойство транзакции переводить БД из одного целостного состояния в другое позволяет использовать понятие транзакции как единицу активности пользователя. В случае одновременного обращения пользователей к БД транзакции, инициируемые разными пользователями, выполняются не параллельно (что невозможно для одной БД), а в соответствии с некоторым планом ставятся в очередь и выполняются последовательно. Таким образом, для пользователя, по инициативе которого образована транзакция, присутствие транзакций других пользователей будет незаметно, если не считать некоторого замедления работы по сравнению с однопользовательским режимом.

Существует несколько базовых алгоритмов планирования очередности транзакций. В централизованных СУБД наиболее распространены алгоритмы,

основанные на синхронизированных захватах объектов БД. При использовании любого алгоритма возможны ситуации конфликтов между двумя или более транзакциями по доступу к объектам БД. В этом случае для поддержания плана необходимо выполнять откат одной или более транзакций. Это один из случаев, когда пользователь многопользовательской СУБД может реально ощутить присутствие в системе транзакций других пользователей.

История развития СУБД тесно связана с совершенствованием подходов к решению задач хранения данных и управления транзакциями. Развитый механизм управления транзакциями в современных СУБД сделал их основным средством построения OLTP-систем, основной задачей которых является обеспечение выполнения операций с БД.

OLTP-системы оперативной обработки транзакций характеризуются большим количеством изменений, одновременным обращением множества пользователей к одним и тем же данным для выполнения разнообразных операций — чтения, записи, удаления или модификации данных. Для нормальной работы множества пользователей применяются блокировки и транзакции. Эффективная обработка транзакций и поддержка блокировок входят в число важнейших требований к системам оперативной обработки транзакций.

К этому классу систем относятся, кстати, и первые СППР — информационные системы руководства (ИСР, Executive Information Systems). Такие системы, как правило, строятся на основе реляционных СУБД, включают в себя подсистемы сбора, хранения и информационно-поискового анализа информации, а также содержат предопределенное множество запросов для повседневной работы. Каждый новый запрос, непредусмотренный при проектировании такой системы, должен быть сначала формально описан, закодирован программистом и только затем выполнен. Время ожидания в этом случае может составлять часы и дни, что неприемлемо для оперативного принятия решений.

## 1.3. Неэффективность использования OLTP-систем для анализа данных

Практика использования OLTP-систем показала неэффективность их применения для полноценного анализа информации. Такие системы достаточно успешно решают задачи сбора, хранения и поиска информации, но они не удовлетворяют требованиям, предъявляемым к современным СППР. Подходы, связанные с наращиванием функциональности OLTP-систем, не дали удовлетворительных результатов. Основной причиной неудачи является противоречивость требований, предъявляемых к системам OLTP и СППР. Перечень основных противоречий между этими системами приведен в табл. 1.1.

Таблица 1.1

| Характеристика | Требования к OLTP-системе | Требования к системе анализа |
|---|---|---|
| Степень детализации хранимых данных | Хранение только детализированных данных | Хранение как детализированных, так и обобщенных данных |
| Качество данных | Допускаются неверные данные из-за ошибок ввода | Не допускаются ошибки в данных |
| Формат хранения данных | Может содержать данные в разных форматах в зависимости от приложений | Единый согласованный формат хранения данных |
| Допущение избыточных данных | Должна обеспечиваться максимальная нормализация | Допускается контролируемая денормализация (избыточность) для эффективного извлечения данных |
| Управление данными | Должна быть возможность в любое время добавлять, удалять и изменять данные | Должна быть возможность периодически добавлять данные |
| Количество хранимых данных | Должны быть доступны все оперативные данные, требующиеся в данный момент | Должны быть доступны все данные, накопленные в течение продолжительного интервала времени |
| Характер запросов к данным | Доступ к данным пользователей осуществляется по заранее составленным запросам | Запросы к данным могут быть произвольными и заранее не оформлены |
| Время обработки обращений к данным | Время отклика системы измеряется в секундах | Время отклика системы может составлять несколько минут |
| Характер вычислительной нагрузки на систему | Постоянно средняя загрузка процессора | Загрузка процессора формируется только при выполнении запроса, но на 100 % |
| Приоритетность характеристик системы | Основными приоритетами являются высокая производительность и доступность | Приоритетными являются обеспечение гибкости системы и независимости работы пользователей |

Рассмотрим требования, предъявляемые к системам OLTP и СППР, более подробно.

- **Степень детализации хранимых данных.** Типичный запрос в OLTP-системе, как правило, выборочно затрагивает отдельные записи в таблицах, которые эффективно извлекаются с помощью индексов. В системах анализа, наоборот, требуется выполнять запросы сразу над большим количеством данных с широким применением группировок и обобщений (суммирования, агрегирования и т. п.).

  Например, в стандартных системах складского учета наиболее часто выполняются операции вычисления текущего количества определенного товара на складе, продажи и оплаты товаров покупателями и т. д. В системах анализа выполняются запросы, связанные с определением общей стоимости товаров, хранящихся на складе, категорий товаров, пользующихся наибольшим и наименьшим спросом, обобщение по категориям и суммирование по всем продажам товаров и т. д.

- **Качество данных.** OLTP-системы, как правило, хранят информацию, вводимую непосредственно пользователями систем (операторами ЭВМ). Присутствие "человеческого фактора" при вводе повышает вероятность ошибочных данных и может создать локальные проблемы в системе. При анализе ошибочные данные могут привести к неправильным выводам и принятию неверных стратегических решений.

- **Формат хранения данных.** OLTP-системы, обслуживающие различные участки работы, не связаны между собой. Они часто реализуются на разных программно-аппаратных платформах. Одни и те же данные в разных базах могут быть представлены в различном виде и могут не совпадать (например, данные о клиенте, который взаимодействовал с разными отделами компании, могут не совпадать в базах данных этих отделов). В процессе анализа такое различие форматов чрезвычайно затрудняет совместный анализ этих данных. Поэтому к системам анализа предъявляется требование единого формата. Как правило, необходимо, чтобы этот формат был оптимизирован для анализа данных (нередко за счет их избыточности).

- **Допущение избыточных данных.** Структура базы данных, обслуживающей OLTP-систему, обычно довольно сложна. Она может содержать многие десятки и даже сотни таблиц, ссылающихся друг на друга. Данные в такой БД сильно нормализованы для оптимизации занимаемых ресурсов. Аналитические запросы к БД очень трудно формулируются и крайне неэффективно выполняются, поскольку содержат представления, объединяющие большое количество таблиц. При проектировании систем анализа стараются максимально упростить схему БД и уменьшить количество таблиц, участвующих в запросе. С этой целью часто допускают денормализацию (избыточность данных) БД.

- **Управление данными.** Основное требование к OLTP-системам — обеспечить выполнение операций модификации над БД. При этом предполагается, что они должны выполняться в реальном режиме, и часто очень интенсивно. Например, при оформлении розничных продаж в систему вводятся соответствующие документы. Очевидно, что интенсивность ввода зависит от интенсивности покупок и в случае ажиотажа будет очень высокой, а любое промедление ведет к потере клиента. В отличие от OLTP-систем данные в системах анализа меняются редко. Единожды попав в систему, данные уже практически не изменяются. Ввод новых данных, как правило, носит эпизодический характер и выполняется в периоды низкой активности системы (например, раз в неделю на выходных).
- **Количество хранимых данных.** Как правило, системы анализа предназначены для анализа временных зависимостей, в то время как OLTP-системы обычно имеют дело с текущими значениями каких-либо параметров. Например, типичное складское приложение OLTP оперирует с текущими остатками товара на складе, в то время как в системе анализа может потребоваться анализ динамики продаж товара. По этой причине в OLTP-системах допускается хранение данных за небольшой период времени (например, за последний квартал). Для анализа данных, наоборот, необходимы сведения за максимально большой интервал времени.
- **Характер запросов к данным.** В OLTP-системах из-за нормализации БД составление запросов является достаточно сложной работой и требует необходимой квалификации. Поэтому для таких систем заранее составляется некоторый ограниченный набор статических запросов к БД, необходимый для работы с системой (например, наличие товара на складе, размер задолженности покупателей и т. п.). Для СППР невозможно заранее определить необходимые запросы, поэтому к ним предъявляется требование обеспечить формирование произвольных запросов к БД аналитиками.
- **Время обработки обращений к данным.** OLTP-системы, как правило, работают в режиме реального времени, поэтому к ним предъявляются жесткие требования по обработке данных. Например, время ввода документов продажи товаров (расходных накладных) и проверки наличия продаваемого товара на складе должно быть минимально, т. к. от этого зависит время обслуживания клиента. В системах анализа, по сравнению с OLTP, обычно выдвигают значительно менее жесткие требования ко времени выполнения запроса. При анализе данных аналитик может потратить больше времени для проверки своих гипотез. Его запросы могут выполняться в диапазоне от нескольких минут до нескольких часов.
- **Характер вычислительной нагрузки на систему.** Как уже отмечалось ранее, работа с OLTP-системами, как правило, выполняется в режиме ре-

ального времени. В связи с этим такие системы нагружены равномерно в течение всего интервала времени работы с ними. Документы продажи или прихода товара оформляются в общем случае постоянно в течение всего рабочего дня. Аналитик при работе с системой анализа обращается к ней для проверки некоторых своих гипотез и получения отчетов, графиков, диаграмм и т. п. При выполнении запросов степень загрузки системы высокая, т. к. обрабатывается большое количество данных, выполняются операции суммирования, группирования и т. п. Таким образом, характер загрузки систем анализа является пиковым. На рис. 1.3 приведены данные фирмы Oracle, отражающие загрузку процессора в течение дня, для систем OLTP, на рис. 1.4 — для систем анализа.

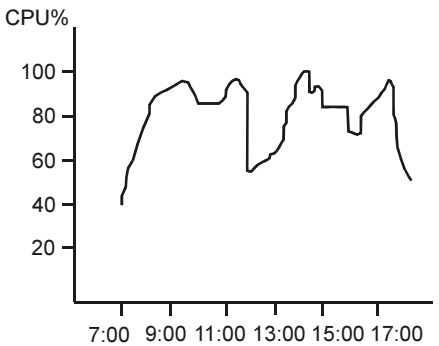



**Рис. 1.3.** Загрузка процессора для систем OLTP

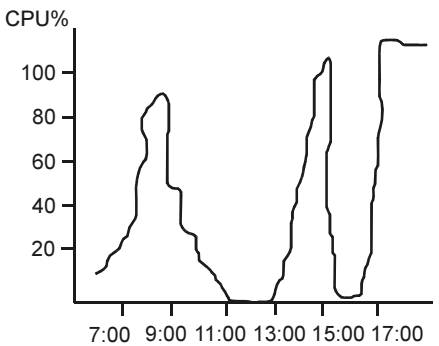



**Рис. 1.4.** Загрузка процессора для систем анализа

- **Приоритетность характеристик системы.** Для OLTP-систем приоритетным является высокая производительность и доступность данных, т. к. работа с ними ведется в режиме реального времени. Для систем анализа более приоритетными являются задачи обеспечения гибкости системы и независимости работы пользователей, т. е. то, что необходимо аналитикам для анализа данных.

Противоречивость требований к OLTP-системам и системам, ориентированным на глубокий анализ информации, усложняет задачу их интеграции как подсистем единой СППР. В настоящее время наиболее популярным решением этой проблемы является подход, ориентированный на использование концепции хранилищ данных.

Общая идея хранилищ данных заключается в разделении БД для OLTP-систем и БД для выполнения анализа и последующем их проектировании с учетом соответствующих требований.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- СППР решают три основные задачи: сбор, хранение и анализ хранимой информации. Задача анализа в общем виде может включать: информационно-поисковый анализ, оперативно-аналитический анализ и интеллектуальный анализ.
- Подсистемы сбора, хранения информации и решения задач информационно-поискового анализа в настоящее время успешно реализуются в рамках ИСР средствами СУБД. Для реализации подсистем, выполняющих оперативно-аналитический анализ, используется концепция многомерного представления данных (OLAP). Подсистема интеллектуального анализа данных реализует методы и алгоритмы Data Mining.
- Исторически выделяют три основные структуры БД: иерархическую, сетевую и реляционную. Первые две не нашли широкого применения на практике. В настоящее время подавляющее большинство БД реализует реляционную структуру представления данных.
- Основной недостаток реляционных БД заключается в невозможности обработки информации, которую нельзя представить в табличном виде. В связи с этим предлагается использовать постреляционные модели, например объектно-ориентированные.
- Для упрощения разработки прикладных программ, использующих БД, создаются системы управления базами данных (СУБД) — программное обеспечение для управления данными, их хранения и безопасности данных.
- В СУБД развит механизм управления транзакциями, что сделало их основным средством создания систем оперативной обработки транзакций (OLTP-систем). К таким системам относятся первые СППР, решающие задачи информационно-поискового анализа — ИСР.
- OLTP-системы не могут эффективно использоваться для решения задач оперативно-аналитического и интеллектуального анализа информации. Основная причина заключается в противоречивости требований к OLTP-системе и к СППР.
- В настоящее время для объединения в рамках одной системы OLTP-подсистем и подсистем анализа используется концепция хранилищ данных. Общая идея заключается в выделении БД для OLTP-подсистем и БД для выполнения анализа.

# ГЛАВА 2

# Хранилище данных

## 2.1. Концепция хранилища данных

Стремление объединить в одной архитектуре СППР возможности OLTP-систем и систем анализа, требования к которым во многом, как следует из табл. 1.1, противоречивы, привело к появлению концепции *хранилищ данных* (ХД).

Концепция ХД так или иначе обсуждалась специалистами в области информационных систем достаточно давно. Первые статьи, посвященные именно ХД, появились в 1988 г., их авторами были Б. Девлин и П. Мэрфи. В 1992 г. У. Инмон подробно описал данную концепцию в своей монографии "Построение хранилищ данных" ("Building the Data Warehouse", second edition — QED Publishing Group, 1996).

В основе концепции ХД лежит идея разделения данных, используемых для оперативной обработки и для решения задач анализа. Это позволяет применять структуры данных, которые удовлетворяют требованиям их хранения с учетом использования в OLTP-системах и системах анализа. Такое разделение позволяет оптимизировать как структуры данных оперативного хранения (оперативные БД, файлы, электронные таблицы и т. п.) для выполнения операций ввода, модификации, удаления и поиска, так и структуры данных, используемые для анализа (для выполнения аналитических запросов). В СППР эти два типа данных называются соответственно *оперативными источниками данных* (ОИД) и хранилищем данных.

В своей работе Инмон дал следующее определение ХД.

> ***Внимание!***
> *Хранилище данных* — предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.

Рассмотрим свойства ХД более подробно.

- **Предметная ориентация.** Это фундаментальное отличие ХД от ОИД. Разные ОИД могут содержать данные, описывающие одну и ту же предметную область с разных точек зрения (например, с точки зрения бухгалтерского учета, складского учета, планового отдела и т. п.). Решение, принятое на основе только одной точки зрения, может быть неэффективным или даже неверным. ХД позволяют интегрировать информацию, отражающую разные точки зрения на одну предметную область.

  Предметная ориентация позволяет также хранить в ХД только те данные, которые нужны для их анализа (например, для анализа нет смысла хранить информацию о номерах документов купли-продажи, в то время как их содержимое — количество, цена проданного товара — необходимо). Это существенно сокращает затраты на носители информации и повышает безопасность доступа к данным.

- **Интеграция.** ОИД, как правило, разрабатываются в разное время несколькими коллективами с собственным инструментарием. Это приводит к тому, что данные, отражающие один и тот же объект реального мира в разных системах, описывают его по-разному. Обязательная интеграция данных в ХД позволяет решить эту проблему, приведя данные к единому формату.

- **Поддержка хронологии.** Данные в ОИД необходимы для выполнения над ними операций в текущий момент времени. Поэтому они могут не иметь привязки ко времени. Для анализа данных часто бывает важно иметь возможность отслеживать хронологию изменений показателей предметной области. Поэтому все данные, хранящиеся в ХД, должны соответствовать последовательным интервалам времени.

- **Неизменяемость.** Требования к ОИД накладывают ограничения на время хранения в них данных. Те данные, которые не нужны для оперативной обработки, как правило, удаляются из ОИД для уменьшения занимаемых ресурсов. Для анализа, наоборот, требуются данные за максимально большой период времени. Поэтому, в отличие от ОИД, данные в ХД после загрузки только читаются. Это позволяет существенно повысить скорость доступа к данным, как за счет возможной избыточности хранящейся информации, так и за счет исключения операций модификации.

При реализации в СППР концепции ХД данные из разных ОИД копируются в единое хранилище. Собранные данные приводятся к единому формату, согласовываются и обобщаются. Аналитические запросы адресуются к ХД (рис. 2.1).

Такая модель неизбежно приводит к дублированию информации в ОИД и в ХД. Однако Инмон в своей работе утверждает, что избыточность данных, хранящихся в СППР, не превышает 1 %! Это можно объяснить следующими причинами.

При загрузке информации из ОИД в ХД данные фильтруются. Многие из них не попадают в ХД, поскольку лишены смысла с точки зрения использования в процедурах анализа.

Информация в ОИД носит, как правило, оперативный характер, и данные, потеряв актуальность, удаляются. В ХД, напротив, хранится историческая информация. С этой точки зрения дублирование содержимого ХД данными ОИД оказывается весьма незначительным. В ХД хранится обобщенная информация, которая в ОИД отсутствует.

Во время загрузки в ХД данные очищаются (удаляется ненужная информация), и после такой обработки они занимают гораздо меньший объем.

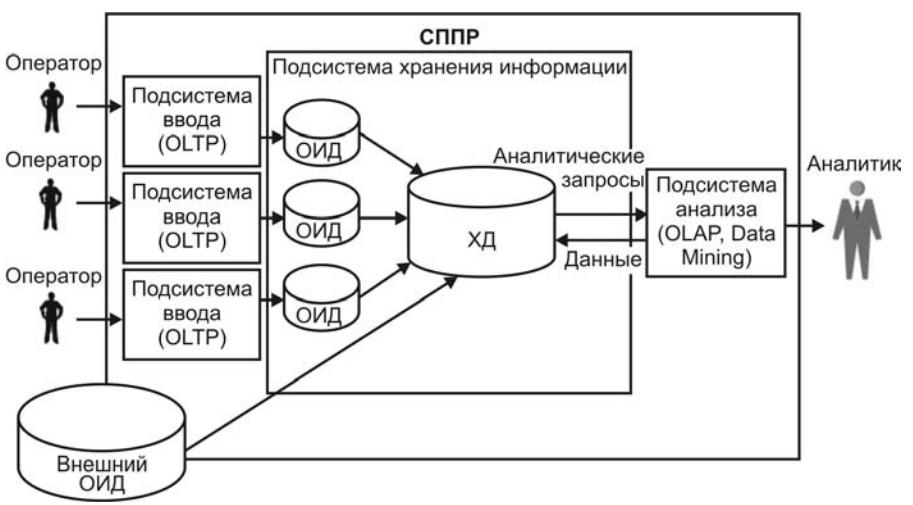



**Рис. 2.1.** Структура СППР с физическим ХД

Избыточность информации можно свести к нулю, используя виртуальное ХД. В данном случае в отличие от классического (физического) ХД данные из ОИД не копируются в единое хранилище. Они извлекаются, преобразуются и интегрируются непосредственно при выполнении аналитических запросов в оперативной памяти компьютера. Фактически такие запросы напрямую адресуются к ОИД (рис. 2.2). Основными достоинствами виртуального ХД являются:

- минимизация объема памяти, занимаемой на носителе информацией;
- работа с текущими, детализированными данными.

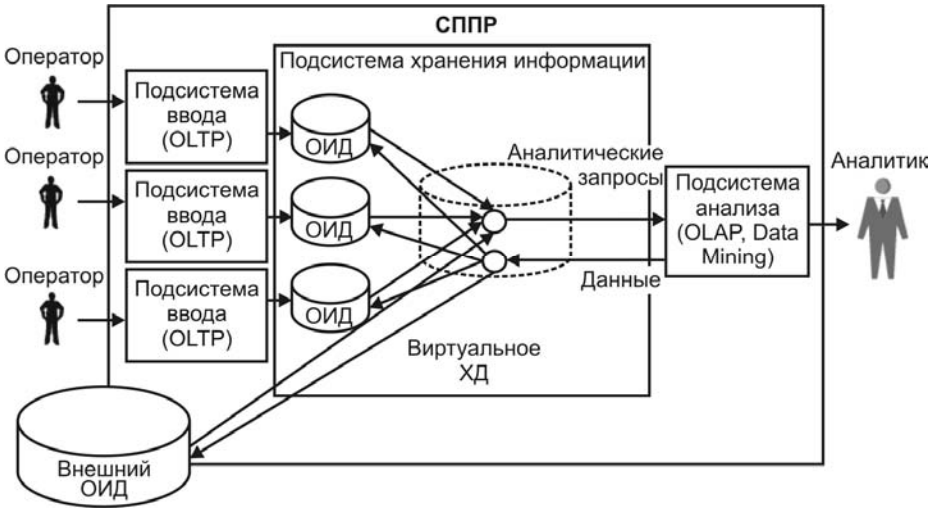



**Рис. 2.2.** Структура СППР с виртуальным ХД

Однако такой подход обладает многими недостатками.

Время обработки запросов к виртуальному ХД значительно превышает соответствующие показатели для физического хранилища. Кроме того, структуры оперативных БД, рассчитанные на интенсивное обновление одиночных записей, в высокой степени нормализованы. Для выполнения же аналитического запроса требуется объединение большого числа таблиц, что также приводит к снижению быстродействия.

Интегрированный взгляд на виртуальное хранилище возможен только при выполнении условия постоянной доступности всех ОИД. Таким образом, временная недоступность хотя бы одного из источников может привести либо к невыполнению аналитического запроса, либо к неверным результатам.

Выполнение сложных аналитических запросов над ОИД требует значительных ресурсов компьютеров. Это приводит к снижению быстродействия OLTP-систем, что недопустимо, т. к. время выполнения операций в таких системах часто весьма критично.

Различные ОИД могут поддерживать разные форматы и кодировки данных. Часто на один и тот же вопрос может быть получено несколько вариантов ответа. Это может быть связано с несинхронностью моментов обновления данных в разных ОИД, отличиями в описании одинаковых объектов и событий предметной области, ошибками при вводе, утерей фрагментов архивов и т. д. В таком случае цель — формирование единого непротиворечивого взгляда на объект управления — может быть не достигнута.

Главным же недостатком виртуального хранилища следует признать практическую невозможность получения данных за долгий период времени. При отсутствии физического хранилища доступны только те данные, которые на момент запроса есть в ОИД. Основное назначение OLTP-систем — оперативная обработка текущих данных, поэтому они не ориентированы на хранение данных за длительный период времени. По мере устаревания данные выгружаются в архив и удаляются из оперативной БД.

Несмотря на преимущества физического ХД перед виртуальным, необходимо признать, что его реализация представляет собой достаточно трудоемкий процесс. Остановимся на основных проблемах создания ХД:

- необходимость интеграции данных из неоднородных источников в распределенной среде;
- потребность в эффективном хранении и обработке очень больших объемов информации;
- необходимость наличия многоуровневых справочников метаданных;
- повышенные требования к безопасности данных.

Рассмотрим эти проблемы более подробно.

**Необходимость интеграции данных из неоднородных источников в распределенной среде.** ХД создаются для интегрирования данных, которые могут поступать из разнородных ОИД, физически размещающихся на разных компьютерах: БД, электронных архивов, публичных и коммерческих электронных каталогов, справочников, статистических сборников. При создании ХД приходится решать задачу построения системы, согласованно функционирующей с неоднородными программными средствами и решениями. При выборе средств реализации ХД приходится учитывать множество факторов, включающих уровень совместимости различных программных компонентов, легкость их освоения и использования, эффективность функционирования и т. д.

**Потребность в эффективном хранении и обработке очень больших объемов информации.** Свойство неизменности ХД предполагает накопление в нем информации за долгий период времени, что должно поддерживаться постоянным ростом объемов дисковой памяти. Ориентация на выполнение аналитических запросов и связанная с этим денормализация данных приводят к нелинейному росту объемов памяти, занимаемой ХД при возрастании объема данных. Исследования, проведенные на основе тестового набора TPC-D, показали, что для баз данных объемом в 100 Гбайт требуется память, в 4,87 раза бóльшая объемом, чем нужно для хранения полезных данных.

**Необходимость многоуровневых справочников метаданных.** Для систем анализа наличие развитых метаданных (данных о данных) и средств их предоставления конечным пользователям является одним из основных условий успешной реализации ХД. Метаданные необходимы пользователям СППР для понимания структуры информации, на основании которой принимается решение. Например, прежде чем менеджер корпорации задаст системе свой вопрос, он должен понять, какая информация имеется, насколько она актуальна, можно ли ей доверять, сколько времени может занять формирование ответа и т. д. При создании ХД необходимо решать задачи хранения и удобного представления метаданных пользователям.

**Повышение требований к безопасности данных.** Собранная вместе и согласованная информация об истории развития корпорации, ее успехах и неудачах, о взаимоотношениях с поставщиками и заказчиками, об истории и состоянии рынка дает возможность анализа прошлой и текущей деятельности корпорации и построения прогнозов для будущего. Очевидно, что подобная информация является конфиденциальной и доступ к ней ограничен в пределах самой компании, не говоря уже о других компаниях. Для обеспечения безопасности данных приходится решать вопросы аутентификации пользователей, защиты данных при их перемещении в хранилище данных из оперативных баз данных и внешних источников, защиты данных при их передаче по сети и т. п.

Снижения затрат на создание ХД можно добиться, создавая его упрощенный вариант — витрину данных (Data Mart).

> ***Внимание!***
> *Витрина данных* (ВД) — это упрощенный вариант ХД, содержащий только тематически объединенные данные.

ВД максимально приближена к конечному пользователю и содержит данные, тематически ориентированные на него (например, ВД для работников отдела маркетинга может содержать данные, необходимые для маркетингового анализа). ВД существенно меньше по объему, чем ХД, и для ее реализации не требуется больших затрат. Они могут быть реализованы как самостоятельно, так и вместе с ХД.

Самостоятельные ВД (рис. 2.3) часто появляются в организации исторически и встречаются в крупных организациях с большим количеством независимых подразделений, решающих собственные аналитические задачи.

Достоинствами такого подхода являются:

- проектирование ВД для ответов на определенный круг вопросов;
- быстрое внедрение автономных ВД и получение отдачи;
- упрощение процедур заполнения ВД и повышение их производительности за счет учета потребностей определенного круга пользователей.

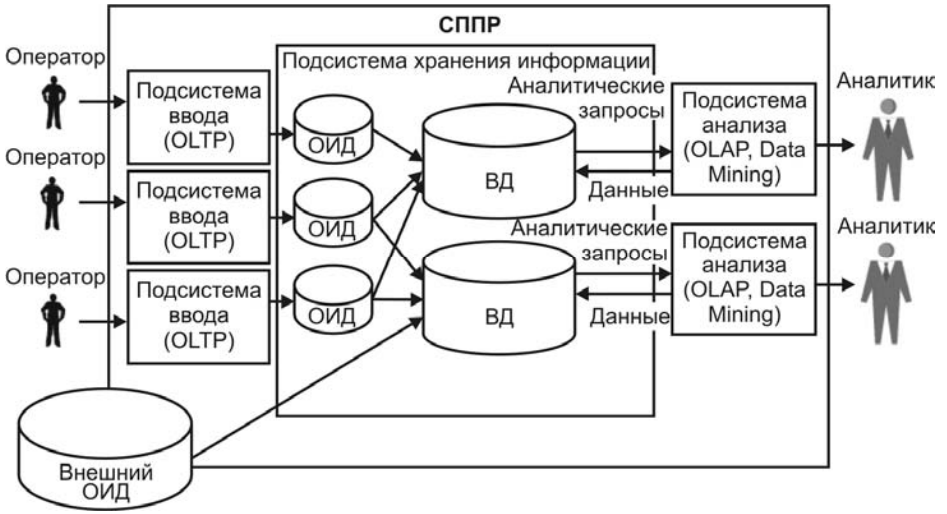



**Рис. 2.3.** Структура СППР с самостоятельными ВД

Недостатками автономных ВД являются:

- многократное хранение данных в разных ВД, что приводит к увеличению расходов на их хранение и потенциальным проблемам, связанным с необходимостью поддержания непротиворечивости данных;
- отсутствие консолидированности данных на уровне предметной области, а следовательно — отсутствие единой картины.

В последнее время все более популярной становится идея совместить ХД и ВД в одной системе. В этом случае ХД используется в качестве единственного источника интегрированных данных для всех ВД (рис. 2.4).

ХД представляет собой единый централизованный источник информации для всей предметной области, а ВД являются подмножествами данных из хранилища, организованными для представления информации по тематическим разделам данной области. Конечные пользователи имеют возможность доступа к детальным данным хранилища, если данных в витрине недостаточно, а также для получения более полной информационной картины.

Достоинствами такого подхода являются:

- простота создания и наполнения ВД, поскольку наполнение происходит из единого стандартизованного надежного источника очищенных данных — из ХД;
- простота расширения СППР за счет добавления новых ВД;
- снижение нагрузки на основное ХД.

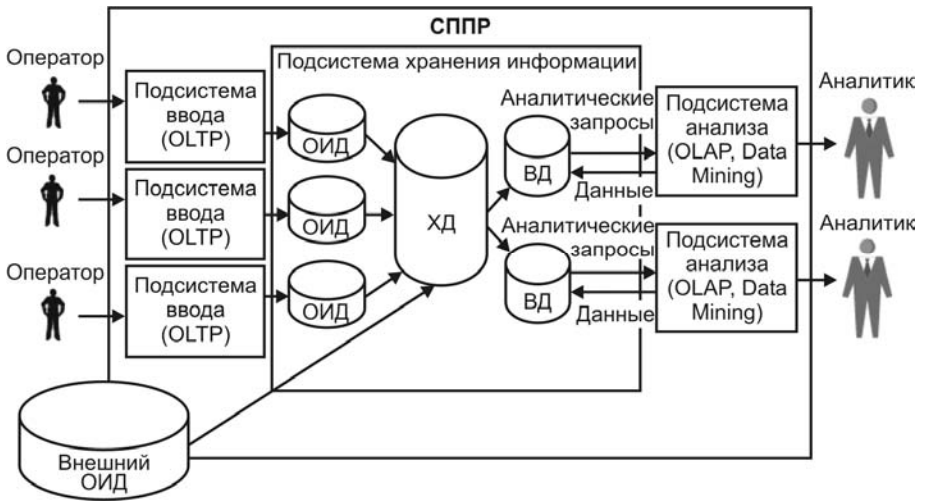



**Рис. 2.4.** Структура СППР с ХД и ВД

К недостаткам относятся:

- избыточность (данные хранятся как в ХД, так и в ВД);
- дополнительные затраты на разработку СППР с ХД и ВД.

Подводя итог анализу путей реализации СППР с использованием концепции ХД, можно выделить следующие архитектуры таких систем:

- СППР с физическим (классическим) ХД (см. рис. 2.1);
- СППР с виртуальным ХД (см. рис. 2.2);
- СППР с ВД (см. рис. 2.3);
- СППР с физическим ХД и с ВД (рис. 2.4).

В случае архитектур с физическим ХД и/или ВД необходимо уделить внимание вопросам организации (архитектуры) ХД и переносу данных из ОИД в ХД.

## 2.2. Организация ХД

Все данные в ХД делятся на три основные категории (рис. 2.5):

- детальные данные;
- агрегированные данные;
- метаданные.

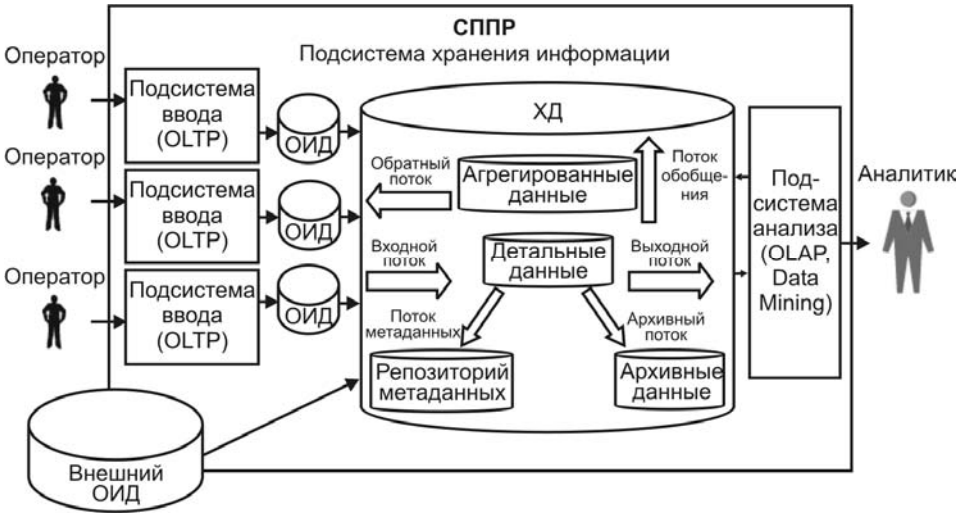



**Рис. 2.5.** Архитектура ХД

*Детальными* являются данные, переносимые непосредственно из ОИД. Они соответствуют элементарным событиям, фиксируемым OLTP-системами (например, продажи, эксперименты и др.). Принято разделять все данные на измерения и факты. *Измерениями* называются наборы данных, необходимые для описания событий (например, города, товары, люди и т. п.). *Фактами* называются данные, отражающие сущность события (например, количество проданного товара, результаты экспериментов и т. п.). Фактические данные могут быть представлены в виде числовых или категориальных значений.

В процессе эксплуатации ХД необходимость в ряде детальных данных может снизиться. Ненужные детальные данные могут храниться в архивах в сжатом виде на более емких накопителях с более медленным доступом (например, на магнитных лентах). Данные в архиве остаются доступными для обработки и анализа. Регулярно используемые для анализа данные должны храниться на накопителях с быстрым доступом (например, на жестких дисках).

На основании детальных данных могут быть получены *агрегированные* (обобщенные) данные. Агрегирование происходит путем суммирования числовых фактических данных по определенным измерениям. В зависимости от возможности агрегировать данные они подразделяются на следующие типы:

- *аддитивные* — числовые фактические данные, которые могут быть просуммированы по всем измерениям;
- *полуаддитивные* — числовые фактические данные, которые могут быть просуммированы только по определенным измерениям;

- *неаддитивные* — фактические данные, которые не могут быть просуммированы ни по одному измерению.

Большинство пользователей СППР работают не с детальными, а с агрегированными данными. Архитектура ХД должна предоставлять быстрый и удобный способ получать интересующую пользователя информацию. Для этого необходимо часть агрегированных данных хранить в ХД, а не вычислять их при выполнении аналитических запросов. Очевидно, что это ведет к избыточности информации и увеличению размеров ХД. Поэтому при проектировании таких систем важно добиться оптимального соотношения между вычисляемыми и хранящимися агрегированными данными. Те данные, к которым редко обращаются пользователи, могут вычисляться в процессе выполнения аналитических запросов. Данные, которые требуются более часто, должны храниться в ХД.

Для удобства работы с ХД необходима информация о содержащихся в нем данных. Такая информация называется *метаданными* (данные о данных). Согласно концепции Дж. Захмана, метаданные должны отвечать на следующие вопросы — что, кто, где, как, когда и почему:

- что (описание *объектов*) — метаданные описывают объекты предметной области, информация о которых хранится в ХД. Такое описание включает: атрибуты объектов, их возможные значения, соответствующие поля в информационных структурах ХД, источники информации об объектах и т. п.;
- кто (описание *пользователей*) — метаданные описывают категории пользователей, использующих данные. Они описывают права доступа к данным, а также включают в себя сведения о пользователях, выполнявших над данными различные операции (ввод, редактирование, загрузку, извлечение и т. п.);
- где (описание *места хранения*) — метаданные описывают местоположение серверов, рабочих станций, ОИД, размещенные на них программные средства и распределение между ними данных;
- как (описание *действий*) — метаданные описывают действия, выполняемые над данными. Описываемые действия могли выполняться как в процессе переноса из ОИД (например, исправление ошибок, расщепление полей и т. п.), так и в процессе их эксплуатации в ХД;
- когда (описание *времени*) — метаданные описывают время выполнения разных операций над данными (например, загрузка, агрегирование, архивирование, извлечение и т. п.);
- почему (описание *причин*) — метаданные описывают причины, повлекшие выполнение над данными тех или иных операций. Такими причинами могут быть требования пользователей, статистика обращений к данным и т. п.

Так как метаданные играют важную роль в процессе работы с ХД, то к ним должен быть обеспечен удобный доступ. Для этого они сохраняются в репозитории метаданных с удобным для пользователя интерфейсом.

Данные, поступающие из ОИД в ХД, перемещаемые внутри ХД и поступающие из ХД к аналитикам, образуют следующие информационные потоки (см. рис. 2.5):

- входной поток (Inflow) — образуется данными, копируемыми из ОИД в ХД;
- поток обобщения (Upflow) — образуется агрегированием детальных данных и их сохранением в ХД;
- архивный поток (Downflow) — образуется перемещением детальных данных, количество обращений к которым снизилось;
- поток метаданных (MetaFlow) — образуется переносом информации о данных в репозиторий данных;
- выходной поток (Outflow) — образуется данными, извлекаемыми пользователями;
- обратный поток (Feedback Flow) — образуется очищенными данными, записываемыми обратно в ОИД.

Самый мощный из информационных потоков — входной — связан с переносом данных из ОИД. Обычно информация не просто копируется в ХД, а подвергается обработке: данные очищаются и обогащаются за счет добавления новых атрибутов. Исходные данные из ОИД объединяются с информацией из внешних источников — текстовых файлов, сообщений электронной почты, электронных таблиц и др. При разработке ХД не менее 60 % всех затрат связано с переносом данных.

Процесс переноса, включающий в себя этапы извлечения, преобразования и загрузки, называют ETL-процессом (E — extraction, T — transformation, L — loading: извлечение, преобразование и загрузка соответственно). Программные средства, обеспечивающие его выполнение, называются ETL-системами. Традиционно ETL-системы использовались для переноса информации из устаревших версий информационных систем в новые. В настоящее время ETL-процесс находит все большее применение для переноса данных из ОИД в ХД и ВД.

Рассмотрим более подробно этапы ETL-процесса (рис. 2.6).

**Извлечение данных.** Чтобы начать ETL-процесс, необходимо извлечь данные из одного или нескольких источников и подготовить их к этапу преобразования. Можно выделить два способа извлечения данных:

1. Извлечение данных вспомогательными программными средствами непосредственно из структур хранения информации (файлов, электронных таб-

лиц, БД и т. п.). Достоинствами такого способа извлечения данных являются:

- отсутствие необходимости расширять OLTP-систему (это особенно важно, если ее структура закрыта);
- данные могут извлекаться с учетом потребностей процесса переноса.

2. Выгрузка данных средствами OLTP-систем в промежуточные структуры. Достоинствами такого подхода являются:
   - возможность использовать средства OLTP-систем, адаптированные к структурам данных;
   - средства выгрузки изменяются вместе с изменениями OLTP-систем и ОИД;
   - возможность выполнения первого шага преобразования данных за счет определенного формата промежуточной структуры хранения данных.

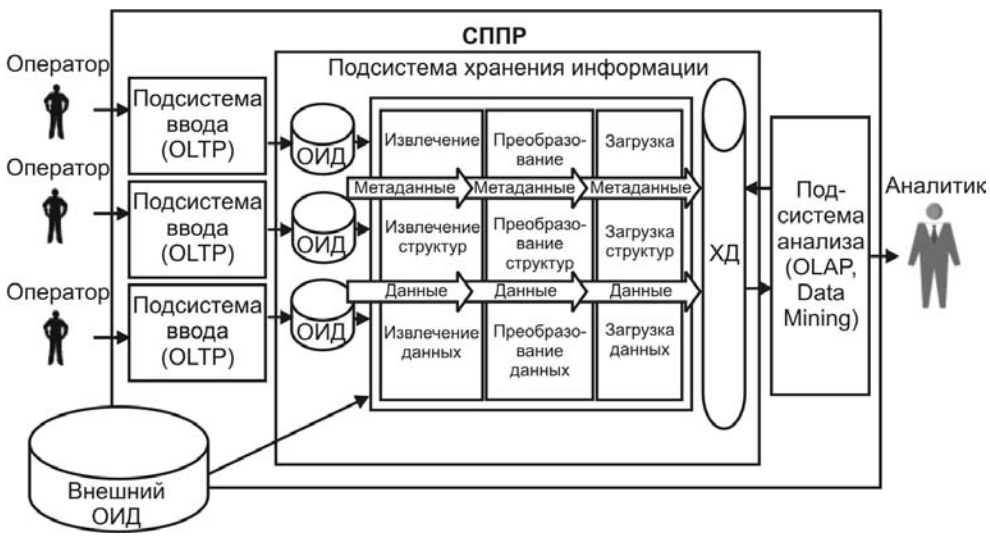



**Рис. 2.6.** ETL-процесс

**Преобразование данных.** После того как сбор данных завершен, необходимо преобразовать их для размещения на новом месте. На этом этапе выполняются следующие процедуры:

- обобщение данных (aggregation) — перед загрузкой данные обобщаются. Процедура обобщения заменяет многочисленные детальные данные относительно небольшим числом агрегированных данных. Например, предположим, что данные о продажах за год занимают в нормализованной базе

данных несколько тысяч записей. После обобщения данные преобразуются в меньшее число кратких записей, которые будут перенесены в ХД;

- перевод значений (value translation) — в ОИД данные часто хранятся в закодированном виде для того, чтобы сократить избыточность данных и память для их хранения. Например, названия товаров, городов, специальностей и т. п. могут храниться в сокращенном виде. Поскольку ХД содержат обобщенную информацию и рассчитаны на простое использование, закодированные данные обычно заменяют на более понятные описания;
- создание полей (field derivation) — при создании полей для конечных пользователей создается и новая информация. Например, ОИД содержит одно поле для указания количества проданных товаров, а второе — для указания цены одного экземпляра. Для исключения операции вычисления стоимости всех товаров можно создать специальное поле для ее хранения во время преобразования данных;
- очистка данных (cleaning) — направлена на выявление и удаление ошибок и несоответствий в данных с целью улучшения их качества. Проблемы с качеством встречаются в отдельных ОИД, например, в файлах и БД могут быть ошибки при вводе, отдельная информация может быть утрачена, могут присутствовать "загрязнения" данных и др. Очистка также применяется для согласования атрибутов полей таким образом, чтобы они соответствовали атрибутам базы данных назначения.

**Загрузка данных.** После того как данные преобразованы для размещения в ХД, осуществляется этап их загрузки. При загрузке выполняется запись преобразованных детальных и агрегированных данных. Кроме того, при записи новых детальных данных часть старых данных может переноситься в архив.

## 2.3. Очистка данных

Одной из важных задач, решаемых при переносе данных в ХД, является их очистка. С одной стороны, данные загружаются постоянно из различных источников, поэтому вероятность попадания "грязных данных" весьма высока. С другой стороны, ХД используются для принятия решений, и "грязные данные" могут стать причиной принятия неверных решений. Таким образом, процедура очистки является обязательной при переносе данных из ОИД в ХД. Ввиду большого спектра возможных несоответствий в данных их очистка считается одной из самых крупных проблем в технологии ХД. Основные проблемы очистки данных можно классифицировать по следующим уровням:

- уровень ячейки таблицы;
- уровень записи;

- уровень таблицы БД;
- уровень одиночной БД;
- уровень множества БД.

Рассмотрим перечисленные уровни и соответствующие им проблемы более подробно.

**Уровень ячейки таблицы.** На данном уровне задача очистки заключается в анализе и исправлении ошибок в данных, хранящихся в ячейках таблиц БД. К таким ошибкам можно отнести следующие.

- Орфографические ошибки (опечатки) — возникают при вводе информации. Они могут привести к неправильному пониманию, а также к искажению реальных данных. Например, при продаже товара вместо количества 1 000 было введено 10 000 или вместо названия товара "Водка" было введено название "Вода".
- Отсутствие данных — такие ошибки происходят из-за отсутствия у оператора соответствующих данных при вводе информации. Главной задачей OLTP-систем является обеспечение ежедневных операций с данными, поэтому оператор может пропустить ввод неизвестных ему данных, а не тратить время на их выяснение. Как следствие, в БД могут оставаться незаполненные ячейки (содержащие значение NULL).
- Фиктивные значения — это значения, введенные оператором, но не имеющие смысла. Наиболее часто такая проблема встречается в полях, обязательных для заполнения, но при отсутствии у оператора реальных данных он вынужден вводить бессмысленные данные, например: номер социального страхования 999-99-9999, или возраст клиента 999, или почтовый индекс 99999. Проблема усугубляется, если существует вероятность появления реальных данных, которые могут быть приняты за фиктивные, например, номер социального страхования 888-88-8888 для указания на статус клиента-иностранца "нерезидент" или месячный доход в размере $99 999,99 для указания на то, что клиент имеет работу.
- Логически неверные значения — значения, не соответствующие логическому смыслу, вкладываемому в данное поле таблицы. Например, в поле "Город" находится значение "Россия", или в поле "температура больного" значение 10.
- Закодированные значения — сокращенная запись или кодировка реальных данных, используемая для уменьшения занимаемого места.
- Составные значения — значения, содержащие несколько логических данных в одной ячейке таблицы. Такая ситуация возможна в полях произвольного формата (например, строковых или текстовых). Проблема усу-

губляется, если отсутствует строгий формат записи информации в такие поля.

**Уровень записи.** На данном уровне возникает проблема противоречивости значений в разных полях записи, описывающей один и тот же объект предметной области, например, когда возраст человека не соответствует его году рождения: `age=22`, `bdate=12.02.50`.

**Уровень таблицы БД.** На данном уровне возникают проблемы, связанные с несоответствием информации, хранящейся в таблице и относящейся к разным объектам. На этом уровне наиболее часто встречаются следующие проблемы.

- *Нарушение уникальности.* Значения, соответствующие уникальным атрибутам разных объектов предметной области, являются одинаковыми.
- *Отсутствие стандартов.* Из-за отсутствия стандартов на формат записи значений могут возникать проблемы, связанные с дублированием данных или с их противоречивостью:
  - дублирующиеся записи (один и тот же человек записан в таблицу два раза, хотя значения полей уникальны):

```
emp1=(name="John Smith", ...);
emp2=(name="J.Smith", ...);
```

  - противоречивые записи (об одном человеке в разных случаях введена разная информация о дате рождения, хотя значения полей уникальны):

```
emp1=(name="John Smith", bdate=12.02.70);
emp2=(name="J.Smith", bdate=12.12.70);
```

**Уровень одиночной БД.** На данном уровне, как правило, возникают проблемы, связанные с нарушением целостности данных.

**Уровень множества БД.** На этом уровне возникают проблемы, связанные с неоднородностью как структур БД, так и хранящейся в них информации. Можно выделить следующие основные проблемы этого уровня:

- различие структур БД (различие наименований полей, типов, размеров и др.);
- в разных БД существуют одинаковые наименования разных атрибутов;
- в разных БД одинаковые данные представлены по-разному;
- в разных БД классификация элементов разная;
- в разных БД временная градация разная;
- в разных БД ключевые значения, идентифицирующие один и тот же объект предметной области, разные и т. п.

При решении задачи очистки данных, прежде всего, необходимо отдавать себе отчет в том, что не все проблемы могут быть устранены. Возможны ситуации, когда данные не существуют и не могут быть восстановлены, вне зависимости от количества приложенных усилий. Встречаются ситуации, когда значения настолько запутаны или найдены в стольких несопоставимых местах с такими на вид различными и противоположными значениями одного и того же факта, что любая попытка расшифровать эти данные может породить еще более неверные результаты, и, возможно, лучшим решением будет отказаться от их обработки. На самом деле не все данные нужно очищать. Как уже отмечалось, процесс очистки требует больших затрат, поэтому те данные, достоверность которых не влияет на процесс принятия решений, могут оставаться неочищенными.

В целом, очистка данных включает в себя несколько этапов:

- выявление проблем в данных;
- определение правил очистки данных;
- тестирование правил очистки данных;
- непосредственная очистка данных.

**Выявление проблем в данных.** Для выявления подлежащих удалению видов ошибок и несоответствий необходим подробный анализ данных. Наряду с ручной проверкой следует использовать аналитические программы. Существует два взаимосвязанных метода анализа: профайлинг данных и Data Mining.

*Профайлинг данных* ориентирован на грубый анализ отдельных атрибутов данных. При этом происходит получение, например, такой информации, как тип, длина, спектр значений, дискретные значения данных и их частота, изменение, уникальность, наличие неопределенных значений, типичных строковых моделей (например, для номеров телефонов) и др., что позволяет обеспечить точное представление различных аспектов качества атрибута.

*Data Mining* помогает найти специфические модели в больших наборах данных, например отношения между несколькими атрибутами. Именно на это направлены так называемые описательные модели Data Mining, включая группировку, обобщение, поиск ассоциаций и последовательностей. При этом могут быть получены ограничения целостности в атрибутах, например, функциональные зависимости или характерные для конкретных приложений бизнес-правила, которые можно использовать для восполнения утраченных и исправления недопустимых значений, а также для выявления дубликатов записей в источниках данных. Например, правило объединения с высокой вероятностью может предсказать проблемы с качеством данных в элементах данных, нарушающих это правило. Таким образом, 99-процентная вероятность правила "итого = количество × единицу" демонстрирует несоответствие

и потребность в более детальном исследовании для оставшегося 1 процента записей.

**Определение правил очистки данных.** В зависимости от числа источников данных, степени их неоднородности и загрязненности, они могут требовать достаточно обширного преобразования и очистки. Первые шаги по очистке данных могут скорректировать проблемы отдельных источников данных и подготовить данные для интеграции. Дальнейшие шаги должны быть направлены на интеграцию данных и устранение проблем множественных источников.

На этом этапе необходимо выработать общие правила преобразования, часть из которых должна быть представлена в виде программных средств очистки.

**Тестирование правил очистки данных.** Корректность и эффективность правил очистки данных должны тестироваться и оцениваться, например, на копиях данных источника. Это необходимо для выяснения целесообразности корректировки правил с целью их улучшения или исправления ошибок.

Этапы определения правил и их тестирование могут выполняться итерационно несколько раз, например, из-за того, что некоторые ошибки становятся заметны только после определенных преобразований.

**Непосредственная очистка данных.** На этом этапе выполняются преобразования в соответствии с определенными ранее правилами. Очистка выполняется в два приема. Сначала устраняются проблемы, связанные с отдельными источниками данных, а затем — проблемы множества БД.

Над отдельными ОИД выполняются следующие процедуры.

- **Расщепление атрибутов.** Данная процедура извлекает значения из атрибутов свободного формата для повышения точности представления и поддержки последующих этапов очистки, таких как сопоставление элементов данных и исключение дубликатов. Необходимые на этом этапе преобразования перераспределяют значения в поле для получения возможности перемещения слов и извлекают значения для расщепленных атрибутов.
- **Проверка допустимости и исправления.** Эта процедура исследует каждый элемент данных источника на наличие ошибок. Обнаруженные ошибки автоматически исправляются (если это возможно). Проверка на наличие орфографических ошибок выполняется на основе просмотра словаря. Словари географических наименований и почтовых индексов помогают корректировать адресные данные. Атрибутивные зависимости (дата рождения — возраст, общая стоимость — цена за шт., город — региональный телефонный код и т. д.) могут использоваться для выявления проблем и замены утраченных или исправления неверных значений.

- **Стандартизация.** Эта процедура преобразует данные в согласованный и унифицированный формат, что необходимо для их дальнейшего согласования и интеграции. Например, записи о дате и времени должны быть оформлены в специальном формате, имена и другие символьные данные должны конвертироваться либо в прописные, либо в строчные буквы и т. д. Текстовые данные могут быть сжаты и унифицированы с помощью выявления основы (шаблона), удаления префиксов, суффиксов и вводных слов. Более того, аббревиатуры и зашифрованные схемы подлежат согласованной расшифровке с помощью специального словаря синонимов или применения предопределенных правил конверсии.

После того как ошибки отдельных источников удалены, очищенные данные должны заменить загрязненные данные в исходных ОИД. Это необходимо для повышения качества данных в ОИД и исключения затрат на очистку при повторном использовании. После завершения преобразований над данными из отдельных источников можно приступать к их интеграции. При этом выполняются следующие процедуры.

- **Сопоставление данных, относящихся к одному элементу.** Эта процедура устраняет противоречивость и дублирование данных из разных источников, относящихся к одному объекту предметной области. Для сопоставления записей из разных источников используются идентификационные атрибуты или комбинация атрибутов. Такими атрибутами могут выступать общие первичные ключи или другие общие уникальные атрибуты. К сожалению, без таких атрибутов процесс сопоставления данных затруднителен.

- **Слияние записей.** Данная процедура объединяет интегрированные записи, относящиеся к одному объекту. Объединение выполняется, если информация из разных записей дополняет или корректирует одна другую.

- **Исключение дубликатов.** Данная процедура удаляет дублирующие записи. Она производится либо над двумя очищенными источниками одновременно, либо над отдельным, уже интегрированным набором данных. Исключение дубликатов требует, в первую очередь, выявления (сопоставления) похожих записей, относящихся к одному и тому же объекту реального окружения.

Очищенные данные сохраняются в ХД и могут использоваться для анализа и принятия на их основе решений. За формирование аналитических запросов к данным и представление результатов их выполнения в СППР отвечают подсистемы анализа. От вида анализа также зависит и непосредственная реализация структур хранения данных в ХД.

## 2.4. Концепция хранилища данных и анализ

Концепция ХД не является законченным архитектурным решением СППР и тем более не является готовым программным продуктом. Цель концепции ХД — определить требования к данным, помещаемым в ХД, общие принципы и этапы построения ХД, основные источники данных, дать рекомендации по решению потенциальных проблем, возникающих при выгрузке, очистке, согласовании, транспортировке и загрузке данных.

Необходимо понимать, что концепция ХД:

- это не концепция анализа данных, скорее, это концепция подготовки данных для анализа;
- не предопределяет архитектуру целевой аналитической системы. Концепция ХД указывает на то, какие процессы должны выполняться в системе, но не где конкретно и как они будут выполняться.

Таким образом, концепция ХД определяет лишь самые общие принципы построения аналитической системы и в первую очередь сконцентрирована на свойствах и требованиях к данным, но не на способах организации и представления данных в целевой БД и режимах их использования. Концепция ХД описывает построение аналитической системы, но не определяет характер ее использования. Она не решает ни одну из следующих проблем:

- выбор наиболее эффективного для анализа способа организации данных;
- организация доступа к данным;
- использование технологии анализа.

Проблемы использования собранных данных решают подсистемы анализа. Как отмечалось в *гл. 1*, такие подсистемы используют следующие технологии:

- регламентированные запросы;
- оперативный анализ данных;
- интеллектуальный анализ данных.

Если регламентированные запросы успешно применялись еще задолго до появления концепции ХД, то оперативный и интеллектуальный анализы в последнее время все больше связывают с ХД.

## Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Концепция ХД предполагает разделение структур хранения данных для оперативной обработки и выполнения аналитических запросов. Это позво-

ляет в рамках одной СППР объединить две подсистемы, удовлетворяющие противоречивым требованиям.

- В соответствии с определением Инмона, ХД — это предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.
- Различают два вида ХД: виртуальное и физическое. В системах, реализующих концепцию виртуального ХД, аналитические запросы адресуются непосредственно к ОИД, а полученные результаты интегрируются в оперативной памяти компьютера. В случае физического ХД данные переносятся из разных ОИД в единое хранилище, к которому адресуются аналитические запросы.
- Облегченным вариантом ХД является ВД, которая содержит только тематически объединенные данные. ВД существенно меньше по объему, чем ХД, и для ее реализации не требуется больших затрат. ВД может быть реализована или самостоятельно, или в комбинации с ХД.
- ХД включает в себя: метаданные, детальные, агрегированные и архивные данные. Перемещающиеся в ХД данные образуют информационные потоки: входной, обобщающий, обратный, выходной и поток метаданных.
- Детальные данные разделяют на два класса: измерения и факты. Измерениями называются наборы данных, необходимые для описания событий. Фактами называются данные, отражающие сущность события.
- Агрегированные данные получаются из детальных данных путем их суммирования по измерениям. Для быстрого доступа к наиболее часто запрашиваемым агрегированным данным они должны сохраняться в ХД, а не вычисляться при выполнении запросов.
- Метаданные необходимы для получения пользователем информации о данных, хранящихся в ХД. Согласно принципам Захмана, метаданные должны описывать объекты предметной области, представленные в ХД, пользователей, работающих с данными, места хранения данных, действия над данными, время обработки данных и причины модификаций данных.
- Наиболее мощным информационным потоком в ХД является входной — поток переноса данных из ОИД в ХД. Процесс переноса, включающий этапы сбора, преобразования и загрузки, называют ETL-процессом.
- Наиболее важной задачей при переносе данных является их очистка. Основные проблемы очистки данных можно классифицировать по следующим уровням: уровень ячейки таблицы, уровень записи, уровень таблицы БД, уровень одиночной БД, уровень множества БД.

- Очистка данных включает следующие этапы: выявление проблем в данных, определение правил очистки, тестирование правил очистки, непосредственная очистка данных. После исправления ошибок отдельных источников очищенные данные должны заменить загрязненные данные в исходных ОИД.
- Очищенные данные сохраняются в ХД и могут использоваться для анализа и принятия на их основе решений. За формирование аналитических запросов к данным и представление результатов их выполнения в СППР отвечают подсистемы анализа. От вида анализа также зависит и непосредственная реализация структур хранения данных в ХД.

# ГЛАВА 3

# OLAP-системы

## 3.1. Многомерная модель данных

Как отмечалось в *гл. 2*, в концепции ХД нет постановки вопросов, связанных с организацией эффективного анализа данных и предоставления доступа к ним. Эти задачи решаются подсистемами анализа. Попробуем разобраться, какой способ работы с данными наиболее подходит пользователю СППР — аналитику.

В процессе принятия решений пользователь генерирует некоторые гипотезы. Для превращения их в законченные решения эти гипотезы должны быть проверены. Проверка гипотез осуществляется на основании информации об анализируемой предметной области. Как правило, наиболее удобным способом представления такой информации для человека является зависимость между некоторыми параметрами. Например, зависимость объемов продаж от региона, времени, категории товара и т. п. Другим примером может служить зависимость количества выздоравливающих пациентов от применяемых средств лечения, возраста и т. п.

В процессе анализа данных, поиска решений часто возникает необходимость в построении зависимостей между различными параметрами. Кроме того, число таких параметров может варьироваться в широких пределах. Как уже отмечалось ранее, традиционные средства анализа, оперирующие данными, которые представлены в виде таблиц реляционной БД, не могут в полной мере удовлетворять таким требованиям. В 1993 г. Е. Кодд — основоположник реляционной модели БД — рассмотрел ее недостатки, указав в первую очередь на невозможность "объединять, просматривать и анализировать данные с точки зрения множественности измерений, т. е. самым понятным для аналитиков способом".

*Измерение* — это последовательность значений одного из анализируемых параметров. Например, для параметра "время" это последовательность календарных дней, для параметра "регион" это может быть список городов.

Множественность измерений предполагает представление данных в виде многомерной модели. По измерениям в многомерной модели откладывают параметры, относящиеся к анализируемой предметной области.

По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) — это множественная перспектива, состоящая из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям определяется как многомерный анализ.

Каждое измерение может быть представлено в виде иерархической структуры. Например, измерение "Исполнитель" может иметь следующие иерархические уровни: "предприятие — подразделение — отдел — служащий". Более того, некоторые измерения могут иметь несколько видов иерархического представления. Например, измерение "Время" может включать две иерархии со следующими уровнями: "год — квартал — месяц — день" и "неделя — день".

На пересечениях осей измерений (Dimensions) располагаются данные, количественно характеризующие анализируемые факты, — меры (Measures). Это могут быть объемы продаж, выраженные в единицах продукции или в денежном выражении, остатки на складе, издержки и т. п.

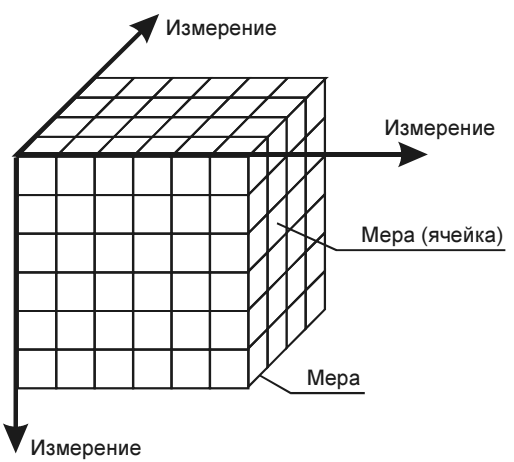



**Рис. 3.1.** Представление данных в виде гиперкуба

Таким образом, многомерную модель данных можно представить как гиперкуб (рис. 3.1) (конечно, название не очень удачное, поскольку под кубом

обычно понимают фигуру с равными ребрами, что в данном случае далеко не так). Ребрами такого гиперкуба являются измерения, а ячейками — меры.

Над таким гиперкубом могут выполняться следующие операции.

- **Срез** (Slice) (рис. 3.2) — формирование подмножества многомерного массива данных, соответствующего единственному значению одного или нескольких элементов измерений, не входящих в это подмножество. Например, при выборе элемента "Факт", измерения "Сценарий" срез данных представляет собой подкуб, в который входят все остальные измерения. Данные, что не вошли в сформированный срез, связаны с теми элементами измерения "Сценарий", которые не были указаны в качестве определяющих (например, "План", "Отклонение", "Прогноз" и т. п.). Если рассматривать термин "срез" с позиции конечного пользователя, то наиболее часто его роль играет двумерная проекция куба.

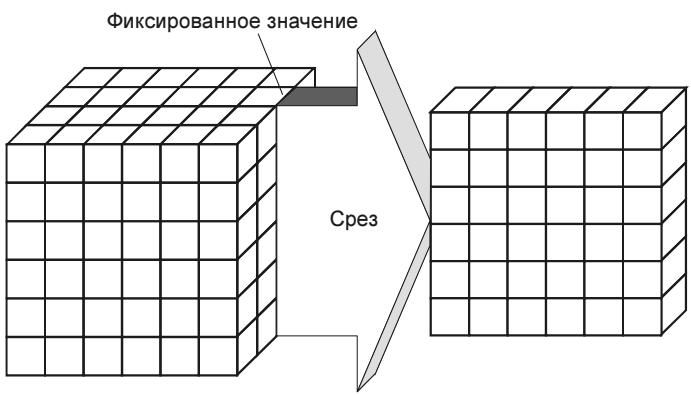



**Рис. 3.2.** Операция среза

- **Вращение** (Rotate) (рис. 3.3) — изменение расположения измерений, представленных в отчете или на отображаемой странице. Например, операция вращения может заключаться в перестановке местами строк и столбцов таблицы или перемещении интересующих измерений в столбцы или строки создаваемого отчета, что позволяет придавать ему желаемый вид. Кроме того, вращением куба данных является перемещение внетабличных измерений на место измерений, представленных на отображаемой странице, и наоборот (при этом внетабличное измерение становится новым измерением строки или измерением столбца). В качестве примера первого случая может служить отчет, для которого элементы измерения "Время" располагаются поперек экрана (являются заголовками столбцов таблицы), а элементы измерения "Продукция" — вдоль экрана (заголовки строк таблицы).

После применения операции вращения отчет будет иметь следующий вид: элементы измерения "Продукция" будут расположены по горизонтали, а элементы измерения "Время" — по вертикали. Примером второго случая может служить преобразование отчета с измерениями "Меры" и "Продукция", расположенными по вертикали, и измерением "Время", расположенным по горизонтали, в отчет, у которого измерение "Меры" располагается по вертикали, а измерения "Время" и "Продукция" — по горизонтали. При этом элементы измерения "Время" располагаются над элементами измерения "Продукция". Для третьего случая применения операции вращения можно привести пример преобразования отчета с расположенными по горизонтали измерением "Время" и по вертикали измерением "Продукция" в отчет, у которого по горизонтали представлено измерение "Время", а по вертикали — измерение "География".

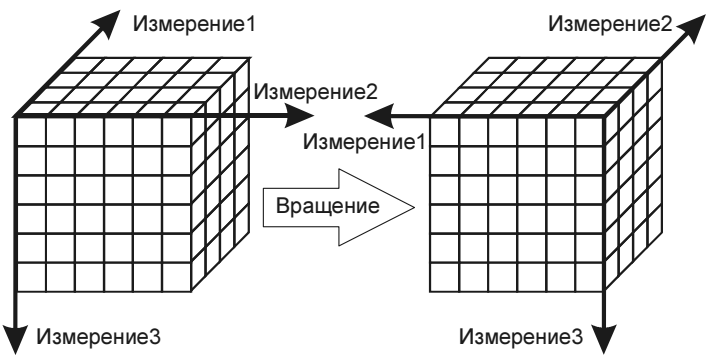



**Рис. 3.3.** Операция вращения

- **Консолидация** (Drill Up) и **детализация** (Drill Down) (рис. 3.4) — операции, которые определяют переход вверх по направлению от детального (down) представления данных к агрегированному (up) и наоборот, соответственно. Направление детализации (обобщения) может быть задано как по иерархии отдельных измерений, так и согласно прочим отношениям, установленным в рамках измерений или между измерениями. Например, если при анализе данных об объемах продаж в Северной Америке выполнить операцию Drill Down для измерения "Регион", то на экране будут отображены такие его элементы, как "Канада", "Восточные Штаты Америки" и "Западные Штаты Америки". В результате дальнейшей детализации элемента "Канада" будут отображены элементы "Торонто", "Ванкувер", "Монреаль" и т. д.

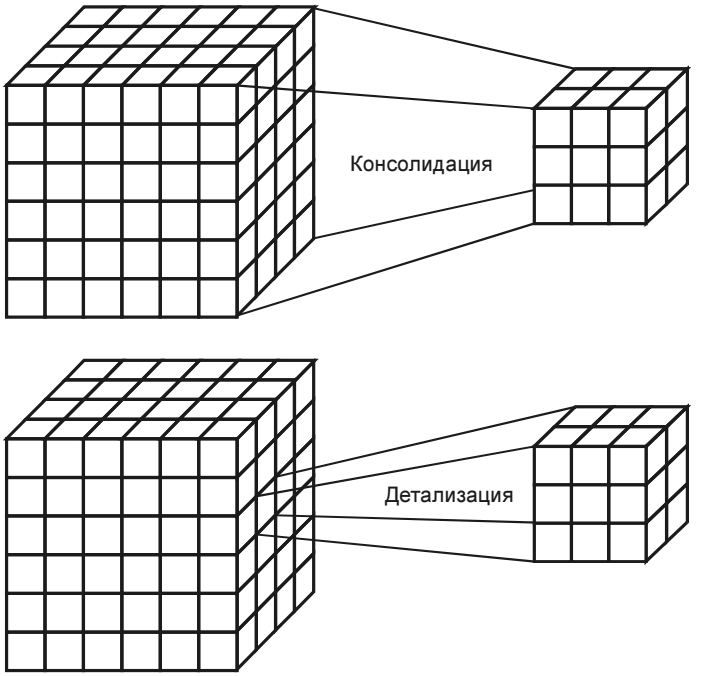



**Рис. 3.4.** Операции консолидации и детализации

# 3.2. Определение OLAP-систем

С концепцией многомерного анализа данных тесно связывают оперативный анализ, который выполняется средствами OLAP-систем.

> ***Внимание!***
> OLAP (On-Line Analytical Processing) — технология оперативной аналитической обработки данных, использующая методы и средства для сбора, хранения и анализа многомерных данных в целях поддержки процессов принятия решений.

Основное назначение OLAP-систем — поддержка аналитической деятельности, произвольных (часто используется термин ad-hoc) запросов пользователей-аналитиков. Цель OLAP-анализа — проверка возникающих гипотез.

У истоков технологии OLAP стоит основоположник реляционного подхода Э. Кодд. В 1993 г. он опубликовал статью под названием "OLAP для пользователей-аналитиков: каким он должен быть". В данной работе изложены основные концепции оперативной аналитической обработки и определены следующие 12 требований, которым должны удовлетворять продукты, позволяющие выполнять оперативную аналитическую обработку.

# 3.3. Концептуальное многомерное представление

## 3.3.1. Двенадцать правил Кодда

Далее перечислены 12 правил, изложенных Коддом и определяющих OLAP:

1. **Многомерность.** OLAP-система на концептуальном уровне должна представлять данные в виде многомерной модели, что упрощает процессы анализа и восприятия информации.
2. **Прозрачность.** OLAP-система должна скрывать от пользователя реальную реализацию многомерной модели, способ организации, источники, средства обработки и хранения.
3. **Доступность.** OLAP-система должна предоставлять пользователю единую, согласованную и целостную модель данных, обеспечивая доступ к данным независимо от того, как и где они хранятся.
4. **Постоянная производительность при разработке отчетов.** Производительность OLAP-систем не должна значительно уменьшаться при увеличении количества измерений, по которым выполняется анализ.
5. **Клиент-серверная архитектура.** OLAP-система должна быть способна работать в среде "клиент-сервер", т. к. большинство данных, которые сегодня требуется подвергать оперативной аналитической обработке, хранятся распределенно. Главной идеей здесь является то, что серверный компонент инструмента OLAP должен быть достаточно интеллектуальным и позволять строить общую концептуальную схему на основе обобщения и консолидации различных логических и физических схем корпоративных БД для обеспечения эффекта прозрачности.
6. **Равноправие измерений.** OLAP-система должна поддерживать многомерную модель, в которой все измерения равноправны. При необходимости дополнительные характеристики могут быть предоставлены отдельным измерениям, но такая возможность должна быть у любого измерения.
7. **Динамическое управление разреженными матрицами.** OLAP-система должна обеспечивать оптимальную обработку разреженных матриц. Скорость доступа должна сохраняться вне зависимости от расположения ячеек данных и быть постоянной величиной для моделей, имеющих разное число измерений и различную степень разреженности данных.
8. **Поддержка многопользовательского режима.** OLAP-система должна предоставлять возможность нескольким пользователям работать совместно с одной аналитической моделью или должна создавать для них различные модели из единых данных. При этом возможны как чтение, так и

запись данных, поэтому система должна обеспечивать их целостность и безопасность.

9. **Неограниченные перекрестные операции.** OLAP-система должна обеспечивать сохранение функциональных отношений, описанных с помощью определенного формального языка между ячейками гиперкуба при выполнении любых операций среза, вращения, консолидации или детализации. Система должна самостоятельно (автоматически) выполнять преобразование установленных отношений, не требуя от пользователя их переопределения.
10. **Интуитивная манипуляция данными.** OLAP-система должна предоставлять способ выполнения операций среза, вращения, консолидации и детализации над гиперкубом без необходимости пользователю совершать множество действий с интерфейсом. Измерения, определенные в аналитической модели, должны содержать всю необходимую информацию для выполнения вышеуказанных операций.
11. **Гибкие возможности получения отчетов.** OLAP-система должна поддерживать различные способы визуализации данных, т. е. средства формирования отчетов должны представлять синтезируемые данные или информацию, следующую из модели данных, в ее любой возможной ориентации. Это означает, что строки, столбцы или страницы должны показывать одновременно от 0 до $N$ измерений, где $N$ — число измерений всей аналитической модели. Кроме того, каждое измерение содержимого, показанное в одной записи, колонке или странице, должно позволять показывать любое подмножество элементов (значений), содержащихся в измерении, в любом порядке.
12. **Неограниченная размерность и число уровней агрегации.** Исследование о возможном числе необходимых измерений, требующихся в аналитической модели, показало, что одновременно могут использоваться до 19 измерений. Отсюда вытекает настоятельная рекомендация, чтобы аналитический инструмент мог одновременно предоставить хотя бы 15, а предпочтительнее — и 20 измерений. Более того, каждое из общих измерений не должно быть ограничено по числу определяемых пользователем-аналитиком уровней агрегации и путей консолидации.

## 3.3.2. Дополнительные правила Кодда

Набор правил Кодда, послуживших де-факто определением OLAP, достаточно часто вызывает различные нарекания, например, правила 1, 2, 3, 6 являются требованиями, а правила 10, 11 — неформализованными пожеланиями. Таким образом, перечисленные 12 правил Кодда не позволяют точно определить OLAP.

В 1995 г. Кодд к приведенному перечню добавил следующие шесть правил:

1. **Пакетное извлечение против интерпретации.** OLAP-система должна в равной степени эффективно обеспечивать доступ как к собственным, так и к внешним данным.
2. **Поддержка всех моделей OLAP-анализа.** OLAP-система должна поддерживать все четыре модели анализа данных, определенные Коддом: категориальную, толковательную, умозрительную и стереотипную.
3. **Обработка ненормализованных данных.** OLAP-система должна быть интегрирована с ненормализованными источниками данных. Модификации данных, выполненные в среде OLAP, не должны приводить к изменениям данных, хранимых в исходных внешних системах.
4. **Сохранение результатов OLAP: хранение их отдельно от исходных данных.** OLAP-система, работающая в режиме чтения-записи, после модификации исходных данных должна сохранять результаты отдельно. Иными словами, должна обеспечиваться безопасность исходных данных.
5. **Исключение отсутствующих значений.** OLAP-система, представляя данные пользователю, должна отбрасывать все отсутствующие значения. Другими словами, отсутствующие значения должны отличаться от нулевых значений.
6. **Обработка отсутствующих значений.** OLAP-система должна игнорировать все отсутствующие значения без учета их источника. Эта особенность связана с 17-м правилом.

Кроме того, Е. Кодд разбил все 18 правил на следующие четыре группы, назвав их *особенностями*. Эти группы получили названия B, S, R и D.

*Основные особенности* (B) включают следующие правила:

- многомерное концептуальное представление данных (правило 1);
- интуитивное манипулирование данными (правило 10);
- доступность (правило 3);
- пакетное извлечение против интерпретации (правило 13);
- поддержка всех моделей OLAP-анализа (правило 14);
- архитектура "клиент-сервер" (правило 5);
- прозрачность (правило 2);
- многопользовательская поддержка (правило 8).

*Специальные особенности* (S):

- обработка ненормализованных данных (правило 15);
- сохранение результатов OLAP: хранение их отдельно от исходных данных (правило 16);

- исключение отсутствующих значений (правило 17);
- обработка отсутствующих значений (правило 18).

*Особенности представления отчетов* (R):

- гибкость формирования отчетов (правило 11);
- постоянная производительность отчетов (правило 4);
- автоматическая настройка физического уровня (измененное оригинальное правило 7).

*Управление измерениями* (D):

- универсальность измерений (правило 6);
- неограниченное число измерений и уровней агрегации (правило 12);
- неограниченные операции между размерностями (правило 9).

## 3.3.3. Тест FASMI

Помимо рассмотренных ранее особенностей известен также тест FASMI (Fast of Analysis Shared Multidimensional Information), созданный в 1995 г. Н. Пендсом и Р. Критом на основе анализа правил Кодда. В данном контексте акцент сделан на скорость обработки, многопользовательский доступ, релевантность информации, наличие средств статистического анализа и многомерность, т. е. на представление анализируемых фактов как функций от большого числа их характеризующих параметров. Таким образом, Пендс и Крит определили OLAP следующими пятью ключевыми словами: FAST (Быстрый), ANALYSIS (Анализ), SHARED (Разделяемой), MULTIDIMENSIONAL (Многомерной), INFORMATION (Информации). Изложим эти пять ключевых представлений более подробно.

- **FAST (Быстрый).** OLAP-система должна обеспечивать выдачу большинства ответов пользователям в пределах приблизительно 5 секунд. При этом самые простые запросы обрабатываются в течение 1 секунды, и очень немногие — более 20 секунд. Недавнее исследование в Нидерландах показало, что конечные пользователи воспринимают процесс неудачным, если результаты не получены по истечении 30 секунд. Они могут нажать комбинацию клавиш <Alt>+<Ctrl>+<Del>, если система не предупредит их, что обработка данных требует большего времени. Даже если система предупредит, что процесс будет длиться существенно дольше, пользователи могут отвлечься и потерять мысль, при этом качество анализа страдает. Такой скорости нелегко достигнуть с большим количеством данных, особенно если требуются специальные вычисления "на лету". Для достижения данной цели используются разные методы, включая применение аппаратных платформ с большей производительностью.

- **ANALYSIS (Анализ).** OLAP-система должна справляться с любым логическим и статистическим анализом, характерным для данного приложения, и обеспечивать его сохранение в виде, доступном для конечного пользователя. Естественно, система должна позволять пользователю определять новые специальные вычисления как часть анализа и формировать отчеты любым желаемым способом без необходимости программирования. Все требуемые функциональные возможности анализа должны обеспечиваться понятным для конечных пользователей способом.

- **SHARED (Разделяемой).** OLAP-система должна выполнять все требования защиты конфиденциальности (возможно, до уровня ячейки хранения данных). Если для записи необходим множественный доступ, обеспечивается блокировка модификаций на соответствующем уровне. Обработка множественных модификаций должна выполняться своевременно и безопасным способом.

- **MULTIDIMENSIONAL (Многомерной).** OLAP-система должна обеспечить многомерное концептуальное представление данных, включая полную поддержку для иерархий и множественных иерархий, обеспечивающих наиболее логичный способ анализа. Это требование не устанавливает минимальное число измерений, которые должны быть обработаны, поскольку этот показатель зависит от приложения. Оно также не определяет используемую технологию БД, если пользователь действительно получает многомерное концептуальное представление информации.

- **INFORMATION (Информации).** OLAP-система должна обеспечивать получение необходимой информации в условиях реального приложения. Мощность различных систем измеряется не объемом хранимой информации, а количеством входных данных, которые они могут обработать. В этом смысле мощность продуктов весьма различна. Большие OLAP-системы могут оперировать по крайней мере в 1 000 раз бóльшим количеством данных по сравнению с простыми версиями OLAP-систем. При этом следует учитывать множество факторов, включая дублирование данных, требуемую оперативную память, использование дискового пространства, эксплуатационные показатели, интеграцию с информационными хранилищами и т. п.

## 3.4. Архитектура OLAP-систем

OLAP-система включает в себя два основных компонента:

- OLAP-сервер — обеспечивает хранение данных, выполнение над ними необходимых операций и формирование многомерной модели на концеп-

туальном уровне. В настоящее время OLAP-серверы объединяют с ХД или ВД;

- OLAP-клиент — представляет пользователю интерфейс к многомерной модели данных, обеспечивая его возможностью удобно манипулировать данными для выполнения задач анализа.

OLAP-серверы скрывают от конечного пользователя способ реализации многомерной модели. Они формируют гиперкуб, с которым пользователи посредством OLAP-клиента выполняют все необходимые манипуляции, анализируя данные. Между тем способ реализации очень важен, т. к. от него зависят такие характеристики, как производительность и занимаемые ресурсы. Выделяют три основных способа реализации:

- *MOLAP* — многомерный (multivariate) OLAP. Для реализации многомерной модели используют многомерные БД;
- *ROLAP* — реляционный (relational) OLAP. Для реализации многомерной модели используют реляционные БД;
- *HOLAP* — гибридный (hybrid) OLAP. Для реализации многомерной модели используют и многомерные, и реляционные БД.

Часто в литературе по OLAP-системам можно встретить аббревиатуры DOLAP и JOLAP:

- *DOLAP* — настольный (desktop) OLAP. Является недорогой и простой в использовании OLAP-системой, предназначенной для локального анализа и представления данных, которые загружаются из реляционной или многомерной БД на машину клиента;
- *JOLAP* — новая, основанная на Java коллективная OLAP-API-инициатива, предназначенная для создания и управления данными и метаданными на серверах OLAP. Основной разработчик — Hyperion Solutions. Другими членами группы, определяющей предложенный API, являются компании IBM, Oracle и др.

## 3.4.1. MOLAP

MOLAP-серверы используют для хранения и управления данными многомерных БД. При этом данные хранятся в виде упорядоченных многомерных массивов. Такие массивы подразделяются на гиперкубы и поликубы.

В *гиперкубе* все хранимые в БД ячейки имеют одинаковую мерность, т. е. находятся в максимально полном базисе измерений.

В *поликубе* каждая ячейка хранится с собственным набором измерений, и все связанные с этим сложности обработки перекладываются на внутренние механизмы системы.

Очевидно, что физически данные, представленные в многомерном виде, хранятся в "плоских" файлах. При этом куб представляется в виде одной плоской таблицы, в которую построчно вписываются все комбинации членов всех измерений с соответствующими им значениями мер (табл. 3.1).

**Таблица 3.1**

| Измерения | | | | Меры | |
|---|---|---|---|---|---|
| **Клиент** | **Время** | **Продавец** | **Продукт** | **Сумма сделки** | **Объем сделки** |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Карандаши | 690 | 30 |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Ручки | 830 | 40 |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Тетради | 500 | 25 |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Фломастеры | 700 | 35 |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Краски | 600 | 15 |
| Женская гимназия № 1 | 07.03.99 | Юрий Т. | Маркеры | 1 500 | 100 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Карандаши | 690 | 30 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Ручки | 830 | 40 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Тетради | 500 | 25 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Фломастеры | 700 | 35 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Краски | 2 000 | 50 |
| Женская гимназия № 1 | 07.03.99 | Дмитрий А. | Маркеры | 2 250 | 150 |
| Женская гимназия № 1 | 07.03.99 | Алексей Ш. | Карандаши | 230 | 10 |
| Женская гимназия № 1 | 07.03.99 | Алексей Ш. | Ручки | 1 000 | 0 |

Можно выделить следующие преимущества использования многомерных БД в OLAP-системах:

- поиск и выборка данных осуществляются значительно быстрее, чем при многомерном концептуальном взгляде на реляционную БД, т. к. многомерная база данных денормализована и содержит заранее агрегированные показатели, обеспечивая оптимизированный доступ к запрашиваемым ячейкам и не требуя дополнительных преобразований при переходе от множества связанных таблиц к многомерной модели;
- многомерные БД легко справляются с задачами включения в информационную модель разнообразных встроенных функций, тогда как объективно существующие ограничения языка SQL делают выполнение этих задач на основе реляционных БД достаточно сложным, а иногда и невозможным.

С другой стороны, имеются также существенные недостатки многомерных БД:

- за счет денормализации и предварительно выполненной агрегации объем данных в многомерной БД, как правило, соответствует (по оценке Кодда) в 2,5 ÷ 100 раз меньшему объему исходных детализированных данных;
- в подавляющем большинстве случаев информационный гиперкуб является сильно разреженным, а поскольку данные хранятся в упорядоченном виде, неопределенные значения удается удалить только за счет выбора оптимального порядка сортировки, позволяющего организовать данные в максимально большие непрерывные группы. Но даже в этом случае проблема решается только частично. Кроме того, оптимальный с точки зрения хранения разреженных данных порядок сортировки, скорее всего, не будет совпадать с порядком, который чаще других используется в запросах. Поэтому в реальных системах приходится искать компромисс между быстродействием и избыточностью дискового пространства, занятого базой данных;
- многомерные БД чувствительны к изменениям в многомерной модели. Так при добавлении нового измерения приходится изменять структуру всей БД, что влечет за собой большие затраты времени.

На основании анализа достоинств и недостатков многомерных БД можно выделить следующие условия, при которых их использование является эффективным:

- объем исходных данных для анализа не слишком велик (не более нескольких гигабайт), т. е. уровень агрегации данных достаточно высок;
- набор информационных измерений стабилен;
- время ответа системы на нерегламентированные запросы является наиболее критичным параметром;

- требуется широкое использование сложных встроенных функций для выполнения кроссмерных вычислений над ячейками гиперкуба, в том числе необходима возможность написания пользовательских функций.

## 3.4.2. ROLAP

ROLAP-серверы используют реляционные БД. По словам Кодда, "реляционные БД были, есть и будут наиболее подходящей технологией для хранения данных. Необходимость существует не в новой технологии БД, а скорее в средствах анализа, дополняющих функции существующих СУБД, и достаточно гибких, чтобы предусмотреть и автоматизировать разные виды интеллектуального анализа, присущие OLAP".

В настоящее время распространены две основные схемы реализации многомерного представления данных с помощью реляционных таблиц: схема "звезда" (рис. 3.5) и схема "снежинка" (рис. 3.6).

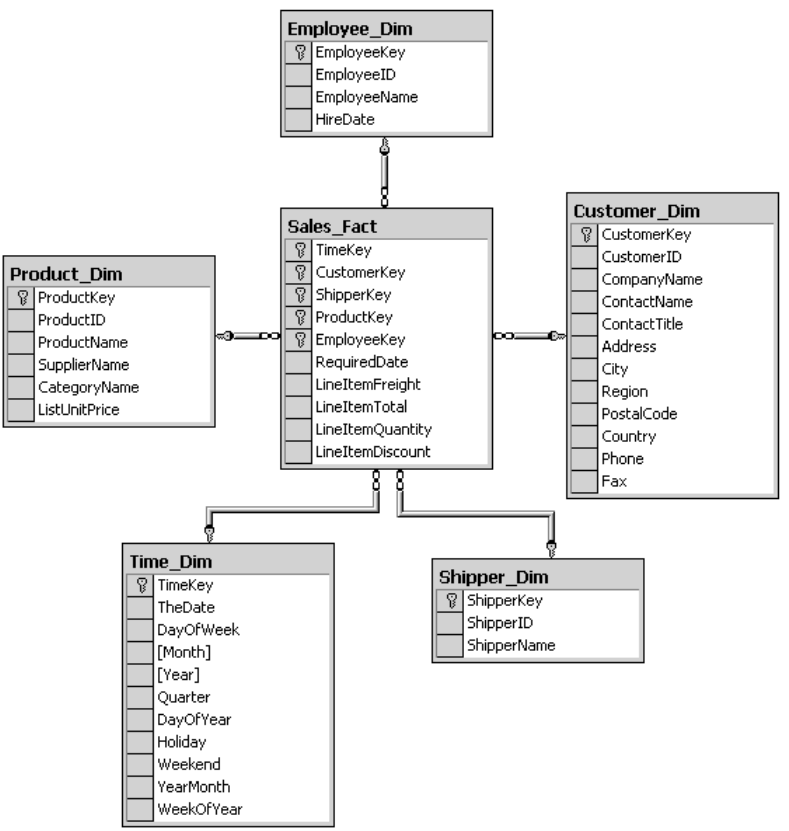



**Рис. 3.5.** Пример схемы "звезда"

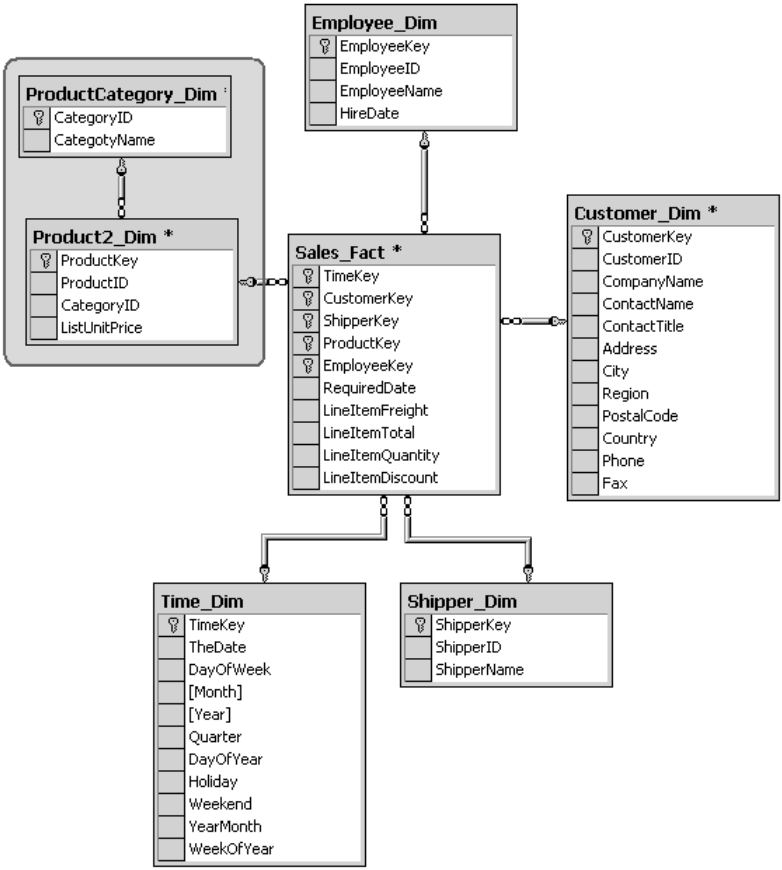



**Рис. 3.6.** Пример схемы "снежинка"

Основными составляющими схемы "звезда" (Star Schema) являются денормализованная таблица фактов (Fact Table) и множество таблиц измерений (Dimension Tables).

*Таблица фактов*, как правило, содержит сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться. Обычно говорят о четырех наиболее часто встречающихся типах фактов. К ним относятся:

- факты, связанные с транзакциями (Transaction facts). Они основаны на отдельных событиях (типичными примерами которых является телефонный звонок или снятие денег со счета с помощью банкомата);
- факты, связанные с "моментальными снимками" (Snapshot facts). Они основаны на состоянии объекта (например, банковского счета) в определенные моменты времени (например, на конец дня или месяца). Типичными

примерами таких фактов является объем продаж за день или дневная выручка;

- факты, связанные с элементами документа (Line-item facts). Они основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, о количестве, цене, проценте скидки);
- факты, связанные с событиями или состоянием объекта (Event or state facts). Они представляют возникновение события без подробностей о нем (например, просто факт продажи или факт отсутствия таковой без иных подробностей).

Таблица фактов, как правило, содержит уникальный составной ключ, объединяющий первичные ключи таблиц измерений. При этом как ключевые, так и некоторые не ключевые поля должны соответствовать измерениям гиперкуба. Помимо этого таблица фактов содержит одно или несколько числовых полей, на основании которых в дальнейшем будут получены агрегатные данные.

Для многомерного анализа пригодны таблицы фактов, содержащие как можно более подробные данные, т. е. соответствующие членам нижних уровней иерархии соответствующих измерений. В таблице фактов нет никаких сведений о том, как группировать записи при вычислении агрегатных данных. Например, в ней есть идентификаторы продуктов или клиентов, но отсутствует информация о том, к какой категории относится данный продукт или в каком городе находится данный клиент. Эти сведения, используемые в дальнейшем для построения иерархий в измерениях куба, содержатся в таблицах измерений.

*Таблицы измерений* содержат неизменяемые или редко изменяемые данные. В подавляющем большинстве случаев эти данные представляют собой по одной записи для каждого члена нижнего уровня иерархии в измерении. Таблицы измерений также содержат как минимум одно описательное поле (обычно с именем члена измерения) и, как правило, целочисленное ключевое поле (обычно это суррогатный ключ) для однозначной идентификации члена измерения. Если измерение, соответствующее таблице, содержит иерархию, то такая таблица также может содержать поля, указывающие на "родителя" данного члена в этой иерархии. Каждая таблица измерений должна находиться в отношении "один-ко-многим" с таблицей фактов.

Скорость роста таблиц измерений должна быть незначительной по сравнению со скоростью роста таблицы фактов. Например, новая запись в таблицу измерений, характеризующую товары, добавляется только при появлении нового товара, не продававшегося ранее.

В сложных задачах с иерархическими измерениями имеет смысл обратиться к расширенной схеме "снежинка" (Snowflake Schema). В этих случаях отдельные таблицы фактов создаются для возможных сочетаний уровней

обобщения различных измерений (см. рис. 3.6). Это позволяет добиться лучшей производительности, но часто приводит к избыточности данных и к значительным усложнениям в структуре базы данных, в которой оказывается огромное количество таблиц фактов.

Увеличение числа таблиц фактов в базе данных определяется не только множественностью уровней различных измерений, но и тем обстоятельством, что в общем случае факты имеют разные множества измерений. При абстрагировании от отдельных измерений пользователь должен получать проекцию максимально полного гиперкуба, причем далеко не всегда значения показателей в ней должны являться результатом элементарного суммирования. Таким образом, при большом числе независимых измерений необходимо поддерживать множество таблиц фактов, соответствующих каждому возможному сочетанию выбранных в запросе измерений, что также приводит к неэкономному использованию внешней памяти, увеличению времени загрузки данных в БД схемы "звезды" из внешних источников и сложностям администрирования.

Использование реляционных БД в OLAP-системах имеет следующие достоинства:

- в большинстве случаев корпоративные хранилища данных реализуются средствами реляционных СУБД, и инструменты ROLAP позволяют производить анализ непосредственно над ними. При этом размер хранилища не является таким критичным параметром, как в случае MOLAP;
- в случае переменной размерности задачи, когда изменения в структуру измерений приходится вносить достаточно часто, ROLAP-системы с динамическим представлением размерности являются оптимальным решением, т. к. в них такие модификации не требуют физической реорганизации БД;
- реляционные СУБД обеспечивают значительно более высокий уровень защиты данных и хорошие возможности разграничения прав доступа.

Главный недостаток ROLAP по сравнению с многомерными СУБД — меньшая производительность. Для обеспечения производительности, сравнимой с MOLAP, реляционные системы требуют тщательной проработки схемы базы данных и настройки индексов, т. е. больших усилий со стороны администраторов БД. Только при использовании схем типа "звезда" производительность хорошо настроенных реляционных систем может быть приближена к производительности систем на основе многомерных баз данных.

## 3.4.3. HOLAP

HOLAP-серверы используют гибридную архитектуру, которая объединяет технологии ROLAP и MOLAP. В отличие от MOLAP, которая работает лучше, когда данные более-менее плотные, серверы ROLAP показывают лучшие параметры в тех случаях, когда данные сильно разрежены. Серверы HOLAP

применяют подход ROLAP для разреженных областей многомерного пространства и подход MOLAP для плотных областей. Серверы HOLAP разделяют запрос на несколько подзапросов, направляют их к соответствующим фрагментам данных, комбинируют результаты, а затем предоставляют результат пользователю.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Для анализа информации наиболее удобным способом ее представления является многомерная модель или гиперкуб, ребрами которого являются измерения. Это позволяет анализировать данные сразу по нескольким измерениям, т. е. выполнять многомерный анализ.
- Измерение — это последовательность значений одного из анализируемых параметров. Измерения могут представлять собой иерархическую структуру. На пересечениях измерений находятся данные, количественно характеризующие анализируемые факты — меры.
- Над многомерной моделью — гиперкубом — могут выполняться операции: среза, вращения, консолидации и детализации. Многомерную модель и эти операции реализуют OLAP-системы.
- OLAP (On-Line Analytical Processing) — технология оперативной аналитической обработки данных. Это класс приложений, предназначенных для сбора, хранения и анализа многомерных данных в целях поддержки принятия решений.
- Для определения OLAP-систем Кодд разработал 12 правил, позднее дополнил к ним еще шесть и разбил 18 правил на четыре группы: основные особенности, специальные особенности, особенности представления отчетов и управление измерениями.
- В 1995 г. Пендсон и Крит на основании правил Кодда разработали тест FASMI, определив OLAP как "Быстрый Анализ Разделяемой Многомерной Информации".
- Архитектура OLAP-системы включает в себя OLAP-сервер и OLAP-клиент. OLAP-сервер может быть реализован на основе многомерных БД (MOLAP), реляционных БД (ROLAP) или сочетания обеих моделей (HOLAP).
- Достоинствами MOLAP являются высокая производительность и простота использования встроенных функций.
- Достоинствами ROLAP являются возможность работы с существующими реляционными БД, более экономичное использование ресурсов и большая гибкость при добавлении новых измерений.

ГЛАВА 4

# Интеллектуальный анализ данных

## 4.1. Добыча данных — Data Mining

OLAP-системы, описанные в *гл. 3*, предоставляют аналитику средства проверки гипотез при анализе данных. При этом основной задачей аналитика является генерация гипотез. Он решает ее, основываясь на своих знаниях и опыте. Однако знания есть не только у человека, но и в накопленных данных, которые подвергаются анализу. Такие знания часто называют "скрытыми", т. к. они содержатся в гигабайтах и терабайтах информации, которые человек не в состоянии исследовать самостоятельно. В связи с этим существует высокая вероятность пропустить гипотезы, которые могут принести значительную выгоду.

Очевидно, что для обнаружения скрытых знаний необходимо применять специальные методы автоматического анализа, при помощи которых приходится практически добывать знания из "завалов" информации. За этим направлением прочно закрепился термин *добыча данных* или *Data Mining*. Классическое определение этого термина дал в 1996 г. один из основателей этого направления — Григорий Пятецкий-Шапиро.

> ***Внимание!***
> Data Mining — 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.

Рассмотрим свойства обнаруживаемых знаний, данные в определении, более подробно.

- **Знания должны быть новые, ранее неизвестные.** Затраченные усилия на открытие знаний, которые уже известны пользователю, не окупаются. Поэтому ценность представляют именно новые, ранее неизвестные знания.

- **Знания должны быть нетривиальны.** Результаты анализа должны отражать неочевидные, неожиданные закономерности в данных, составляющие так называемые скрытые знания. Результаты, которые могли бы быть получены более простыми способами (например, визуальным просмотром), не оправдывают привлечение мощных методов Data Mining.
- **Знания должны быть практически полезны.** Найденные знания должны быть применимы, в том числе и на новых данных, с достаточно высокой степенью достоверности. Полезность заключается в том, чтобы эти знания могли принести определенную выгоду при их применении.
- **Знания должны быть доступны для понимания человеку.** Найденные закономерности должны быть логически объяснимы, в противном случае существует вероятность, что они являются случайными. Кроме того, обнаруженные знания должны быть представлены в понятном для человека виде.

В Data Mining для представления полученных знаний служат *модели*. Виды моделей зависят от методов их создания. Наиболее распространенными являются: правила, деревья решений, кластеры и математические функции.

# 4.2. Задачи Data Mining

## 4.2.1. Классификация задач Data Mining

Методы Data Mining помогают решить многие задачи, с которыми сталкивается аналитик. Из них основными являются: классификация, регрессия, поиск ассоциативных правил и кластеризация. Далее приведено краткое описание основных задач анализа данных.

- Задача классификации сводится к определению класса объекта по его характеристикам. Необходимо заметить, что в этой задаче множество классов, к которым может быть отнесен объект, известно заранее.
- Задача регрессии подобно задаче классификации позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значением параметра является не конечное множество классов, а множество действительных чисел.
- При поиске ассоциативных правил целью является нахождение частых зависимостей (или ассоциаций) между объектами или событиями. Найденные зависимости представляются в виде правил и могут быть использованы как для лучшего понимания природы анализируемых данных, так и для предсказания появления событий.
- Задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик во всем множестве анализируемых данных. Реше-

ние этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет сократить их число, а следовательно, и облегчить анализ.

Перечисленные задачи по назначению делятся на описательные и предсказательные.

**Описательные (descriptive) задачи** уделяют внимание улучшению понимания анализируемых данных. Ключевой момент в таких моделях — легкость и прозрачность результатов для восприятия человеком. Возможно, обнаруженные закономерности будут специфической чертой именно конкретных исследуемых данных и больше нигде не встретятся, но это все равно может быть полезно и потому должно быть известно. К такому виду задач относятся кластеризация и поиск ассоциативных правил.

Решение **предсказательных (predictive) задач** разбивается на два этапа. На первом этапе на основании набора данных с известными результатами строится модель. На втором этапе она используется для предсказания результатов на основании новых наборов данных. При этом, естественно, требуется, чтобы построенные модели работали максимально точно. К данному виду задач относят задачи классификации и регрессии. Сюда можно отнести и задачу поиска ассоциативных правил, если результаты ее решения могут быть использованы для предсказания появления некоторых событий.

По способам решения задачи разделяют на *supervised learning* (обучение с учителем) и *unsupervised learning* (обучение без учителя). Такое название произошло от термина Machine Learning (машинное обучение), часто используемого в англоязычной литературе и обозначающего все технологии Data Mining.

В случае supervised learning задача анализа данных решается в несколько этапов. Сначала с помощью какого-либо алгоритма Data Mining строится модель анализируемых данных — классификатор. Затем классификатор подвергается обучению. Другими словами, проверяется качество его работы, и, если оно неудовлетворительное, происходит дополнительное обучение классификатора. Так продолжается до тех пор, пока не будет достигнут требуемый уровень качества или не станет ясно, что выбранный алгоритм не работает корректно с данными, либо же сами данные не имеют структуры, которую можно выявить. К этому типу задач относят задачи классификации и регрессии.

Unsupervised learning объединяет задачи, выявляющие описательные модели, например закономерности в покупках, совершаемых клиентами большого магазина. Очевидно, что если эти закономерности есть, то модель должна их представить и неуместно говорить об ее обучении. Достоинством таких задач является возможность их решения без каких-либо предварительных знаний

об анализируемых данных. К этим задачам относятся кластеризация и поиск ассоциативных правил.

### 4.2.2. Задача классификации и регрессии

При анализе часто требуется определить, к какому из известных классов относятся исследуемые объекты, т. е. классифицировать их. Например, когда человек обращается в банк за предоставлением ему кредита, банковский служащий должен принять решение: кредитоспособен ли потенциальный клиент или нет. Очевидно, что такое решение принимается на основании данных об исследуемом объекте (в данном случае — о человеке): его место работы, размер заработной платы, возраст, состав семьи и т. п. В результате анализа этой информации банковский служащий должен отнести человека к одному из двух известных классов: "кредитоспособен" и "некредитоспособен".

Другим примером задачи классификации является фильтрация электронной почты. В этом случае программа фильтрации должна классифицировать входящее сообщение как спам (spam — нежелательная электронная почта) или как письмо. Данное решение принимается на основании частоты появления в сообщении определенных слов (например, имени получателя, безличного обращения, слов и словосочетаний: "приобрести", "заработать", "выгодное предложение" и т. п.).

В общем случае количество классов в задачах классификации может быть более двух. Например, в задаче распознавания образа цифр таких классов может быть 10 (по количеству цифр в десятичной системе счисления). В такой задаче объектом классификации является матрица пикселов, представляющая образ распознаваемой цифры. При этом цвет каждого пиксела является характеристикой анализируемого объекта.

В Data Mining задачу классификации рассматривают как задачу определения значения одного из параметров анализируемого объекта на основании значений других параметров. Определяемый параметр часто называют зависимой переменной, а параметры, участвующие в его определении, — независимыми переменными. В рассмотренных примерах независимыми переменными являлись:

- зарплата, возраст, количество детей и т. д.;
- частота появления определенных слов;
- значения цвета пикселов матрицы.

Зависимыми переменными в этих же примерах являлись соответственно:

- кредитоспособность клиента (возможные значения этой переменной — "да" и "нет");

- тип сообщения (возможные значения этой переменной — "spam" и "mail");
- цифра образа (возможные значения этой переменной — 0, 1, ... , 9).

Необходимо обратить внимание, что во всех рассмотренных примерах независимая переменная принимала значение из конечного множества значений: {"да", "нет"}, {"spam", "mail"}, {0, 1, ... , 9}. Если значениями независимых и зависимой переменных являются действительные числа, то задача называется *задачей регрессии*. Примером задачи регрессии может служить задача определения суммы кредита, которая может быть выдана банком клиенту.

Задача классификации и регрессии решается в два этапа. На первом выделяется обучающая выборка. В нее входят объекты, для которых известны значения как независимых, так и зависимых переменных. В описанных ранее примерах такими обучающими выборками могут быть:

- информация о клиентах, которым ранее выдавались кредиты на разные суммы, и информация об их погашении;
- сообщения, классифицированные вручную как спам или как письмо;
- распознанные ранее матрицы образов цифр.

На основании обучающей выборки строится модель определения значения зависимой переменной. Ее часто называют функцией классификации или регрессии. Для получения максимально точной функции к обучающей выборке предъявляются следующие основные требования:

- количество объектов, входящих в выборку, должно быть достаточно большим. Чем больше объектов, тем точнее будет построенная на ее основе функция классификации или регрессии;
- в выборку должны входить объекты, представляющие все возможные классы в случае задачи классификации или всю область значений в случае задачи регрессии;
- для каждого класса в задаче классификации или для каждого интервала области значений в задаче регрессии выборка должна содержать достаточное количество объектов.

На втором этапе построенную модель применяют к анализируемым объектам (к объектам с неопределенным значением зависимой переменной).

Задача классификации и регрессии имеет геометрическую интерпретацию. Рассмотрим ее на примере с двумя независимыми переменными, что позволит представить ее в двумерном пространстве (рис. 4.1). Каждому объекту ставится в соответствие точка на плоскости. Символы "+" и "–" обозначают принадлежность объекта к одному из двух классов. Очевидно, что данные имеют четко выраженную структуру: все точки класса "+" сосредоточены в центральной области. Построение классификационной функции сводится

к построению поверхности, которая обводит центральную область. Она определяется как функция, имеющая значения "+" внутри обведенной области и "–" — вне ее.

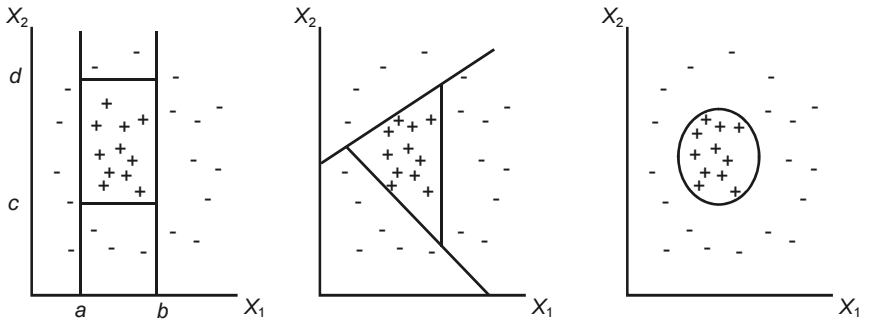



**Рис. 4.1.** Классификация в двумерном пространстве

Как видно из рисунка, есть несколько возможностей для построения обводящей области. Вид функции зависит от применяемого алгоритма.

Основные проблемы, с которыми сталкиваются при решении задач классификации и регрессии, — это неудовлетворительное качество исходных данных, в которых встречаются как ошибочные данные, так и пропущенные значения, различные типы атрибутов — числовые и категорические, разная значимость атрибутов, а также так называемые проблемы overfitting и underfitting. Суть первой из них заключается в том, что классификационная функция при построении "слишком хорошо" адаптируется к данным и встречающиеся в них ошибки и аномальные значения пытается интерпретировать как часть внутренней структуры данных. Очевидно, что в дальнейшем такая модель будет некорректно работать с другими данными, где характер ошибок будет несколько иной. Термином underfitting обозначают ситуацию, когда слишком велико количество ошибок при проверке классификатора на обучающем множестве. Это означает, что особых закономерностей в данных не было обнаружено, и либо их нет вообще, либо необходимо выбрать иной метод их обнаружения.

## 4.2.3. Задача поиска ассоциативных правил

Поиск ассоциативных правил является одним из самых популярных приложений Data Mining. Суть задачи заключается в определении часто встречающихся наборов объектов в большом множестве таких наборов. Данная задача является частным случаем задачи классификации. Первоначально она решалась при анализе тенденций в поведении покупателей в супермаркетах. Ана-

лизу подвергались данные о совершаемых ими покупках, которые покупатели складывают в тележку (корзину). Это послужило причиной второго часто встречающегося названия — анализ рыночных корзин (Basket Analysis). При анализе этих данных интерес прежде всего представляет информация о том, какие товары покупаются вместе, в какой последовательности, какие категории потребителей какие товары предпочитают, в какие периоды времени и т. п. Такая информация позволяет более эффективно планировать закупку товаров, проведение рекламной кампании и т. д.

Например, из набора покупок, совершаемых в магазине, можно выделить следующие наборы товаров, которые покупаются вместе: {чипсы, пиво}; {вода, орехи}. Следовательно, можно сделать вывод, что если покупаются чипсы или орехи, то, как правило, покупаются и пиво или вода соответственно. Обладая такими знаниями, можно разместить эти товары рядом, объединить их в один пакет со скидкой или предпринять другие действия, стимулирующие покупателя приобрести товар.

Задача поиска ассоциативных правил актуальна не только в сфере торговли. Например, в сфере обслуживания интерес представляет информация о том, какими услугами клиенты предпочитают пользоваться в совокупности. Для получения этой информации задача решается применительно к данным об услугах, которыми пользуется один клиент в течение определенного времени (месяца, года). Это помогает определить, например, как наиболее выгодно составить пакеты услуг, предлагаемых клиенту.

В медицине анализу могут подвергаться симптомы и болезни, наблюдаемые у пациентов. В этом случае знания о том, какие сочетания болезней и симптомов встречаются наиболее часто, помогают в будущем правильно ставить диагноз.

При анализе часто вызывает интерес последовательность происходящих событий. При обнаружении закономерностей в таких последовательностях можно с некоторой долей вероятности предсказывать появление событий в будущем, что позволяет принимать более правильные решения. Такая задача является разновидностью задачи поиска ассоциативных правил и называется *сиквенциальным анализом*.

Основным отличием задачи сиквенциального анализа от поиска ассоциативных правил является установление 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.

Сиквенциальный анализ широко используется, например, в телекоммуникационных компаниях, для анализа данных об авариях на различных узлах

сети. Информация о последовательности совершения аварий может помочь в обнаружении неполадок и предупреждении новых аварий. Например, если известна последовательность сбоев: $\{e_5, e_2, e_7, e_{13}, e_6, e_1, \ldots\}$, где $e_i$ — код сбоя, то на основании факта появления сбоя $e_2$ можно сделать вывод о скором появлении сбоя $e_7$. Зная это, можно предпринять профилактические меры, устраняющие причины возникновения сбоя. Если дополнительно обладать и знаниями о времени между сбоями, то можно предсказать не только факт его появления, но и время, что часто не менее важно.

## 4.2.4. Задача кластеризации

Задача кластеризации состоит в разделении исследуемого множества объектов на группы "похожих" объектов, называемых *кластерами* (cluster). Слово cluster переводится с английского как сгусток, пучок, группа. Родственные понятия, используемые в литературе, — класс, таксон, сгущение. Часто решение задачи разбиения множества элементов на кластеры называют *кластерным анализом*.

Кластеризация может применяться практически в любой области, где необходимо исследование экспериментальных или статистических данных. Рассмотрим пример из области маркетинга, в котором данная задача называется сегментацией.

Концептуально сегментирование основано на предпосылке, что все потребители разные. У них разные потребности, разные требования к товару, они ведут себя по-разному: в процессе выбора товара, в процессе приобретения товара, в процессе использования товара, в процессе формирования реакции на товар. В связи с этим необходимо по-разному подходить к работе с потребителями: предлагать им различные по своим характеристикам товары, по-разному продвигать и продавать товары. Для того чтобы определить, чем отличаются потребители друг от друга и как эти отличия отражаются на требованиях к товару, и производится сегментирование потребителей.

В маркетинге критериями (характеристиками) сегментации являются: географическое местоположение, социально-демографические характеристики, мотивы совершения покупки и т. п.

На основании результатов сегментации маркетолог может определить, например, такие характеристики сегментов рынка, как реальная и потенциальная емкость сегмента, группы потребителей, чьи потребности не удовлетворяются в полной мере ни одним производителем, работающим на данном сегменте рынка, и т. п. На основании этих параметров маркетолог может сделать вывод о привлекательности работы фирмы в каждом из выделенных сегментов рынка.

Для научных исследований изучение результатов кластеризации, а именно выяснение причин, по которым объекты объединяются в группы, способно открыть новые перспективные направления. Традиционным примером, который обычно приводят для этого случая, является периодическая таблица элементов. В 1869 г. Дмитрий Менделеев разделил 60 известных в то время элементов на кластеры или периоды. Элементы, попавшие в одну группу, обладали схожими характеристиками. Изучение причин, по которым элементы разбивались на явно выраженные кластеры, в значительной степени определило приоритеты научных изысканий на годы вперед. Но лишь спустя 50 лет квантовая физика дала убедительные объяснения периодической системы.

Кластеризация отличается от классификации тем, что для проведения анализа не требуется иметь выделенную зависимую переменную, поэтому она относится к классу unsupervised learning. Эта задача решается на начальных этапах исследования, когда о данных мало что известно. Ее решение помогает лучше понять данные, и с этой точки зрения задача кластеризации является описательной.

Для задачи кластеризации характерно отсутствие каких-либо различий как между переменными, так и между объектами. Напротив, ищутся группы наиболее близких, похожих объектов. Методы автоматического разбиения на кластеры редко используются сами по себе, а только для получения групп схожих объектов. После определения кластеров используются другие методы Data Mining, чтобы попытаться установить, что означает такое разбиение, чем оно вызвано.

Кластерный анализ позволяет рассматривать достаточно большой объем информации и резко сокращать, сжимать большие массивы информации, делать их компактными и наглядными.

Отметим ряд особенностей, присущих задаче кластеризации.

Во-первых, решение сильно зависит от природы объектов данных (и их атрибутов). Так, с одной стороны, это могут быть однозначно определенные, количественно очерченные объекты, а с другой — объекты, имеющие вероятностное или нечеткое описание.

Во-вторых, решение в значительной степени зависит и от представления кластеров и предполагаемых отношений объектов данных и кластеров. Так, необходимо учитывать такие свойства, как возможность/невозможность принадлежности объектов к нескольким кластерам. Необходимо определение самого понятия принадлежности кластеру: однозначная (принадлежит/не принадлежит), вероятностная (вероятность принадлежности), нечеткая (степень принадлежности).

# 4.3. Практическое применение Data Mining

## 4.3.1. Интернет-технологии

В системах электронного бизнеса, где особую важность имеют вопросы привлечения и удержания клиентов, технологии Data Mining часто применяются для построения рекомендательных систем интернет-магазинов и для решения проблемы персонализации посетителей Web-сайтов. Рекомендации товаров и услуг, построенные на основе закономерностей в покупках клиентов, обладают огромной убеждающей силой. Статистика показывает, что почти каждый посетитель магазина Amazon не упускает возможности посмотреть на то, что же купили "Customers who bought this book also bought: ... " ("Те, кто купил эту книгу, также купили ..."). Персонализация клиентов или, другими словами, автоматическое распознание принадлежности клиента к определенной целевой аудитории позволяет компании проводить более гибкую маркетинговую политику. Поскольку в электронной коммерции деньги и платежные системы тоже электронные, то важной задачей становится обеспечение безопасности при операциях с пластиковыми карточками. Data Mining позволяет обнаруживать случаи мошенничества (fraud detection). В области электронной коммерции также остаются справедливыми все методологии Data Mining, разработанные для обычного маркетинга. С другой стороны, эта область тесно связана с понятием Web Mining.

Специфика Web Mining заключается в применении традиционных технологий Data Mining для анализа крайне неоднородной, распределенной и значительной по объему информации, содержащейся на Web-узлах. Здесь можно выделить два направления: Web Content Mining и Web Usage Mining. В первом случае речь идет об автоматическом поиске и извлечении качественной информации из перегруженных "информационным шумом" источников Интернета, а также о всевозможных средствах автоматической классификации и аннотировании документов. Данное направление также называют Text Mining (более подробно см. *гл. 9*). Web Usage Mining направлен на обнаружение закономерностей в поведении пользователей конкретного Web-узла (группы узлов), в частности на то, какие страницы в какой временной последовательности и какими группами пользователей запрашиваются. Web Mining более подробно описан в *гл. 14*.

## 4.3.2. Торговля

Для успешного продвижения товаров всегда важно знать, что и как продается, а также кто является потребителем. Исчерпывающий ответ на первый вопрос дают такие средства Data Mining, как анализ рыночных корзин и сиквенциальный анализ. Зная связи между покупками и временные закономер-

ности, можно оптимальным образом регулировать предложение. С другой стороны, маркетинг имеет возможность непосредственно управлять спросом, но для этого необходимо знать как можно больше о потребителях — целевой аудитории маркетинга. Data Mining позволяет решать задачи выделения групп потребителей со схожими стереотипами поведения, т. е. сегментировать рынок. Для этого можно применять такие технологии Data Mining, как кластеризация и классификация.

Сиквенциальный анализ помогает торговым предприятиям принимать решения о создании товарных запасов. Он дает ответы на вопросы типа "Если сегодня покупатель приобрел видеокамеру, то через какое время он вероятнее всего купит новые батарейки и пленку?".

## 4.3.3. Телекоммуникации

Телекоммуникационный бизнес является одной из наиболее динамически развивающихся областей современной экономики. Возможно, поэтому традиционные проблемы, с которыми сталкивается в своей деятельности любая компания, здесь ощущаются особо остро. Приведем некоторые цифры. Телекоммуникационные компании работают в условиях жесткой конкуренции, что проявляется в ежегодном оттоке около 25 % клиентов. При этом известно, что удержать клиента в 4—5 раз дешевле, чем привлечь нового, а вот вернуть ушедшего клиента будет стоить уже в 50—100 раз больше, чем его удержать. Далее, как и в целом в экономике, справедливо правило Парето — только 20 % клиентов приносят компании основной доход. Помимо этого существует ряд клиентов, наносящих компании прямой вред. 10 % всего дохода телекоммуникационной индустрии в год теряется из-за случаев мошенничества, что составляет $4 млрд. Таким образом, использование технологий Data Mining, направленных как на анализ доходности и риска клиентов (churn prevention), так и на защиту от мошенничества, сэкономит компании огромные средства.

Еще один из распространенных способов использования методов Data Mining — это анализ записей о подробных характеристиках вызовов. Назначение такого анализа — выявление категорий клиентов с похожими стереотипами пользования услугами и разработка привлекательных наборов цен и услуг.

## 4.3.4. Промышленное производство

Промышленное производство создает идеальные условия для применения технологий Data Mining. Причина — в самой природе технологического процесса, который должен быть воспроизводимым и контролируемым. Все отклонения в течение процесса, влияющие на качество выходного результата, также находятся в заранее известных пределах. Таким образом, создается статистическая стабильность, первостепенную важность которой отмечают в

работах по классификации. Естественно, что в таких условиях использование Data Mining способно дать лучшие результаты, чем, к примеру, при прогнозировании ухода клиентов телекоммуникационных компаний. В последнем случае причинами ухода могут стать не предрасположенности к смене мест, присущие целым группам абонентов, а внешние, совершенно случайные, и поэтому не образующие никаких закономерностей обстоятельства (например, удачно проведенная конкурентами рекламная кампания, экономический кризис и т. д.). Опыт работы компаний, предлагающих решения Data Mining для промышленного производства, также свидетельствует об успешности такой интеграции. Примером применения Data Mining в промышленности может быть прогнозирование качества изделия в зависимости от замеряемых параметров технологического процесса.

## 4.3.5. Медицина

В медицинских и биологических исследованиях, равно как и в практической медицине, спектр решаемых задач настолько широк, что возможно использование любых методологий Data Mining. Примером может служить построение диагностической системы или исследование эффективности хирургического вмешательства.

Известно много экспертных систем для постановки медицинских диагнозов. Они построены главным образом на основе правил, описывающих сочетания различных симптомов отдельных заболеваний. С помощью таких правил узнают не только, чем болен пациент, но и как нужно его лечить. Правила помогают выбирать средства медикаментозного воздействия, определять показания/противопоказания, ориентироваться в лечебных процедурах, создавать условия наиболее эффективного лечения, предсказывать исходы назначенного курса лечения и т. п. Технологии Data Mining позволяют обнаруживать в медицинских данных шаблоны, составляющие основу указанных правил.

Одним из наиболее передовых направлений медицины является биоинформатика — область науки, разрабатывающая и применяющая вычислительные алгоритмы для анализа и систематизации генетической информации с целью выяснения структуры и функции макромолекул, последующего использования этих знаний для объяснения различных биологических явлений и создания новых лекарственных препаратов (Drug Design). Объектом исследования биоинформатики являются огромные объемы информации о последовательностях ДНК и первичной структуре белков, появившиеся в результате изучения структуры геномов микроорганизмов, млекопитающих и человека. Абстрагируясь от конкретного содержания этой информации, ее можно рассматривать как набор генетических текстов, состоящих из протяженных символьных последовательностей. Выявление структурных закономерностей в

таких последовательностях входит в число задач, эффективно решаемых средствами Data Mining, например, с помощью сиквенциального и ассоциативного анализа. Основная область практического применения биоинформатики — это разработка лекарств нового поколения, которые полностью преобразят современную медицину. Сегодня разработка одного препарата в США занимает в среднем 10—12 лет, а стоимость составляет $300—500 млн. Биоинформатика сокращает эти цифры вдвое. Опираясь на аппарат Data Mining биоинформатика может еще больше ускорить и удешевить дофармакологическую фазу исследования новых препаратов.

### 4.3.6. Банковское дело

Классическим примером применения Data Mining на практике является решение проблемы о возможной некредитоспособности клиентов банка. Этот вопрос, тревожащий любого сотрудника кредитного отдела банка, можно разрешить и интуитивно. Если образ клиента в сознании банковского служащего соответствует его представлению о кредитоспособном клиенте, то кредит выдавать можно, иначе — отказать. По схожей схеме, но более продуктивно и полностью автоматически работают установленные в тысячах американских банков системы поддержки принятия решений (Decision System Support) со встроенной функциональностью Data Mining. Лишенные субъективной предвзятости, они опираются в своей работе только на историческую базу данных банка, где записывается детальная информация о каждом клиенте и, в конечном итоге, факт его кредитоспособности. Классификационные алгоритмы Data Mining обрабатывают эти данные, а полученные результаты используются далее для принятия решений.

Анализ кредитного риска заключается, прежде всего, в оценке кредитоспособности заемщика. Эта задача решается на основе анализа накопленной информации, т. е. кредитной истории "старых" клиентов. С помощью инструментов Data Mining (деревья решений, кластерный анализ, нейронные сети и др.) банк может получить профили добросовестных и неблагонадежных заемщиков. Кроме того, можно классифицировать заемщика по группам риска, а значит, не только решить вопрос о возможности кредитования, но и установить лимит кредита, проценты по нему и срок возврата.

Мошенничество с кредитными карточками представляет собой серьезную проблему, т. к. убытки от него измеряются миллионами долларов ежегодно, а рост количества мошеннических операций составляет, по оценкам экспертов, от 15 до 25 % ежегодно.

В борьбе с мошенничеством технология Data Mining использует стереотипы подозрительных операций, созданные в результате анализа огромного количества транзакций — как законных, так и неправомерных. Исследуется не

только отдельно взятая операция, но и совокупность последовательных во времени транзакций. Кроме того, алгоритмы и модели (например, нейронные сети), имеющиеся в составе продуктов Data Mining, способны тестироваться и самообучаться. При попытке совершения подозрительной операции средства интеллектуального анализа данных оперативно выдают предупреждение об этом, что позволяет банку предотвратить незаконные действия, а не устранять их последствия. Использование технологии Data Mining позволяет сократить число нарушений на 20—30 %.

### 4.3.7. Страховой бизнес

В страховании, также как в банковском деле и маркетинге, возникает задача обработки больших объемов информации для определения типичных групп (профилей) клиентов. Эта информация используется для того, чтобы предлагать определенные услуги страхования с наименьшим для компании риском и, возможно, с пользой для клиента. С помощью технологий Data Mining также решается такая часто встречающаяся в страховании задача, как определение случаев мошенничества.

### 4.3.8. Другие области применения

Data Mining может применяться практически везде, где возникает задача автоматического анализа данных. В качестве примера приведем такие популярные направления, как анализ и последующая фильтрация спама, а также разработка так называемых виртуальных собеседников. Последние сейчас являются не более чем экзотическим дополнением к интерфейсу некоторых сайтов, но предполагается, что в будущем они могут заменить собой call-центры компаний.

## 4.4. Модели Data Mining

Цель технологии Data Mining — нахождение в данных таких моделей, которые не могут быть найдены обычными методами. Существуют два вида моделей: *предсказательные* и *описательные*.

### 4.4.1. Предсказательные модели

Предсказательные (predictive) модели строятся на основании набора данных с известными результатами. Они используются для предсказания результатов на основании других наборов данных. При этом, естественно, требуется, чтобы модель работала максимально точно, была статистически значима и оправданна и т. д.

К таким моделям относятся следующие:

- модели классификации — описывают правила или набор правил, в соответствии с которыми можно отнести описание любого нового объекта к одному из классов. Такие правила строятся на основании информации о существующих объектах путем разбиения их на классы;
- модели последовательностей — описывают функции, позволяющие прогнозировать изменение непрерывных числовых параметров. Они строятся на основании данных об изменении некоторого параметра за прошедший период времени.

## 4.4.2. Описательные модели

Описательные (descriptive) модели уделяют внимание сути зависимостей в наборе данных, взаимному влиянию различных факторов, т. е. построению эмпирических моделей различных систем. Ключевой момент в таких моделях — легкость и прозрачность для восприятия человеком. Возможно, обнаруженные закономерности будут специфической чертой именно конкретных исследуемых данных и больше нигде не встретятся, но это все равно может быть полезно, и потому должно быть известно.

К таким моделям относятся следующие виды:

- регрессионные модели — описывают функциональные зависимости между зависимыми и независимыми показателями и переменными в понятной человеку форме. Необходимо заметить, что такие модели описывают функциональную зависимость не только между непрерывными числовыми параметрами, но и между категориальными параметрами;
- модели кластеров — описывают группы (кластеры), на которые можно разделить объекты, данные о которых подвергаются анализу. Группируются объекты (наблюдения, события) на основе данных (свойств), описывающих сущность объектов. Объекты внутри кластера должны быть "похожими" друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем сильнее "похожи" объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация;
- модели исключений — описывают исключительные ситуации в записях (например, отдельных пациентов), которые резко отличаются чем-либо от основного множества записей (группы больных). Знание исключений может быть использовано двояким образом. Возможно, эти записи представляют собой случайный сбой, например ошибки операторов, вводивших данные в компьютер. Характерный случай: если оператор, ошибаясь, ставит десятичную точку не в том месте, то такая ошибка сразу дает резкий

"всплеск" на порядок. Подобную "шумовую" случайную составляющую имеет смысл отбросить, исключить из дальнейших исследований, поскольку большинство методов, которые будут рассмотрены в данной главе, очень чувствительно к наличию "выбросов" — резко отличающихся точек, редких, нетипичных случаев. С другой стороны, отдельные, исключительные записи могут представлять самостоятельный интерес для исследования, т. к. они могут указывать на некоторые редкие, но важные аномальные заболевания. Даже сама идентификация этих записей, не говоря об их последующем анализе и детальном рассмотрении, может оказаться очень полезной для понимания сущности изучаемых объектов или явлений;

- итоговые модели — выявление ограничений на данные анализируемого массива. Например, при изучении выборки данных по пациентам не старше 30 лет, перенесшим инфаркт миокарда, обнаруживается, что все пациенты, описанные в этой выборке, либо курят более 5 пачек сигарет в день, либо имеют вес не ниже 95 кг. Подобные ограничения важны для понимания данных массива, по сути дела это новое знание, извлеченное в результате анализа. Таким образом, построение итоговых моделей заключается в нахождении каких-либо фактов, которые верны для всех или почти всех записей в изучаемой выборке данных, но которые достаточно редко встречались бы во всем мыслимом многообразии записей такого же формата и, например, характеризовались бы теми же распределениями значений полей. Если взять для сравнения информацию по всем пациентам, то процент либо сильно курящих, либо чрезмерно тучных людей будет весьма невелик. Можно сказать, что решается как бы неявная задача классификации, хотя фактически задан только один класс, представленный имеющимися данными;

- ассоциативные модели — выявление закономерностей между связанными событиями. Примером такой закономерности служит правило, указывающее, что из события *X* следует событие *Y*. Такие правила называются ассоциативными.

Для построения рассмотренных моделей используются различные методы и алгоритмы Data Mining. Ввиду того, что технология Data Mining развивалась и развивается на стыке таких дисциплин, как статистика, теория информации, машинное обучение и теория баз данных, вполне закономерно, что большинство алгоритмов и методов Data Mining были разработаны на основе различных технологий и концепций. Далее рассмотрим технологии, наиболее часто реализуемые методами Data Mining.

# 4.5. Методы Data Mining

## 4.5.1. Базовые методы

К базовым методам Data Mining принято относить, прежде всего, алгоритмы, основанные на переборе. Простой перебор всех исследуемых объектов требует $O(2^N)$ операций, где $N$ — количество объектов. Следовательно, с увеличением количества данных объем вычислений растет экспоненциально, что при большом объеме делает решение любой задачи таким методом практически невозможным.

Для сокращения вычислительной сложности в таких алгоритмах, как правило, используют разного вида эвристики, приводящие к сокращению перебора. Оптимизация подобных алгоритмов сводится к приведению зависимости количества операций от количества исследуемых данных к функции линейного вида. В то же время, зависимость от количества атрибутов, как правило, остается экспоненциальной. При условии, что их немного (в подавляющем большинстве случаев их значительно меньше, чем данных), такая зависимость является приемлемой.

Основным достоинством данных алгоритмов является их простота, как с точки зрения понимания, так и реализации. К недостаткам можно отнести отсутствие формальной теории, на основании которой строятся такие алгоритмы, а следовательно, и сложности, связанные с их исследованием и развитием.

К базовым методам Data Mining можно отнести также и подходы, использующие элементы теории статистики. В связи с тем, что Data Mining является развитием статистики, таких методов достаточно много. Их основная идея сводится к корреляционному, регрессионному и другим видам статистического анализа. Главным недостатком является усреднение значений, что приводит к потере информативности данных. Это в свою очередь приводит к уменьшению количества добываемых знаний.

## 4.5.2. Нечеткая логика

Основным способом исследования задач анализа данных является их отображение на формализованный язык и последующий анализ полученной модели. Неопределенность по объему отсутствующей информации у системного аналитика можно разделить на три большие группы:

1. Неизвестность.
2. Неполнота (недостаточность, неадекватность).
3. Недостоверность.

Недостоверность бывает *физической* (источником ее является внешняя среда) и *лингвистической* (возникает в результате словесного обобщения и обуслов-

ливается необходимостью описания бесконечного числа ситуаций ограниченным числом слов за ограниченное время).

Выделяют два вида физической неопределенности:

1. Неточность (неточность измерений значений определенной величины, выполняемых физическими приборами).
2. Случайность (или наличие во внешней среде нескольких возможностей, каждая из которых случайным образом может стать действительностью; предполагается знание соответствующего закона распределения вероятностей).

Выделяют два вида лингвистической неопределенности:

1. Неопределенность значений слов (многозначность, расплывчатость, неясность, нечеткость). Она возникает в случае, если отображаемые одним и тем же словом объекты задачи управления различны.
2. Неоднозначность смысла фраз (выделяют синтаксическую и семантическую).

Для обработки физических неопределенностей успешно используются методы теории вероятностей и классическая теория множеств. Однако с развитием систем, использующих методы теории искусственного интеллекта, в которых требуется обрабатывать понятия и отношения естественного языка, возникла необходимость расширения множества формальных методов с целью учета лингвистической неопределенности задач.

Основной сферой применения нечеткой логики было и во многом остается управление. Не случайно основоположником теории нечетких множеств стал известный специалист в области управления Л. Заде. Дело в том, что в исходную идею о нечеткой логике очень хорошо укладывались представления об управлении и процессах принятия решений. А поскольку подобные задачи возникают почти во всех технологических процессах, потребности в развитии данной теории и возможности ее приложения достаточно широки.

С увеличением размеров и сложности системы существенно усложняется ее моделирование с помощью известных математических выражений. Это связано с увеличением числа переменных и параметров, повышением сложности измерения отдельных переменных. В результате, создание адекватной модели становится практически невозможным. Вместо этого Л. Заде предложил лингвистическую модель, которая использует не математические выражения, а слова, отражающие качество. Применение словесной модели не обеспечивает точность, аналогичную математическому моделированию, однако создание хорошей, качественной модели возможно. В этом случае предметом обсуждения становится нечеткость слов языка описания системы.

Человеку в процессе управления сложными объектами свойственно оперировать понятиями и отношениями с расплывчатыми границами. Источником расплывчатости является существование классов объектов, степень принадлежности к которым — величина, непрерывно изменяющаяся от полной принадлежности к нему до полной непринадлежности. Обычное математическое понятие множества, основанное на бинарной характеристической функции, не позволяет формализовать такое описание.

Введение Л. Заде двух основных исходных понятий — нечеткого множества и лингвистической переменной — существенно расширило возможности формализации описаний подобных сложных систем. Такие модели стали называться лингвистическими.

Рассмотрим основные достоинства нечеткой логики, наиболее ярко проявляющиеся на примере общей задачи нечеткого управления. Если говорить кратко, то нечеткая логика позволяет удачно представить мышление человека. Очевидно, что в повседневной деятельности человек никогда не пользуется формальным моделированием на основе математических выражений, не ищет одного универсального закона, описывающего все окружающее. Он использует нечеткий естественный язык. В процессе принятия решения человек легко овладевает ситуацией, разделяя ее на события, находит решение сложных проблем, применяя для отдельных событий соответствующие, по опыту, правила принятия решений, используя при этом большое количество иногда даже противоречивых качественных критериев. Таким образом, перед человеком возникает ряд локальных моделей, описывающих свойства фрагментов объектов в определенных условиях. Крайне важным является то, что все модели обладают некой общностью и очень просты для понимания на качественном уровне. Ярким примером каркаса подобной словесной модели является конструкция "если ... , то ... ".

Теперь определим три основные особенности нечеткой логики:

1. Правила принятия решений являются условными высказываниями типа "если ... , то ... " и реализуются с помощью механизма логического вывода.
2. Вместо одного четкого обобщенного правила нечеткая логика оперирует со множеством частных правил. При этом для каждой локальной области распределенного информационного пространства, для каждой регулируемой величины, для каждой цели управления задаются свои правила. Это позволяет отказываться от трудоемкого процесса свертки целей и получения обобщенного целевого критерия, что, в свою очередь, дает возможность оперировать даже с противоположными целями.
3. Правила в виде "если ... , то ... " позволяют решать задачи классификации в режиме диалога с оператором, что способствует повышению качества классификатора уже в процессе эксплуатации.

Таким образом, нетрудно заметить существенные общие черты нечеткой логики и мышления человека, поэтому методы управления на основе нечеткой логики можно считать во многом эвристическими. Эвристические приемы решения задач основаны не на строгих математических моделях и алгоритмах, а на соображениях "здравого смысла".

Развитием эвристических алгоритмов обработки нечетких данных можно считать самоорганизующиеся системы. В любом случае исходным ядром последних является обработка нечеткостей, а следовательно, используются принципы мышления человека. Однако самоорганизующиеся системы идут дальше и начинают развиваться, настраиваться на объект, в определенном смысле, самостоятельно, используя получаемую в процессе работы информацию об объекте управления.

В общем случае можно предложить следующую схему реализации процесса управления: распознавание → предсказание → идентификация → принятие решения → управление.

Можно показать, что все эти задачи относятся к одному классу и могут быть решены самоорганизующимися системами.

## 4.5.3. Генетические алгоритмы

*Генетические алгоритмы* (ГА) относятся к числу универсальных методов оптимизации, позволяющих решать задачи различных типов (комбинаторные, общие задачи с ограничениями и без ограничений) и различной степени сложности. При этом ГА характеризуются возможностью как однокритериального, так и многокритериального поиска в большом пространстве, ландшафт которого является негладким.

В последние годы резко возросло число работ, прежде всего зарубежных ученых, посвященных развитию теории ГА и вопросам их практического использования. Результаты данных исследований показывают, в частности, что ГА могут получить более широкое распространение при интеграции с другими методами и технологиями. Появились работы, в которых доказывается эффективность интеграции ГА и методов теории нечеткости, а также нейронных вычислений и систем.

Эффективность такой интеграции нашла практическое подтверждение в разработке соответствующих инструментальных средств (ИС). Так, фирма Attar Software включила ГА-компонент, ориентированный на решение задач оптимизации, в свои ИС, предназначенные для разработки экспертной системы. Фирма California Scientific Software связала ИС для нейронных сетей с ГА-компонентами, обеспечивающими автоматическую генерацию и настройку нейронной сети. Фирма NIBS Inc. включила в свои ИС для нейронных

сетей, ориентированные на прогнозирование рынка ценных бумаг, ГА-компоненты, которые, по мнению финансовых экспертов, позволяют уточнять прогнозирование.

Несмотря на известные общие подходы к такой интеграции ГА и нечеткой логики, по-прежнему актуальна задача определения наиболее значимых параметров операционного базиса ГА с целью их адаптации в процессе работы ГА за счет использования нечеткого продукционного алгоритма (НПА).

Перечисленные далее причины коммерческого успеха инструментальных средств в области искусственного интеллекта могут рассматриваться как общие требования к разработке систем анализа данных, используемых ГА:

- интегрированность — разработка ИС, легко интегрирующихся с другими информационными технологиями и средствами;
- открытость и переносимость — разработка ИС в соответствии со стандартами, обеспечивающими возможность исполнения в разнородном программно-аппаратном окружении, и переносимость на другие платформы без перепрограммирования;
- использование языков традиционного программирования — переход к ИС, реализованным на языках традиционного программирования (C, C++ и т. д.), что упрощает обеспечение интегрированности, снижает требования приложений к быстродействию ЭВМ и к объемам оперативной памяти;
- архитектура "клиент-сервер" — разработка ИС, поддерживающих распределенные вычисления в архитектуре "клиент-сервер", что позволяет снизить стоимость оборудования, используемого в приложениях, децентрализовать приложения и повысить их производительность.

Перечисленные требования обусловлены необходимостью создания интегрированных приложений, т. е. приложений, объединяющих в рамках единого комплекса традиционные программные системы с системами искусственного интеллекта и ГА в частности.

Интеграция ГА и нейронных сетей позволяет решать проблемы поиска оптимальных значений весов входов нейронов, а интеграция ГА и нечеткой логики позволяет оптимизировать систему продукционных правил, которые могут быть использованы для управления операторами ГА (двунаправленная интеграция).

Одним из наиболее востребованных приложений ГА в области Data Mining является поиск наиболее оптимальной модели (поиск алгоритма, соответствующего специфике конкретной области).

В *приложении 2* представлены базовые принципы организации и выполнения ГА.

## 4.5.4. Нейронные сети

*Нейронные сети* — это класс моделей, основанных на биологической аналогии с мозгом человека и предназначенных после прохождения этапа так называемого обучения на имеющихся данных для решения разнообразных задач анализа данных. При применении этих методов, прежде всего, встает вопрос выбора конкретной архитектуры сети (числа "слоев" и количества "нейронов" в каждом из них). Размер и структура сети должны соответствовать (например, в смысле формальной вычислительной сложности) существу исследуемого явления. Поскольку на начальном этапе анализа природа явления обычно известна плохо, выбор архитектуры является непростой задачей и часто связан с длительным процессом "проб и ошибок" (однако в последнее время стали появляться нейронно-сетевые программы, в которых для решения трудоемкой задачи поиска наилучшей архитектуры сети применяются методы искусственного интеллекта).

Затем построенная сеть подвергается процессу так называемого обучения. На этом этапе нейроны сети итеративно обрабатывают входные данные и корректируют свои веса так, чтобы сеть наилучшим образом прогнозировала (в традиционных терминах следовало бы сказать "осуществляла подгонку") данные, на которых выполняется "обучение". После обучения на имеющихся данных сеть готова к работе и может использоваться для построения прогнозов.

Нейронная сеть, полученная в результате "обучения", выражает закономерности, присутствующие в данных. При таком подходе она оказывается функциональным эквивалентом некоторой модели зависимостей между переменными, подобной тем, которые строятся в традиционном моделировании. Однако, в отличие от традиционных моделей, в случае нейронных сетей эти зависимости не могут быть записаны в явном виде, подобно тому, как это делается в статистике (например, "$A$ положительно коррелированно с $B$ для наблюдений, у которых величина $C$ мала, а $D$ велика"). Иногда нейронные сети выдают прогноз очень высокого качества, однако они представляют собой типичный пример нетеоретического подхода к исследованию (иногда это называют "черным ящиком"). При таком подходе сосредотачиваются исключительно на практическом результате, в данном случае на точности прогнозов и их прикладной ценности, а не на сути механизмов, лежащих в основе явления, или на соответствии полученных результатов какой-либо имеющейся теории.

Следует, однако, отметить, что методы нейронных сетей могут применяться и в исследованиях, направленных на построение объясняющей модели явления, поскольку нейронные сети помогают изучать данные с целью поиска значимых переменных или групп таких переменных, и полученные результаты мо-

гут облегчить процесс последующего построения модели. Более того, сейчас имеются нейросетевые программы, которые с помощью сложных алгоритмов могут находить наиболее важные входные переменные, что уже непосредственно помогает строить модель.

Одно из главных преимуществ нейронных сетей состоит в том, что они, по крайней мере, теоретически могут аппроксимировать любую непрерывную функцию, и поэтому исследователю нет необходимости заранее принимать какие-либо гипотезы относительно модели и даже в ряде случаев о том, какие переменные действительно важны. Однако существенным недостатком нейронных сетей является то обстоятельство, что окончательное решение зависит от начальных установок сети и, как уже отмечалось, его практически невозможно интерпретировать в традиционных аналитических терминах, которые обычно применяются при построении теории явления.

Некоторые авторы отмечают тот факт, что нейронные сети используют или, точнее, предполагают использование вычислительных систем с массовым параллелизмом. Например, Haykin в 1994 г. определил нейронную сеть следующим образом.

> ***Внимание!***
>
> *Нейронная сеть* — это процессор с массивным распараллеливанием операций, обладающий естественной способностью сохранять экспериментальные знания и делать их доступными для последующего использования. Он похож на мозг в двух отношениях:
>
> 1. Сеть приобретает знания в результате процесса обучения.
> 2. Для хранения информации используются величины интенсивности межнейронных соединений, которые называются *синаптическими весами*.

Однако, как отмечает Риплей (1996 г.), большинство существующих нейросетевых программ работают на однопроцессорных компьютерах. По его мнению, существенно ускорить работу можно не только за счет разработки программного обеспечения, использующего преимущества многопроцессорных систем, но и с помощью создания более эффективных алгоритмов обучения.

# 4.6. Процесс обнаружения знаний

## 4.6.1. Основные этапы анализа

Для обнаружения знаний в данных недостаточно просто применить методы Data Mining, хотя, безусловно, этот этап является основным в процессе интеллектуального анализа. Весь процесс состоит из нескольких этапов. Рассмотрим основные из них, чтобы продемонстрировать, что без специальной подготовки аналитика методы Data Mining сами по себе не решают сущест-

вующих проблем. Итак, весь процесс можно разбить на следующие этапы (рис. 4.2):

1. Понимание и формулировка задачи анализа.
2. Подготовка данных для автоматизированного анализа (препроцессинг).
3. Применение методов Data Mining и построение моделей.
4. Проверка построенных моделей.
5. Интерпретация моделей человеком.


**Рис. 4.2.** Этапы интеллектуального анализа данных

На первом этапе выполняется осмысление поставленной задачи и уточнение целей, которые должны быть достигнуты методами Data Mining. Важно правильно сформулировать цели и выбрать необходимые для их достижения методы, т. к. от этого зависит дальнейшая эффективность всего процесса.

Второй этап состоит в приведении данных к форме, пригодной для применения конкретных методов Data Mining. Данный процесс далее будет описан более подробно, здесь заметим только, что вид преобразований, совершаемых над данными, во многом зависит от используемых методов, выбранных на предыдущем этапе.

Третий этап — это собственно применение методов Data Mining. Сценарии этого применения могут быть самыми различными и могут включать сложную комбинацию разных методов, особенно если используемые методы позволяют проанализировать данные с разных точек зрения.

Следующий этап — проверка построенных моделей. Очень простой и часто используемый способ заключается в том, что все имеющиеся данные, которые необходимо анализировать, разбиваются на две группы. Как правило, одна из них большего размера, другая — меньшего. На большей группе, применяя те или иные методы Data Mining, получают модели, а на меньшей — проверяют их. По разнице в точности между тестовой и обучающей группами можно судить об адекватности построенной модели.

Последний этап — интерпретация полученных моделей человеком в целях их использования для принятия решений, добавление получившихся правил и зависимостей в базы знаний и т. д. Этот этап часто подразумевает использование методов, находящихся на стыке технологии Data Mining и технологии экспертных систем. От того, насколько эффективным он будет, в значительной степени зависит успех решения поставленной задачи.

Этим этапом завершается цикл Data Mining. Окончательная оценка ценности добытого нового знания выходит за рамки анализа, автоматизированного или традиционного, и может быть проведена только после претворения в жизнь решения, принятого на основе добытого знания, после проверки нового знания практикой. Исследование достигнутых практических результатов завершает оценку ценности добытого средствами Data Mining нового знания.

## 4.6.2. Подготовка исходных данных

Как уже отмечалось ранее, для применения того или иного метода Data Mining к данным их необходимо подготовить к этому. Например, поставлена задача построить фильтр электронной почты, не пропускающий спам. Письма представляют собой тексты в электронном виде. Практически ни один из существующих методов Data Mining не может работать непосредственно с текстами. Чтобы работать с ними, необходимо из исходной текстовой информации предварительно получить некие производные параметры, например: частоту встречаемости ключевых слов, среднюю длину предложений, параметры, характеризующие сочетаемость тех или иных слов в предложении, и т. д. Другими словами, необходимо выработать некий четкий набор числовых или нечисловых параметров, характеризующих письмо. Эта задача наименее автоматизирована в том смысле, что выбор системы данных параметров производится человеком, хотя, конечно, их значения могут вычисляться автоматически. После выбора описывающих параметров изучаемые данные могут быть представлены в виде прямоугольной таблицы, где каждая

строка представляет собой отдельный случай, объект или состояние изучаемого объекта, а каждая колонка — параметры, свойства или признаки всех исследуемых объектов. Большинство методов Data Mining работают только с подобными прямоугольными таблицами.

Полученная прямоугольная таблица пока еще является слишком сырым материалом для применения методов Data Mining, и входящие в нее данные необходимо предварительно обработать. Во-первых, таблица может содержать параметры, имеющие одинаковые значения для всей колонки. Если бы исследуемые объекты характеризовались только такими признаками, они были бы абсолютно идентичны, значит, эти признаки никак не индивидуализируют исследуемые объекты. Следовательно, их надо исключить из анализа. Во-вторых, таблица может содержать некоторый категориальный признак, значения которого во всех записях различны. Ясно, что мы никак не можем использовать это поле для анализа данных и его надо исключить. Наконец, просто этих полей может быть очень много, и если все их включить в исследование, то это существенно увеличит время вычислений, поскольку практически для всех методов Data Mining характерна сильная зависимость времени от количества параметров (квадратичная, а нередко и экспоненциальная). В то же время зависимость времени от количества исследуемых объектов линейна или близка к линейной. Поэтому в качестве предварительной обработки данных необходимо, во-первых, выделить то множество признаков, которые наиболее важны в контексте данного исследования, отбросить явно неприменимые из-за константности или чрезмерной вариабельности и выделить те, которые наиболее вероятно войдут в искомую зависимость. Для этого, как правило, используются статистические методы, основанные на применении корреляционного анализа, линейных регрессий и т. д. Такие методы позволяют быстро, хотя и приближенно оценить влияние одного параметра на другой.

Мы обсудили очистку данных по столбцам таблицы (признакам). Точно так же бывает необходимо провести предварительную очистку данных по строкам таблицы (записям). Любая реальная база данных обычно содержит ошибки, очень приблизительно определенные значения, записи, соответствующие каким-то редким, исключительным ситуациям, и другие дефекты, которые могут резко понизить эффективность методов Data Mining, применяемых на следующих этапах анализа. Такие записи необходимо отбросить. Даже если подобные "выбросы" не являются ошибками, а представляют собой редкие исключительные ситуации, они все равно вряд ли могут быть использованы, поскольку по нескольким точкам статистически значимо судить об искомой зависимости невозможно. Эта предварительная обработка или препроцессинг данных и составляет второй этап процесса обнаружения знаний.

# 4.7. Управление знаниями (Knowledge Management)

Итак, основной целью технологии Data Mining является извлечение знаний из больших объемов данных. Однако если полученные знания не будут использоваться, то они, а также и все затраты на их извлечение будут бесполезны. По этой причине одной из задач анализа информации является управление полученными знаниями.

*Управление знаниями* (Knowledge Management) основано на интегральном подходе к созданию, накоплению и управлению кодифицированными и не кодифицированными знаниями.

Знания можно классифицировать, например, следующим образом:

- embodied — физические и физиологические знания, в т. ч. навыки, например, знание парикмахера, который делает прическу;
- embrained — знания, хранилищем которых является сознание, например, знания консультантов;
- encodied — кодифицированные знания, содержащиеся на разнообразных носителях информации: на бумаге, в базах данных и т. д.;
- embedded — материализованные знания, содержащиеся в технологиях, архитектуре, процедурах;
- encultured — общие интеллектуальные модели, разделяемые коллегами.

Существуют и другие виды знаний, однако в рамках Управления знаниями рассматриваются перечисленные ранее.

Знания также можно разделить на две большие группы:

- явные (explicit), например, кодифицированные;
- неявные (tacit), то есть личностные, которые по мнению специалистов не поддаются кодификации.

В управлении знаниями выделяют два основных подхода:

- метод, ориентированный на продукты — здесь в центре внимания находятся документы, хранение данных, истории событий и шаблоны решений. В данном случае знания рассматриваются без учета тех людей, которые их создают (или обнаруживают), и без тех, кто их использует;
- метод, ориентированный на процессы — это более целостный подход к управлению знаниями за счет выделения среды, в которой генерируются и распространяются знания. Его можно рассматривать как процесс социальной коммуникации. Это означает, что знания концентрируются у тех, кто их обнаруживает, а распространение информации производится путем

личных контактов. В процессе формируются самоорганизующиеся группы — сообщества, которые участвуют в развивающемся естественным образом общении.

## 4.8. Средства Data Mining

В настоящее время технология Data Mining представлена целым рядом коммерческих и свободно распространяемых программных продуктов. Достаточно полный и регулярно обновляемый список этих продуктов можно найти на сайте **www.kdnuggets.com**, посвященном Data Mining. Классифицировать программные продукты Data Mining можно по тем же принципам, что положены в основу классификации самой технологии. Однако подобная классификация не будет иметь практической ценности. Вследствие высокой конкуренции на рынке и стремления к полноте технических решений многие из продуктов Data Mining охватывают буквально все аспекты применения аналитических технологий. Поэтому целесообразнее классифицировать продукты Data Mining по тому, каким образом они реализованы и, соответственно, какой потенциал для интеграции они предоставляют. Очевидно, что и это условность, поскольку такой критерий не позволяет очертить четкие границы между продуктами. Однако у подобной классификации есть одно несомненное преимущество. Она позволяет быстро принять решение о выборе того или иного готового решения при инициализации проектов в области анализа данных, разработки систем поддержки принятия решений, создания хранилищ данных и т. д.

Итак, продукты Data Mining условно можно разделить на три больших категории:

- входящие, как неотъемлемая часть, в системы управления базами данных;
- библиотеки алгоритмов Data Mining с сопутствующей инфраструктурой;
- коробочные или настольные решения ("черные ящики").

Продукты первых двух категорий предоставляют наибольшие возможности для интеграции и позволяют реализовать аналитический потенциал практически в любом приложении в любой области. Коробочные приложения, в свою очередь, могут предоставлять некоторые уникальные достижения в области Data Mining или быть специализированными для какой-либо конкретной сферы применения. Однако в большинстве случаев их проблематично интегрировать в более широкие решения.

Включение аналитических возможностей в состав коммерческих систем управления базами данных является закономерной и имеющей огромный потенциал тенденцией. Действительно, где, как ни в местах концентрации данных, имеет наибольший смысл размещать средства их обработки. Исходя из

этого принципа, функциональность Data Mining в настоящий момент реализована в следующих коммерческих базах данных:

- Oracle;
- Microsoft SQL Server;
- IBM DB2.

Каждая из названных СУБД позволяет решать основные задачи, связанные с анализом данных, и имеет хорошие возможности для интеграции. Однако только Oracle может считаться действительно аналитической платформой. Помимо реализации функциональности Data Mining, Oracle имеет мощные средства для анализа неструктурированной текстовой информации (Oracle Text), информации, имеющей сетевую модель организации (Oracle Network Data Models), и информации, имеющей пространственные атрибуты (Oracle Spatial Topology). Таким образом, используя СУБД Oracle как платформу для построения аналитической системы, можно решить практически любую поставленную задачу: от построения рекомендательных систем для интернет-магазинов до многофункциональных систем поддержки принятия решений для различных государственных и силовых ведомств. Далее приводится обзор основных аналитических возможностей Oracle.

Oracle Data Mining включает в себя четыре наиболее мощных и зарекомендовавших себя алгоритма классификации (supervised learning):

- Naive Bayes (NB) — использует теорию вероятности для классификации объектов;
- Decision Trees — классифицирует объекты путем построения деревьев решений;
- Adaptive Bayes Networks (ABN) — расширенный вариант алгоритма NB;
- Support Vector Machines — использует теорию вычисления близости векторов для классификации объектов.

Для решения задач кластеризации и ассоциативного анализа (unsupervised learning) в Oracle применяется пять алгоритмов:

- Enhanced k-Means Clustering — для обнаружения групп схожих объектов;
- Orthogonal Partitioning Clustering — для кластеризации методом ортогонального деления;
- Anomaly Detection — для обнаружения редких, вызывающих подозрение событий (аномалий);
- Association Rules — для обнаружения шаблонов в событиях;
- Nonnegative Matrix Factorization (NMF) — для уменьшения количества атрибутов.

Oracle также включает в себя алгоритм Minimum Description Length (MDL) для решения проблемы важности атрибутов. С его помощью можно определить те атрибуты, которые имеют наибольшее влияние на зависимые поля или атрибуты.

В последние годы особую важность приобрели задачи, связанные с обработкой больших массивов генетической информации. Для их решения Oracle предлагает алгоритм BLAST (Basic Local Alignment Search Technique), позволяющий в геномных данных отыскивать последовательности, которые наиболее точно соответствуют определенным последовательностям.

Доступ к функциональности Data Mining в Oracle осуществляется либо посредством расширения языка SQL, либо с помощью прикладного программного интерфейса на Java. Отметим, что данный интерфейс совместим со спецификацией JDM.

Анализ текстовой информации в Oracle представлен целым рядом технологий, которые обеспечивают следующие возможности:

- качественный полнотекстовый поиск:
  - сортировка результатов поиска по релевантности;
  - автоматическое приведение слов в запросе к нормальной грамматической форме (стемминг);
  - поиск с указанием расположения фрагмента в тексте (заголовках, параграфах);
  - поиск фраз и точных совпадений слов;
  - использование логических операторов (И, ИЛИ и т. д.) при составлении запросов;
  - автоматическая фильтрация шумовых слов в запросе (союзы, частицы и др.);
  - автоматическое расширение запросов семантически близкими словами (синонимы и др.);
  - автоматическое расширение запросов клавиатурно близкими словами (опечатки);
  - расширение запросов с помощью масок (wildcard-символы: '*', '?');
  - поиск документов по указанной теме;
  - нечеткий поиск (созвучные слова, типичные ошибки и т. д.);
- автоматическое определение темы документа;
- автоматическое аннотирование документов;

- управление базами знаний системы (тезаурусы, словари тем, синонимы и т. д.);
- автоматическая классификация входящих документов;
- поиск групп схожих документов (кластеризация);
- поддержка нескольких естественных языков (русский, английский);
- автоматическое распознавание распространенных форматов (Word, XML, PDF и т. д.).

Таким образом, Oracle Text позволяет строить системы обработки неструктурированной информации любого уровня: от поискового портала до интеллектуальных систем документооборота. В настоящее время Oracle Text поддерживает множество языков, в том числе русский.

Анализ сетевой информации в Oracle позволяет выявлять неявные связи между объектами, организованными в сетевые структуры или графы. На первый взгляд это может показаться достаточно абстрактной областью применения, однако подобные возможности являются крайне востребованными, например, в силовых ведомствах. Всевозможные службы разведки, расследования налоговых преступлений и обнаружения нелегального финансирования терроризма, полицейские организации и т. д. располагают обширными базами данных, где фиксируются определенные аспекты деятельности объектов. В частности, их непосредственные и видимые связи друг с другом. Очевидно, что помимо видимых и зафиксированных связей существуют также неявные, опосредованные связи. Они то и представляют наибольший интерес для расследований.

Для решения такой задачи Oracle Network Data Models содержит следующую функциональность по анализу графов:

- определить все пути, соединяющие два данных узла;
- найти все узлы, достижимые с данного узла;
- найти все узлы, из которых возможно попасть в данный узел;
- определить кратчайший путь между узлами;
- определить наиболее эффективный путь, включающий указанные узлы;
- определить узлы, в которые можно попасть с данного узла в пределах указанной стоимости пути;
- определить, достижим ли целевой узел с данного узла;
- определить минимальное покрывающее дерево;
- определить ближайших соседей (по количеству) для данного узла.

Технологии анализа пространственной информации (Oracle Spatial Topology) содержат весь необходимый инструментарий для построения геоинформационных систем разных уровней сложности.

Среди многочисленных библиотек Data Mining, существующих в настоящий момент на рынке, особо следует выделить систему с открытым кодом Weka. Это динамично развивающаяся, обширная коллекция разнообразных алгоритмов, разработанная в новозеландском университете Waikato. Она реализована на языке Java, имеет достаточно простой программный интерфейс, снабжена графической оболочкой и хорошо документирована. Все это, включая свободное распространение, делает библиотеку Weka исключительно популярной. Среди недостатков библиотеки можно отметить недостаточное внимание разработчиков к проблеме масштабируемости. Это означает, что работа со сверхбольшими данными, требующая специфического подхода к разработке и оптимизации существующих алгоритмов, в Weka практически не проделана.

Использование коробочных коммерческих продуктов имеет наибольший смысл в тех проектах, где интеллектуальный анализ данных является основной целью, а не сопровождает построение более обширной системы, например, системы поддержки принятия решений. В этом случае выбор конкретного продукта может определяться множеством факторов — от предметной ориентации до цены. Чтобы упростить проблему выбора, приведем наиболее объективные и несколько неформальные данные о популярности Data Mining средств на основе ежегодных опросов на сайте **www.kdnuggets.com** (табл. 4.1).

**Таблица 4.1**

| **2005 год** | **2006 год** | **2007 год** | **2008 год** |
|---|---|---|---|
| SPSS Clementine | CART/MARS/Tree Net/RF | SPSS Clementine | SPSS Clementine |
| CART/MARS/ TreeNet/RF | SPSS Clementine | Salford CART/ MARS/TreeNet/RF | Salford CART, MARS, TreeNet, RF |
| SAS | KXEN | Excel | SPSS |
| SAS E- Miner | SAS | SPSS | Excel |
| Insightful Miner/ S-Plus | Weka | SAS | SAS |
| Statsoft Statistica | R | Angoss | KXEN |
| Weka | MATLAB | KXEN | SAS Enterprise Miner |

**Таблица 4.1** *(окончание)*

| 2005 год | 2006 год | 2007 год | 2008 год |
|---|---|---|---|
| ThinkAnalytics | SAS E-Miner | SQL Server | MATLAB |
| C4.5/C5.0/See5 | Microsoft SQL Server | MATLAB | SQL Server |
| R | Oracle Data Mining | SAS E-Miner | Other commercial tools |
| Microsoft SQL Server | Insightful Miner/ S-Plus | Other commercial tools | Angoss |
| MATLAB | C4.5/C5.0/See5 | Statsoft Statistica | Oracle DM |
| Mineset (PurpleInsight) | Statsoft Statistica | Oracle DM | Statsoft Statistica |
| Xelopes | Angoss | Tiberius | Insightful Miner/ S-Plus |
| Oracle Data Mining | Mineset (PurpleInsight) | FairIsaac Model Builder | Viscovery |
| Gornik | Eudaptics Viscovery | Xelopes | Xelopes |
| KXEN | Xelopes | Miner3D | Tiberius |
| IBM Iminer | Visumap | Bayesia | Miner3D |
| Angoss | IBM I-miner | Megaputer | Megaputer |
| Equbits | Equbits | | FairIsaac Model Builder |
| Fair Isaac | | | Bayesia |

Списки продуктов в таблице упорядочены по степени убывания "популярности". Как видно, из года в год лидируют одни и те же инструменты, хотя общий список насчитывает не менее сотни наименований. Очевидно, что при выборе подходящего средства Data Mining в первую очередь следует рассматривать и сравнивать между собой перечисленные продукты.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Интеллектуальный анализ данных позволяет автоматически, основываясь на большом количестве накопленных данных, генерировать гипотезы, которые могут быть проверены другими средствами анализа (например, OLAP).

- Data Mining — исследование и обнаружение машиной (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны и доступны для интерпретации человеком.
- Методами Data Mining решаются три основные задачи: задача классификации и регрессии, задача поиска ассоциативных правил и задача кластеризации. По назначению они делятся на описательные и предсказательные. По способам решения задачи разделяют на supervised learning (обучение с учителем) и unsupervised learning (обучение без учителя).
- Задача классификации и регрессии сводится к определению значения зависимой переменной объекта по его независимым переменным. Если зависимая переменная принимает численные значения, то говорят о задаче регрессии, в противном случае — о задаче классификации.
- При поиске ассоциативных правил целью является нахождение частых зависимостей (или ассоциаций) между объектами или событиями. Найденные зависимости представляются в виде правил и могут быть использованы как для лучшего понимания природы анализируемых данных, так и для предсказания событий.
- Задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик во всем множестве анализируемых данных. Решение этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет сократить их число, а следовательно, и облегчить анализ.
- Методы Data Mining находятся на стыке разных направлений информационных технологий и технологий искусственного интеллекта: статистики, нейронных сетей, нечетких множеств, генетических алгоритмов и др.
- Интеллектуальный анализ включает в себя следующие этапы: понимание и формулировка задачи анализа, подготовка данных для автоматизированного анализа, применение методов Data Mining и построение моделей, проверка построенных моделей, интерпретация моделей человеком.
- Перед применением методов Data Mining исходные данные должны быть преобразованы. Вид преобразований зависит от применяемых методов.
- Полученными в результате анализа данных знаниями должны управляться, как и любым другим ресурсом организации. Для управления знаниями могут использоваться подходы, входящие в технологию Knowledge Management.
- Методы Data Mining могут эффективно использоваться в различных областях человеческой деятельности: в бизнесе, медицине, науке, телекоммуникациях и т. д.

# ГЛАВА 5

# Классификация и регрессия

## 5.1. Постановка задачи

В задаче классификации и регрессии требуется определить значение зависимой переменной объекта на основании значений других переменных, характеризующих данный объект. Формально задачу классификации и регрессии можно описать следующим образом. Имеется множество объектов:

$$I = \{i_1, i_2, ..., i_j, ..., i_n\},$$

где $i_j$ — исследуемый объект. Примером таких объектов может быть информация о проведении игр при разных погодных условиях (табл. 5.1).

**Таблица 5.1**

| Наблюдение | Температура | Влажность | Ветер | Игра |
|---|---|---|---|---|
| Солнце | Жарко | Высокая | Нет | Нет |
| Солнце | Жарко | Высокая | Есть | Нет |
| Облачность | Жарко | Высокая | Нет | Да |
| Дождь | Норма | Высокая | Нет | Да |
| Дождь | Холодно | Норма | Нет | Да |
| Дождь | Холодно | Норма | Есть | Нет |
| Облачность | Холодно | Норма | Есть | Да |
| Солнце | Норма | Высокая | Нет | Нет |
| Солнце | Холодно | Норма | Нет | Да |
| Дождь | Норма | Норма | Нет | Да |
| Солнце | Норма | Норма | Есть | Да |

**Таблица 5.1** *(окончание)*

| Наблюдение | Температура | Влажность | Ветер | Игра |
|---|---|---|---|---|
| Облачность | Норма | Высокая | Есть | Да |
| Облачность | Жарко | Норма | Нет | Да |
| Дождь | Норма | Высокая | Есть | Нет |

Каждый объект характеризуется набором переменных:

$$I_j = \{x_1, x_2, ..., x_h, ..., x_m, y\},$$

где $x_h$ — независимые переменные, значения которых известны и на основании которых определяется значение зависимой переменной $y$. В данном примере независимыми переменными являются: `наблюдение`, `температура`, `влажность` и `ветер`. Зависимой переменной является `игра`.

В Data Mining часто набор независимых переменных обозначают в виде вектора:

$$X = \{x_1, x_2, ..., x_h, ..., x_m\}.$$

Каждая переменная $x_h$ может принимать значения из некоторого множества:

$$C_h = \{c_{h1}, c_{h2}, ...\}.$$

Если значениями переменной являются элементы конечного множества, то говорят, что она имеет категориальный тип. Например, переменная `наблюдение` принимает значения на множестве значений `{солнце, облачность, дождь}`.

Если множество значений $C = \{c_1, c_2, ..., c_r, ..., c_k\}$ переменной $y$ конечное, то задача называется задачей классификации. Если переменная $y$ принимает значение на множестве действительных чисел $R$, то задача называется задачей регрессии.

# 5.2. Представление результатов

## 5.2.1. Правила классификации

В задачах классификации и регрессии обнаруженная функциональная зависимость между переменными может быть представлена одним из следующих способов:

- классификационные правила;
- деревья решений;
- математические функции.

Классификационные правила состоят из двух частей: условия и заключения:

```
если (условие) то (заключение).
```

Условием является проверка одной или нескольких независимых переменных. Проверки нескольких переменных могут быть объединены с помощью операций "и", "или" и "не". Заключением является значение зависимой переменной или распределение ее вероятности по классам, например:

```
если (наблюдение = солнце и температура = жарко) то (игра = нет);
если (наблюдение = облачность и температура = холодно) то (игра = да).
```

Основным достоинством правил является легкость их восприятия и запись на естественном языке. Еще одно преимущество — их относительная независимость. В набор правил легко добавить новое правило без необходимости изменять уже существующие. Относительность независимости правил связана с возможной их противоречивостью друг другу. Если переменные, характеризующие некоторый объект, удовлетворяют условным частям правил с разными заключениями, то возникает неопределенность со значением его зависимой переменной. Например, пусть имеются правила:

```
если (наблюдение = солнце) то (игра = нет);
если (наблюдение = облачность и температура = холодно) то (игра = да).
```

В них объекты, удовлетворяющие условиям второго правила, удовлетворяют и условиям первого правила. Однако вывод делается разный. Другими словами, в соответствии с этими правилами при одинаковых обстоятельствах будут получены противоречивые указания, что неприемлемо.

## 5.2.2. Деревья решений

Деревья решений — это способ представления правил в иерархической, последовательной структуре. На рис. 5.1 изображен пример дерева решений для данных, представленных в табл. 5.1.

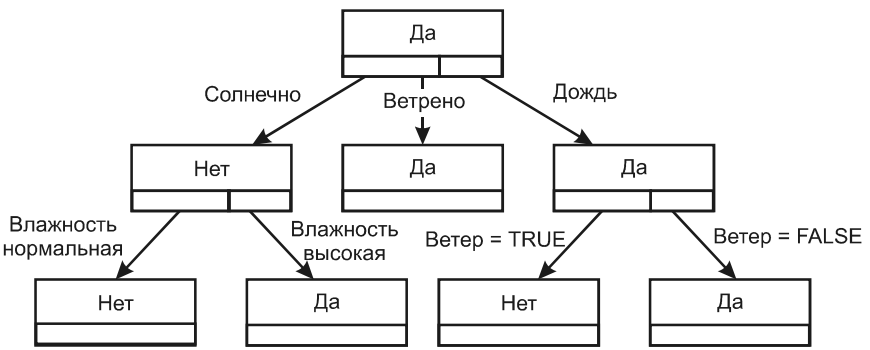



**Рис. 5.1.** Пример дерева решений

Обычно каждый узел дерева включает проверку определенной независимой переменной. Иногда в узле дерева две независимые переменные сравниваются друг с другом или определяется некоторая функция от одной или нескольких переменных.

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 несколько интервалов. В этом случае выполняется проверка на попадание значения в один из интервалов.

Листья деревьев соответствуют значениям зависимой переменной, т. е. классам. Объект принадлежит определенному классу, если значения его независимых переменных удовлетворяют условиям, записанным в узлах дерева на пути от корня к листу, соответствующему этому классу.

Если какая-либо независимая переменная классифицируемого объекта не имеет значения, то возникает проблема, связанная с неопределенностью пути, по которому необходимо двигаться по дереву. В некоторых случаях пропущенные значения можно заменять значениями по умолчанию. Если такой подход неприемлем, то необходимо предусмотреть специальные способы обработки таких ситуаций (например, перемещаться по ветви, которая ведет к бóльшему количеству объектов из обучающей выборки). Другой вариант обработки может быть связан с добавлением специальной ветви к узлу для пропущенных значений.

Деревья решений легко преобразуются в правила. В условную часть таких правил записывается условие, описанное в узлах дерева на пути к листу, в заключительную часть — значение, определенное в листе. Например, для дерева, приведенного на рис. 5.1, могут быть построены следующие правила:

```
если наблюдение = солнечно и влажность = высокая то игра = нет;
если наблюдение = солнечно и влажность = нормально то игра = да;
если наблюдение = дождь и ветер = да то игра = нет;
если наблюдение = дождь и ветер = нет то игра = да;
```

Необходимо заметить, что обратное преобразование от правил к дереву не всегда возможно. Это связано с бóльшей свободой записи правил. Например, при использовании операции "или" в построенном по такому правилу дереву возникнет необходимость в дублировании поддеревьев.

## 5.2.3. Математические функции

*Математическая функция* выражает отношение зависимой переменной от независимых переменных. В этом случае анализируемые объекты рассматриваются как точки в $(m + 1)$-мерном пространстве. Тогда переменные объекта

$i_j = \{x_1, x_2, ..., x_h, ..., x_m, y\}$, рассматриваются как координаты, а функция имеет следующий вид:

$$y_j = \{\omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_m x_m\},$$

где $\omega_0, \omega_1, ..., \omega_m$ — веса независимых переменных, в поиске которых и состоит задача нахождения классификационной функции.

Очевидно, что все переменные должны быть представлены в виде числовых параметров. Для преобразования логических и категориальных переменных к числовым используют разные способы.

Логические типы, как правило, кодируют цифрами 1 и 0. Истине ставят в соответствие значение 1, а ложь обозначают 0.

Значениями категориальных переменных являются имена возможных состояний изучаемого объекта. Разумеется, таких состояний может быть больше двух. Их имена должны быть перечислены и пронумерованы в списке. Каждое имя из списка может быть представлено своим номером. В итоге категориальная переменная преобразуется в числовую переменную. Например, значение переменной `наблюдение = {солнце, облачность, дождь}` можно заменить значениями `{0, 1, 2}`.

Другой способ представления исходно категориальной переменной в системе — это замена возможных значений набором двоичных признаков. В наборе столько двоичных признаков, сколько имен содержится в списке возможных состояний объекта. При анализе объекта значение 1 присваивается тому двоичному признаку, который соответствует состоянию объекта. Остальным присваивается значение 0. Например, для переменной наблюдения такими значениями будут `{001, 010, 100}`.

Разные алгоритмы решения задачи классификации и регрессии строят и используют различные способы определения значения зависимой переменной.

# 5.3. Методы построения правил классификации

## 5.3.1. Алгоритм построения 1-правил

Рассмотрим простейший алгоритм формирования элементарных правил для классификации объекта. Он строит правила по значению одной независимой переменной, поэтому в литературе его часто называют "1-правило" (1-rule) или кратко 1R-алгоритм.

Идея алгоритма очень проста. Для любого возможного значения каждой независимой переменной формируется правило, которое классифицирует объ-

екты из обучающей выборки. При этом в заключительной части правила указывается значение зависимой переменной, которое наиболее часто встречается у объектов с выбранным значением независимой переменной. В этом случае ошибкой правила является количество объектов, имеющих то же значение рассматриваемой переменной, но не относящихся к выбранному классу.

Таким образом, для каждой переменной будет получен набор правил (для каждого значения). Оценив степень ошибки каждого набора, выбирается переменная, для которой построены правила с наименьшей ошибкой.

Для примера, представленного в табл. 5.1, в результате будут получены правила и их оценки, приведенные в табл. 5.2.

**Таблица 5.2**

| **Правило** | **Ошибка** |
|---|---|
| Если (наблюдение = солнце) то (игра = нет) | 2/5 |
| Если (наблюдение = облачно) то (игра = да) | 0/4 |
| Если (наблюдение = дождь) то (игра = да) | 2/5 |
| Если (температура = жарко) то (игра = нет) * | 2/4 |
| Если (температура = норма) то (игра = да) | 2/6 |
| Если (температура = холодно) то (игра = да) | 1/4 |
| Если (влажность = высокая) то (игра = нет) | 3/7 |
| Если (влажность = норма) то (игра = да) | 1/7 |
| Если (ветер = нет) то (игра = да) | 2/8 |
| Если (ветер = есть) то (игра = нет) * | 3/6 |

Если в обучающей выборке встречаются объекты с пропущенными значениями независимых переменных, то 1R-алгоритм подсчитывает такие объекты для каждого возможного значения переменной.

Другой проблемой для рассматриваемого алгоритма являются численные значения переменных. Очевидно, что если переменная имеет вещественный тип, то количество возможных значений может быть бесконечно. Для решения этой проблемы всю область значений такой переменной разбивают на интервалы таким образом, чтобы каждый из них соответствовал определенному классу в обучающей выборке. В результате будет получен набор дискретных значений, с которыми может работать данный алгоритм.

Предположим, что данные переменной температура, приведенные в табл. 5.1, имеют следующие числовые значения и соответствующие им значения зависимой переменной:

```
 4   5   8   9  10  11  12  12  15  15  20  21  23  25
да |нет|да  да  да |нет нет|да  да  да |нет|да  да |нет
```

В этом случае диапазон значений можно было бы разбить на интервалы следующим образом:

```
{до 4,5; 4,5-7,5; 7,5-10,5; 10,5-12; 12-17,5; 17,5-20,5; 20,5-24;
более 24 }
```

Более серьезная проблема рассматриваемого алгоритма — это сверхчувствительность (overfitting). Дело в том, что алгоритм будет выбирать переменные, принимающие наибольшее количество возможных значений, т. к. для них ошибка будет наименьшей. Например, для переменной, являющейся ключом (т. е. для каждого объекта свое уникальное значение), ошибка будет равна нулю. Однако для таких переменных правила будут абсолютно бесполезны, поэтому при формировании обучающей выборки для данного алгоритма важно правильно выбрать набор независимых переменных.

В заключение необходимо отметить, что 1R-алгоритм, несмотря на свою простоту, во многих случаях на практике оказывается достаточно эффективным. Это объясняется тем, что многие объекты действительно можно классифицировать лишь по одному атрибуту. Кроме того, немногочисленность формируемых правил позволяет легко понять и использовать полученные результаты.

## 5.3.2. Метод Naive Bayes

Рассмотренный ранее 1R-алгоритм формирует правила для принятия решений лишь по одной переменной объекта. Однако это не всегда приемлемо. Нередко для классификации необходимо рассмотреть несколько независимых переменных. Такую классификацию позволяет выполнять алгоритм Naive Bayes, использующий формулу Байеса для расчета вероятности. Название naive (наивный) происходит от наивного предположения, что все рассматриваемые переменные независимы друг от друга. В действительности это не всегда так, но на практике данный алгоритм находит применение.

Вероятность того, что некоторый объект $i_j$ относится к классу $c_r$ (т. е. $y = c_r$), обозначим как $P(y = c_r)$. Событие, соответствующее равенству независимых переменных определенным значениям, обозначим как $E$, а вероятность его наступления $P(E)$. Идея алгоритма заключается в расчете услов-

ной вероятности принадлежности объекта к $c_r$ при равенстве его независимых переменных определенным значениям. Из теории вероятности известно, что ее можно вычислить по формуле:

$$P(y = c_r \mid E) = P(E \mid y = c_r) \cdot P(y = c_r) / P(E).$$

Другими словами, формируются правила, в условных частях которых сравниваются все независимые переменные с соответствующими возможными значениями. В заключительной части присутствуют все возможные значения зависимой переменной:

$$\text{если } x_1 = c_p^1 \text{ и } x_2 = c_d^2 \text{ и ... } x_m = c_b^m, \text{ тогда } y = c_r .$$

Для каждого из этих правил по формуле Байеса определяется его вероятность. Предполагая, что переменные принимают значения независимо друг от друга, выразим вероятность $P(E \mid y = c_r)$ через произведение вероятностей для каждой независимой переменной:

$$P(E \mid y = c_r) = P(x_1 = c_p^1 \mid y = c_r) \times P(x_2 = c_d^2 \mid y = c_r) \times ... \times$$
$$\times P(x_m = c_b^m \mid y = c_r).$$

Тогда вероятность для всего правила можно определить по формуле:

$$P(y = c_r \mid E) = P(x_1 = c_p^1 \mid y = c_r) \times P(x_2 = c_d^2 \mid y = c_r) \times ... \times$$
$$\times P(x_m = c_b^m \mid y = c_r) \times P(y = c_r) / P(E).$$

Вероятность принадлежности объекта к классу $c_r$ при условии равенства его переменной $x_h$ некоторому значению $c_d^h$ определяется по формуле:

$$P(x_h = c_d^h \mid y = c_r) = P(x_h = c_d^h \text{ и } y = c_r) / P(y = c_r),$$

т. е. равно отношению количества объектов в обучающей выборке, у которых $x_h = c_d^h$ и $y = c_r$, к количеству объектов, относящихся к классу $c_r$. Например, для объектов из табл. 5.1 получаем следующие вероятности для значений независимой переменной `наблюдение`:

$$P(\texttt{наблюдение = солнце} \mid \texttt{игра = да}) = 2/9;$$
$$P(\texttt{наблюдение = облачно} \mid \texttt{игра = да}) = 4/9;$$
$$P(\texttt{наблюдение = дождь} \mid \texttt{игра = да}) = 3/9;$$
$$P(\texttt{наблюдение = солнце} \mid \texttt{игра = нет}) = 3/5;$$
$$P(\texttt{наблюдение = облачно} \mid \texttt{игра = нет}) = 0/5;$$
$$P(\texttt{наблюдение = дождь} \mid \texttt{игра = нет}) = 2/5.$$

Вероятность $P(y = c_r)$ есть отношение объектов из обучающей выборки, принадлежащих классу $c_r$, к общему количеству объектов в выборке. В данном примере это:

$$P(\text{игра = да}) = 9/14;$$

$$P(\text{игра = нет}) = 5/14.$$

Таким образом, если необходимо определить, состоится ли игра при следующих значениях независимых переменных (событии $E$):

```
наблюдение = солнечно;
температура = холодно;
влажность = высокая;
ветер = есть,
```

то надо вычислить следующие условные вероятности:

$$P(\text{игра = да} \mid E) = P(\text{наблюдение = солнечно} \mid \text{игра = да}) \times$$

$$\times P(\text{температура = холодно} \mid \text{игра = да}) \times$$

$$\times P(\text{влажность = высокая} \mid \text{игра = да}) \times$$

$$\times P(\text{ветер = есть} \mid \text{игра = да}) \times P(\text{игра = да}) / P(E);$$

$$P(\text{игра = нет} \mid E) = P(\text{наблюдение = солнечно} \mid \text{игра = нет}) \times$$

$$\times P(\text{температура = холодно} \mid \text{игра = нет}) \times$$

$$\times P(\text{влажность = высокая} \mid \text{игра = нет}) \times$$

$$\times P(\text{ветер = есть} \mid \text{игра = нет}) \times P(\text{игра = нет}) / P(E).$$

Подставляя соответствующие вероятности, получим следующие значения:

$$P(\text{игра = да} \mid E) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 / P(E) = 0{,}0053/P(E);$$

$$P(\text{игра = нет} \mid E) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 / P(E) = 0{,}0206/P(E).$$

Вероятность $P(E)$ не учитывается, т. к. при нормализации вероятностей для каждого из возможных правил она исчезает. Нормализованная вероятность для правила вычисляется по формуле:

$$P'(y = c_r \mid E) = P(y = c_r \mid E) / \sum P(y = c_r \mid E).$$

В данном случае можно утверждать, что при указанных условиях игра состоится с вероятностью:

$$P'(\text{игра = да} \mid E) = 0{,}0053/(0{,}0053 + 0{,}0206) = 0{,}205;$$

и не состоится с вероятностью:

$$P'(\text{игра = нет} \mid E) = 0{,}0206/(0{,}0053 + 0{,}0206) = 0{,}795.$$

Таким образом, при указанных условиях более вероятно, что игра не состоится.

При использовании формулы Байеса для оценки достоверности правила возникает проблема, связанная с тем, что в обучающей выборке может не быть ни одного объекта, имеющего значение $c_d^h$ переменной $x_h$ и относящегося к классу $c_r$. В этом случае соответствующая вероятность будет равна 0, а следовательно, и вероятность такого правила равна 0. Чтобы избежать этого, к каждой вероятности добавляется некоторое значение, отличное от нуля. Такая методика называется оценочной функцией Лапласа.

Одним из действительных преимуществ данного метода является то, что пропущенные значения не создают никакой проблемы. При подсчете вероятности они просто пропускаются для всех правил, и это не влияет на соотношение вероятностей.

Числовые значения независимых переменных обычно обрабатываются с учетом того, что они имеют нормальное или Гауссово распределение вероятностей. Для них определяются математическое ожидание и среднеквадратичное отклонение.

В данном случае под математическим ожиданием понимается просто среднее число значений, т. е. сумма, разделенная на число объектов. Среднеквадратичное отклонение — это квадратный корень из типовой дисперсии.

Функция плотности вероятности для нормального распределения с математическим ожиданием $\mu$ и среднеквадратичным отклонением $\sigma$:

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

Функция плотности вероятности для объекта тесно связана с его вероятностью, однако это не совсем то же самое. Реальный смысл функции плотности $f(x)$ — вероятность того, что количество значений зависимой переменной в пределах небольшой области вокруг $x$ (например, между $x - e/2$ и $x + e/2$) равна $f(x)$.

# 5.4. Методы построения деревьев решений

## 5.4.1. Методика "разделяй и властвуй"

Общий принцип построения деревьев решений, основанный на методике "разделяй и властвуй", заключается в рекурсивном разбиении множества объектов из обучающей выборки на подмножества, содержащие объекты, относящиеся к одинаковым классам.

Относительно обучающей выборки $T$ и множества классов $C$ возможны три ситуации:

- множество $T$ содержит один или более объектов, относящихся к одному классу $c_r$. Тогда дерево решений для $T$ — это лист, определяющий класс $c_r$;
- множество $T$ не содержит ни одного объекта (пустое множество). Тогда это снова лист, и класс, ассоциированный с листом, выбирается из другого множества, отличного от $T$, например, из множества, ассоциированного с родителем;
- множество $T$ содержит объекты, относящиеся к разным классам. В этом случае следует разбить множество $T$ на некоторые подмножества. Для этого выбирается одна из независимых переменных $x_h$, имеющая два и более отличных друг от друга значений $c_h^1$, $c_h^2$, ..., $c_h^l$; множество $T$ разбивается на подмножества $T_1$, $T_2$, …, $T_n$ где каждое подмножество $T_i$ содержит все объекты, имеющие значение $c_h^l$ для выбранного признака. Эта процедура будет рекурсивно продолжаться до тех пор, пока в конечном множестве не окажутся объекты только одного класса.

Очевидно, что при использовании данной методики построение дерева решений будет происходить сверху вниз. Описанная процедура лежит в основе многих алгоритмов построения деревьев решений. Большинство из них являются "жадными алгоритмами". Это значит, что если один раз переменная была выбрана и по ней было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другую переменную, которая дала бы лучшее разбиение. Поэтому на этапе построения нельзя сказать, даст ли выбранная переменная, в конечном итоге, оптимальное разбиение.

При построении деревьев решений особое внимание уделяется выбору переменной, по которой будет выполняться разбиение. Для построения дерева на каждом внутреннем узле необходимо найти такое условие (проверку), которое бы разбивало множество, ассоциированное с этим узлом, на подмножества. В качестве такой проверки должна быть выбрана одна из независимых переменных. Общее правило для выбора можно сформулировать следующим образом: выбранная переменная должна разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т. е. чтобы количество объектов из других классов ("примесей") в каждом из этих множеств было минимальным. Разные алгоритмы реализуют разные способы выбора.

Другой проблемой при построении дерева является проблема остановки его разбиения. В дополнение к основному методу построения деревьев решений были предложены следующие правила:

- использование статистических методов для оценки целесообразности дальнейшего разбиения, так называемая "ранняя остановка" (prepruning).

В конечном счете "ранняя остановка" процесса построения привлекательна в плане экономии времени обучения, но здесь уместно сделать одно важное предостережение: этот подход строит менее точные классификационные модели, и поэтому "ранняя остановка" крайне нежелательна. Признанные авторитеты в этой области Л. Брейман и Р. Куинлен советуют буквально следующее: "вместо остановки используйте отсечение";

- ограничить глубину дерева. Нужно остановить дальнейшее построение, если разбиение ведет к дереву с глубиной, превышающей заданное значение;
- разбиение должно быть нетривиальным, т. е. получившиеся в результате узлы должны содержать не менее заданного количества объектов.

Этот список эвристических правил можно продолжить, но на сегодняшний день не существует такого, которое бы имело большую практическую ценность. К этому вопросу следует подходить осторожно, т. к. многие из правил применимы в каких-то частных случаях.

Очень часто алгоритмы построения деревьев решений дают сложные деревья, которые "переполнены данными" и имеют много узлов и ветвей. В таких "ветвистых" деревьях очень трудно разобраться. К тому же, ветвистое дерево, имеющее много узлов, разбивает обучающее множество на все большее количество подмножеств, состоящих из все меньшего количества объектов. Ценность правила, справедливого, например, для 2—3 объектов, крайне низка, и в целях анализа данных такое правило практически непригодно. Гораздо предпочтительнее иметь дерево, состоящее из малого количества узлов, которым бы соответствовало большое количество объектов из обучающей выборки. И тут возникает вопрос: а не построить ли все возможные варианты деревьев, соответствующие обучающему множеству, и из них выбрать дерево с наименьшей глубиной? К сожалению, Л. Хайфилем и Р. Ривестом было показано, что данная задача является NP-полной, а как известно, этот класс задач не имеет эффективных методов решения.

Для решения описанной проблемы часто применяется так называемое отсечение ветвей (pruning).

Пусть под точностью (распознавания) дерева решений понимается отношение правильно классифицированных объектов при обучении к общему количеству объектов из обучающего множества, а под ошибкой — количество неправильно классифицированных объектов. Предположим, что известен способ оценки ошибки дерева, ветвей и листьев. Тогда можно использовать следующее простое правило:

- построить дерево;
- отсечь или заменить поддеревом те ветви, которые не приведут к возрастанию ошибки.

В отличие от процесса построения, отсечение ветвей происходит снизу вверх, двигаясь с листьев дерева, отмечая узлы как листья либо заменяя их поддеревом. Хотя отсечение не является панацеей, но в большинстве практических задач дает хорошие результаты, что позволяет говорить о правомерности использования подобной методики.

## Алгоритм ID3

Рассмотрим более подробно критерий выбора переменной, используемый в алгоритмах ID3 и C4.5. Очевидно, что полный набор вариантов разбиения описывается множеством $|X|$ (количество независимых переменных).

Рассмотрим проверку переменной $x_h$ (в качестве проверки может быть выбрана любая переменная), которая принимает $m$ значений $c_{h1}$, $c_{h2}$, ..., $c_{hm}$. Тогда разбиение $T$ по проверке переменной $x_h$ даст подмножества $T_1$, $T_2$, …, $T_m$. Единственная доступная информация — каким образом классы распределены в множестве $T$ и его подмножествах, получаемых при разбиении. Именно она и используется при выборе переменной. В данном случае существует четыре варианта разбиения дерева (рис. 5.2).

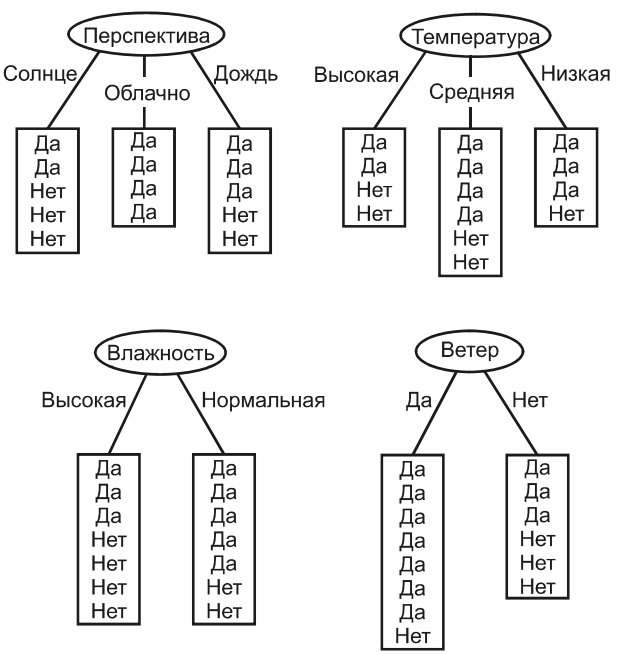



**Рис. 5.2.** Четыре варианта первоначального разбиения дерева для разных переменных

Пусть $\text{freq}(c_r, I)$ — количество объектов из множества $I$, относящихся к одному и тому же классу $c_r$. Тогда вероятность того, что случайно выбранный объект из множества $I$ будет принадлежать классу $c_r$, равна:

$$P = \frac{\text{freq}(\tilde{n}_r, I)}{|I|}.$$

Так, для примера, рассмотренного в табл. 5.1, вероятность того, что в случайно выбранный день игра состоится, равна 9/14.

Согласно теории информации оценку среднего количества информации, необходимого для определения класса объекта из множества $T$, дает выражение:

$$\text{Info}(T) = -\sum_{j=1}^{k} \text{freq}\left(c_r, \frac{T}{|T|}\right) \log_2\left(\frac{\text{freq}(c_r, T)}{|T|}\right).$$

Поскольку используется логарифм с двоичным основанием, то это выражение дает количественную оценку в битах. Для данного примера имеем:

$$\text{Info}(I) = -9/14 \cdot \log_2(9/14) - 5/14 \cdot \log_2(5/14) = 0{,}940 \text{ бит}.$$

Ту же оценку, но только уже после разбиения множества $T$ по $x_h$, дает следующее выражение:

$$\text{Info}_{x_h}(T) = \sum_{i=1}^{m} T_i / |T| \, \text{Info}(T_i).$$

Например, для переменной `наблюдение` оценка будет следующей:

$$\text{Info}_{\text{наблюдение}} = (5/14) \cdot 0{,}971 + (4/14) \cdot 0 + (5/14) \cdot 0{,}971 = 0{,}693 \text{ бит}.$$

Критерием для выбора атрибута будет являться следующая формула:

$$\text{Gain}(x_h) = \text{Info}(T) - \text{Info}_{x_h}(T).$$

Этот критерий считается для всех независимых переменных. В данном примере:

$$\text{Gain}(\texttt{наблюдение}) = 0{,}247 \text{ бит};$$

$$\text{Gain}(\texttt{температура}) = 0{,}029 \text{ бит};$$

$$\text{Gain}(\texttt{влажность}) = 0{,}152 \text{ бит};$$

$$\text{Gain}(\texttt{ветер}) = 0{,}048 \text{ бит}.$$

Выбирается переменная с максимальным значением Gain. Она и будет являться проверкой в текущем узле дерева, а затем по ней будет производиться дальнейшее построение дерева. То есть в узле будет проверяться значение по

этой переменной и дальнейшее движение по дереву будет производиться в зависимости от полученного ответа. Таким образом, для случая с определением игры будет выбрана переменная наблюдение.

Такие же рассуждения можно применить к полученным подмножествам $T_1$, $T_2$, …, $T_m$ и продолжить рекурсивно процесс построения дерева до тех пор, пока в узле не окажутся объекты из одного класса.

Так для множества, полученного при значении солнечно переменной наблюдение, для остальных трех переменных будут следующие значения:

$$\text{Gain}(\text{температура}) = 0{,}571 \text{ бит};$$

$$\text{Gain}(\text{влажность}) = 0{,}971 \text{ бит};$$

$$\text{Gain}(\text{ветер}) = 0{,}020 \text{ бит}.$$

Таким образом, следующей переменной, по которой будет разбиваться подмножество $T_{солнечно}$, окажется влажность. Построенное дерево будет выглядеть так, как изображено на рис. 5.3.

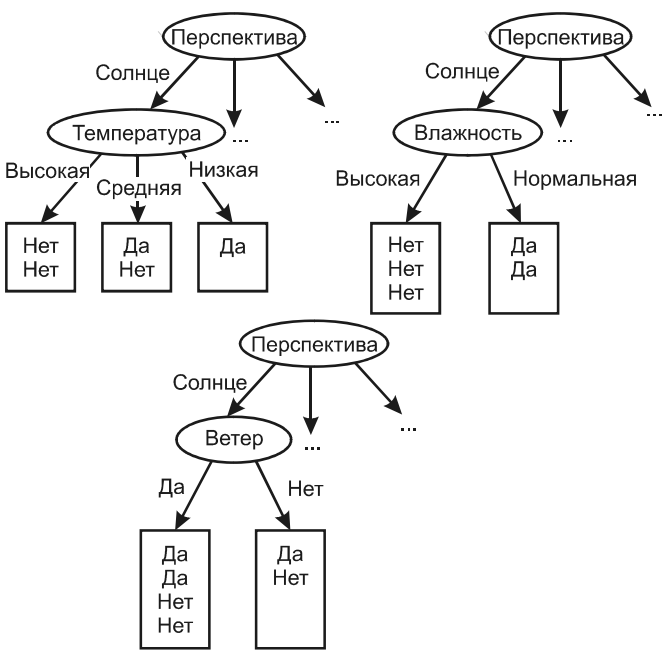



**Рис. 5.3.** Разбиение дерева на второй итерации для подмножества $T_{солнечно}$

Одно важное замечание: если в процессе работы алгоритма получен узел, ассоциированный с пустым множеством (ни один объект не попал в данный

узел), то он помечается как лист, и в качестве решения листа выбирается наиболее часто встречающийся класс у непосредственного предка данного листа.

Здесь следует пояснить, почему критерий $\mathrm{Gain}(X)$ должен максимизироваться. Из свойств энтропии известно, что максимально возможное значение энтропии достигается в том случае, когда все сообщения множества равновероятны. В данном случае энтропия $\mathrm{Info}_x$ достигает своего максимума, когда частота появления классов в множестве $T$ равновероятна. Необходимо выбрать такую переменную, чтобы при разбиении по ней один из классов имел наибольшую вероятность появления. Это возможно в том случае, когда энтропия $\mathrm{Info}_x$ имеет минимальное значение и, соответственно, критерий $\mathrm{Gain}(X)$ достигает своего максимума.

## Алгоритм C4.5

Рассмотренный способ выбора переменной использует алгоритм ID3. Однако он подвержен сверхчувствительности (overfitting), т. е. "предпочитает" переменные, которые имеют много значений. Например, если переменная уникально идентифицирует объекты, то ввиду уникальности каждого значения этой переменной при разбиении множества объектов по ней получаются подмножества, содержащие только по одному объекту. Так как все эти множества "однообъектные", то и объект относится, соответственно, к одному-единственному классу, поэтому:

$$\mathrm{Info}_x(T) = 0.$$

В этом случае $\mathrm{Gain}(X)$ принимает свое максимальное значение, и, несомненно, именно эта переменная будет выбрана алгоритмом. Однако если рассмотреть проблему под другим углом — с точки зрения предсказательных способностей построенной модели, то становится очевидным вся бесполезность такой модели.

В алгоритме C4.5 проблема решается введением некоторой нормализации. Пусть суть информации сообщения, относящегося к объекту, указывает не на класс, к которому объект принадлежит, а на выход. Тогда, по аналогии с определением $\mathrm{Info}(T)$, имеем:

$$\text{split info}(x_h) = -\sum_{i=1}^{m} |T_i|/|T| \cdot \log_2\left(|T_i|/|T|\right).$$

Это выражение оценивает потенциальную информацию, получаемую при разбиении множества $T$ на $m$ подмножеств.

Рассмотрим следующее выражение:

$$\text{gain ratio}\left(x_h\right) = \mathrm{Gain}\left(x_h\right)/\text{split info}\left(x_h\right).$$

Пусть это выражение является критерием выбора переменной.

Очевидно, что переменная, идентифицирующая объект, не будет высоко оценена критерием gain ratio. Пусть имеется $k$ классов, тогда числитель выражения максимально будет равен $\log_2 k$, и пусть $n$ — количество объектов в обучающей выборке и одновременно количество значений переменных, тогда знаменатель максимально равен $\log_2 n$. Если предположить, что количество объектов заведомо больше количества классов, то знаменатель растет быстрее, чем числитель и, соответственно, значение выражения будет небольшим.

Несмотря на улучшение критерия выбора атрибута для разбиения, алгоритм может создавать узлы и листья, содержащие незначительное количество примеров. Чтобы избежать этого, следует воспользоваться еще одним эвристическим правилом: при разбиении множества $T$ по крайней мере два подмножества должны иметь не меньше заданного минимального количества объектов $k(k>1)$; обычно оно равно двум. В случае невыполнения данного правила дальнейшее разбиение этого множества прекращается и соответствующий узел отмечается как лист. При таком ограничении возможна ситуация, когда объекты, ассоциированные с узлом, относятся к разным классам. В качестве решения листа выбирается класс, который наиболее часто встречается в узле. Если же примеров равное количество из всех классов, то решение дает класс, наиболее часто встречающийся у непосредственного предка данного листа.

Рассматриваемый алгоритм построения деревьев решений предполагает, что для переменной, выбираемой в качестве проверки, существуют все значения, хотя явно это нигде не утверждалось. То есть для любого примера из обучающей выборки существует значение по этой переменной.

Первое решение, которое лежит на поверхности, — не учитывать объекты с пропущенными значениями. Следует подчеркнуть, что крайне нежелательно отбрасывать весь объект только потому, что по одной из переменных пропущено значение, поскольку можно потерять много полезной информации.

Тогда необходимо выработать процедуру работы с пропущенными данными.

Пусть $T$ — обучающая выборка и $X$ — проверка по некоторой переменной $x$. Обозначим через $U$ количество неопределенных значений переменной $x$. Изменим формулы таким образом, чтобы учитывать только те объекты, у которых существуют значения по переменной $x$:

$$\mathrm{Info}(T) = -\sum_{j=1}^{k} \mathrm{freq}\left(c_r, T\right) / \left(|T| - U\right) \cdot \log_2 \left(\mathrm{freq}\left(c_r, T\right) / \left(|T| - U\right)\right);$$

$$\mathrm{Info}_{x_h}(T) = \sum_{i=1}^{m} |T_i| / \left(|T| - U\right) \cdot \mathrm{Info}\left(T_i\right).$$

В этом случае при подсчете $\text{freq}(c_r, T)$ учитываются только объекты с существующими значениями переменной $x$.

Тогда критерий можно переписать:

$$\text{Gain}(x) = (|T| - U)/|T| \cdot (\text{Info}(T) - \text{Info}_x(T)).$$

Подобным образом изменяется и критерий gain ratio. Если проверка имеет $n$ выходных значений, то критерий gain ratio считается так же, как и в случае, когда исходное множество разделено на $n+1$ подмножеств.

Пусть теперь проверка $x_h$ с выходными значениями $c_{h1}$, $c_{h2}$, ..., $c_{hm}$ выбрана на основе модифицированного критерия.

Предстоит решить, что делать с пропущенными данными. Если объект из множества $T$ с известным выходом $c_{hi}$ ассоциирован с подмножеством $T_i$, вероятность того, что пример из множества $T_i$, равна 1. Пусть тогда каждый объект из подмножества $T_i$ имеет вес, указывающий на вероятность того, что объект принадлежит $T_i$. Если объект имеет значение по переменной $x$, тогда вес равен 1, в противном случае объект ассоциируется со всеми множествами $T_1$, $T_2$, ..., $T_m$ с соответствующими весами:

$$|T_1|/(|T| - U),\ |T_2|/(|T| - U),\ ...,\ |T_m|/(|T| - U).$$

Легко убедиться, что:

$$\sum_{i=1}^{m} |T_1|/(|T| - U) = 1.$$

Вкратце этот подход можно сформулировать таким образом: предполагается, что пропущенные значения по переменной вероятностно распределены пропорционально частоте появления существующих значений.

## 5.4.2. Алгоритм покрытия

Рассмотренные ранее методы построения деревьев работают сверху вниз, разбивая на каждом шаге всю обучающую выборку на подмножества. Целью такого разбиения является получение подмножеств, соответствующих всем классам.

Альтернативой подходу "разделяй и властвуй" является подход, который заключается в построении деревьев решений для каждого класса по отдельности. Он называется алгоритмом покрытия, т. к. на каждом этапе генерируется проверка узла дерева, который покрывает несколько объектов обучающей выборки.

Идею алгоритма можно представить графически (рис. 5.4). При наличии у объектов только двух переменных их можно представить в виде точек двумерного пространства. Объекты, относящиеся к разным классам, отмечаются знаками "+" и "–". Как видно из рисунка, при разбиении множества на подмножества строится дерево, покрывающее только объекты выбранного класса.

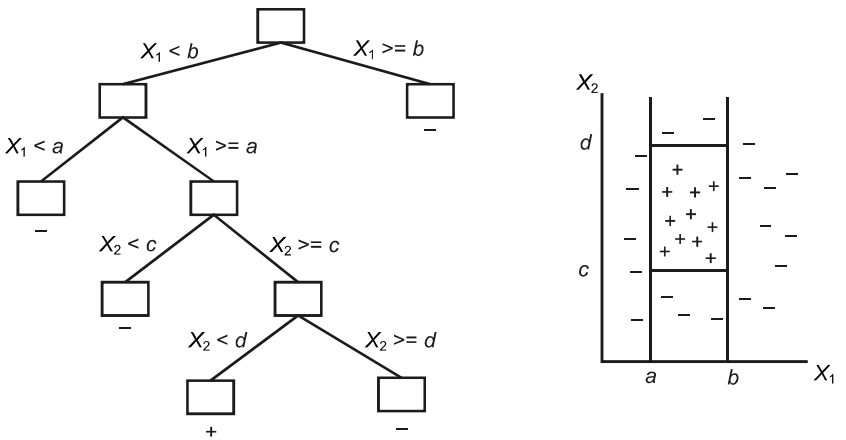



**Рис. 5.4.** Геометрическая интерпретация идеи алгоритма покрытия

Для построения правил с помощью данного алгоритма в обучающей выборке должны присутствовать всевозможные комбинации значений независимых переменных. Например, данные, позволяющие рекомендовать тип контактных линз, представлены в табл. 5.3.

**Таблица 5.3**

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Юный | Близорукость | Нет | Пониженный | Нет |
| Юный | Близорукость | Нет | Нормальный | Мягкие |
| Юный | Близорукость | Да | Пониженный | Нет |
| Юный | Близорукость | Да | Нормальный | Жесткие |
| Юный | Дальнозоркость | Нет | Пониженный | Нет |
| Юный | Дальнозоркость | Нет | Нормальный | Мягкие |
| Юный | Дальнозоркость | Да | Пониженный | Нет |
| Юный | Дальнозоркость | Да | Нормальный | Жесткие |

**Таблица 5.3** *(окончание)*

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Пожилой | Близорукость | Нет | Пониженный | Нет |
| Пожилой | Близорукость | Нет | Нормальный | Мягкие |
| Пожилой | Близорукость | Да | Пониженный | Нет |
| Пожилой | Близорукость | Да | Нормальный | Жесткие |
| Пожилой | Дальнозоркость | Нет | Пониженный | Нет |
| Пожилой | Дальнозоркость | Нет | Нормальный | Мягкие |
| Пожилой | Дальнозоркость | Да | Пониженный | Нет |
| Пожилой | Дальнозоркость | Да | Нормальный | Нет |
| Старческий | Близорукость | Нет | Пониженный | Нет |
| Старческий | Близорукость | Нет | Нормальный | Нет |
| Старческий | Близорукость | Да | Пониженный | Нет |
| Старческий | Близорукость | Да | Нормальный | Жесткие |
| Старческий | Дальнозоркость | Нет | Пониженный | Нет |
| Старческий | Дальнозоркость | Нет | Нормальный | Мягкие |
| Старческий | Дальнозоркость | Да | Пониженный | Нет |
| Старческий | Дальнозоркость | Да | Нормальный | Нет |

На каждом шаге алгоритма выбирается значение переменной, которое разделяет все множество на два подмножества. Разделение должно выполняться так, чтобы все объекты класса, для которого строится дерево, принадлежали одному подмножеству. Такое разбиение производится до тех пор, пока не будет построено подмножество, содержащее только объекты одного класса.

Для выбора независимой переменной и ее значения, которое разделяет множество, выполняются следующие действия:

1. Из построенного на предыдущем этапе подмножества (для первого этапа это вся обучающая выборка), включающего объекты, относящиеся к выбранному классу для каждой независимой переменной, выбираются все значения, встречающиеся в этом подмножестве.
2. Для каждого значения каждой переменной подсчитывается количество объектов, удовлетворяющих этому условию и относящихся к выбранному классу.

3. Выбираются условия, покрывающие наибольшее количество объектов выбранного класса.
4. Выбранное условие является условием разбиения подмножества на два новых.

После построения дерева для одного класса таким же образом строятся деревья для других классов.

Приведем пример для данных, представленных в табл. 5.3. Предположим, что нужно построить правило для определения условий, при которых необходимо рекомендовать жесткие линзы:

```
если (?) то рекомендация = жесткие
```

Выполним оценку каждой независимой переменной и всех их возможных значений:

```
возраст = юный — 2/8;
возраст = пожилой — 1/8;
возраст = старческий — 1/8;
предписание = близорукость — 3/12;
предписание = дальнозоркость — 1/12;
астигматизм = нет — 0/12;
астигматизм = да — 4/12;
степень износа = низкая — 0/12;
степень износа = нормальная — 4/12.
```

Выбираем переменную и значение с максимальной оценкой `астигматизм = да`. Таким образом, получаем уточненное правило следующего вида:

```
если (астигматизм = да и ?) то рекомендация = жесткие.
```

Данное правило образует подмножество, в которое входят все объекты, относящиеся к классу `жесткие`. Кроме них в него входят и другие объекты, следовательно, правило должно уточняться (табл. 5.4).

**Таблица 5.4**

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Юный | Близорукость | Да | Пониженный | Нет |
| Юный | Близорукость | Да | Нормальный | Жесткие |
| Юный | Дальнозоркость | Да | Пониженный | Нет |
| Юный | Дальнозоркость | Да | Нормальный | Жесткие |
| Пожилой | Близорукость | Да | Пониженный | Нет |
| Пожилой | Близорукость | Да | Нормальный | Жесткие |
| Пожилой | Дальнозоркость | Да | Пониженный | Нет |

**Таблица 5.4** *(окончание)*

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Пожилой | Дальнозоркость | Да | Нормальный | Нет |
| Старческий | Близорукость | Да | Пониженный | Нет |
| Старческий | Близорукость | Да | Нормальный | Жесткие |
| Старческий | Дальнозоркость | Да | Пониженный | Нет |
| Старческий | Дальнозоркость | Да | Нормальный | Нет |

Выполним повторную оценку для оставшихся независимых переменных и их значений, но уже на новом множестве:

```
возраст = юный — 2/4;
возраст = пожилой — 1/4;
возраст = старческий — 1/4;
предписание = близорукость —3/6;
предписание = дальнозоркость — 1/6;
степень износа = низкая —0/6;
степень износа = нормальная —4/6.
```

После уточнения получим правило и множество, представленное в табл. 5.5:

```
если (астигматизм = да и степень износа = нормальная)
     то рекомендация = жесткие.
```

**Таблица 5.5**

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Юный | Близорукость | Да | Нормальный | Жесткие |
| Юный | Дальнозоркость | Да | Нормальный | Жесткие |
| Пожилой | Близорукость | Да | Нормальный | Жесткие |
| Пожилой | Дальнозоркость | Да | Нормальный | Нет |
| Старческий | Близорукость | Да | Нормальный | Жесткие |
| Старческий | Дальнозоркость | Да | Нормальный | Нет |

Так как в полученном множестве все еще остаются объекты, не относящиеся к классу `жесткий`, то необходимо выполнить уточнение:

```
возраст = юный — 2/2;
возраст = пожилой —1/2;
```

```
возраст = старческий —1/2;
предписание = близорукость —3/3;
предписание = дальнозоркость — 1/3.
```

Очевидно, что уточненное правило будет иметь следующий вид:

```
если (астигматизм = да и степень износа = нормальная и предписание =
близорукость) то рекомендация = жесткие.
```

Однако в полученном подмножестве отсутствует один из объектов, относящихся к классу `жесткие`, поэтому необходимо решить, какое из последних двух правил более приемлемо для аналитика.

# 5.5. Методы построения математических функций

## 5.5.1. Общий вид

Методы, рассмотренные для правил и деревьев решений, работают наиболее естественно с категориальными переменными. Их можно адаптировать для работы с числовыми переменными, однако существуют методы, которые наиболее естественно работают с ними.

При построении математической функции классификации или регрессии основная задача сводится к выбору наилучшей функции из всего множества вариантов. Дело в том, что может существовать множество функций, одинаково классифицирующих одну и ту же обучающую выборку. Данная проблема проиллюстрирована на рис. 5.5.

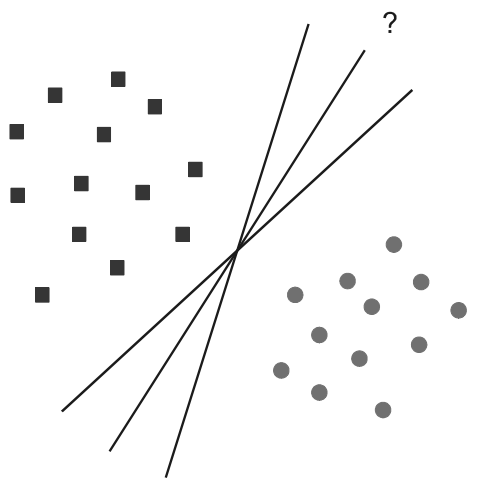



**Рис. 5.5.** Варианты линейного разделения обучающей выборки

Каждая из трех линий успешно разделяет все точки на два класса (представленные на рисунке квадратами и кружками), однако модель должна быть представлена одной функцией, которая наилучшим образом решит задачу для новых объектов.

В результате задачу построения функции классификации и регрессии в простейшей форме можно формально описать как задачу выбора функции с минимальной степенью ошибки:

$$\min_{f \in F} R(f) = \min_{f \in F} \frac{1}{m} \sum_{i=1}^{m} c\left(y_i, f\left(x_i\right)\right), \tag{5.1}$$

где $F$ — множество всех возможных функций; $c\left(y_i, f(x_i)\right)$ — функция потерь (loss function), в которой $f(x_i)$ — значение зависимой переменной, найденное с помощью функции $f$ для вектора $x_i \in T$, а $y_i$ — ее точное (известное) значение.

Следует отметить, что функция потерь принимает неотрицательные значения. Это означает, что невозможно получить "вознаграждение" за очень хорошее предсказание. Если выбранная функция потерь все же принимает отрицательные значения, то это легко исправить, введя положительный сдвиг (возможно, зависимый от $x$). Такими же простыми средствами можно добиться нулевых потерь при абсолютно точном предсказании $f(x) = y$. Преимущества подобного ограничения функции потерь заключаются в том, что всегда известен минимум и известно, что он достижим (по крайней мере, для данной пары $x, y$).

Для задач классификации и регрессии такие функции имеют разный вид. Так, в случае бинарной классификации (принадлежности объекта к одному из двух классов; первый класс далее обозначается через +1, а второй класс — через –1) простейшая функция потерь (называемая "0-1 loss" в англоязычной литературе) принимает значение 1 в случае неправильного предсказания и 0 в противном случае:

$$c\left(x, y, f(x)\right) = \begin{cases} 0, & y = f(x); \\ 1, & y \neq f(x). \end{cases}$$

Здесь не учитывается ни тип ошибки ($f(x) = 1$, $y = -1$ — положительная ошибка, $f(x) = -1$, $y = 1$ — отрицательная ошибка), ни ее величина.

Небольшое изменение позволяет учесть характер ошибки:

$$c\left(x, y, f(x)\right) = \begin{cases} 0, & y = f(x); \\ c'\left(x, y, f(x)\right), & y \neq f(x). \end{cases}$$

Здесь $c'\big(x, y, f(x)\big)$ может учитывать многие параметры классифицируемого объекта и характер ошибки.

Ситуация усложняется в случае классификации с количеством классов более двух. Каждый тип ошибки классификации в общем случае вносит свой тип потерь таким образом, что получается матрица размера $k \times k$ (где $k$ — число классов).

При оценке величин, принимающих вещественные значения, целесообразно использовать разность $f(x) - y$ для оценки качества классификации. Эта разность в случае регрессии имеет вполне определенный смысл (например, размер финансовых потерь при неправильной оценке стоимости финансового инструмента на рынке ценных бумаг). Учитывая условие независимости от положения, функция потерь будет иметь вид:

$$c\big(x, y, f(x)\big) = c'\big(f(x) - y\big).$$

Чаще всего применяется минимизация квадратов разностей $f(x) - y$. Этот вариант соответствует наличию аддитивного нормально распределенного шума, влияющего на результаты наблюдений $y_i$. Соответствующим образом минимизируем:

$$c\big(x, y, f(x)\big) = \big(f(x) - y\big)^2. \tag{5.2}$$

## 5.5.2. Линейные методы. Метод наименьших квадратов

Различают два вида функций: линейные и нелинейные. В первом случае функции множества *F* имеют вид:

$$y = \omega_0 + \omega_1 x^1 + \omega_2 x^2 + \ldots + \omega_d x^d = \omega_0 + \sum_{j=1}^{d} \omega_j x^j,$$

где $\omega_0$, $\omega_1$, ... , $\omega_d$ — коэффициенты при независимых переменных.

Задача заключается в отыскании таких коэффициентов $\omega$, чтобы удовлетворить условие (5.1). Например, при решении задачи регрессии коэффициенты $\omega$ можно вычислить, используя квадратичную функцию потерь (5.2) и множество линейных функций $F$:

$$F := \left\{ f \,\middle|\, f(x) = \sum_{i=1}^{n} \omega_i f_i(x), \quad \omega_i \in \Re \right\},$$

где $f_i : X \to \Re$.

Необходимо найти решение следующей задачи:

$$\min_{f \in F} R(f) = \min_{f \in \Re^d} \frac{1}{m} \sum_{i=1}^{m} \left( y_i - \sum_{j=1}^{d} \omega_j f_j(x_i) \right)^2.$$

Вычисляя производную $R(f)$ по $\omega$ и вводя обозначение $Y_{ij} := f_j(x_i)$, получаем, что минимум достижим при условии:

$$Y^T y = Y^T Y \omega.$$

Решением этого выражения будет:

$$\omega = (Y^T Y)^{-1} Y^T y.$$

Откуда и получаются искомые коэффициенты $\omega$. Рассмотренный пример иллюстрирует поиск оптимальной функции $f$ методом наименьших квадратов.

## 5.5.3. Нелинейные методы

Нелинейные модели лучше классифицируют объекты, однако их построение более сложно. Задача также сводится к минимизации выражения (5.1). При этом множество $F$ содержит нелинейные функции.

В простейшем случае построение таких функций все-таки сводится к построению линейных моделей. Для этого исходное пространство объектов преобразуется к новому:

$$\hat{O}: X \to X'.$$

В новом пространстве строится линейная функция, которая в исходном пространстве является нелинейной. Для использования построенной функции выполняется обратное преобразование в исходное пространство (рис. 5.6).

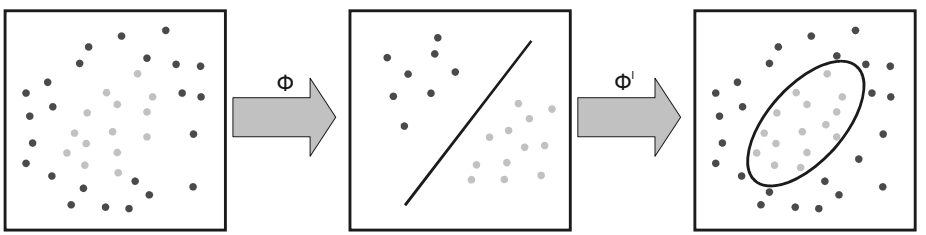



**Рис. 5.6.** Графическая интерпретация прямого и обратного преобразований из линейного пространства в нелинейное

Описанный подход имеет один существенный недостаток. Процесс преобразования достаточно сложен с точки зрения вычислений, причем вычислительная сложность растет с увеличением числа данных. Если учесть, что преобразование выполняется два раза (прямое и обратное), то количество вычислений возрастает. В связи с этим построение нелинейных моделей с таким подходом будет неэффективным. Альтернативой ему может служить метод Support Vector Machines (SVM), не выполняющий отдельных преобразований всех объектов, а учитывающий это в расчетах.

## 5.5.4. Support Vector Machines (SVM)

В 1974 г. вышла первая книга Вапника и Червоненкиса "Теория распознавания образов", положившая начало целой серии их работ в этой области. Предложенные авторами методы распознавания образов и статистическая теория обучения, лежащая в их основе, оказались весьма успешными и прочно вошли в арсенал методов Data Mining. Алгоритмы классификации и регрессии под общим названием SVM во многих случаях успешно заменили нейронные сети и в данное время применяются очень широко.

Идея метода основывается на предположении о том, что наилучшим способом разделения точек в $m$-мерном пространстве является $m-1$ плоскость (заданная параметризация $f(x)=0$), равноудаленная от точек, принадлежащих разным классам. Для двумерного пространства эту идею можно представить в виде, изображенном на рис. 5.7.

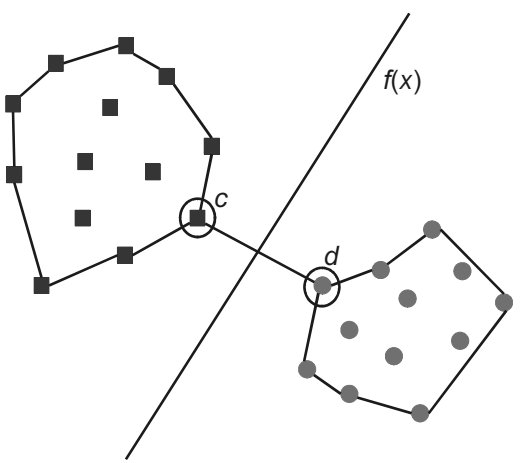



**Рис. 5.7.** Графическая интерпретация идеи метода SVM

Как можно заметить, для решения этой задачи достаточно провести плоскость, равноудаленную от ближайших друг к другу точек, относящихся к

разному классу. На рисунке такими точками являются точки $c$ и $d$. Данный метод интерпретирует объекты (и соответствующие им в пространстве точки) как векторы размера $m$. Другими словами, независимые переменные, характеризующие объекты, являются координатами векторов. Ближайшие друг к другу векторы, относящиеся к разным классам, называются векторами поддержки (support vectors).

Формально данную задачу можно описать как поиск функции, отвечающей следующим условиям:

$$\begin{cases} y_i(<\omega, x_i> + b) \geq 1 - \xi_i \ \forall i; \\ \xi_i \geq 0, \sum \xi_i \leq C \end{cases}$$

для некоторого конечного значения ошибки $\varepsilon \in \Re$.

Если $f(x)$ линейна, то ее можно записать в виде:

$$f(x) = <\omega, x> + b, \omega \in \Re^d, b \in \Re,$$

где $<\omega, x>$ — скалярное произведение векторов $\omega$ и $x$; $b$ — константа, заменяющая коэффициент $\omega_0$.

Введем понятие *плоскости функции* таким образом, что большему значению плоскости соответствует меньшее значение евклидовой нормы вектора $\omega$:

$$\| \omega \| = \sqrt{<\omega, \omega>}.$$

Тогда задачу нахождения функции $f(x)$ можно сформулировать следующим образом: минимизировать значение $\frac{1}{2}\| \omega \|^2$ при условии:

$$\begin{cases} y_i(<\omega, x_i> + b) \geq 1 - \xi_i \ \forall i; \\ \xi_i \geq 0, \sum \xi_i \leq C. \end{cases}$$

Решением данной задачи является функция вида:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i < x, x_i > + b, \tag{5.3}$$

где $\alpha_i$ — положительная константа, минимизирующая функцию Лагранжа

$$L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j < x_i, x_j >$$

и удовлетворяющая следующим условиям:

$$\begin{cases} \sum_{i=1}^{m} \alpha_i y_i = 0; \\ \alpha_i \in [0, C]. \end{cases}$$

Константа $C$ задает соотношение между плоскостью функции $f(x)$ и допустимым значением нарушения границы $\varepsilon$.

Несмотря на то, что рассмотрен случай с линейной функцией $f(x)$, метод SVM может быть использован и для построения нелинейных моделей. Для этого скалярное произведение двух векторов $(x_i, x)$ необходимо заменить на скалярное произведение преобразованных векторов:

$$k(x, x') := \langle \Phi(x), \Phi(x') \rangle.$$

Функция $k(x, y)$ называется ядром.

Тогда выражение (5.3) можно переписать в виде:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b.$$

Отличие от линейного варианта SVM здесь в том, что $\omega$ теперь находится не непосредственно, а с использованием преобразования $\Phi$. Необходимо также заметить, что при создании нелинейных моделей с использованием метода SVM не выполняется прямое, а затем обратное преобразование объектов из нелинейного в линейное пространство. Преобразование заложено в самой формуле расчета, что значительно снижает вычислительные затраты.

Вид преобразования, а точнее функция $k(x_i, x)$ может быть различного типа и выбирается в зависимости от структуры данных. В табл. 5.6 приведены основные виды функций классификации, применяемых в SVM-методе.

**Таблица 5.6**

| **Ядро** | **Название** |
|---|---|
| $k(x, y) = xy$ | Линейная |
| $k(x, y) = (\gamma\, xy + c_0)^d$ | Полиномиал степени $d$ |
| $k(x, y) = \exp(-\gamma \| x - y \|)$ | Базовая радиальная функция Гаусса |
| $k(x, y) = \tanh(\gamma\, xy + c_0)$ | Сигмоидальная |

К достоинствам метода SVM можно отнести следующие факторы:

- теоретическая и практическая обоснованность метода;
- общий подход ко многим задачам. Используя разные функции $k(x, y)$, можно получить решения для разных задач;
- устойчивые решения, нет проблем с локальными минимумами;
- не подвержен проблеме overfitting;
- работает в любом количестве измерений.

Недостатками метода являются:

- невысокая производительность по сравнению с более простыми методами;
- отсутствие общих рекомендаций по подбору параметров и выбору ядра;
- побочные эффекты нелинейных преобразований;
- сложности с интерпретацией результата;
- работает с небольшим количеством векторов.

## 5.5.5. Регуляризационные сети (Regularization Networks)

В *разд. 5.1* были описали простейшие проблемы классификации и регрессии. Однако этот подход часто страдает от излишней подгонки, так как методы решения, описанные в условии (5.1), пытаются корректно классифицировать *все* векторы данных — независимо от степени сложности функции $f$. Кроме того, описанные подходы приводят к множеству решений (как показано на рис. 5.5) и возникает проблема *некорректности*, которая также численно нестабильна.

Чтобы преодолеть эту проблему, был применен подход регуляризации типа Тихонова, в результате чего было получено обобщение (5.1), названное *регуляризационными сетями*. Эта работа, обобщая такие широко известные подходы, как гребневая регрессия (Ridge Regression) и сглаживающие сплайны (Smoothing Splines), была главным образом выведена Ф. Гироси (F. Girosi) в 90-х годах [150]. Гироси показал, что многие современные методы классификации и регрессии (включая SVM) могут быть интерпретированы как методы решения для следующей проблемы регуляризации:

$$\min_{f \in F} R(f) = \min_{f \in F} \frac{1}{m} \sum_{i=1}^{m} c\left(y_i, f\left(x_i\right)\right) + \lambda \varphi(f), \tag{5.4}$$

где $\varphi(f)$ — сглаживающий оператор, $\varphi(f) = \|Pf\|^2$, $e.g.\ Pf = \nabla f$.

$\lambda$ — параметр регуляризации.

Здесь первый терм $\frac{1}{m}\sum_{i=1}^{m} c\left(y_i, f\left(x_i\right)\right)$ гарантирует хорошую адаптацию классификационной функции тренировочным данным, тогда как второй терм $\varphi(f)$ удерживает классификационную функцию настолько гладкой, насколько возможно и гарантирует хорошую обобщающую способность. Таким образом, первый терм склоняется к "переподгонке", а второй — к "недоподгонке". Через параметр регуляризации $\lambda$ между этими двумя термами создается компромисс. Существуют разные подходы для нахождения оптимального значения $\lambda$. Обычно подходящий параметр регуляризации выбирается посредством оценки подмножества или перекрестной проверкой.

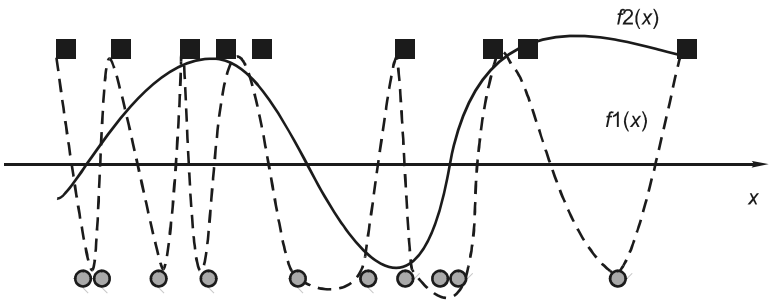



**Рис. 5.8.** Графическая интерпретация регуляризационных сетей

Случай одномерного пространства и двухразрядной задачи проиллюстрирован на рис. 5.8.

Предположим, что для сглаживания мы используем дифференцирующий оператор первого порядка $Pf = \frac{df}{dx}$. Тогда функция $f1$ представляет слабый параметр регуляризации $\lambda$, и здесь преобладает точная классификация, тогда как $f2$ представляет высокий $\lambda$, ведущий к преобладанию сглаживающего терма. Сравнивая две функции, можно сказать, что $f1$ выполняется лучше на тренировочных данных, но имеет более сложную структуру, чем $f$.

Гироси доказал, что проблема SVM *(см. разд. 5.5.4)* может быть переформулирована как регуляризационная сеть (5.4) при использовании комплексных операторов регуляризации $P$. Заметьте, что в случае SVM параметр $C$ играет роль параметра регуляризации. Многие другие схемы аппроксимации, такие как аддитивные модели, функции с радиальным базисом и некоторые типы нейронных сетей, могут быть выведены выбором специального оператора регуляризации. Таким образом, регуляризационные сети являются важным инструментом в изучении различных классификационных и регрессивных методов.

## 5.5.6. Дискретизации и редкие сетки

Все нелинейные классификационные и регрессионные методы для изученных до настоящего момента функций решают свои операторные уравнения (5.1) или (5.4) точно. Цена этого — большое количество вычислений, по меньшей мере, с квадратичной размерностью и количеством векторов данных $m$. Как результат, нелинейные классификационные и регрессионные методы могут быть использованы только для относительно небольших наборов данных.

Для преодоления этой проблемы все больше и больше изучаются методы приблизительного решения (5.4), основанные на технике дискретизации. Для интегральных и дифференциальных уравнений этот способ применяется многие годы, и в основном для аппроксимации функции используется *метод конечных элементов* (МКЭ) (Finite Element Methods (FEM)).

Суть метода — определить конечномерное подпространство $F_n \subset F$ изначальной функции пространства $F$, и затем разрешить задачу в подпространстве $F_n$. Подпространство строится путем задания сетки над пространством атрибутов $\Re^d$ и привязыванием базисных функций к узлам сетки. Наиболее распространенной базисной функцией является линейная (или Куранта (Courant)) базисная функция, изображенная на рис. 5.9 для одномерного пространства атрибутов.

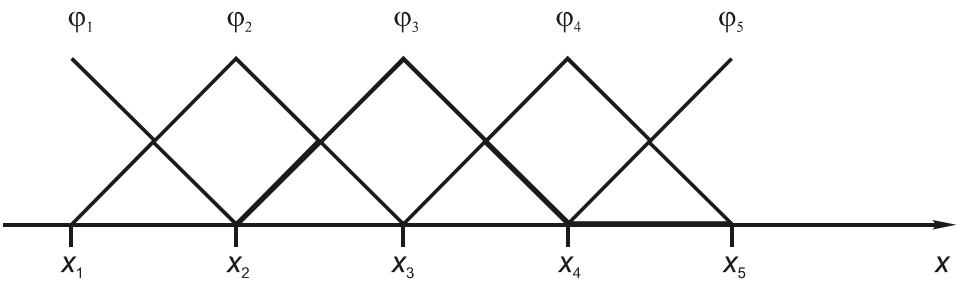



**Рис. 5.9.** Базис МКЭ (FEM) с линейными базисными функциями на интервале $[x_1, x_5]$; функция $x_5$ $x_2$ $\phi_3$ изображена жирной линией

Важным свойством базисных функций КЭ является то, что каждая функция имеет комплексное основание (основание — это область определения с ненулевыми значениями функции). В примере $\phi_3$ за пределами интервала $[x_2, x_4]$ всегда равна нулю. Это свойство, объясняющее термин "конечный элемент", очень важно, поскольку оно позволяет оперировать большим количеством узлов.

Используя базисные функции $\{\phi_j\}_{j=1}^{n}$ из пространства функций $F_n$, мы можем представить $f$ как

$$f(x) = \sum_{j=1}^{n} \alpha_j \phi_j(x). \tag{5.5}$$

После подстановки (5.5) в (5.4) с функцией потерь (5.2) и дифференцирования по параметру $\alpha_j$ получаем систему уравнений

$$B^T y = \left(B^T B + \lambda G\right)\alpha. \tag{5.6}$$

Здесь $Y$ имеет вхождения $B_{ij} := \phi_j(x_i)$ и $G_{jk} := m \cdot < P\phi_j, P\phi_k >$. Это аналогично системе уравнений линейных методов из *разд. 5.5.2*, кроме того, что здесь добавлен регуляризационный элемент. Теперь, так как мы имеем редкий базис $\{\phi_j\}_{j=1}^{n}$ в МКЭ, матрицы $B$ и $G$ (для большинства дифференцирующих операторов $P$) также являются редкими, и вычислительные затраты на решение системы уравнений будут линейными в $m$. Таким образом, мы имеем возможность обработать почти неограниченное количество векторов данных.

Однако все подходы к проблеме, полученные в этом направлении до сих пор, были разрушены тем фактом, что количество точек сетки растет экспоненциально с увеличением размерности пространства атрибутов, и, таким образом, эти методы могут быть применены только для размерностей 3 и 4.

К счастью, уже существуют первые многомерные методы дискретизации. Наиболее известен метод *редких сеток* (Sparse Grids). Основанная на подходе русского математика Смоляка (Smolyak) [151], техника редких сеток была разработана в университетах Мюнхена и Бонна в начале 90-х гг. и впервые позволила аппроксимацию многомерных сглаживающих функций. Метод редких сеток основан на особой иерархической комбинации анизотропных сеток (рис. 5.10).

Таким образом, идея состоит в том, чтобы построить специальные сетки для функций с иерархическим базисом (так называемых вейвлетов), требующие меньше узлов, чем сетки МКЭ, но имеющие сопоставимые аппроксимирующие свойства. Вот почему эти сетки называются "редкие". На самом деле, редкие сетки позволяют обрабатывать размерности порядка 20—25, а в адаптивных вариантах даже больше. В настоящее время редкие сетки используются как основной инструмент в решении многомерных интегральных и дифференциальных уравнений.

Применяя редкие сетки, мы можем решить (5.6) для любого количества векторов данных $m$ для умеренной разрядности $d$ [149]. Другое преимущество

редких сеток в возможности иметь дело с гораздо более широкими классами операторных уравнений, чем одни регуляризационные сети (5.4). Тем не менее, редкие сетки развиты в теории и распространены на практике. Текущие исследования сфокусированы на разработке адаптивных к размерности техниках, которые смогут работать с более чем 100 измерениями [148].

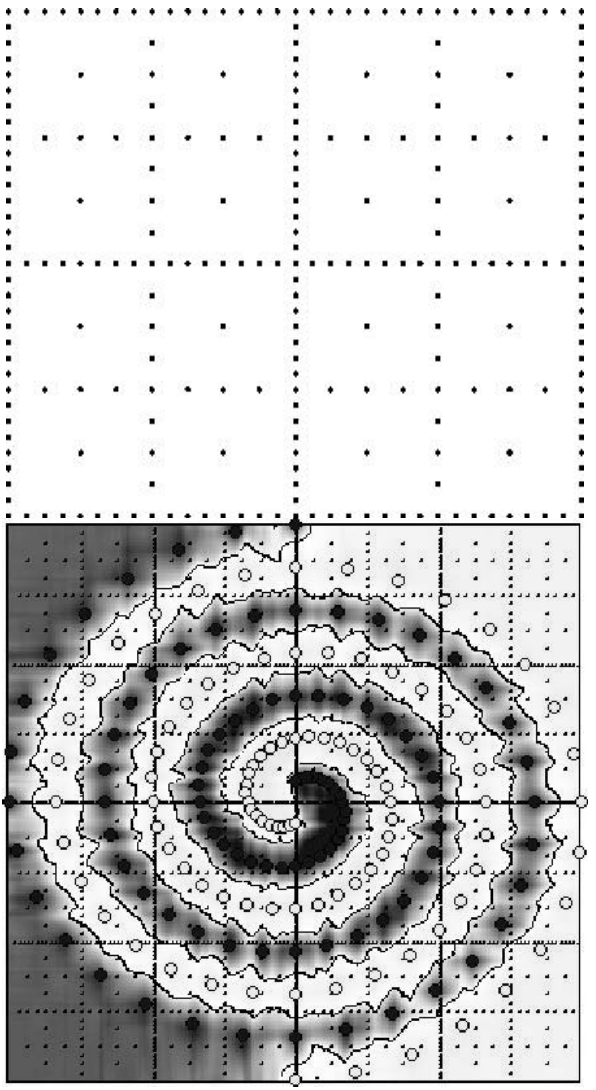


**Рис. 5.10.** Пример структуры и классификационной функции редких сеток

# 5.6. Прогнозирование временных рядов

## 5.6.1. Постановка задачи

Частным случаем задачи классификации является задача прогнозирования временных рядов.

*Временным рядом* называется последовательность событий, упорядоченных по времени их наблюдения. События обычно фиксируются через равные интервалы времени $T$ и представляются в виде последовательности:

$$\{e_1, e_2, ..., e_i, ..., e_n\},$$

где $e_i$ — событие в момент времени $t_i$, $n$ — общее количество событий.

Событие может характеризоваться несколькими атрибутами:

$$e_i = \left\{x_1^i, x_1^i, ..., x_j^i, ..., x_m^i\right\},$$

где $x_j^i$ — $j$-й атрибут, характеризующий событие в момент времени $t_i$. Если один из этих атрибутов может быть определен из значений других атрибутов в текущий или предыдущие моменты времени, то такой атрибут является *зависимым* (или *целевым*). Атрибуты, через которые можно выразить зависимый атрибут, называются *независимыми*.

Задачу построения прогноза по временному ряду можно сформулировать следующим образом: пусть дан временной ряд $\{e_1, e_2, ..., e_i, ..., e_n\}$, требуется на его основании определить значение $e_{n-k}$ при $k > 0$.

## 5.6.2. Методы прогнозирования временных рядов

Прогнозирование временных рядов осуществляется в три этапа:

1. Построение модели, характеризующей временной ряд. Для этого применяются различные методы статистики и классификации.
2. Оценка построенной модели. Для оценки модели имеющиеся данные разбиваются на два множества: обучающую и тестовую. Построение модели выполняется на обучающем множестве, а затем с ее помощью строят прогноз на тестовом множестве. Спрогнозированные результаты сравнивают с реальными данными и по степени ошибки оценивают модель.
3. Если построенная на первом этапе модель получила удовлетворительную оценку, то ее можно использовать для прогноза будущих событий.

Данная задача может решаться как методами математической статистики, такими как экстраполяция, экспоненциальное сглаживание и др., так и методами Data Mining, например методом скользящего окна.

Метод экстраполяции предполагает построение по имеющимся данным функции, выражающей зависимость искомых значений от времени:

$$e_{n-k} = f(t).$$

Вид функции $f$ может быть как линейный, так и нелинейный. В общем виде ее можно записать следующим образом:

$$f(t) = \sum \alpha_i t_i^p,$$

где $\alpha_i$ — искомые коэффициенты, подбираемые так, чтобы построенная функция имела бы минимальную ошибку прогноза.

Метод экспоненциального сглаживания строит адаптивные модели прогнозирования, т. е. модели, способные приспосабливать свою структуру и параметры к изменению условий. Общая идея построения таких моделей может быть представлена следующим образом:

1. По нескольким первым уровням ряда оцениваются значения параметров модели.
2. По имеющейся модели строится прогноз на один шаг вперед, причем его отклонение от фактических уровней ряда расценивается как ошибка прогнозирования, которая учитывается в соответствии со схемой корректировки модели.
3. Далее по модели со скорректированными параметрами рассчитывается прогнозная оценка на следующий момент времени и т. д.

Таким образом, модель постоянно учитывает новую информацию и к концу периода обучения отражает тенденцию развития процесса, существующую на данный момент. Вид такой модели можно представить на примере модели Холта:

$$S_k = \alpha e_k + (1-\alpha) S_{k-1},$$

где $S_k$ — предсказанное значение, $\alpha$ — параметр, определяющий сглаживание ряда. Если $\alpha$ равно 1, то предыдущие наблюдения полностью игнорируются. Если $\alpha$ равно 0, то игнорируются текущие наблюдения. Значения $\alpha$ между 0 и 1 дают промежуточные результаты.

Основной идеей метода скользящего окна является гипотеза о том, что существует некий закон, по которому можно определить значение очередного члена ряда как функцию от нескольких предыдущих членов. Обычно из каких-то соображений фиксируют число $k$ и предполагают, что только $k$

предшествующих членов влияют на дальнейшее поведение ряда, а зависимостью от остальных пренебрегают, т. е.:

$$e_{n+1} = f(e_n, e_{n-1}, ..., e_{n-k}).$$

При этом говорят об "окне" размером $k$, в пределах которого рассматривается ряд. Для нахождения функции $f$ временной ряд нарезается на множество окон (каждое из которых сдвигается на один элемент). На полученном множестве выполняется поиск искомой функции.

Необходимо заметить, что если функция $f$ используется для предсказания численных значений, то говорят о задаче регрессии. В случае категориальных значений ряда мы имеем дело с классической задачей классификации. Решения этих задач получаются методами Data Mining. Например, задача регрессии может быть решена методом SVM, а задача классификации — методами построения деревьев решений. При использовании метода деревьев решений полученные результаты легко представить в виде правил <если—то>. В этом случае в условной части (если) указываются уже прошедшие события, а в заключительной (то) — предсказываемые.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- В задаче классификации и регрессии требуется определить значение зависимой переменной объекта на основании значений других переменных, характеризующих его.
- Наиболее распространенные модели, отражающие результаты классификации, — это классификационные правила, деревья решений, математические (линейные и нелинейные) функции.
- Классификационные правила состоят из двух частей: условия и заключения. Они могут быть построены, например, такими методами, как 1R и Naive Bayes.
- Деревья решений — это способ представления правил в иерархической последовательной структуре. Они строятся такими алгоритмами, как ID3, C4.5 и алгоритмом покрытия.
- Математическая функция выражает отношение зависимой переменной от независимых, строится статистическими методами, а также методом SVM.
- Идея 1R-алгоритма заключается в формировании для каждого возможного значения каждой независимой переменной правила, которое классифицирует объекты из обучающей выборки.

- Идея метода Naive Bayes заключается в расчете условной вероятности принадлежности объекта к классу при равенстве его независимых переменных определенным значениям.
- Алгоритмы ID3 и C4.5 основаны на методе "разделяй и властвуй", суть которого заключается в рекурсивном разбиении множества объектов из обучающей выборки на подмножества, содержащие объекты, относящиеся к одинаковым классам.
- Идея алгоритма покрытия заключается в построении деревьев решений для каждого класса по отдельности.
- Идея метода SVM основывается на предположении, что наилучшим способом разделения точек в $m$-мерном пространстве является $m-1$ плоскость (заданная функцией $f(x)$), равноудаленная от точек, принадлежащих разным классам.
- Для преодоления проблем, присущих алгоритмам классификации и регрессии, был предложен подход регуляризации типа Тихонова, в результате чего было получено обобщение, названное регуляризационными сетями.
- Для преодоления проблемы работы с небольшим количеством векторов применяются методы приблизительного решения, основанные на технике дискретизации.
- Временным рядом называется последовательность событий, упорядоченных по времени их наблюдения, при этом предсказание заключается в определении будущих событий по уже произошедшим событиям методами математической статистики или методами классификации Data Mining.

# ГЛАВА 6

# Поиск ассоциативных правил

## 6.1. Постановка задачи

### 6.1.1. Формальная постановка задачи

Одной из наиболее распространенных задач анализа данных является определение часто встречающихся наборов объектов в большом множестве наборов.

Опишем эту задачу в обобщенном виде. Для этого обозначим объекты, составляющие исследуемые наборы (itemsets), следующим множеством:

$$I = \{i_1, i_2, ..., i_j, ..., i_n\},$$

где $i_j$ — объекты, входящие в анализируемые наборы; $n$ — общее количество объектов.

В сфере торговли, например, такими объектами являются товары, представленные в прайс-листе (табл. 6.1).

**Таблица 6.1**

| Идентификатор | Наименование товара | Цена |
|---|---|---|
| 0 | Шоколад | 30.00 |
| 1 | Чипсы | 12.00 |
| 2 | Кокосы | 10.00 |
| 3 | Вода | 4.00 |
| 4 | Пиво | 14.00 |
| 5 | Орехи | 15.00 |

Они соответствуют следующему множеству объектов:

$$I = \{\text{шоколад, чипсы, кокосы, вода, пиво, орехи}\}.$$

Наборы объектов из множества $I$, хранящиеся в БД и подвергаемые анализу, называются *транзакциями*. Опишем транзакцию как подмножество множества $I$:

$$T = \{i_j \mid i_i \in I\}.$$

Такие транзакции в магазине соответствуют наборам товаров, покупаемых потребителем и сохраняемых в БД в виде товарного чека или накладной. В них перечисляются приобретаемые покупателем товары, их цена, количество и др. Например, следующие транзакции соответствуют покупкам, совершаемым потребителями в супермаркете:

$$T_1 = \{\text{чипсы, вода, пиво}\};$$
$$T_2 = \{\text{кокосы, вода, орехи}\}.$$

Набор транзакций, информация о которых доступна для анализа, обозначим следующим множеством:

$$D = \{T_1, T_2, ..., T_r, ..., T_m\},$$

где $m$ — количество доступных для анализа транзакций.

Например, в магазине таким множеством будет:

$$D = \{\{\text{чипсы, вода, пиво}\},$$
$$\{\text{кокосы, вода, орехи}\},$$
$$\{\text{орехи, кокосы, чипсы, кокосы, вода}\},$$
$$\{\text{кокосы, орехи, кокосы}\}\}.$$

Для использования методов Data Mining множество $D$ может быть представлено в виде табл. 6.2.

**Таблица 6.2**

| Номер транзакции | Номер товара | Наименование товара | Цена |
|---|---|---|---|
| 0 | 1 | Чипсы | 12,00 |
| 0 | 3 | Вода | 4,00 |
| 0 | 4 | Пиво | 14,00 |
| 1 | 2 | Кокосы | 10,00 |
| 1 | 3 | Вода | 4,00 |
| 1 | 5 | Орехи | 15,00 |

**Таблица 6.2** *(окончание)*

| Номер транзакции | Номер товара | Наименование товара | Цена |
|---|---|---|---|
| 2 | 5 | Орехи | 15,00 |
| 2 | 2 | Кокосы | 10,00 |
| 2 | 1 | Чипсы | 12,00 |
| 2 | 2 | Кокосы | 10,00 |
| 2 | 3 | Вода | 4,00 |
| 3 | 2 | Кокосы | 10,00 |
| 3 | 5 | Орехи | 15,00 |
| 3 | 2 | Кокосы | 10,00 |

Множество транзакций, в которые входит объект $i_j$, обозначим следующим образом:

$$D_{i_j} = \left\{T_r \mid i_j \in T_r;\ j = 1..n;\ r = 1..m\right\} \subseteq D.$$

В данном примере множеством транзакций, содержащих объект вода, является следующее множество:

$$D_{\text{вода}} = \{\{\text{чипсы, вода, пиво}\},$$
$$\{\text{кокосы, вода, орехи}\},$$
$$\{\text{орехи, кокосы, чипсы, кокосы, вода}\}\}.$$

Некоторый произвольный набор объектов (itemset) обозначим следующим образом:

$$F = \left\{i_j \mid i_j \in I;\ j = 1..n\right\}.$$

Например,

$$F = \{\text{кокосы, вода}\}.$$

Набор, состоящий из $k$ объектов, называется *k*-элементным набором (в данном примере это 2-элементный набор).

Множество транзакций, в которые входит набор $F$, обозначим следующим образом:

$$D_F = \left\{T_r \mid F \subseteq T_r;\ r = 1..m\right\} \subseteq D.$$

В данном примере:

$$D_{\{\text{кокосы, вода}\}} = \{\{\text{кокосы, вода, орехи}\},$$
$$\{\text{орехи, кокосы, чипсы, кокосы, вода}\}\}.$$

Отношение количества транзакций, в которое входит набор $F$, к общему количеству транзакций называется *поддержкой (support) набора* $F$ и обозначается $\mathrm{Supp}(F)$:

$$\mathrm{Supp}(F) = \frac{|D_F|}{|D|}.$$

Для набора {кокосы, вода} поддержка будет равна 0,5, т. к. данный набор входит в две транзакции (с номерами 1 и 2), а всего транзакций 4.

При поиске аналитик может указать минимальное значение поддержки интересующих его наборов $\mathrm{Supp}_{\min}$. Набор называется *частым* (large itemset), если значение его поддержки больше минимального значения поддержки, заданного пользователем:

$$\mathrm{Supp}(F) > \mathrm{Supp}_{\min}.$$

Таким образом, при поиске ассоциативных правил требуется найти множество всех частых наборов:

$$L = \{F \mid \mathrm{Supp}(F) > \mathrm{Supp}_{\min}\}.$$

В данном примере частыми наборами при $\mathrm{Supp}_{\min} = 0{,}5$ являются следующие:

{чипсы} $\mathrm{Supp}_{\min} = 0{,}5$;
{чипсы, вода} $\mathrm{Supp}_{\min} = 0{,}5$;
{кокосы} $\mathrm{Supp}_{\min} = 0{,}75$;
{кокосы, вода} $\mathrm{Supp}_{\min} = 0{,}5$;
{кокосы, вода, орехи} $\mathrm{Supp}_{\min} = 0{,}5$;
{кокосы, орехи} $\mathrm{Supp}_{\min} = 0{,}75$;
{вода} $\mathrm{Supp}_{\min} = 0{,}75$;
{вода, орехи} $\mathrm{Supp}_{\min} = 0{,}5$;
{орехи} $\mathrm{Supp}_{\min} = 0{,}75$.

## 6.1.2. Секвенциальный анализ

При анализе часто вызывает интерес последовательность происходящих событий. При обнаружении закономерностей в таких последовательностях можно с некоторой долей вероятности предсказывать появление событий в будущем, что позволяет принимать более правильные решения.

Последовательностью называется упорядоченное множество объектов. Для этого на множестве должно быть задано отношение порядка.

Тогда последовательность объектов можно описать в следующем виде:

$$S = \{..., i_p, ..., i_q\}, \text{ где } p < q.$$

Например, в случае с покупками в магазинах таким отношением порядка может выступать время покупок. Тогда последовательность

$$S = \{(\text{вода}, 02.03.2003), (\text{чипсы}, 05.03.2003), (\text{пиво}, 10.03.2003)\}$$

можно интерпретировать как покупки, совершаемые одним человеком в разное время (вначале была куплена вода, затем чипсы, а потом пиво).

Различают два вида последовательностей: с циклами и без циклов. В первом случае допускается вхождение в последовательность одного и того же объекта на разных позициях:

$$S = \{..., i_p, ..., i_q, ...\}, \text{ где } p < q, \ i_q = i_p.$$

Говорят, что транзакция $T$ содержит последовательность $S$, если $S \subseteq T$ и объекты, входящие в $S$, входят и в множество $T$ с сохранением отношения порядка. При этом допускается, что в множестве $T$ между объектами из последовательности $S$ могут находиться другие объекты.

*Поддержкой последовательности S* называется отношение количества транзакций, в которое входит последовательность $S$, к общему количеству транзакций. Последовательность является *частой*, если ее поддержка превышает минимальную поддержку, заданную пользователем:

$$\text{Supp}(S) > \text{Supp}_{\min}.$$

Задачей секвенциального анализа является поиск всех частых последовательностей:

$$L = \{S \mid \text{Supp}(S) > \text{Supp}_{\min}\}.$$

Основным отличием задачи секвенциального анализа от поиска ассоциативных правил является установление отношения порядка между объектами множества *I*. Данное отношение может быть определено разными способами. При анализе последовательности событий, происходящих во времени, объектами множества $I$ являются события, а отношение порядка соответствует хронологии их появления.

Например, при анализе последовательности покупок в супермаркете наборами являются покупки, совершаемые в разное время одними и теми же покупателями, а отношением порядка в них является хронология покупок:

$$D = \{\{(\text{вода}), (\text{пиво})\},$$
$$\{(\text{кокосы}, \text{вода}), (\text{пиво}), (\text{вода}, \text{шоколад}, \text{кокосы})\},$$
$$\{(\text{пиво}, \text{чипсы}, \text{вода}), (\text{пиво})\}\}.$$

Естественно, что в этом случае возникает проблема идентификации покупателей. На практике она решается введением дисконтных карт, имеющих уникальный идентификатор. Данное множество можно представить в виде табл. 6.3.

**Таблица 6.3**

| ID покупателя | Последовательность покупок |
|---|---|
| 0 | (Пиво), (вода) |
| 1 | (Кокосы, вода), (пиво), (вода, шоколад, кокосы) |
| 2 | (Пиво, чипсы, вода), (пиво) |

Интерпретировать такую последовательность можно следующим образом: покупатель с идентификатором 1 вначале купил кокосы и воду, затем купил пиво, в последний раз он купил воду, шоколад и кокосы.

Поддержка, например, последовательности {(пиво), (вода)} составит 2/3, т. к. она встречается у покупателей с идентификаторами 0 и 1. У последнего покупателя также встречается набор {(пиво), (вода)}, но не сохраняется последовательность (он купил вначале воду, а затем пиво).

Секвенциальный анализ актуален и для телекоммуникационных компаний. Основная проблема, для решения которой он используется, — это анализ данных об авариях на различных узлах телекоммуникационной сети. Информация о последовательности совершения аварий может помочь в обнаружении неполадок и предупреждении новых аварий. Данные для такого анализа могут быть представлены, например, в виде табл. 6.4.

**Таблица 6.4**

| Дата | Время | Источник ошибки | Код источника | Код ошибки | ... |
|---|---|---|---|---|---|
| 01.01.03 | 15:04:23 | Станция 1 | 1001 | *a* | ... |
| 01.01.03 | 16:45:46 | Станция 1 | 1001 | *f* | ... |
| 01.01.03 | 18:32:26 | Станция 4 | 1004 | *z* | ... |
| 01.01.03 | 20:07:11 | Станция 5 | 1005 | *h* | ... |
| 01.01.03 | 20:54:43 | Станция 1 | 1001 | *q* | ... |
| ... | ... | ... | ... | ... | ... |

В данной задаче объектами множества $I$ являются коды ошибок, возникающих в процессе работы телекоммуникационной сети. Последовательность $S_{\text{sid}}$ содержит сбои, происходящие на станции с идентификатором sid. Их можно представить в виде пар $(eid,t)$, где $eid$ — код ошибки, а $t$ — время, когда она произошла. Таким образом, последовательность сбоев на станции с идентификатором sid будет иметь следующий вид:

$$S_{\text{sid}} = \{(eid_1,t_1),(eid_2,t_2),...,(eid_n,t_n)\}.$$

Для данных, приведенных в табл. 6.4, транзакции будут следующие:

$T_{1001}$ = {(*a*, 15:04:23 01.01.03), (*f*, 16:45:46 01.01.03), (*q*, 20:54:43 01.01.03), ... };

$T_{1004}$ = {(*z*, 18:32:26 01.01.03), ... };

$T_{1005}$ = {(*h*, 20:07:11 01.01.03), ... }.

Отношение порядка в данном случае задается относительно времени появления ошибки (значения $t$).

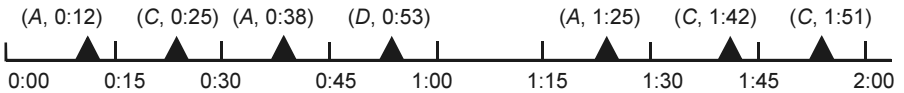



**Рис. 6.1.** Последовательность сбоев на телекоммуникационной станции

При анализе такой последовательности важным является определение интервала времени между сбоями. Оно позволяет предсказать момент и характер новых сбоев, а следовательно, предпринять профилактические меры. По этой причине при анализе данных интерес вызывает не просто последовательность событий, а сбои, которые происходят друг за другом. Например, на рис. 6.1 изображена временная шкала последовательности сбоев, происходящих на одной станции. При определении поддержки, например, для последовательности {*A*, *C*} учитываются только следующие наборы: {(*A*, 0:12), (*C*, 0:25)}, {(*A*, 0:38), (*C*, 1:42)}, {(*A*, 1:25), (*C*, 1:42)}. При этом не учитываются следующие последовательности: {(*A*, 0:12), (*C*, 1:42)}, {(*A*, 0:12), (*C*, 1:51)}, {(*A*, 0:38), (*C*, 1:51)} и {(*A*, 1:25), (*C*, 1:51)}, т. к. они не следуют непосредственно друг за другом.

## 6.1.3. Разновидности задачи поиска ассоциативных правил

Во многих прикладных областях объекты множества $I$ естественным образом объединяются в группы, которые в свою очередь также могут объединяться в более общие группы, и т. д. Таким образом, получается иерархическая структура объектов, представленная на рис. 6.2.

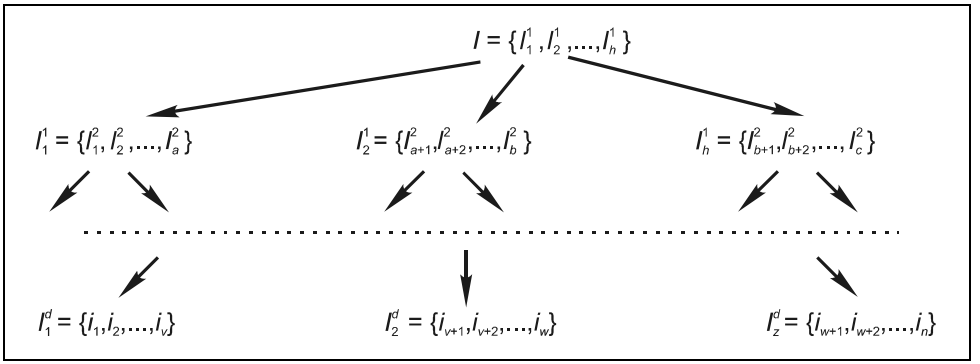



**Рис. 6.2.** Иерархическое представление объектов множества $I$

Для примера, приведенного в табл. 6.1, такой иерархии может быть следующая категоризация товаров:

- напитки;
  - алкогольные;
    - пиво;
  - безалкогольные;
    - вода;
- еда;
  - шоколад;
  - чипсы;
  - кокосы;
  - орехи.

Наличие иерархии изменяет представление о том, когда объект $i$ присутствует в транзакции $T$. Очевидно, что поддержка не отдельного объекта, а группы, в которую он входит, больше:

$$\text{Supp}(I_q^g) \geq \text{Supp}(i_j),$$

где $i_j \in I_q^g$.

Это связано с тем, что при анализе групп подсчитываются не только транзакции, в которые входит отдельный объект, но и транзакции, содержащие все объекты анализируемой группы. Например, если поддержка $\text{Supp}_{\{\text{кокосы, вода}\}} = 2/4$, то поддержка $\text{Supp}_{\{\text{еда, напитки}\}} = 2/4$, т. к. объекты групп `еда` и `напитки` входят в транзакции с идентификаторами 0, 1 и 2.

Использование иерархии позволяет определить связи, входящие в более высокие уровни иерархии, поскольку поддержка набора может увеличиваться, если подсчитывается вхождение группы, а не ее объекта. Кроме поиска наборов, часто встречающихся в транзакциях, состоящих из объектов $F = \{i \mid i \in I\}$ или групп одного уровня иерархии $F = \left\{I^g \mid I^g \in I^{g+1}\right\}$, можно рассматривать

также смешанные наборы объектов и групп $F = \left\{ i, I^g \mid i \in I^g \in I^{g+1} \right\}$. Это позволяет расширить анализ и получить дополнительные знания.

При иерархическом построении объектов можно варьировать характер поиска изменяя анализируемый уровень. Очевидно, что чем больше объектов во множестве *I*, тем больше объектов в транзакциях *T* и частых наборах. Это в свою очередь увеличивает время поиска и усложняет анализ результатов. Уменьшить или увеличить количество данных можно с помощью иерархического представления анализируемых объектов. Перемещаясь вверх по иерархии, обобщаем данные и уменьшаем их количество, и наоборот.

Недостатком обобщения объектов является меньшая полезность полученных знаний, т. к. в этом случае они относятся к группам товаров, что не всегда приемлемо. Для достижения компромисса между анализом групп и анализом отдельных объектов часто поступают следующим образом: сначала анализируют группы, а затем, в зависимости от полученных результатов, исследуют объекты заинтересовавших аналитика групп. В любом случае можно утверждать, что наличие иерархии в объектах и ее использование в задаче поиска ассоциативных правил позволяет выполнять более гибкий анализ и получать дополнительные знания.

В рассмотренной задаче поиска ассоциативных правил наличие объекта в транзакции определялось только его присутствием в ней $(i_j \in T)$ или отсутствием $(i_j \notin T)$. Часто объекты имеют дополнительные атрибуты, как правило, численные. Например, товары в транзакции имеют атрибуты: цена и количество. При этом наличие объекта в наборе может определяться не просто фактом его присутствия, а выполнением условия по отношению к определенному атрибуту. Например, при анализе транзакций, совершаемых покупателем, может интересовать не просто наличие покупаемого товара, а товара, покупаемого по некоторой цене.

Для расширения возможностей анализа с помощью поиска ассоциативных правил в исследуемые наборы можно добавлять дополнительные объекты. В общем случае они могут иметь природу, отличную от основных объектов. Например, для определения товаров, имеющих больший спрос в зависимости от месторасположения магазина, в транзакции можно добавить объект, характеризующий район.

## 6.2. Представление результатов

Решение задачи поиска ассоциативных правил, как и любой задачи, сводится к обработке исходных данных и получению результатов. Обработка над исходными данными выполняется по некоторому алгоритму Data Mining.

Результаты, получаемые при решении этой задачи, принято представлять в виде ассоциативных правил. В связи с этим при их поиске выделяют два основных этапа:

- нахождение всех частых наборов объектов;
- генерация ассоциативных правил из найденных частых наборов объектов.

Ассоциативные правила имеют следующий вид:

```
если (условие) то (результат)
```

где `условие` — обычно не логическое выражение (как в классификационных правилах), а набор объектов из множества $I$, с которыми связаны (ассоциированы) объекты, включенные в `результат` данного правила.

Например, ассоциативное правило:

```
если (кокосы, вода) то (орехи)
```

означает, что если потребитель покупает кокосы и воду, то он покупает и орехи.

Как уже отмечалось, в ассоциативных правилах условие и результат являются объектами множества $I$:

```
если X то Y
```

где $X \in I, Y \in I, X \cup Y = \varphi$.

Ассоциативное правило можно представить как импликацию над множеством $X \Rightarrow Y$, где $X \in I, Y \in I, X \cup Y = \varphi$.

Основным достоинством ассоциативных правил является их легкое восприятие человеком и простая интерпретация языками программирования. Однако они не всегда полезны. Выделяют три вида правил:

- *полезные правила* — содержат действительную информацию, которая ранее была неизвестна, но имеет логичное объяснение. Такие правила могут быть использованы для принятия решений, приносящих выгоду;
- *тривиальные правила* — содержат действительную и легко объяснимую информацию, которая уже известна. Такие правила, хотя и объяснимы, но не могут принести какой-либо пользы, т. к. отражают или известные законы в исследуемой области, или результаты прошлой деятельности. Иногда такие правила могут использоваться для проверки выполнения решений, принятых на основании предыдущего анализа;
- *непонятные правила* — содержат информацию, которая не может быть объяснена. Такие правила могут быть получены или на основе аномальных значений, или глубоко скрытых знаний. Напрямую такие правила нельзя использовать для принятия решений, т. к. их необъяснимость может привести к непредсказуемым результатам. Для лучшего понимания требуется дополнительный анализ.

Ассоциативные правила строятся на основе частых наборов. Так, правила, построенные на основании набора $F$ (т. е. $X \cup Y = F$), являются всеми возможными комбинациями объектов, входящих в него.

Например, для набора {кокосы, вода, орехи} могут быть построены следующие правила:

```
если (кокосы) то (вода);
если (кокосы) то (орехи);
если (кокосы) то (вода, орехи);
если (вода, орехи) то (кокосы);
если (вода) то (кокосы);
если (вода) то (орехи);
если (вода) то (кокосы, орехи);
если (кокосы, орехи) то (вода);
если (орехи) то (вода);
если (орехи) то (кокосы);
если (орехи) то (вода, кокосы);
если (вода, кокосы) то (орехи).
```

Таким образом, количество ассоциативных правил может быть очень большим и трудновоспринимаемым для человека. К тому же, не все из построенных правил несут в себе полезную информацию. Для оценки их полезности вводятся следующие величины.

**Поддержка** (support) — показывает, какой процент транзакций поддерживает данное правило. Так как правило строится на основании набора, то, значит, правило $X \Rightarrow Y$ имеет поддержку, равную поддержке набора $F$, который составляют $X$ и $Y$:

$$\mathrm{Supp}_{X \Rightarrow Y} = \mathrm{Supp}_F = \frac{|D_{F=X \cup Y}|}{|D|}.$$

Очевидно, что правила, построенные на основании одного и того же набора, имеют одинаковую поддержку, например, поддержка

$$\mathrm{Supp}_{\text{если (вода, кокосы) то (орехи)}} = \mathrm{Supp}_{\{\text{вода, кокосы, орехи}\}} = 2/4.$$

**Достоверность** (confidence) — показывает вероятность того, что из наличия в транзакции набора $X$ следует наличие в ней набора $Y$. Достоверностью правила $X \Rightarrow Y$ является отношение числа транзакций, содержащих наборы $X$ и $Y$, к числу транзакций, содержащих набор $X$:

$$\mathrm{Conf}_{X \Rightarrow Y} = \frac{|D_{F=X \cup Y}|}{|D_X|} = \frac{\mathrm{Supp}_{X \cup Y}}{\mathrm{Supp}_X}.$$

Очевидно, что чем больше достоверность, тем правило лучше, причем у правил, построенных на основании одного и того же набора, достоверность будет разная, например:

$$\mathrm{Conf}_{\text{если (вода) то (орехи)}} = 2/3;$$
$$\mathrm{Conf}_{\text{если (орехи) то (вода)}} = 2/3;$$

$$\text{Conf}_{\text{если (вода, кокосы) то (орехи)}} = 1;$$
$$\text{Conf}_{\text{если (вода) то (орехи, кокосы)}} = 2/3.$$

К сожалению, достоверность не позволяет оценить полезность правила. Если процент наличия в транзакциях набора $Y$ при условии наличия в них набора $X$ меньше, чем процент безусловного наличия набора $Y$, т. е.:

$$\text{Conf}_{X \Rightarrow Y} = \frac{\text{Supp}_{X \cup Y}}{\text{Supp}_X} < \text{Supp}_Y,$$

это значит, что вероятность случайно угадать наличие в транзакции набора $Y$ больше, чем предсказать это с помощью правила $X \Rightarrow Y$. Для исправления такой ситуации вводится мера — *улучшение*.

**Улучшение** (improvement) — показывает, полезнее ли правило случайного угадывания. Улучшение правила является отношением числа транзакций, содержащих наборы $X$ и $Y$, к произведению количества транзакций, содержащих набор $X$, и количества транзакций, содержащих набор $Y$:

$$\text{impr}_{X \Rightarrow Y} = \frac{|D_{F=X \cup Y}|}{|D_X||D_Y|} = \frac{\text{Supp}_{X \cup Y}}{\text{Supp}_X \cdot \text{Supp}_Y}.$$

Например,

$$\text{impr}_{\text{если (вода, кокосы) то (орехи)}} = 0{,}5/(0{,}5 \cdot 0{,}5) = 2.$$

Если улучшение больше единицы, то это значит, что с помощью правила предсказать наличие набора $Y$ вероятнее, чем случайное угадывание, если меньше единицы, то наоборот.

В последнем случае можно использовать отрицающее правило, т. е. правило, которое предсказывает отсутствие набора $Y$:

$$X \Rightarrow \text{не } Y.$$

У такого правила улучшение будет больше единицы, т. к. $\text{Supp}_{\text{не } Y} = 1 - \text{Supp}_Y$.

Таким образом, можно получить правило, которое предсказывает результат лучше, чем случайным образом. Правда, на практике такие правила мало применимы. Например, правило:

```
если (вода, орехи) то не пиво
```

мало полезно, т. к. слабо выражает поведение покупателя.

Данные оценки используются при генерации правил. Аналитик при поиске ассоциативных правил задает минимальные значения перечисленных величин. В результате те правила, которые не удовлетворяют этим условиям, отбрасываются и не включаются в решение задачи. С этой точки зрения нельзя

объединять разные правила, хотя и имеющие общую смысловую нагрузку. Например, следующие правила:

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_3\},$$

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_4\}$$

нельзя объединить в одно:

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_3, i_4\},$$

т. к. достоверности их будут разные, следовательно, некоторые из них могут быть исключены, а некоторые — нет.

Если объекты имеют дополнительные атрибуты, которые влияют на состав объектов в транзакциях, а следовательно, и в наборах, то они должны учитываться в генерируемых правилах. В этом случае условная часть правил будет содержать не только проверку наличия объекта в транзакции, но и более сложные операции сравнения: больше, меньше, включает и др. Результирующая часть правил также может содержать утверждения относительно значений атрибутов. Например, если у товаров рассматривается цена, то правила могут иметь следующий вид:

```
если пиво.цена < 10 то чипсы.цена < 7.
```

Данное правило говорит о том, что если покупается пиво по цене меньше 10 р., то, вероятно, будут куплены чипсы по цене меньше 7 р.

# 6.3. Алгоритмы

## 6.3.1. Алгоритм Apriori

Выявление частых наборов объектов — операция, требующая большого количества вычислений, а следовательно, и времени. Алгоритм Apriori описан в 1994 г. Срикантом Рамакришнан (Ramakrishnan Srikant) и Ракешом Агравалом (Rakesh Agrawal). Он использует одно из свойств поддержки, гласящее: поддержка любого набора объектов не может превышать минимальной поддержки любого из его подмножеств:

$$\text{Supp}_F \le \text{Supp}_E \text{ при } E \subset F.$$

Например, поддержка 3-объектного набора {пиво, вода, чипсы} будет всегда меньше или равна поддержке 2-объектных наборов {пиво, вода}, {вода, чипсы}, {пиво, чипсы}. Это объясняется тем, что любая транзакция, содержащая {пиво, вода, чипсы}, содержит также и наборы {пиво, вода}, {вода, чипсы}, {пиво, чипсы}, причем обратное неверно.

Алгоритм Apriori определяет часто встречающиеся наборы за несколько этапов. На $i$-м этапе определяются все часто встречающиеся $i$-элементные наборы. Каждый этап состоит из двух шагов: формирования кандидатов (candidate generation) и подсчета поддержки кандидатов (candidate counting).

Рассмотрим $i$-й этап. На шаге формирования кандидатов алгоритм создает множество кандидатов из $i$-элементных наборов, чья поддержка пока не вычисляется. На шаге подсчета кандидатов алгоритм сканирует множество транзакций, вычисляя поддержку наборов-кандидатов. После сканирования отбрасываются кандидаты, поддержка которых меньше определенного пользователем минимума, и сохраняются только часто встречающиеся $i$-элементные наборы. Во время 1-го этапа выбранное множество наборов-кандидатов содержит все 1-элементные частые наборы. Алгоритм вычисляет их поддержку во время шага подсчета кандидатов.

Описанный алгоритм можно записать в виде следующего псевдокода:

```
L1 = {часто встречающиеся 1-элементные наборы}
для (k=2; Lk-1 <> ϕ; k++)
       Ck = Apriorigen(Fk-1) // генерация кандидатов
       для всех транзакций t ∈ D выполнить
              Ct = subset(Ck, t) // удаление избыточных правил
              для всех кандидатов c ∈ Ct выполнить
                     c.count ++
              конец для всех
       конец для всех
       Lk = { c ∈ Ck | c.count >= Suppmin} // отбор кандидатов
конец для
Результат = ⋃k Lk
```

Опишем обозначения, используемые в алгоритме:

- $L_k$ — множество $k$-элементных частых наборов, чья поддержка не меньше заданной пользователем. Каждый член множества имеет набор упорядоченных ($i_j < i_p$, если $j < p$) элементов $F$ и значение поддержки набора $\mathrm{Supp}_F > \mathrm{Supp}_{\min}$:

$$L_k = \{(F_1, \mathrm{Supp}_1), (F_2, \mathrm{Supp}_2), ..., (F_q, \mathrm{Supp}_q)\},$$

  где $F_j = \{i_1, i_2, ..., i_k\}$;

- $C_k$ — множество кандидатов $k$-элементных наборов потенциально частых. Каждый член множества имеет набор упорядоченных ($i_j < i_p$, если $j < p$) элементов $F$ и значение поддержки набора $\mathrm{Supp}$.

Опишем данный алгоритм по шагам.

**Шаг 1.** Присвоить $k = 1$ и выполнить отбор всех 1-элементных наборов, у которых поддержка больше минимально заданной пользователем $\text{Supp}_{\min}$.

**Шаг 2.** $k = k + 1$.

**Шаг 3.** Если не удается создавать *k*-элементные наборы, то завершить алгоритм, иначе выполнить следующий шаг.

**Шаг 4.** Создать множество *k*-элементных наборов кандидатов в частые наборы. Для этого необходимо объединить в *k*-элементные кандидаты $(k-1)$-элементные частые наборы. Каждый кандидат $c \in C_k$ будет формироваться путем добавления к $(k-1)$-элементному частому набору — $p$ элемента из другого $(k-1)$-элементного частого набора — $q$. Причем добавляется последний элемент набора $q$, который по порядку выше, чем последний элемент набора $p$ ($\text{p.item}_{k-1} < \text{q.item}_{k-1}$). При этом первые все $(k-2)$ элемента обоих наборов одинаковы ($\text{p.item}_1 = \text{q.item}_1$, $\text{p.item}_2 = \text{q.item}_2$, ..., $\text{p.item}_{k-2} = \text{q.item}_{k-2}$).

Это может быть записано в виде следующего SQL-подобного запроса.

```
insert into Ck
select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
from Lk-1 p, Lk-1 q
where p.item1 = q.item1, p.item2 = q.item2, ..., p.itemk-2 = q.itemk-2,
p.itemk-1 < q.itemk-1
```

**Шаг 5.** Для каждой транзакции $T$ из множества $D$ выбрать кандидатов $C_t$ из множества $C_k$, присутствующих в транзакции $T$. Для каждого набора из построенного множества $C_k$ удалить набор, если хотя бы одно из его $(k-1)$ подмножеств не является часто встречающимся, т. е. отсутствует во множестве $L_{k-1}$. Это можно записать в виде следующего псевдокода:

```
для всех наборов c ∈ Ck выполнить
        для всех (k-1) - поднаборов s из c выполнить
              если (s ∉ Lk-1) то
                    удалить c из Ck
```

**Шаг 6.** Для каждого кандидата из множества $C_k$ увеличить значение поддержки на единицу.

**Шаг 7.** Выбрать только кандидатов $L_k$ из множества $C_k$, у которых значение поддержки больше заданной пользователем $\text{Supp}_{\min}$. Вернуться к шагу 2.

Результатом работы алгоритма является объединение всех множеств $L_k$ для всех *k*.

Рассмотрим работу алгоритма на примере, приведенном в табл. 6.1, при $\text{Supp}_{\min} = 0,5$. На первом шаге имеем следующее множество кандидатов $C_1$ (указываются идентификаторы товаров) (табл. 6.5).

**Таблица 6.5**

| № | Набор | Supp |
|---|---|---|
| 1 | {0} | 0 |
| 2 | {1} | 0,5 |
| 3 | {2} | 0,75 |
| 4 | {4} | 0,25 |
| 5 | {3} | 0,75 |
| 6 | {5} | 0,75 |

Заданной минимальной поддержке удовлетворяют только кандидаты 2, 3, 5 и 6, следовательно:

$$L_1 = \{\{1\}, \{2\}, \{3\}, \{5\}\}.$$

На втором шаге увеличиваем значение $k$ до двух. Так как можно построить 2-элементные наборы, то получаем множество $C_2$ (табл. 6.6).

**Таблица 6.6**

| № | Набор | Supp |
|---|---|---|
| 1 | {1, 2} | 0,25 |
| 2 | {1, 3} | 0,5 |
| 3 | {1, 5} | 0,25 |
| 4 | {2, 3} | 0,5 |
| 5 | {2, 5} | 0,75 |
| 6 | {3, 5} | 0,5 |

Из построенных кандидатов заданной минимальной поддержке удовлетворяют только кандидаты 2, 4, 5 и 6, следовательно:

$$L_2 = \{\{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 5\}\}.$$

На третьем шаге перейдем к созданию 3-элементных кандидатов и подсчету их поддержки. В результате получим следующее множество $C_3$ (табл. 6.7).

**Таблица 6.7**

| № | Набор | Supp |
|---|---|---|
| 1 | {2, 3, 5} | 0,5 |

Данный набор удовлетворяет минимальной поддержке, следовательно:

$$L_3 = \{\{2, 3, 5\}\}.$$

Так как 4-элементные наборы создать не удастся, то результатом работы алгоритма является множество:

$$L = L_1 \cup L_2 \cup L_3 = \{\{1\}, \{2\}, \{3\}, \{5\}, \{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 5\}, \{2, 3, 5\}\}.$$

Для подсчета поддержки кандидатов нужно сравнить каждую транзакцию с каждым кандидатом. Очевидно, что количество кандидатов может быть очень большим и нужен эффективный способ подсчета. Гораздо быстрее и эффективнее использовать подход, основанный на хранении кандидатов в хэш-дереве. Внутренние узлы дерева содержат хэш-таблицы с указателями на потомков, а листья — на кандидатов. Это дерево используется при быстром подсчете поддержки для кандидатов.

Хэш-дерево строится каждый раз, когда формируются кандидаты. Первоначально дерево состоит только из корня, который является листом, и не содержит никаких кандидатов-наборов. Каждый раз, когда формируется новый кандидат, он заносится в корень дерева, и так до тех пор, пока количество кандидатов в корне-листе не превысит некоего порога. Как только это происходит, корень преобразуется в хэш-таблицу, т. е. становится внутренним узлом, и для него создаются потомки-листья. Все кандидаты распределяются по узлам-потомкам согласно хэш-значениям элементов, входящих в набор. Каждый новый кандидат хэшируется на внутренних узлах, пока не достигнет первого узла-листа, где он и будет храниться, пока количество наборов опять же не превысит порога.

После того как хэш-дерево с кандидатами-наборами построено, легко подсчитать поддержку для каждого кандидата. Для этого нужно "пропустить" каждую транзакцию через дерево и увеличить счетчики для тех кандидатов, чьи элементы также содержатся и в транзакции, $C_k \cap T_i = C_k$. На корневом уровне хэш-функция применяется к каждому объекту из транзакции. Далее, на втором уровне, хэш-функция применяется ко вторым объектам и т. д. На $k$-м уровне хэшируется $k$ -элемент, и так до тех пор, пока не достигнем листа. Если кандидат, хранящийся в листе, является подмножеством рассматриваемой транзакции, увеличиваем счетчик поддержки этого кандидата на единицу.

После того как каждая транзакция из исходного набора данных "пропущена" через дерево, можно проверить, удовлетворяют ли значения поддержки кандидатов минимальному порогу. Кандидаты, для которых это условие выполняется, переносятся в разряд часто встречающихся. Кроме того, следует запомнить и поддержку набора, которая пригодится при извлечении правил. Эти же действия применяются для нахождения ($k$ + 1)-элементных наборов и т. д.

## 6.3.2. Разновидности алгоритма Apriori

Алгоритм *AprioriTid* является разновидностью алгоритма Apriori. Отличительной чертой данного алгоритма является подсчет значения поддержки кандидатов не при сканировании множества $D$, а с помощью множества $\overline{C}_k$, являющегося множеством кандидатов ($k$-элементных наборов) потенциально частых, в соответствие которым ставится идентификатор TID транзакций, в которых они содержатся.

Каждый член множества $\overline{C}_k$ является парой вида $< \text{TID}, F_k >$, где каждый $F_k$ является потенциально частым $k$-элементным набором, представленным в транзакции с идентификатором TID. Множество $\overline{C}_1 = D$ соответствует множеству транзакций, хотя каждый объект в транзакции соответствует однообъектному набору в множестве $\overline{C}_1$, содержащем этот объект. Для $k > 1$ множество $\overline{C}_k$ генерируется в соответствии с алгоритмом, описанным ниже. Член множества $\overline{C}_k$, соответствующий транзакции $T$, является парой следующего вида:

$$< T.\text{TID}, \{c \in C_k \mid c \in T\} >.$$

Подмножество наборов в $\overline{C}_k$ с одинаковыми TID (т. е. содержатся в одной и той же транзакции) называется *записью*. Если транзакция не содержит ни одного $k$-элементного кандидата, то $\overline{C}_k$ не будет иметь записи для этой транзакции. То есть количество записей в $\overline{C}_k$ может быть меньше, чем в $D$, особенно для больших значений $k$. Кроме того, для больших значений $k$ каждая запись может быть меньше, чем соответствующая ей транзакция, т. к. в транзакции будет содержаться мало кандидатов. Однако для малых значений $k$ каждая запись может быть больше, чем соответствующая транзакция, т. к. $\overline{C}_k$ включает всех кандидатов $k$-элементных наборов, содержащихся в транзакции.

Другой разновидностью алгоритма Apriori является алгоритм *MSAP* (Mining Sequential Alarm Patterns), специально разработанный для выполнения секвенциального анализа сбоев телекоммуникационной сети.

Он использует следующее свойство поддержки последовательностей: для любой последовательности $L_k$ ее поддержка будет меньше, чем поддержка последовательностей из множества $L_{k-1}$.

Алгоритм MSAP для поиска событий, следующих друг за другом, использует понятие "срочного окна" (Urgent Window). Это позволяет выявлять не просто одинаковые последовательности событий, а следующие друг за другом. В остальном данный алгоритм работает по тому же принципу, что и Apriori.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Задачей поиска ассоциативных правил является определение часто встречающихся наборов объектов в большом множестве наборов.
- Секвенциальный анализ заключается в поиске частых последовательностей. Основным отличием задачи секвенциального анализа от поиска ассоциативных правил является установление отношения порядка между объектами.
- Наличие иерархии в объектах и ее использование в задаче поиска ассоциативных правил позволяет выполнять более гибкий анализ и получать дополнительные знания.
- Результаты решения задачи представляются в виде ассоциативных правил, условная и заключительная часть которых содержит наборы объектов.
- Основными характеристиками ассоциативных правил являются поддержка, достоверность и улучшение.
- Поддержка (support) показывает, какой процент транзакций поддерживает данное правило.
- Достоверность (confidence) показывает, какова вероятность того, что из наличия в транзакции набора условной части правила следует наличие в ней набора заключительной части.
- Улучшение (improvement) показывает, полезнее ли правило случайного угадывания.
- Задача поиска ассоциативных правил решается в два этапа. На первом выполняется поиск всех частых наборов объектов. На втором из найденных частых наборов объектов генерируются ассоциативные правила.
- Алгоритм Apriori использует одно из свойств поддержки, гласящее: поддержка любого набора объектов не может превышать минимальной поддержки любого из его подмножеств.

# ГЛАВА 7

# Кластеризация

## 7.1. Постановка задачи кластеризации

Первые публикации по кластерному анализу появились в конце 30-х гг. прошлого столетия, но активное развитие этих методов и их широкое использование началось в конце 60-х—начале 70-х гг. В дальнейшем это направление многомерного анализа интенсивно развивалось. Появились новые методы, модификации уже известных алгоритмов, существенно расширилась область применения кластерного анализа. Если первоначально методы многомерной классификации использовались в психологии, археологии, биологии, то сейчас они стали активно применяться в социологии, экономике, статистике, в исторических исследованиях. Особенно расширилось их использование в связи с появлением и развитием ЭВМ и, в частности, персональных компьютеров. Это связано прежде всего с трудоемкостью обработки больших массивов информации (вычисление и обращение матриц больших размеров).

Большое достоинство кластерного анализа в том, что он позволяет осуществлять разбиение объектов не по одному параметру, а по целому набору признаков. Кроме того, кластерный анализ, в отличие от большинства математико-статистических методов, не накладывает никаких ограничений на вид рассматриваемых объектов и позволяет рассматривать множество исходных данных практически произвольной природы. Это имеет большое значение, например, для прогнозирования конъюнктуры, при наличии разнородных показателей, затрудняющих применение традиционных эконометрических подходов.

Кластерный анализ позволяет рассматривать достаточно большой объем информации и резко сокращать, сжимать большие массивы информации, делать их компактными и наглядными.

Задача кластеризации состоит в разделении исследуемого множества объектов на группы "похожих" объектов, называемых кластерами. Слово кластер

английского происхождения (cluster), переводится как сгусток, пучок, группа. Родственные понятия, используемые в литературе, — класс, таксон, сгущение. Часто решение задачи разбиения множества элементов на кластеры называют кластерным анализом.

Решением задачи классификации является отнесение каждого из объектов данных к одному (или нескольким) из заранее определенных классов и построение в конечном счете одним из методов классификации модели данных, определяющей разбиение множества объектов данных на классы.

В задаче кластеризации отнесение каждого из объектов данных осуществляется к одному (или нескольким) из заранее неопределенных классов. Разбиение объектов данных по кластерам осуществляется при одновременном их формировании. Определение кластеров и разбиение по ним объектов данных выражается в итоговой модели данных, которая является решением задачи кластеризации.

Ввиду особого положения задачи кластеризации в списке задач интеллектуального анализа данных было разработано множество способов ее решения. Один из них — построение набора характеристических функций классов, которые показывают, относится ли объект данных к данному классу или нет. Характеристическая функция класса может быть двух типов:

1. Дискретная функция, принимающая одно из двух определенных значений, смысл которых в принадлежности/непринадлежности объекта данных заданному классу.
2. Функция, принимающая вещественные значения, например из интервала 0 ... 1. Чем ближе значение функции к единице, тем больше объект данных принадлежит заданному классу.

Общий подход к решению задачи кластеризации стал возможен после развития Л. Заде теории нечетких множеств. В рамках данного подхода удается формализовать качественные понятия, неопределенность, присущую реальным данным и процессам. Успех этого подхода объясняется еще и тем, что в процессе анализа данных участвует человек, оценки и суждения которого расплывчаты и субъективны. Уместно привести высказывание Л. Заде, основоположника теории нечетких множеств: "...нужна новая точка зрения, новый комплекс понятий и методов, в которых нечеткость принимается как универсальная реальность человеческого существования".

Применяя теорию нечетких множеств для решения задачи кластеризации, возможны различные варианты введения нечеткости в методы, решающие данную задачу. Нечеткость может учитываться как в представлении данных, так и при описании их взаимосвязи. Кроме того, данные могут как обладать, так и не обладать количественной природой. Тем не менее во многих практических задачах данные, которые необходимо исследовать, являются результатом накопленного опыта в той или иной сфере человеческой деятельности

и часто имеют количественное представление. Учет нечеткости самих исследуемых данных, в общем случае, — серьезная проблема. Поэтому как в существующих алгоритмах, так и в подходе, предлагаемом в данном издании, не делается никаких допущений о нечеткости самих исходных данных. Считается, что данные являются четкими и выражены количественно.

Описывать нечеткие взаимосвязи данных можно разными способами. Одним из таких способов, нашедших широкое распространение в используемых в настоящее время алгоритмах нечеткой кластеризации данных, является описание взаимосвязи данных через их отношение к некоторым эталонным образцам — центрам кластеров. В данных алгоритмах нечеткость проявляется в описании кластеров как нечетких множеств, имеющих ядро в центре кластера. С другой стороны, взаимосвязь данных в условиях неопределенности можно учитывать при помощи аппарата нечетких отношений между отдельными образцами данных, не прибегая при этом к понятию центра кластера. Такой подход не нашел еще широкого распространения на практике, хотя, очевидно, является более универсальным. В данном издании делается попытка восполнить этот пробел и показать те преимущества, которые дает использование указанного понятия для задачи кластеризации. Итак, перейдем к постановке задачи кластеризации.

## 7.1.1. Формальная постановка задачи

Дано — набор данных со следующими свойствами:

- каждый экземпляр данных выражается четким числовым значением;
- класс для каждого конкретного экземпляра данных неизвестен.

Найти:

- способ сравнения данных между собой (меру сходства);
- способ кластеризации;
- разбиение данных по кластерам.

Формально задача кластеризации описывается следующим образом.

Дано множество объектов данных $I$, каждый из которых представлен набором атрибутов. Требуется построить множество кластеров $C$ и отображение $F$ множества $I$ на множество $C$, т. е. $F: I \to C$. Отображение $F$ задает модель данных, являющуюся решением задачи. Качество решения задачи определяется количеством верно классифицированных объектов данных.

Множество $I$ определим следующим образом:

$$I = \{i_1, i_2, ..., i_j, ..., i_n\},$$

где $i_j$ — исследуемый объект.

Примером такого множества может быть набор данных об ирисах, с которыми в середине 30-х гг. прошлого столетия работал известный статистик Р. А. Фишер (эти данные часто называют ирисы Фишера). Он рассмотрел три класса ирисов: Iris setosa, Iris versicolor и Iris virginica. Для каждого из них было представлено по 50 экземпляров с разными значениями четырех параметров: длина и ширина чашелистника, длина и ширина лепестка. В табл. 7.1 представлены данные по пяти экземплярам для каждого класса.

**Таблица 7.1**

| № | Длина чашелистника | Ширина чашелистника | Длина лепестка | Ширина лепестка | Класс |
|---|---|---|---|---|---|
| 1 | 5,1 | 3,5 | 1,4 | 0,2 | Iris setosa |
| 2 | 4,9 | 3,0 | 1,4 | 0,2 | Iris setosa |
| 3 | 4,7 | 3,2 | 1,3 | 0,2 | Iris setosa |
| 4 | 4,6 | 3,1 | 1,5 | 0,2 | Iris setosa |
| 5 | 5,0 | 3,6 | 1,4 | 0,2 | Iris setosa |
| 51 | 7,0 | 3,2 | 4,7 | 1,4 | Iris versicolor |
| 52 | 6,4 | 3,2 | 4,5 | 1,5 | Iris versicolor |
| 53 | 6,9 | 3,1 | 4,9 | 1,5 | Iris versicolor |
| 54 | 5,5 | 2,3 | 4,0 | 1,3 | Iris versicolor |
| 55 | 6,5 | 2,8 | 4,6 | 1,5 | Iris versicolor |
| 101 | 6,3 | 3,3 | 6,0 | 2,5 | Iris virginica |
| 102 | 5,8 | 2,7 | 5,1 | 1,9 | Iris virginica |
| 103 | 7,1 | 3,0 | 5,9 | 2,1 | Iris virginica |
| 104 | 6,3 | 2,9 | 5,6 | 1,8 | Iris virginica |
| 105 | 6,5 | 3,0 | 5,8 | 2,2 | Iris virginica |

Каждый из объектов характеризуется набором параметров:

$$i_j = \{x_1, x_2, ..., x_h, ..., x_m\}.$$

В примере с ирисами, как уже отмечалось, такими параметрами являются длина и ширина чашелистника, длина и ширина лепестка.

Каждая переменная $x_h$ может принимать значения из некоторого множества:

$$x_h = \{v_h^1, v_h^2, ...\}.$$

В данном примере значениями являются действительные числа.

Задача кластеризации состоит в построении множества:

$$C = \{c_1, c_2, ..., c_k, ..., c_g\}.$$

Здесь $c_k$ — кластер, содержащий похожие друг на друга объекты из множества $I$:

$$c_k = \{i_j, i_p \mid i_j \in I, i_p \in I \text{ и } d(i_j, i_p) < \sigma\},$$

где $\sigma$ — величина, определяющая меру близости для включения объектов в один кластер; $d(i_j, i_p)$ — мера близости между объектами, называемая расстоянием.

Неотрицательное значение $d(i_j, i_p)$ называется расстоянием между элементами $i_j$ и $i_p$, если выполняются следующие условия:

1. $d(i_j, i_p) \geq 0$, для всех $i_j$ и $i_p$.
2. $d(i_j, i_p) = 0$, тогда и только тогда, когда $i_j = i_p$.
3. $d(i_j, i_p) = d(i_p, i_j)$.
4. $d(i_j, i_p) \leq d(i_j, i_r) + d(i_r, i_p)$.

Если расстояние $d(i_j, i_p)$ меньше некоторого значения $\sigma$, то говорят, что элементы близки и помещаются в один кластер. В противном случае говорят, что элементы отличны друг от друга и их помещают в разные кластеры.

Большинство популярных алгоритмов, решающих задачу кластеризации, используют в качестве формата входных данных матрицу отличия $D$. Строки и столбцы матрицы соответствуют элементам множества $I$. Элементами матрицы являются значения $d(i_j, i_p)$ в строке $j$ и столбце $p$. Очевидно, что на главной диагонали значения будут равны нулю:

$$D = \begin{pmatrix} 0 & d(e_1, e_2) & d(e_1, e_n) \\ d(e_2, e_1) & 0 & d(e_2, e_n) \\ d(e_n, e_1) & d(e_n, e_2) & 0 \end{pmatrix}.$$

Большинство алгоритмов работают с симметричными матрицами. Если матрица несимметрична, то ее можно привести к симметричному виду путем следующего преобразования:

$$(D + D^m)/2.$$

## 7.1.2. Меры близости, основанные на расстояниях, используемые в алгоритмах кластеризации

Расстояния между объектами предполагают их представление в виде точек $m$-мерного пространства $R^m$. В этом случае могут быть использованы различные подходы к вычислению расстояний.

Рассмотренные далее меры определяют расстояния между двумя точками, принадлежащими пространству входных переменных. Используются следующие обозначения:

- $X_Q \subseteq R^m$ — множество данных, являющееся подмножеством $m$-мерного вещественного пространства;
- $x_i = (x_{i1}, ..., x_{im}) \in X_Q$, $i = \overline{1, Q}$ — элементы множества данных;
- $\bar{x} = \frac{1}{Q}\sum_{i=1}^{Q} x_i$ — среднее значение точек данных;
- $S = \frac{1}{Q-1}\sum_{i=1}^{Q}(x_i - \bar{x})(x_i - \bar{x})^t$ — ковариационная матрица ($m \times m$).

Итак, приведем наиболее известные меры близости.

**Евклидово расстояние.** Иногда может возникнуть желание возвести в квадрат стандартное евклидово расстояние, чтобы придать большие веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом *(см. также замечания в разд. 7.1.1)*:

$$d_2(x_i, x_j) = \sqrt{\sum_{t=1}^{m}(x_{it} - x_{jt})^2}\ . \qquad (7.1)$$

**Расстояние по Хеммингу.** Это расстояние является просто средним разностей по координатам. В большинстве случаев данная мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида, однако для нее влияние отдельных больших разностей (выбросов) уменьшается (т. к. они не возводятся в квадрат). Расстояние по Хеммингу вычисляется по формуле:

$$d_H(x_i, x_j) = \sum_{t=1}^{m} | x_{it} - x_{jt} |. \qquad (7.2)$$

**Расстояние Чебышева.** Это расстояние может оказаться полезным, когда желают определить два объекта как "различные", если они различаются по какой-либо одной координате (каким-либо одним измерением). Расстояние Чебышева вычисляется по формуле:

$$d_\infty(x_i, x_j) = \max_{1 \le t \le m} | x_{it} - x_{jt} |. \qquad (7.3)$$

**Расстояние Махаланобиса** преодолевает этот недостаток, но данная мера расстояния плохо работает, если ковариационная матрица высчитывается на

всем множестве входных данных. В то же время, будучи сосредоточенной на конкретном классе (группе данных), данная мера расстояния показывает хорошие результаты:

$$d_M(x_i, x_j) = (x_i - x_j)S^{-1}(x_i - x_j)^t. \quad (7.4)$$

**Пиковое расстояние** предполагает независимость между случайными переменными, что говорит о расстоянии в ортогональном пространстве. Но в практических приложениях эти переменные не являются независимыми:

$$d_L(x_i, x_j) = \frac{1}{m}\sum_{t=1}^{m}\frac{|x_{it} - x_{jt}|}{x_{it} + x_{jt}}. \quad (7.5)$$

Любую из приведенных мер расстояния можно выбирать с уверенностью лишь в том случае, если имеется информация о характере данных, подвергаемых кластеризации.

Так, например, пиковое расстояние предполагает независимость между случайными переменными, что говорит о расстоянии в ортогональном пространстве. Но в практических приложениях эти переменные не являются независимыми.

## 7.2. Представление результатов

Результатом кластерного анализа является набор кластеров, содержащих элементы исходного множества. Кластерная модель должна описывать как сами кластеры, так и принадлежность каждого объекта к одному из них.

Для небольшого числа объектов, характеризующихся двумя переменными, результаты кластерного анализа изображают графически. Элементы представляются точками, кластеры разделяются прямыми, которые описываются линейными функциями. Для примера с данными из табл. 7.1 результат кластеризации можно представить диаграммой, изображенной на рис. 7.1.

Если кластеры нельзя разделить прямыми, то рисуются ломаные линии, которые описываются нелинейными функциями.

В случае если элемент может принадлежать нескольким кластерам, то можно использовать Венские диаграммы, например, как на рис. 7.2.

Некоторые алгоритмы не просто относят элемент к одному из кластеров, а определяют вероятность его принадлежности. В этом случае удобнее представлять результат их работы в виде таблицы. В ней строки соответствуют элементам исходного множества, столбцы — кластерам, а в ячейках указывается вероятность принадлежности элемента к кластеру.

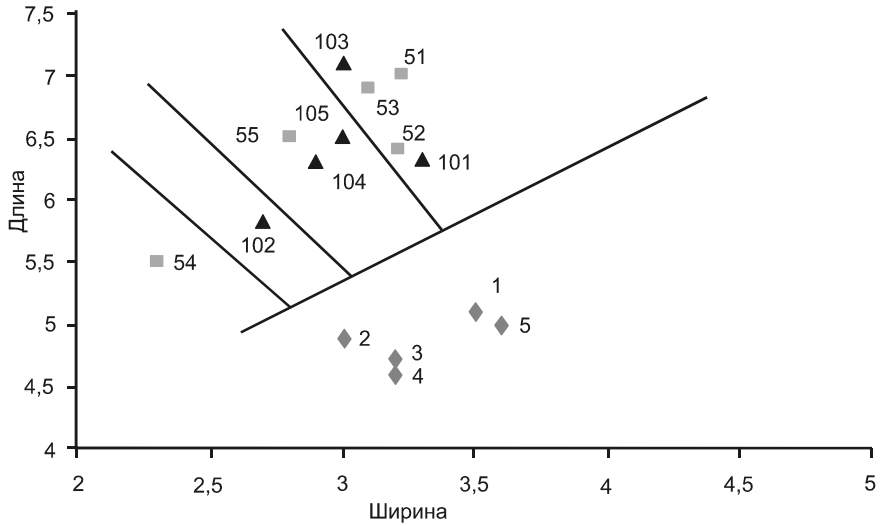



**Рис. 7.1.** Разделение ирисов на кластеры линиями

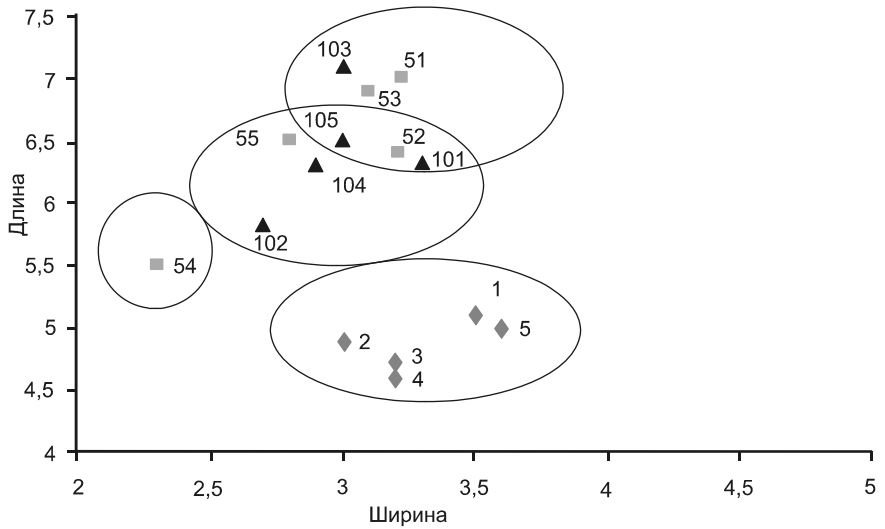



**Рис. 7.2.** Разделение ирисов на кластеры с использованием Венских диаграмм

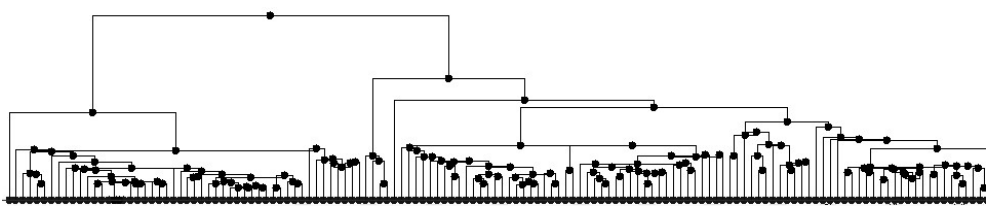


**Рис. 7.3.** Дендрограмма, построенная для данных из табл. 7.1

Ряд алгоритмов кластеризации строят иерархические структуры кластеров. В таких структурах самый верхний уровень соответствует всему множеству объектов, т. е. одному-единственному кластеру. На следующем уровне он делится на несколько подкластеров. Каждый из них делится еще на несколько и т. д. Построение такой иерархии может происходить до тех пор, пока кластеры не будут соответствовать отдельным объектам. Такие диаграммы называются *дендрограммами* (dendrograms). Этот термин подчеркивает древовидную структуру диаграмм (от греч. *dendron* — дерево).

Существует много способов построения дендрограмм. Для примера с ирисами дендрограмма будет выглядеть так, как показано на рис. 7.3.

# 7.3. Базовые алгоритмы кластеризации

## 7.3.1. Классификация алгоритмов

При выполнении кластеризации важно, сколько в результате должно быть построено кластеров. Предполагается, что кластеризация должна выявить естественные локальные сгущения объектов. Поэтому число кластеров является параметром, часто существенно усложняющим вид алгоритма, если предполагается неизвестным, и существенно влияющим на качество результата, если оно известно.

Проблема выбора числа кластеров весьма нетривиальна. Достаточно сказать, что для получения удовлетворительного теоретического решения часто требуется сделать весьма сильные предположения о свойствах некоторого заранее заданного семейства распределений. Но о каких предположениях может идти речь, когда, особенно в начале исследования, о данных практически ничего неизвестно? Поэтому алгоритмы кластеризации обычно строятся как некоторый способ перебора числа кластеров и определения его оптимального значения в процессе перебора.

Число методов разбиения множества на кластеры довольно велико. Все их можно подразделить на иерархические и неиерархические.

В неиерархических алгоритмах характер их работы и условие остановки необходимо заранее регламентировать часто довольно большим числом параметров, что иногда затруднительно, особенно на начальном этапе изучения материала. Но в таких алгоритмах достигается большая гибкость в варьировании кластеризации и обычно определяется число кластеров.

С другой стороны, когда объекты характеризуются большим числом признаков (параметров), то приобретает важное значение задача группировки признаков. Исходная информация содержится в квадратной матрице связей при-

знаков, в частности в корреляционной матрице. Основой успешного решения задачи группировки является неформальная гипотеза о небольшом числе скрытых факторов, которые определяют структуру взаимных связей между признаками.

В иерархических алгоритмах фактически отказываются от определения числа кластеров, строя полное дерево вложенных кластеров (дендрограмму). Число кластеров определяется из предположений, в принципе, не относящихся к работе алгоритмов, например по динамике изменения порога расщепления (слияния) кластеров. Трудности таких алгоритмов хорошо изучены: выбор мер близости кластеров, проблема инверсий индексации в дендрограммах, негибкость иерархических классификаций, которая иногда весьма нежелательна. Тем не менее, представление кластеризации в виде дендрограммы позволяет получить наиболее полное представление о структуре кластеров.

Иерархические алгоритмы связаны с построением дендрограмм и делятся:

- на агломеративные, характеризуемые последовательным объединением исходных элементов и соответствующим уменьшением числа кластеров (построение кластеров снизу вверх);
- на дивизимные (делимые), в которых число кластеров возрастает, начиная с одного, в результате чего образуется последовательность расщепляющих групп (построение кластеров сверху вниз).

## 7.3.2. Иерархические алгоритмы

### Агломеративные алгоритмы

На первом шаге все множество $I$ представляется как множество кластеров:

$$c_1 = \{i_1\}, \quad c_2 = \{i_2\}, \quad ..., \quad c_m = \{i_m\}.$$

На следующем шаге выбираются два наиболее близких друг к другу (например, $c_p$ и $c_q$) и объединяются в один общий кластер. Новое множество, состоящее уже из $(m-1)$ кластеров, будет:

$$c_1 = \{i_1\}, \quad c_2 = \{i_2\}, \quad ..., \quad c_p = \{i_p, i_q\}, \quad \ldots, \quad c_m = \{i_m\}.$$

Повторяя процесс, получим последовательные множества кластеров, состоящие из $(m-2)$, $(m-3)$, $(m-4)$ и т. д.

В конце процедуры получится кластер, состоящий из $m$ объектов и совпадающий с первоначальным множеством *I*.

Для определения расстояния между кластерами можно выбрать разные способы. В зависимости от этого получают алгоритмы с различными свойствами.

Существует несколько методов пересчета расстояний с использованием старых значений расстояний для объединяемых кластеров, отличающихся коэффициентами в формуле:

$$d_{rs} = \alpha_p d_{ps} + \alpha_q d_{qs} + \beta d_{pq} + \gamma \left| d_{ps} - d_{qs} \right|.$$

Если кластеры $p$ и $q$ объединяются в кластер $r$ и требуется рассчитать расстояние от нового кластера до кластера $s$, применение того или иного метода зависит от способа определения расстояния между кластерами, эти методы различаются значениями коэффициентов $\alpha_p$, $\alpha_q$, $\beta$ и $\gamma$.

В табл. 7.2 приведены коэффициенты пересчета расстояний между кластерами $\alpha_p$, $\alpha_q$, $\beta$ и $\gamma$.

**Таблица 7.2**

| **Название метода** | $\alpha_p$ | $\alpha_q$ | $\beta$ | $\gamma$ |
|---|---|---|---|---|
| Расстояние между ближайшими соседями-ближайшими объектами кластеров (Nearest neighbor) | 1/2 | 1/2 | 0 | ~1/2 |
| Расстояние между самыми далекими соседями (Furthest neighbor) | 1/2 | 1/2 | 0 | 1/2 |
| Метод медиан — тот же центроидный метод, но центр объединенного кластера вычисляется как среднее всех объектов (Median clustering) | 1/2 | 1/2 | ~1/4 | 0 |
| Среднее расстояние между кластерами (Between-groups linkage) | 1/2 | 1/2 | 0 | 0 |
| Среднее расстояние между всеми объектами пары кластеров с учетом расстояний внутри кластеров (Within-groups linkage) | $\frac{k_p}{k_p + k_q}$ | $\frac{k_q}{k_p + k_q}$ | 0 | 0 |
| Расстояние между центрами кластеров (Centroid clustering), или центроидный метод. Недостатком этого метода является то, что центр объединенного кластера вычисляется как среднее центров объединяемых кластеров, без учета их объема | $\frac{k_p}{k_p + k_q}$ | $\frac{k_p}{k_p + k_q}$ | $\frac{-k_p k_q}{k_p + k_q}$ | 0 |

**Таблица 7.2** *(окончание)*

| Название метода | $\alpha_p$ | $\alpha_q$ | $\beta$ | $\gamma$ |
|---|---|---|---|---|
| Метод Уорда (Ward's method). В качестве расстояния между кластерами берется прирост суммы квадратов расстояний объектов до центров кластеров, получаемый в результате их объединения | $\frac{k_r + k_p}{k_r + k_p + k_q}$ | $\frac{k_r + k_p}{k_r + k_p + k_q}$ | $\frac{-k_r}{k_r + k_p + k_q}$ | 0 |

## Дивизимные алгоритмы

Дивизимные кластерные алгоритмы, в отличие от агломеративных, на первом шаге представляют все множество элементов $I$ как единственный кластер. На каждом шаге алгоритма один из существующих кластеров рекурсивно делится на два дочерних. Таким образом итерационно образуются кластеры сверху вниз. Этот подход не так подробно описывается в литературе по кластерному анализу, как агломеративные алгоритмы. Его применяют, когда необходимо разделить все множество объектов $I$ на относительно небольшое количество кластеров.

Один из первых дивизимных алгоритмов был предложен Смитом Макнаотоном в 1965 г.

На первом шаге все элементы помещаются в один кластер $C_1 = I$.

Затем выбирается элемент, у которого среднее значение расстояния от других элементов в этом кластере наибольшее. Среднее значение может быть вычислено, например, с помощью формулы:

$$D_{C1} = 1 / N_{C1} \times \sum \sum d(i_p, i_q) \forall i_p, i_q \in C.$$

Выбранный элемент удаляется из кластера $C_1$ и формирует первый член второго кластера $C_2$.

На каждом последующем шаге элемент в кластере $C_1$, для которого разница между средним расстоянием до элементов, находящихся в $C_2$, и средним расстоянием до элементов, остающихся в $C_1$, наибольшая, переносится в $C_2$.

Переносы элементов из $C_1$ в $C_2$ продолжаются до тех пор, пока соответствующие разницы средних не станут отрицательными, т. е. пока существуют элементы, расположенные к элементам кластера $C_2$ ближе, чем к элементам кластера $C_1$.

В результате один кластер делится на два дочерних, один из которых расщепляется на следующем уровне иерархии. Каждый последующий уровень применяет процедуру разделения к одному из кластеров, полученных на пре-

дыдущем уровне. Выбор расщепляемого кластера может выполняться по-разному.

В 1990 г. Кауфман и Роузеув предложили выбирать на каждом уровне кластер для расщепления с наибольшим диаметром, который вычисляется по формуле

$$D_C = \max d(i_p, i_q) \forall i_p, i_q \in C.$$

Рекурсивное разделение кластеров продолжается, пока все кластеры или не станут сиглетонами (т. е. состоящими из одного объекта), или пока все члены одного кластера не будут иметь нулевое отличие друг от друга.

## 7.3.3. Неиерархические алгоритмы

Большую популярность при решении задач кластеризации приобрели алгоритмы, основанные на поиске оптимального в определенном смысле разбиения множества данных на кластеры (группы). Во многих задачах в силу своих достоинств используются именно алгоритмы построения разбиения. Данные алгоритмы пытаются сгруппировать данные (в кластеры) таким образом, чтобы целевая функция алгоритма разбиения достигала экстремума (минимума). Рассмотрим три основных алгоритма кластеризации, основанных на методах разбиения [9]. В данных алгоритмах используются следующие базовые понятия:

- обучающее множество (входное множество данных) $M$, на котором строится разбиение;
- метрика расстояния:

$$d_A^2\left(m_j, c^{(i)}\right) = \left\| m_j - c^{(i)} \right\|_A^2 = \left(m_j - c^{(i)}\right)^t A\left(m_j - c^{(i)}\right), \qquad (7.6)$$

  где матрица $A$ определяет способ вычисления расстояния. Например, для единичной матрицы будем использовать расстояние по Евклиду;
- вектор центров кластеров $C$;
- матрица разбиения по кластерам $U$;
- целевая функция $J = J(M, d, C, U)$;
- набор ограничений.

*Исследуемые данные* представляются как множество векторов в многомерном пространстве, описывающих некоторые объекты предметной области. Каждый вектор состоит из набора координат, именуемых атрибутами. Каждый атрибут имеет свою природу (числовую или категориальную) и свое множество значений. Множество значений в случае числовой природы атрибута задается в виде одного или нескольких интервалов числовой оси. В случае категориальной природы атрибута множество значений задается перечис-

лением возможных значений. Большое значение при подготовке данных к кластеризации имеет их очистка и нормировка. Очистка данных производится как по входному множеству в целом, так и по множеству атрибутов. Очистка данных по множеству атрибутов необходима для исключения зависимых атрибутов, которые не окажут влияния на результат, но увеличат время, необходимое на обработку данных. Нормировка необходима для того, чтобы на результаты кластеризации не оказывали влияния различные области значений отдельных атрибутов. При этом числовые и категориальные атрибуты нормируются по-разному. В результате каждый атрибут представляется значением из единичного интервала.

*Метрика расстояния* — возможно важнейшее понятие, используемое в кластеризации. Именно с помощью расстояния между входными векторами определяется их сходство или различие. Есть множество способов вычисления расстояния: евклидово, Хемминга, расстояние Махаланобиса и др. Выбор способа вычисления расстояния зависит от природы исследуемых объектов и непосредственно влияет на результат.

*Целевая функция* — это функция, минимизация которой дает решение задачи кластеризации. Последовательность действий, реализующая поиск минимума целевой функции, является алгоритмом кластеризации.

*Матрица разбиения* — это основной результат неиерархической кластеризации. Матрица разбиения представляет собой таблицу, где каждая ячейка содержит значение функции принадлежности данного вектора заданному кластеру. На основании этой матрицы и получается итоговое разбиение. В большинстве алгоритмов помимо матрицы принадлежности в качестве результата порождается множество центров кластеров. Центр кластера — это вектор, степень принадлежности которого заданному кластеру максимальна. Как правило, центров кластеров нет в исходном множестве.

*Набор ограничений* связан с условиями, налагаемыми на значения элементов матрицы принадлежности. Эти ограничения определяются алгоритмом, используемым для кластеризации.

## Алгоритм *k*-means (Hard-c-means)

Рассмотрим более подробно алгоритм на примере данных из табл. 7.1. Для большей наглядности ограничимся двумя параметрами — длиной и шириной чашелистника. Это позволит представить данные в двумерном пространстве (рис. 7.4). Точки отмечены номерами объектов.

Вначале выбирается $k$ произвольных исходных центров — точек в пространстве всех объектов. Не очень критично, какие именно это будут центры, процедура выбора исходных точек отразится, главным образом, только на времени счета. Например, это могут быть первые $k$ объектов множества $I$. В данном примере это точки 1, 2 и 3.

Дальше итерационно выполняется операция двух шагов.

На первом шаге все объекты разбиваются на $k$ групп, наиболее близких к одному из центров. Близость определяется расстоянием, которое вычисляется одним из описанных ранее способов (например, берется евклидово расстояние). Рис. 7.5 иллюстрирует разбиение ирисов на три кластера.

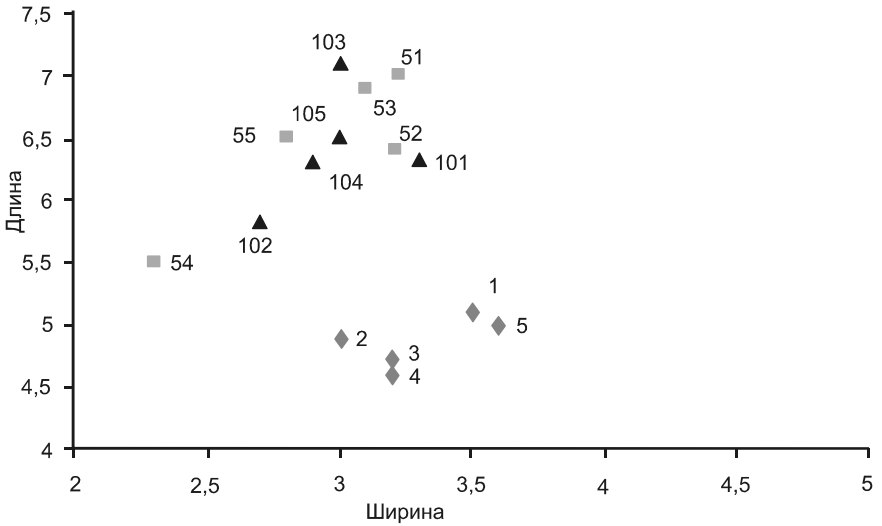



**Рис. 7.4.** Представление ирисов в двумерном пространстве

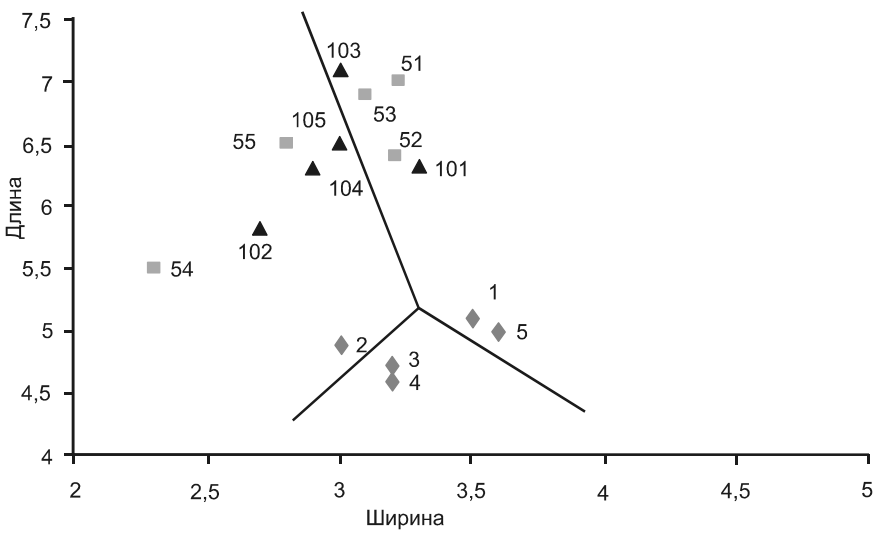



**Рис. 7.5.** Начальное разбиение ирисов на три кластера

На втором шаге вычисляются новые центры кластеров. Центры можно вычислить как средние значения переменных объектов, отнесенных к сформированным группам. Новые центры, естественно, могут отличаться от предыдущих. На рис. 7.6 отмечены новые центры и новое разделение в соответствии с ними. Естественно, что некоторые точки, ранее относящиеся к одному кластеру, при новом разбиении попадают в другой (в данном случае такими точками являются 1, 2, 5 и др.). Новые центры на рисунке помечены символом "x", обведенным в кружок.

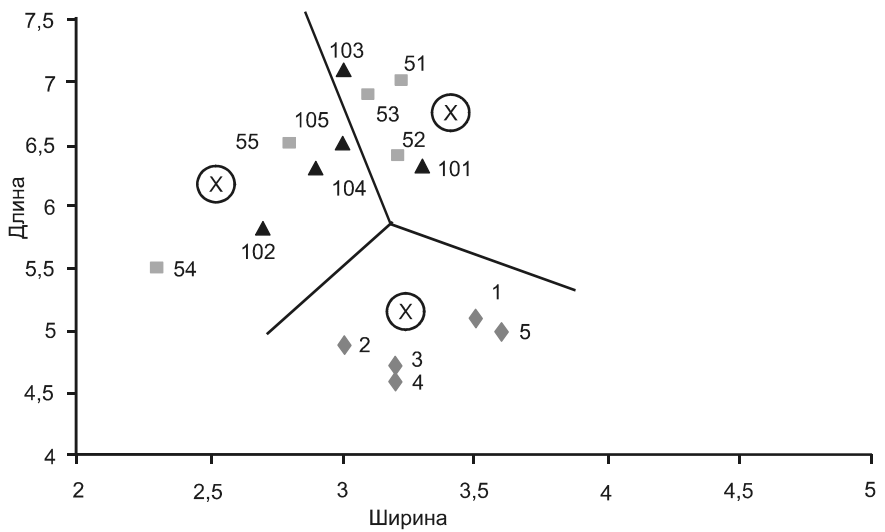



**Рис. 7.6.** Второе разбиение множества ирисов на кластеры

Рассмотренная операция повторяется рекурсивно до тех пор, пока центры кластеров (соответственно, и границы между ними) не перестанут меняться. В нашем примере второе разбиение является последним. Если посмотреть на результаты кластеризации, то можно заметить, что нижний кластер полностью совпадает с классом Iris setosa. В остальных кластерах имеются представители обоих классов. В данном случае это произошло по причине кластеризации только по двум параметрам, а не по всем четырем.

После того как объекты отражены в точки многомерного пространства, процедура автоматического разбиения на кластеры весьма проста. Проблема в том, что исходные объекты не всегда можно представить в виде точек. В геометрии все переменные равнозначны, в реальных же данных изменение одной из них на некоторое значение по смыслу может значить существенно больше, чем такое же изменение другой переменной. Действительные переменные можно преобразовать к примерно равнозначному масштабу, разделив на их характерный естественный масштаб или, если он неизвестен, на сред-

нее значение этой переменной, на диапазон ее изменения (разность между максимальным и минимальным значениями переменной) или на ее стандартное отклонение. Тогда геометрическое расстояние между точками будет примерно соответствовать интуитивным представлениям о близости записей. Введение метрики, расстояния между категориальными переменными или отношениями порядка несколько сложнее.

Необходимо отметить, что метод $k$ -средних хорошо работает, если данные по своей естественной природе делятся на компактные, примерно сферические группы.

Данный алгоритм является прообразом практически всех алгоритмов нечеткой кластеризации, и его формализация поможет лучшему пониманию принципов, заложенных в более сложные алгоритмы.

Базовые определения и понятия в рамках данного алгоритма имеют вид:

- обучающее множество $M = \{m_j\}_{j=1}^{d}$, $d$ — количество точек (векторов) данных;
- метрика расстояния, рассчитываемая по формуле (7.6);
- вектор центров кластеров $C = \{c^{(i)}\}_{i=1}^{c}$, где:

$$c^{(i)} = \frac{\sum_{j=1}^{d} u_{ij} m_j}{\sum_{j=1}^{d} u_{ij}}, \quad 1 \le i \le c; \tag{7.7}$$

- матрица разбиения $U = \{u_{ij}\}$, где:

$$u_{ij}^{(l)} = \begin{cases} 1 & \text{при } d(m_j, c_i^{(l)}) = \min\limits_{1 \le k \le c} d(m_j, c_k^{(l)}), \\ 0 & \text{в остальных случаях,} \end{cases} \tag{7.8}$$

- целевая функция:

$$J(M,U,C) = \sum_{i=1}^{c} \sum_{j=1}^{d} u_{ij} d_A^2\left(m_j, c^{(i)}\right); \tag{7.9}$$

- набор ограничений:

$$u_{ij} \in \{0,\ 1\}; \quad \sum_{i=1}^{c} u_{ij} = 1; \quad 0 < \sum_{j=1}^{d} u_{ij} < d, \tag{7.10}$$

который определяет, что каждый вектор данных может принадлежать только одному кластеру и не принадлежать остальным. В каждом кластере содержится не менее одной точки, но менее общего количества точек.

Конструктивно алгоритм представляет собой итерационную процедуру следующего вида.

**Шаг 1.** Проинициализировать начальное разбиение (например, случайным образом), выбрать точность $\delta$ (используется в условии завершения алгоритма), проинициализировать номер итерации $l = 0$.

**Шаг 2.** Определить центры кластеров по следующей формуле:

$$c_l^{(i)} = \frac{\sum_{j=1}^{d} u_{ij}^{(l-1)} \cdot m_j}{\sum_{j=1}^{d} u_{ij}^{(l-1)}}, \quad 1 \le i \le c. \tag{7.11}$$

**Шаг 3.** Обновить матрицу разбиения с тем, чтобы минимизировать квадраты ошибок, используя формулу

$$u_{ij}^{(l)} = \begin{cases} 1 & \text{при } d(m_j, c_i^{(l)}) = \min\limits_{1 \le k \le c} d(m_j, c_k^{(l)}), \\ 0 & \text{в остальных случаях.} \end{cases} \tag{7.12}$$

**Шаг 4.** Проверить условие $\left\| U^{(l)} - U^{(l-1)} \right\| < \delta$. Если условие выполняется, завершить процесс, если нет — перейти к шагу 2 с номером итерации $l = l + 1$.

Основной недостаток, присущий данному алгоритму в силу дискретного характера элементов матрицы разбиения, — большой размер пространства разбиения.

Одним из способов устранения данного недостатка является представление элементов матрицы разбиения числами из единичного интервала. То есть принадлежность элемента данных кластеру должна определяться функцией принадлежности — элемент данных может принадлежать нескольким кластерам с различной степенью принадлежности. Данный подход нашел свое воплощение в алгоритме нечеткой кластеризации Fuzzy C-Means.

## Алгоритм Fuzzy C-Means

Данный алгоритм является обобщением предыдущего алгоритма. Его отличие состоит в том, что кластеры теперь являются нечеткими множествами и каждая точка принадлежит различным кластерам с различной степенью принадлежности. Точка относится к тому или иному кластеру по критерию максимума принадлежности данному кластеру.

Базовые понятия в данном случае имеют вид:

- обучающее множество $M = \{m_j\}_{j=1}^{d}$, $d$ — количество точек (векторов) данных;

- метрика расстояния (см. формулу 7.6);
- вектор центров кластеров $C = \{c^{(i)}\}_{i=1}^{c}$, где:

$$c^{(i)} = \frac{\sum_{j=1}^{d} \left(u_{ij}\right)^{w} \cdot m_j}{\sum_{j=1}^{d} \left(u_{ij}\right)^{w}}, \quad 1 \le i \le c; \tag{7.13}$$

- матрица разбиения $U = \{u_{ij}\}$, где:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d_A^2(m_j, c^{(i)})}{d_A^2(m_j, c^{(k)})} \right)^{\frac{1}{w-1}}}; \tag{7.14}$$

- целевая функция:

$$J(M,U,C) = \sum_{i=1}^{c} \sum_{j=1}^{d} u_{ij}^{w} d_A^2 \left(m_j, c^{(i)}\right), \tag{7.15}$$

  где $w \in (1,\infty)$ — показатель нечеткости (взвешивающий коэффициент), регулирующий нечеткость разбиения. Обычно используется $w = 2$;

- набор ограничений:

$$u_{ij} \in [0,\ 1]; \quad \sum_{i=1}^{c} u_{ij} = 1; \quad 0 < \sum_{j=1}^{d} u_{ij} < d, \tag{7.16}$$

  который определяет, что каждый вектор данных может принадлежать различным кластерам с разной степенью принадлежности, сумма принадлежностей элемента данных всем кластерам пространства разбиения равна единице.

Конструктивно алгоритм представляет собой итерационную процедуру следующего вида.

**Шаг 1.** Выбрать количество кластеров $2 \le c \le d$.

**Шаг 2.** Выбрать скалярную метрику для отображения векторов данных на вещественную ось.

**Шаг 3.** Выбрать параметр остановки $\delta$.

**Шаг 4.** Выбрать коэффициент нечеткости $w \in (1,\infty)$, например $w = 2$.

**Шаг 5.** Проинициализировать матрицу разбиения (например, случайными значениями).

**Шаг 6.** Вычислить прототипы (центры) кластеров по формуле:

$$c_l^{(i)} = \frac{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w m_j}{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w}, \quad 1 \le i \le c\,. \tag{7.17}$$

**Шаг 7.** Для всех элементов данных высчитать квадраты расстояний до всех (центров) кластеров по формуле:

$$d_A^2(m_j, c_l^{(i)}) = (c_l^{(i)} - m_j)^t\, A(c_l^{(i)} - m_j)\,. \tag{7.18}$$

**Шаг 8.** Обновить матрицу разбиения по следующей формуле:

$$u_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d_A^2(m_j, c^{(i)})}{d_A^2(m_j, c^{(k)})} \right)^{\frac{1}{w-1}}} \text{ для всех } 1 \le i \le c\,,\ 1 \le j \le d\,, \tag{7.19}$$

учитывая ограничения из формулы (7.16).

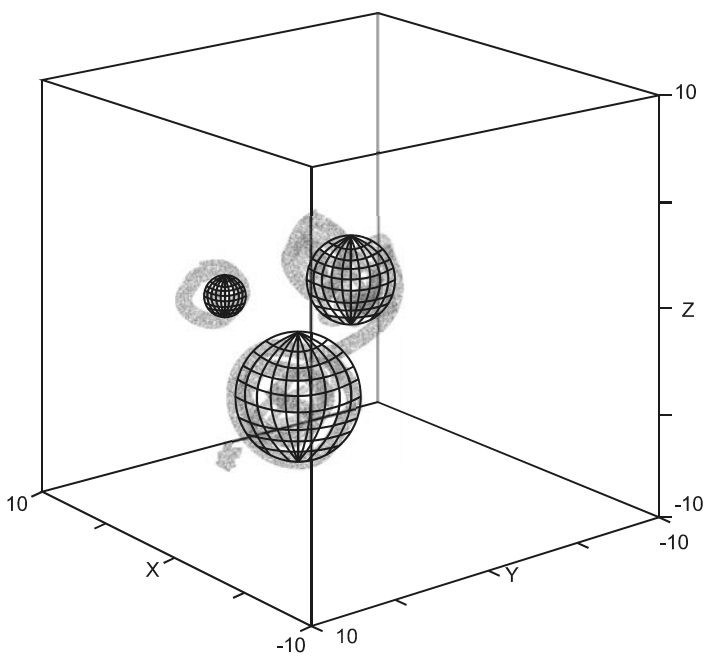



**Рис. 7.7.** Форма кластеров в алгоритме Fuzzy C-Means

**Шаг 9.** Проверить условие $\left\|U^{(l)} - U^{(l-1)}\right\| < \delta$. Если условие выполняется, завершить процесс, если нет — перейти к шагу 7 с номером итерации $l = l + 1$.

Данный алгоритм имеет преимущества перед алгоритмом $k$-means, но обладает тем недостатком, что ищет кластеры сферической формы (рис. 7.7), что подходит далеко не для всех задач и поэтому зачастую неоправданно огрубляет результаты. От данного недостатка свободен следующий алгоритм.

## Кластеризация по Гюстафсону-Кесселю

Данный алгоритм нечеткой кластеризации ищет кластеры в форме эллипсоидов (рис. 7.8), что делает его более гибким при решении различных задач.

Перед тем как описать данный алгоритм, обозначим дополнительные понятия, используемые в алгоритме.

Кластер характеризуется не только своим центром, но и ковариационной матрицей:

$$F^{(i)} = \frac{\sum_{j=1}^{d} (u_{ij})^w (m_j - c^{(i)})(m_j - c^{(i)})^t}{\sum_{j=1}^{d} (u_{ij})^w}, \qquad (7.20)$$

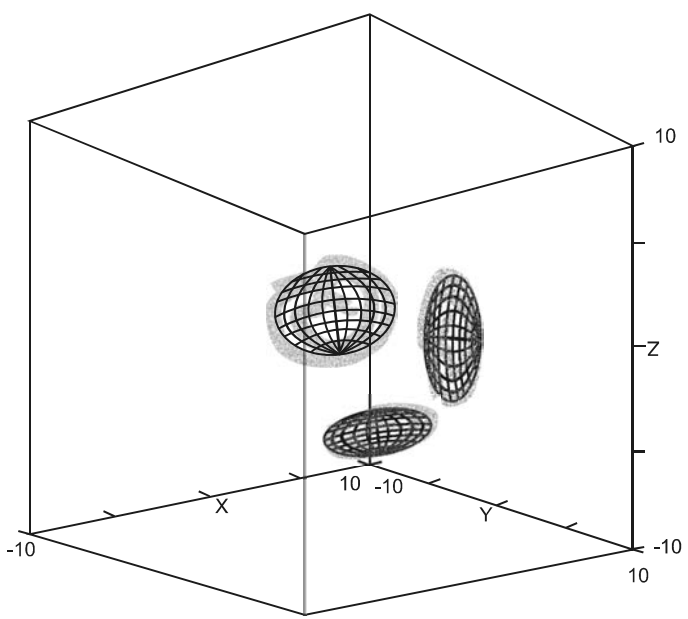



**Рис. 7.8.** Форма кластера в алгоритме кластеризации по Гюстафсону-Кесселю

где $\lambda_{ik}$ обозначает $k$-е собственное значение матрицы $F^{(i)}$, а $\Phi_{ik}$ — $k$-й единичный собственный вектор $F^{(i)}$. Собственные значения $\lambda_{ik}$ упорядочены в порядке убывания. Тогда собственные векторы $\Phi_{i1}$ ... $\Phi_{i(n-1)}$ охватывают линейное подпространство $i$-го кластера, а $\Phi_{in}$ является нормалью к этому линейному подпространству. Все изложенное иллюстрирует рис. 7.9.

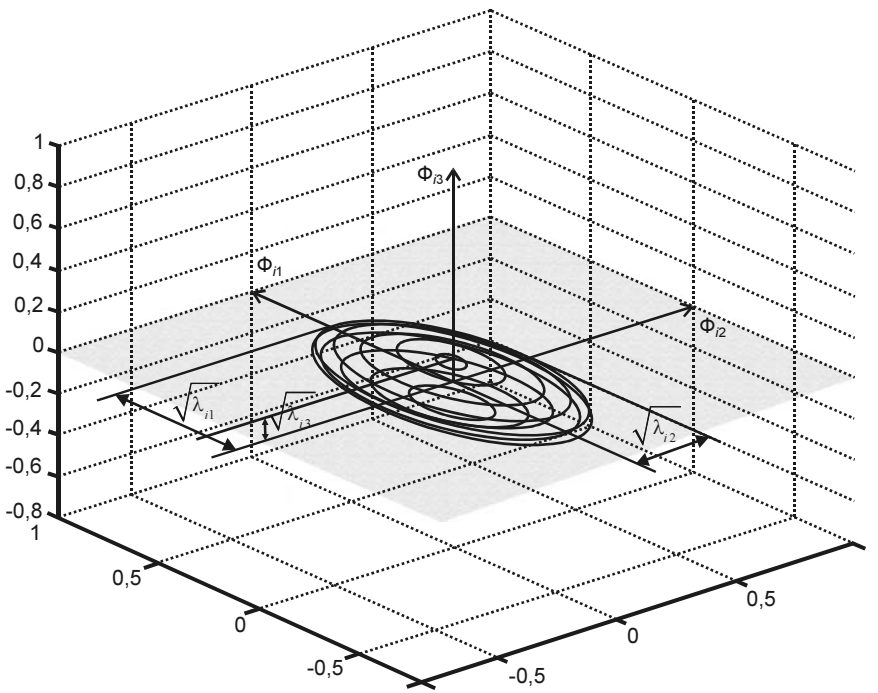



**Рис. 7.9.** Геометрическая иллюстрация к алгоритму кластеризации по Гюстафсону-Кесселю

Данный алгоритм использует свою нормирующую матрицу для вычисления расстояния. Выражения для вычисления выглядят следующим образом:

$$d^2_{A^{(i)}} = \left(c^{(i)} - m_j\right)^t A^{(i)} \left(c^{(i)} - m_j\right), \tag{7.21}$$

где:

$$A^{(i)} = \left(\left|F^{(i)}\right|\right)^{\frac{1}{r+1}} \left(F^{(i)}\right)^{-1}, \tag{7.22}$$

а $F^{(i)}$ определяется по формуле (7.20).

Обобщим базовые понятия:

- обучающее множество $M = \{m_j\}_{j=1}^{d}$, $d$ — количество точек (векторов) данных;
- метрика расстояния, вычисляемая по формулам (7.21) и (7.22);
- вектор центров кластеров $C = \{c^{(i)}\}_{i=1}^{c}$, где:

$$c^{(i)} = \frac{\sum_{j=1}^{d} \left(u_{ij}\right)^{w} m_j}{\sum_{j=1}^{d} \left(u_{ij}\right)^{w}}, \quad 1 \le i \le c; \tag{7.23}$$

- матрица разбиения $U = \{u_{ij}\}$, где:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d^2_{A^{(i)}}(m_j, c^{(i)})}{d^2_{A^{(i)}}(m_j, c^{(k)})} \right)^{\frac{1}{w-1}}}; \tag{7.24}$$

- целевая функция:

$$J(M,U,C) = \sum_{i=1}^{c} \sum_{j=1}^{d} u_{ij}^{w} d^2_{A^{(i)}} \left(m_j, c^{(i)}\right), \tag{7.25}$$

  где $w \in (1,\infty)$ — показатель нечеткости (взвешивающий коэффициент);

- набор ограничений:

$$u_{ij} \in [0,\ 1]; \quad \sum_{i=1}^{c} u_{ij} = 1; \quad 0 < \sum_{j=1}^{d} u_{ij} < d, \tag{7.26}$$

  который означает, что каждый вектор данных может принадлежать различным кластерам с разной степенью принадлежности, суммарная принадлежность элемента данных всем кластерам пространства разбиения равна единице.

Конструктивно алгоритм выглядит следующим образом.

**Шаг 1.** Определить количество кластеров $2 \le c \le d$.

**Шаг 2.** Определить критерий остановки $\delta > 0$.

**Шаг 3.** Определить параметр нечеткости $w \in (1,\infty)$, например 2.

**Шаг 4.** Проинициализировать матрицу разбиения, например случайными значениями.

**Шаг 5.** Рассчитать прототипы кластеров по формуле:

$$c_l^{(i)} = \frac{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w m_j}{\sum_{j=1}^{d} \left(u_{ij}^{(l-1)}\right)^w}, \quad 1 \le i \le c\,. \tag{7.27}$$

**Шаг 6.** Рассчитать ковариационные матрицы кластеров по формуле:

$$F^{(i)} = \frac{\sum_{j=1}^{d} (u_{ij}^{(l-1)})^w (m_j - c^{(i)})(m_j - c^{(i)})^t}{\sum_{j=1}^{d} (u_{ij}^{(l-1)})^w}. \tag{7.28}$$

**Шаг 7.** Рассчитать расстояния по формуле:

$$d^2_{F^{(i)}}\left(c_l^{(i)}, m_j\right) = \left(c_l^{(i)} - m_j\right)^t \left[\left|F^{(i)}\right|^{\frac{1}{r+1}} \left(F^{(i)}\right)^{-1}\right] \left(c_l^{(i)} - m_j\right). \tag{7.29}$$

**Шаг 8.** Обновить матрицу разбиения по формуле:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d^2_{F^{(i)}}(m_j, c^{(i)})}{d^2_{F^{(i)}}(m_j, c^{(k)})} \right)^{\frac{1}{w-1}}} \tag{7.30}$$

с учетом следующих ограничений из формулы (7.26).

**Шаг 9.** Проверить условие $\left\| U^{(l)} - U^{(l-1)} \right\| < \delta$. Если условие выполняется, завершить процесс, если нет — перейти к шагу 5 с номером итерации $l = l + 1$.

В заключение данного этапа следует отметить, что приведенные алгоритмы не отличаются друг от друга подходом к кластеризации. Это становится очевидным при сравнении целевых функций, минимизация которых составляет сущность данных алгоритмов. Отличие заключается лишь в разных способах вычисления расстояний между точками в пространстве входных данных. Алгоритмы расположены в порядке их усложнения. Так, каждый последующий алгоритм пытается учитывать все больше аспектов взаимосвязи данных. Существует некоторое количество алгоритмов, подобных описанным, единственное отличие которых заключается в дополнительных слагаемых целевой функции, которые учитывают некоторые другие аспекты взаимосвязи данных (взаимное расположение кластеров, допущение о случайном характере рас-

пределения точек внутри кластера, учет принадлежности тому или иному кластеру ближайших соседей данной точки и др.). Однако важно заметить, что неизменным слагаемым в этих целевых функциях является определенная Бездеком двойная сумма $\sum_{i=1}^{c}\sum_{j=1}^{d} u_{ij}^{w} d_{A^{(i)}}^{2}\left(m_j, c^{(i)}\right)$, что свидетельствует о неизменности основных допущений, на основании которых построены целевые функции. Основные из этих допущений выглядят следующим образом:

- кластеры в общем случае имеют форму эллипсоида;
- из предыдущего пункта следует, что у кластера всегда есть центр;
- отнесение точек к кластерам (разбиение) базируется на некотором расстоянии точек до центров кластера.

Уже этих трех пунктов достаточно для определения недостатков данных алгоритмов:

- допущение о том, что все кластеры всегда имеют некоторую, определяемую алгоритмом форму, а это, очевидно, далеко не всегда выполняется. Аппроксимация пространства входных данных некоторыми заданными фигурами на данных, имеющих сложное взаимное расположение, может привести к неинтерпретируемым результатам;
- допущение о том, что в кластере всегда есть некоторая узловая точка (центр кластера), степень принадлежности которой кластеру равна единице, в то время как остальные точки (не равные центру кластера) не могут принадлежать кластеру с такой же высокой степенью принадлежности, что, опять же, при сложном взаимном расположении точек данных является неприемлемым;
- данные алгоритмы строятся не на основе взаимного расположения точек, а лишь на отношении точек к центрам кластеров.

Интересной иллюстрацией слабых сторон подобных алгоритмов кластеризации является случай, когда входные данные имеют форму двух вложенных сфер. Алгоритм Fuzzy C-Means, строящий сферические кластеры, ни при каких условиях не разобьет пространство данных на два кластера, содержащих эти сферы.

Из перечисленных недостатков следует, что необходимо разработать такой подход к кластеризации данных, который бы учитывал взаимосвязь между точками данных, а не взаимосвязь точек и центров кластеров (которых в общем случае может в принципе не существовать). Такой подход может быть получен при помощи аппарата нечетких отношений, который еще не нашел широкого применения в алгоритмах кластеризации.

Для реализации указанного подхода необходимо исследовать как способы построения нечетких отношений, так и их свойства. Желательным результатом явилось бы получение аналога понятия классов эквивалентности, существующего в теории множеств для случая нечетких отношений.

# 7.4. Адаптивные методы кластеризации

## 7.4.1. Выбор наилучшего решения и качество кластеризации

В предыдущем разделе были рассмотрены различные методы кластеризации. Основным результатом любого из них является набор кластеров. Для того чтобы алгоритм кластеризации построил этот набор, необходимо знать количество кластеров. Меняя его, можно получить множество равноценных (с формальной точки зрения) результатов. Тем не менее подразумевается, что существует небольшое количество практически полезных решений задачи кластеризации (чаще всего одно) для заданного множества данных. Поэтому, когда о количестве кластеров нет информации (это самая распространенная ситуация), возникает проблема выбора наилучшего разбиения, а это нетривиальная задача. Облегчить ее решение можно, добавив в алгоритм кластеризации некоторый адаптивный механизм выбора оптимального решения среди множества возможных. Выбор оптимального решения будем основывать на понятии качества кластеризации. Качеством кластеризации назовем степень приближения результата кластеризации к идеальному решению. Поскольку идеальное решение задачи кластеризации неизвестно, то оценить качество можно двумя способами — экспертным и формальным. Экспертный выбор наилучшего решения задачи заключается в оценке решения специалистами в данной предметной области. Но экспертная оценка зачастую объективно невозможна из-за большого объема и сложности данных. Поэтому важную роль играют формальные критерии оценки качества кластеризации.

## 7.4.2. Использование формальных критериев качества в адаптивной кластеризации

Формальные критерии оценивают качество кластеризации по некоторому показателю, вычисленному на основании результатов кластеризации. Наилучшим в терминах выбранного критерия является решение, для которого значение критерия достигает экстремального значения.

Адаптивная составляющая хорошо сочетается с неиерархическими алгоритмами, особенно с алгоритмами нечеткой кластеризации. Алгоритмы неиерархической кластеризации, как правило, реализуют итерационную процедуру

приближения к решению задачи. Типовая процедура поиска решения уже была изложена в *разд. 7.3.3* (например, Fuzzy C-Means). В результате решения основным результатом является матрица принадлежности — на ее основе получается разбиение на кластеры. Другим важным результатом является множество центров кластеров — векторов, принадлежность которых соответствующим кластерам максимальна. Таким образом, для построения критерия необходимо использовать один или оба этих результата. Построив критерий (или систему критериев), можно будет применять адаптивный механизм кластеризации.

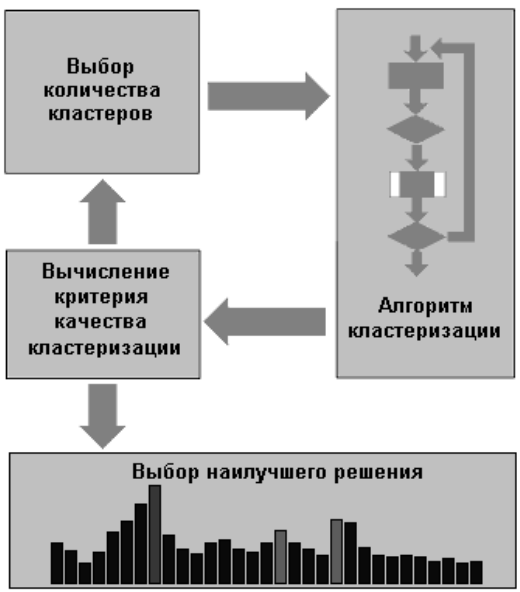



**Рис. 7.10.** Обобщенная схема процедуры адаптивной кластеризации

Ключевым элементом в адаптивной кластеризации является выбор критерия, по которому будет оцениваться качество кластеризации. Приведем некоторые из них.

## Показатели четкости

Показатели четкости достигают максимума при наиболее четком разбиении.

- *Коэффициент разбиения*:

$$PC = \frac{\sum_{q=1}^{Q}\sum_{k=1}^{K} u_{ij}^2}{Q}, \quad PC \in \left[\frac{1}{K}, 1\right].$$

- *Модифицированный коэффициент разбиения*:

$$PC_M = \frac{\sum_{q=1}^{Q}\sum_{k=1}^{K} u_{ij}^2}{Q} - \frac{1}{K}, \quad PC \in \left[0, \frac{K-1}{K}\right].$$

Оба критерия обладают общим недостатком — их области значений напрямую зависят от выбранного количества кластеров. Модифицированный коэффициент разбиения, обеспечивая примерно одинаковую шкалу значений критерия в ее начале, снижает влияние этого недостатка, т. к. на практике оптимальное количество кластеров обычно намного меньше количества элементов исходного множества.

- *Индекс четкости*:

$$CI = \frac{K \cdot PC - 1}{K - 1}, \quad CI \in [0,1].$$

## Энтропийные критерии

Энтропия известна как численное выражение упорядоченности системы. Энтропия разбиения достигает минимума при наибольшей упорядоченности в системе (в случае четкого разбиения энтропия равна нулю). То есть чем больше степень принадлежности элемента одному кластеру (и меньше степень принадлежности всем остальным кластерам), тем меньше значение энтропии и тем более качественно выполнена кластеризация.

- *Энтропия разбиения*:

$$PE = -\frac{\sum_{q=1}^{Q}\sum_{k=1}^{K} u_{qk} \ln(u_{qk})}{Q}, \quad PE \in [0, \ln K].$$

Анализируя формулу и учитывая свойства функции принадлежности, очевидно, что в общем случае разбиение на меньшее количество кластеров даст меньшее значение энтропии. Чтобы учесть этот факт, данный критерий видоизменяют.

- *Модифицированная энтропия*:

$$PE_M = \frac{\sum_{q=1}^{Q}\sum_{k=1}^{K} u_{qk} \ln(u_{qk})}{Q \ln K} = \frac{PE}{\ln K}, \quad PE_M \in [0,1].$$

### Другие критерии

- *Показатель компактности и изолированности*:

$$CS = \frac{\sum_{q=1}^{Q}\sum_{k=1}^{K} u_{qk}^2 \cdot d^2(x_q, c_k)}{Q \cdot \min\left\{ d^2(c_i, c_j) \middle| i, j \in \overline{1,K}, i \neq j \right\}}.$$

  Меньшие значения этого индикатора соответствуют более компактным, хорошо отделимым кластерам.

- *Индекс эффективности*.

  Максимум этого критерия даст оптимальное количество кластеров. Критерий строится из двух составных частей:

  - межкластерные отличия (велики при оптимальном $K$):

$$\sum_{k=1}^{K}\sum_{q=1}^{Q} u_{qk}^2 d^2(c_k, \overline{x}),$$

  - внутрикластерные отличия (малы при оптимальном $K$):

$$\sum_{k=1}^{K}\sum_{q=1}^{Q} u_{qk}^2 d^2(x_q, c_k).$$

  Комбинируя эти части, получаем критерий:

$$PI = \sum_{k=1}^{K}\sum_{q=1}^{Q} u_{qk}^2 \left( d^2(c_k, \overline{x}) - d^2(x_q, c_k) \right).$$

  Здесь $\overline{x}$ — среднее арифметическое всех входных векторов.

## 7.4.3. Пример адаптивной кластеризации

Для иллюстрации использования адаптивной кластеризации приведем пример. Исходными данными является множество из табл. 7.1 — классический пример, используемый для проверки методов анализа данных. Множество состоит из 3 классов по 50 элементов в каждом. Каждый из классов — это некоторый вид ириса. Один класс линейно отделим от двух других. Другие два класса линейно неотделимы друг от друга. Каждый входной вектор имеет четыре атрибута:

- длина чашелистника (в сантиметрах);
- ширина чашелистника (в сантиметрах);

- длина лепестка (в сантиметрах);
- ширина лепестка (в сантиметрах).

Иллюстрация четырех проекций данных в трехмерное пространство представлена на рис. 7.11.

Критериями качества выберем два из приведенных критериев: модифицированную энтропию и индекс эффективности. При помощи адаптивной процедуры кластеризации будем осуществлять поиск оптимального количества кластеров. Диапазон поиска выбран из общих рекомендаций, которые говорят о том, что минимальное количество кластеров равно двум, а максимальное — порядка квадратного корня из мощности входного множества. Будем использовать евклидово расстояние. На рис. 7.12—7.15 показаны зависимости значений критериев от количества кластеров. Красным столбцом показаны экстремальные значения критериев.

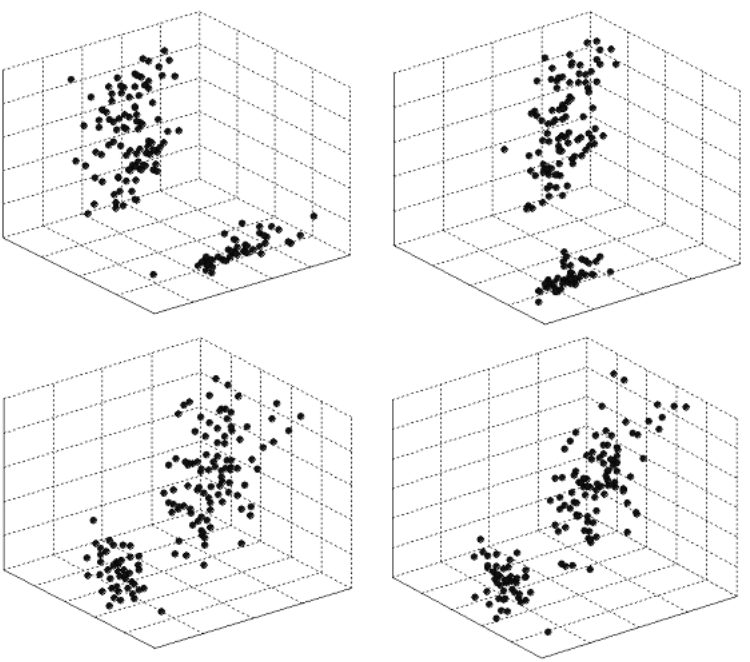


**Рис. 7.11.** Четыре проекции данных в трехмерном пространстве

Из приведенных рисунков видно, что критерии указывают на разное значение кластеров. В данном случае индекс эффективности и модифицированный коэффициент разбиения показали лучшие результаты, сумев различить все три

кластера, которые есть во входных данных, в том числе и два линейно неразделимых кластера. Тем не менее, в других задачах использование этих критериев может дать другой результат.

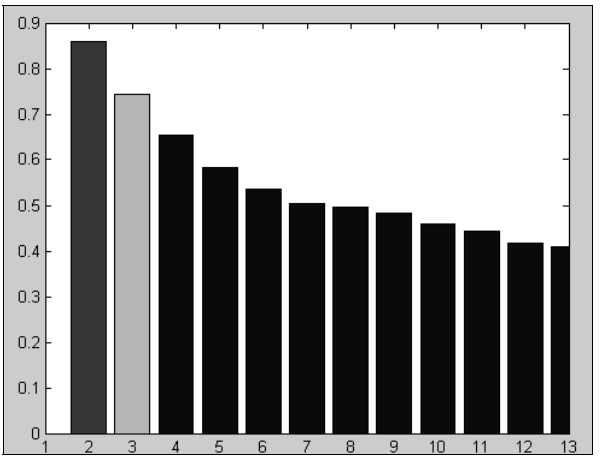



**Рис. 7.12.** Зависимость значений критериев от количества кластеров. Коэффициент разбиения

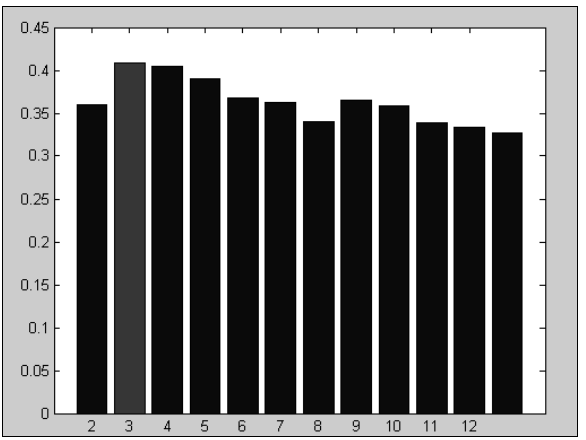



**Рис. 7.13.** Зависимость значений критериев от количества кластеров. Модифицированный коэффициент разбиения

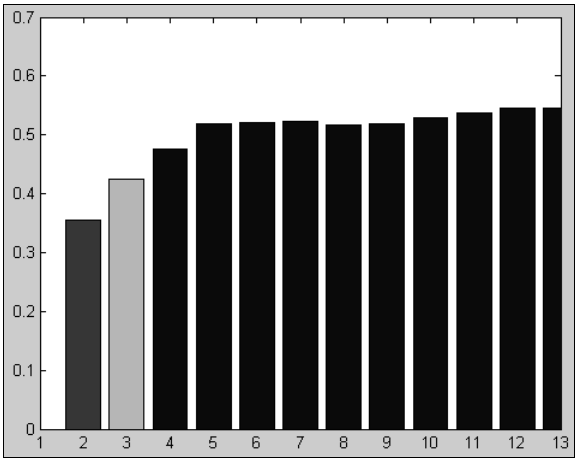



**Рис 7.14.** Зависимость значений критериев от количества кластеров. Модифицированная энтропия разбиения

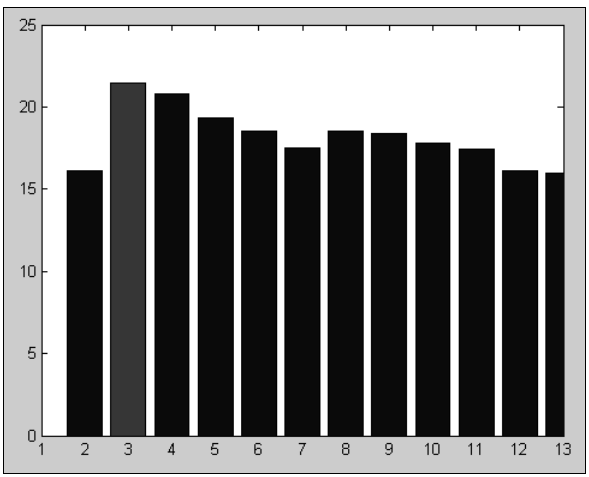



**Рис. 7.15.** Зависимость значений критериев от количества кластеров. Индекс эффективности

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Задача кластеризации состоит в разделении исследуемого множества объектов на группы похожих объектов, называемых кластерами.

- Для определения "похожести" объектов вводится мера близости, называемая расстоянием. Существуют разные способы вычисления расстояний: евклидово, манхеттенское, Чебышева и др.
- Результаты кластеризации могут быть представлены разными способами. Одним из наиболее популярных является дендрограмма — отображение последовательного процесса кластеризации.
- Базовые методы кластеризации делятся на иерархические и неиерархические. Первые строят дендрограммы или снизу вверх (агломеративные), или сверху вниз (дивизимные).
- Наиболее популярный из неиерархических алгоритмов — алгоритм $k$-средних и его разновидности. Идея метода заключается в определении центров $k$ кластеров и отнесения к каждому кластеру объектов, наиболее близко находящихся к этим центрам.
- Применение адаптивной кластеризации может помочь более эффективно решать задачу кластеризации и более взвешенно подходить к оценке результата. Тем не менее, выбор критерия оценки качества может оказаться критичным для решения задачи.

ГЛАВА 8

# Визуальный анализ данных — Visual Mining

## 8.1. Выполнение визуального анализа данных

Результаты, получаемые при анализе данных с помощью методов Data Mining, не всегда удобны для восприятия человеком. Во множестве классификационных или ассоциативных правил, в математических формулах человеку достаточно сложно быстро и легко найти новые и полезные знания. Из-за сложности информации это не всегда возможно и в простейших графических видах представления знаний, таких как деревья решений, дейтограммы, двумерные графики и т. п. В связи с этим возникает необходимость в более сложных средствах отображения результатов анализа. К ним относятся средства визуального анализа данных, которые в зарубежной литературе часто называют термином Visual Mining.

Основной идеей визуального анализа данных является представление данных в некоторой визуальной форме, позволяющей человеку погрузиться в данные, работать с их визуальным представлением, понять их суть, сделать выводы и напрямую взаимодействовать с данными.

До недавнего времени визуальный анализ данных для отображения результатов на обычных мониторах использовал только двумерную или очень простую трехмерную графику. Более сложные графические образы отображать в реальном времени было достаточно сложно и дорого. Однако прогресс в области аппаратных средств вывода изображений способствовал и совершенствованию средств визуального анализа данных. В настоящее время существует достаточно большое количество различных видов графических образов, позволяющих представлять результаты анализа в виде, удобном для понимания человеком.

С помощью новых технологий пользователи способны оценивать: большие объекты или маленькие, далеко они находятся или близко. Пользователь в реальном времени может двигаться вокруг объектов или кластеров объектов и рассматривать их со всех сторон. Это позволяет использовать для анализа естественные человеческие перцепционные навыки в обнаружении неопределенных образцов в визуальном трехмерном представлении данных.

Визуальный анализ данных особенно полезен, когда о самих данных мало известно и цели исследования до конца непонятны. За счет того, что пользователь напрямую работает с данными, представленными в виде визуальных образов, которые он может рассматривать с разных сторон и под любыми углами зрения, в прямом смысле этого слова, он может получить дополнительную информацию, которая поможет ему более четко сформулировать цели исследования.

Таким образом, визуальный анализ данных можно представить как процесс генерации гипотез. При этом сгенерированные гипотезы можно проверить или автоматическими средствами (методами статистического анализа или методами Data Mining), или средствами визуального анализа. Кроме того, прямое вовлечение пользователя в визуальный анализ имеет два основных преимущества перед автоматическими методами:

- визуальный анализ данных позволяет легко работать с неоднородными и зашумленными данными, в то время как не все автоматические методы могут работать с такими данными и давать удовлетворительные результаты;
- визуальный анализ данных интуитивно понятен и не требует сложных математических или статистических алгоритмов.

Следствием этих преимуществ является то, что визуальный анализ выполняется быстрее и в некоторых случаях дает лучший результат, чем автоматические методы анализа.

Еще одним важным достоинством визуального анализа является высокая степень конфиденциальности полученных сведений, т. к. они целиком сосредоточены в голове аналитика и не сохраняются даже в оперативной памяти компьютера.

Все перечисленные преимущества позволяют еще больше облегчить работу аналитика и повысить качество получаемых знаний при совместном использовании визуального анализа и методов автоматического анализа.

Визуальный анализ данных обычно выполняется в три этапа:

- беглый анализ — позволяет идентифицировать интересные шаблоны и сфокусироваться на одном или нескольких из них;

- увеличение и фильтрация — идентифицированные на предыдущем этапе шаблоны отфильтровываются и рассматриваются в большем масштабе;
- детализация по необходимости — если пользователю нужно получить дополнительную информацию, он может визуализировать более детальные данные.

Процесс визуализации изображен на рис. 8.1. Так же как и при анализе данных, информация извлекается из некоторого источника, например, из базы данных или из файлов. Затем к ней могут быть применены алгоритмы Data Mining для выявления скрытых закономерностей (классификации, кластеризации и т. п.). Как результаты применения алгоритмов, так и исходные данные подвергаются обработке в ядре визуализации. Основной целью обработки является приведение многомерных данных к такому виду, который можно было бы представить на экране монитора. Виды визуальных образов будут описаны в следующих разделах.

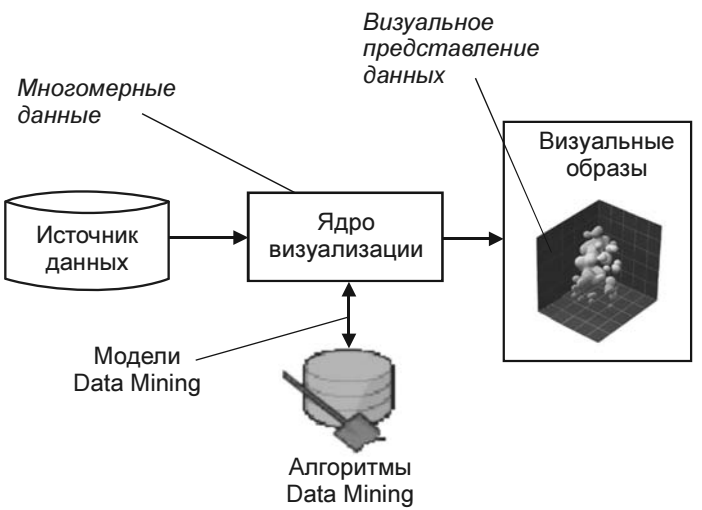



**Рис. 8.1.** Процесс визуализации данных

# 8.2. Характеристики средств визуализации данных

Существует достаточно большое количество средств визуализации данных, предоставляющих различные возможности. Для выбора таких средств рассмотрим более подробно три основные характеристики, описанные в статье [39]:

- характер данных, которые нужно визуализировать с помощью данного средства;

- методы визуализации и образы, в виде которых могут быть представлены данные;
- возможности взаимодействия с визуальными образами и методами для лучшего анализа данных.

Наборы визуализируемых данных, как и в Data Mining, представляют собой матрицы, в которых ряды являются данными (например, записями об экспериментах, покупки в магазине и т. п.), а колонки — атрибутами данных. При этом данные могут характеризоваться одним или несколькими атрибутами. Кроме того, сами данные могут иметь более сложную структуру: иерархическую, текстовую, графическую и т. п. Таким образом, выделяют следующие виды данных, с которыми могут работать средства визуализации:

- одномерные данные — одномерные массивы, временные ряды и т. п.;
- двумерные данные — точки двумерных графиков, географические координаты и т. п.;
- многомерные данные — финансовые показатели, результаты экспериментов и т. п.;
- тексты и гипертексты — газетные статьи, Web-документы и т. п.;
- иерархические и связанные — структура подчиненности в организации, электронная переписка людей, гиперссылки документов и т. п.;
- алгоритмы и программы — информационные потоки, отладочные операции и т. п.

*Одномерные данные* обычно характеризуются одним атрибутом. Типичным примером одномерных данных являются хронологические данные (временные ряды), где единственным атрибутом является время. Необходимо обратить внимание, что с одной отметкой времени может быть связано одно или несколько значений данных.

*Двумерные данные* имеют два отдельных измерения. Типичным примером могут служить географические данные, характеризующиеся двумя атрибутами: долгота и широта. Обычно такие данные отображаются в виде точек в двумерной системе координат.

Кажущаяся простота визуализации одномерных и двумерных данных может быть обманчива. При большом объеме данных их визуализация может не дать желаемого эффекта. Например, при визуализации большого количества данных на длинном временном интервале, отображение их на экран приведет к необходимости значительного уменьшения масштаба и, как следствие, потери наглядности.

Многие наборы данных характеризуются более чем тремя измерениями, из-за чего просто отобразить их в виде двумерных или даже трехмерных точек не-

возможно. Примерами *многомерных данных* являются реляционные таблицы баз данных, где атрибутами данных являются колонки. Для отображения таких данных на экране монитора требуются специальные методы визуализации, например, параллельные координаты *(см. разд. 8.3.1)*.

Не все данные могут быть описаны в терминах измерений. К таким данным относятся *тексты, гипертексты* и т. п. Они характеризуются тем, что не могут быть описаны количественными характеристиками, и, как следствие, не все методы визуализации могут быть для них использованы. Поэтому перед визуализацией требуются дополнительные преобразования, подготавливающие тексты к виду, пригодному для использования методов визуализации.

Данные могут состоять в некоторых отношениях с другими частями информации. Для представления таких взаимосвязей широко используются *графы*. Граф состоит из набора узлов и соединяющих их линий, называемых *дугами*. Примерами таких данных могут являться: электронная почта, пересылаемая между людьми, гиперссылки между Web-документами, файловая структура диска и т. п. Для отображения таких связей используются специальные методы визуализации.

Другой класс данных — *алгоритмы и программы*. Копирование больших программных проектов является проблемой. Целью визуализации является поддержка в разработке программных средств, например, отображение потока информации в программе для лучшего понимания алгоритмов и написанного кода, отображение структуры тысячи строк исходного кода в виде графа и поддержки процесса отладки, визуализация ошибок. Существует большое количество инструментов и систем, которые реализуют эти задачи.

Для визуализации перечисленных типов данных используются различные визуальные образы и методы их создания [39, 40]. Очевидно, что количество визуальных образов, которыми могут представляться данные, ограничивается только человеческой фантазией. Основное требование к ним — это наглядность и удобство анализа данных, которые они представляют. Методы визуализации могут быть как самые простые (линейные графики, диаграммы, гистограммы и т. п.), так и более сложные, основанные на сложном математическом аппарате. Кроме того, при визуализации могут использоваться комбинации различных методов. Выделяют следующие типы методов визуализации:

- стандартные 2D/3D-образы — гистограммы, линейные графики и т. п.;
- геометрические преобразования — диаграмма разброса данных, параллельные координаты и т. п.;
- отображение иконок — линейчатые фигуры (needle icons) и звезды (star icons);

- методы, ориентированные на пикселы — рекурсивные шаблоны, циклические сегменты и т. п.;
- иерархические образы — древовидные карты и наложение измерений.

Простейшие методы визуализации, к которым относятся *2D/3D-образы*, широко используются в существующих системах (например, в Microsoft Excel). К этим методам относятся: графики, диаграммы, гистограммы и т. п. Основным их недостатком является невозможность приемлемой визуализации сложных данных и большого количества данных.

Методы *геометрических преобразований* визуальных образов направлены на трансформацию многомерных наборов данных с целью отображения их в декартовом и в недекартовом геометрических пространствах. Данный класс методов включает в себя математический аппарат статистики. К нему относятся такие популярные методы, как диаграмма разброса данных (scatter plot), параллельные координаты (Parallel Coordinates), гипердоли (Hyperslice) и др. Более подробно эти методы будут описаны в *разд. 8.3.1*.

Другим классом методов визуализации данных являются методы *отображения иконок*. Их основной идеей является отображение значений элементов многомерных данных в свойства образов. Такие образы могут представлять собой: человеческие лица, стрелки, звезды и т. п. Визуализация генерируется отображением атрибутов элементов данных в свойства образов. Такие образы можно группировать для целостного анализа данных. Результирующая визуализация представляет собой шаблоны текстур, которые имеют различия, соответствующие характеристикам данных и обнаруживаемые преаттентивным[1] восприятием. Более подробно эти методы будут описаны в *разд. 8.3.2*.

Основной идеей методов, *ориентированных на пикселы*, является отображение каждого измерения значения в цветной пиксел и их группировка по принадлежности к измерению. Так как один пиксел используется для отображения одного значения, то, следовательно, данный метод позволяет визуализировать большое количество данных (свыше одного миллиона значений). Более подробно эти методы будут рассматриваться в *разд. 8.3.3*.

Методы *иерархических образов* предназначены для представления данных, имеющих иерархическую структуру. В случае многомерных данных должны быть правильно выбраны измерения, которые используются для построения иерархии. Более подробно эти методы описаны в *разд. 8.3.4*.

В результате применения методов визуализации будут построены визуальные образы, отражающие данные. Однако этого не всегда бывает достаточно для

[1] *Преаттентивным восприятием* называется способность человека обрабатывать (воспринимать) информацию, не сознавая, что он вообще обратил на нее внимание. В процессе такой обработки информации неосознанно формируется и отношение к ней.

полного анализа. Пользователь должен иметь возможность работать с образами: видеть их с разных сторон, в разном масштабе и т. п. Для этого у него должны быть соответствующие возможности взаимодействия с образами:

- динамическое проецирование;
- интерактивная фильтрация;
- масштабирование образов;
- интерактивное искажение;
- интерактивное комбинирование.

Основная идея *динамического проецирования* заключается в динамическом изменении проекций при проведении исследования многомерных наборов данных. Примером может служить проецирование в двумерную плоскость всех интересующих проекций многомерных данных в виде диаграмм разброса (scatter plots). Необходимо обратить внимание, что количество возможных проекций экспоненциально увеличивается с ростом числа измерений, и следовательно, при большом количестве измерений проекции будут тяжело воспринимаемы.

При исследовании большого количества данных важно иметь возможность разделять наборы данных и выделять интересующие поднаборы — *фильтровать* образы. При этом важно, чтобы данная возможность предоставлялась в режиме реального времени работы с визуальными образами (т. е. интерактивно). Выбор поднабора может осуществляться или напрямую из списка, или с помощью определения свойств интересующего поднабора. Выбор из списка неудобен при большом количестве поднаборов, в то же время запросы не всегда позволяют получить желаемый результат.

Примером *масштабирования образов* является "магическая линза" (Magic Lenses). Ее основная идея состоит в использовании инструмента, похожего на увеличительное стекло, чтобы выполнять фильтрацию непосредственно при визуализации. Данные, попадающие под увеличительное стекло, обрабатываются фильтром, и результат отображается отдельно от основных данных. Линза показывает модифицированное изображение выбранного региона, тогда как остальные визуализированные данные не детализируются.

*Масштабирование* — это хорошо известный метод взаимодействия, используемый во многих приложениях. При работе с большим объемом данных этот метод хорош для представления данных в сжатом общем виде, и, в то же время, он предоставляет возможность отображения любой их части в более детальном виде. Масштабирование может заключаться не только в простом увеличении объектов, но в изменении их представления на разных уровнях. Так, например, на нижнем уровне объект может быть представлен пикселом,

на более высоком уровне — неким визуальным образом, а на следующем — текстовой меткой.

Метод *интерактивного искажения* поддерживает процесс исследования данных с помощью искажения масштаба данных при частичной детализации. Основная идея этого метода заключается в том, что часть данных отображается с высокой степенью детализации, а одновременно с этим остальные данные показываются с низким уровнем детализации. Наиболее популярные методы — это гиперболическое и сферическое искажения, которые часто используются на иерархиях и графах, но могут применяться и в других визуальных образах.

Существует достаточно много методов визуализации, но все они имеют как достоинства, так и недостатки. Основная идея *комбинирования* заключается в объединении различных методов визуализации для преодоления недостатков одного из них. Различные проекции рассеивания точек, например, могут быть скомбинированы с методами окрашивания и компоновки точек во всех проекциях. Такой подход может быть использован для любых методов визуализации. Окраска точек во всех методах визуализации дает возможность определить зависимости и корреляции в данных. Таким образом, комбинирование нескольких методов визуализации обеспечивает бóльшую информативность, чем в общем независимое использование методов. Типичными примерами визуальных образов, которые могут комбинироваться, являются: точки рассеивания, гистограммы, параллельные координаты, отображаемые пикселы и карты.

Любое средство визуализации может быть классифицировано по всем трем параметрам, т. е. по виду данных, с которым оно работает, по визуальным образам, которые оно может предоставлять, и по возможностям взаимодействия с этими визуальными образами. Очевидно, что одно средство визуализации может поддерживать разные виды данных, разные визуальные образы и разные способы взаимодействия с образами.

# 8.3. Методы визуализации

## 8.3.1. Методы геометрических преобразований

Основная идея методов геометрических преобразований — визуализировать преобразования и проекции данных в декартовом и в недекартовом геометрических пространствах. К этим методам относятся:

- точки и матрицы;
- гипердоли;
- поверхностные и объемные графики, контуры;

- параллельные координаты;
- текстуры и растры.

*Матрица диаграмм разброса* (Scatterplot Matrix) является комбинацией отдельных диаграмм разброса, что позволяет отображать более одного атрибута. Значения атрибутов отображаются в диагональных ячейках матрицы, а остальные ячейки представляют собой отношения между ними. Например, на рис. 8.2 показана матрица 5×5. Вдоль диагонали изображаются гистограммы пяти атрибутов, а, например, ячейка (2, 3) представляет отношение атрибута 2 с атрибутом 3. Соответственно, ячейка (3, 2) представляет отношение атрибута 3 с атрибутом 2.

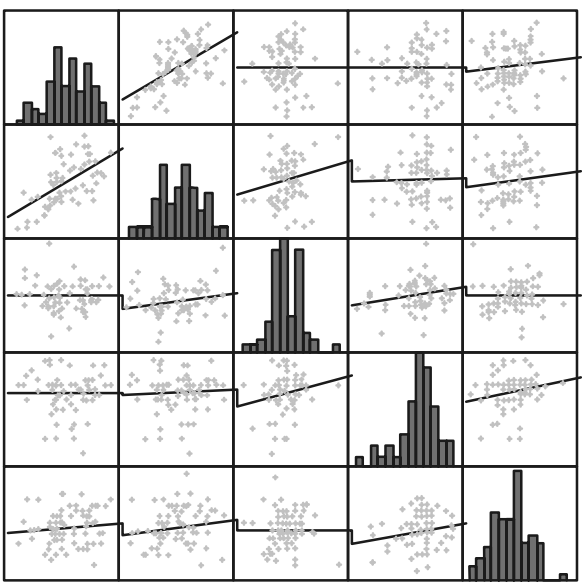


**Рис. 8.2.** Пример матрицы диаграмм разброса

В данном методе визуализации могут быть использованы такие типы взаимодействия, как соприкосновение и связывание. Например, когда пользователь наводит курсор, или щелкает мышью на определенной точке, или выбирает несколько точек в одной из ячеек, представляющих отношение, то в остальных ячейках матрицы могут подсвечиваться эквивалентные точки.

*Гипердоли* являются модификацией матрицы диаграмм разброса. Основная концепция та же, за исключением того, что в ячейках матрицы отображаются скалярные функции. Таким образом, в диагональных ячейках матрицы отображается скалярная функция, представляющая отдельные атрибуты, а в остальных ячейках — скалярное отношение нескольких атрибутов.

Пользователь может взаимодействовать с данным представлением, описав визуальный фокус и диапазон значений (например, так, как в ячейке (2, 3) на рис. 8.3). При этом отображаться будут только данные в заданном диапазоне. Перемещая фокус, пользователь может быстро исследовать другие данные из близлежащих диапазонов.

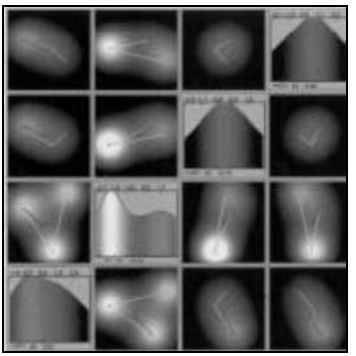


**Рис. 8.3.** Пример гипердолей

На ранних фазах визуального анализа большие величины непрерывных данных могут отображаться с помощью объема. *Объемный* рендеринг позволяет пользователю видеть внутреннюю часть объемных графиков. Цвета, яркость и полупрозрачность используются, чтобы изобразить различия распределений и значения атрибутов. Подвижность объемных графиков используется, чтобы визуализировать различные их слои.

Объемные графики (рис. 8.4) представляют собой 3D-плоскость, на которой отображается отношение между данными. Контурные линии используются для соединения точек, соответствующих данным с одинаковыми атрибутами. Однако представление большого количества данных с помощью этого метода может быть затруднено из-за густоты точек и, как следствие, затемненности и неясности изображения.

Еще одним распространенным методом геометрических преобразований является *метод параллельных координат*. Данный метод предполагает представление атрибутов параллельными линиями на недекартовой плоскости. Данные представляются кривыми линиями, которые пересекают линии атрибутов. Точки пересечений соответствуют значениям соответствующих атрибутов отображаемых данных. На рис. 8.5 приведен пример для данных, характеризующихся 10-ю измерениями.

Это достаточно простой способ представления многомерных данных, но при большом количестве линий получается большая зашумленность изображения, что приводит к неинформативности визуализации.

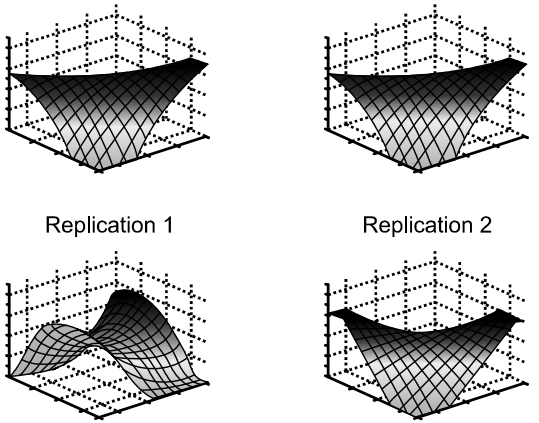



**Рис. 8.4.** Пример объемных графиков

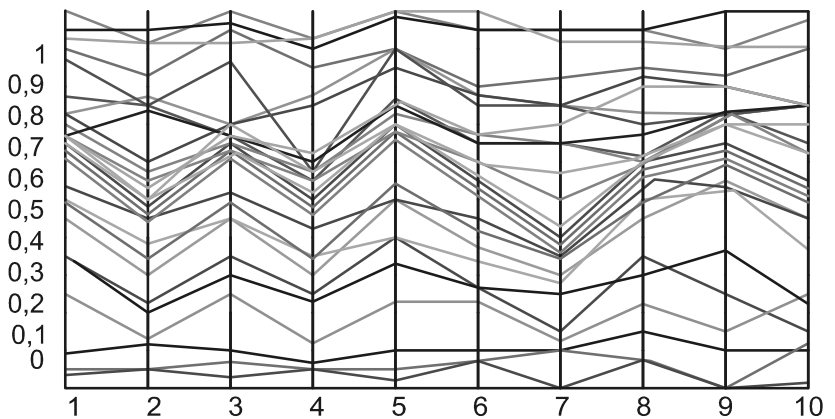



**Рис. 8.5.** Пример параллельных координат

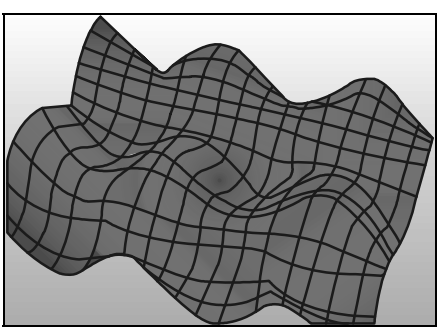


**Рис. 8.6.** Пример отображения текстур

*Текстурная* и *растровая визуализации* используют способность человека к преаттентивному (подсознательному) восприятию информации. Такой метод в совокупности с различными визуальными свойствами (такими как подсветка и интенсивность) позволяет отобразить большое количество атрибутов. Например, на рис. 8.6 с помощью текстуры представляется векторная и контурные диаграммы на плоскости.

## 8.3.2. Отображение иконок

Подход, основанный на отображении иконок, предполагает каждому объекту данных ставить в соответствие некоторую иконку. При этом атрибуты объекта должны отображаться различными визуальными свойствами иконок. Иконки могут комбинироваться в матрицы или графики и, таким образом, предоставляют возможность анализировать все объекты в целом.

Использование иконок предполагает следующие методы визуализации:

- ❑ линейчатые фигуры;
- ❑ "лица Чернова";
- ❑ цветные иконки;
- ❑ глифы [1] и др.

*Линейчатая фигура* представляет собой иконку с некоторым количеством ветвей (линий). Например, на рис. 8.7 представлены две фигуры, имеющие тело (длинная линия) и ветви (четыре коротких линии).

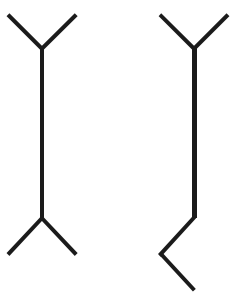


**Рис. 8.7.** Пример линейчатой фигуры

Каждый объект представляется отдельной фигурой. Атрибуты объекта отображаются с разными наклонами и местоположением линий (относительно

[1] Глиф — визуальное представление символа шрифта, образ символа шрифта, а также печатное изображение символа шрифта.

тела). В этом методе можно также использовать цветовую гамму для представления атрибутов.

Для анализа всех данных целиком линейчатые фигуры могут группироваться и создавать текстурное изображение. На рис. 8.8 представлен пример такого изображения для данных с 20-ю атрибутами.

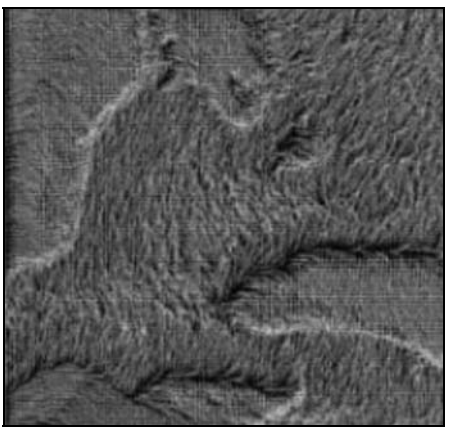


**Рис. 8.8.** Пример группировки линейчатых фигур

Другим хорошо известным методом отображения иконок является метод *"лиц Чернова"*. Этот метод предполагает использовать для представления объектов образы человеческих лиц (рис. 8.9). При этом каждый атрибут отображается определенной характеристикой человеческого лица: длиной, формой и т. п.

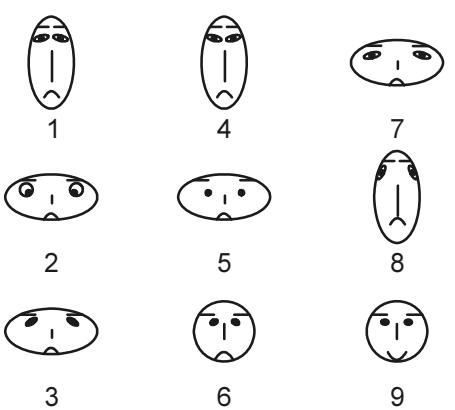



**Рис. 8.9.** Пример изображений лиц

*Цветные иконки* представляют атрибуты объектов цветом, формой, размером, границами, ориентацией (рис. 8.10).

Существует два подхода к раскрашиванию иконок:

1. Закрашивается линия, которая соответствует отдельному атрибуту.
2. Закрашивается часть иконки, соответствующая атрибуту.

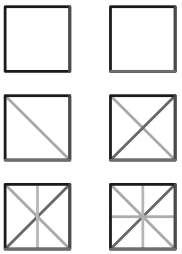


**Рис. 8.10.** Пример раскраски цветных иконок

*Глифы* представляют собой пикселы с более чем одним измерением. Они размещаются на 2D-площадке и их позиции описываются двумя атрибутами, тогда как другие атрибуты представляются цветом и формой. Некоторые модификации метода применяют глифы в виде цветов, звезд и др. На рис. 8.11 представлен пример скалярного глифа.

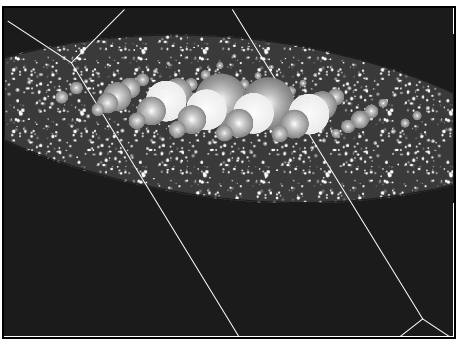


**Рис. 8.11.** Пример скалярного глифа

## 8.3.3. Методы, ориентированные на пикселы

Методы, ориентированные на пикселы, используют для представления каждого элемента данных цветные пикселы. Выделяют следующие подобные методы:

- заполнение пространства;
- рекурсивные шаблоны;
- мозаика.

В *методе заполнения пространства* каждый атрибут представляется пикселом. Цвет пиксела определяется диапазоном значений атрибута. Наборы пикселов для каждого объекта организовываются в определенные шаблоны: спирали, линейные шлейфы и т. п.

*Метод рекурсивных шаблонов* является комбинацией метода пиксельного заполнения пространства на множестве экранов. В рекурсивных шаблонах пикселы позиционируются в петли и спирали. Порядок заполнения начинается от центра и ведется к внешней границе шаблона (рис. 8.12).

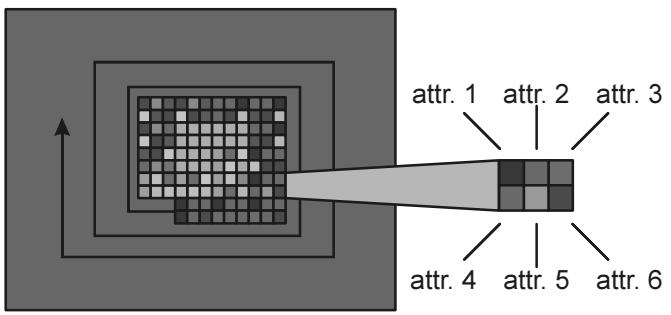



**Рис. 8.12.** Пример заполнения пространства по спирали

Идея мозаичного представления данных была предложена в 1981 г. Дж. А. Хартингом и Б. Кляйнером. Метод заключается в графическом представлении многовариантной таблицы сопряженности, что является естественным расширением одномерных спиндиаграмм, которые в свою очередь являются модификацией гистограмм. Спиндиаграммы одного атрибута группируются вместе. Такие группы отображаются на экране (рис. 8.13).

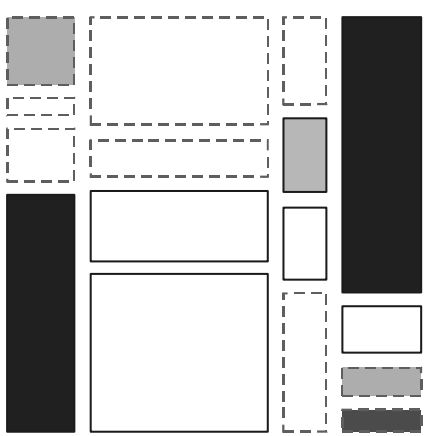


**Рис. 8.13.** Пример мозаики

## 8.3.4. Иерархические образы

Иерархические образы используются для отображения иерархий и отношений в данных. Они применяются в следующих методах:

- иерархические оси;
- наложение измерений;
- деревья.

*Оси*, представляющие каждый атрибут, накладываются горизонтально, при этом первое место в иерархии отводится наиболее изменяемому атрибуту. Такой метод может отображать до 20 атрибутов на одном экране. Для большого количества данных метод может использовать подпространственное масштабирование и, тем самым, походить на древовидную структуру.

На рис. 8.14 представлен пример иерархического расположения гистограмм. На первой гистограмме высота черных прямоугольников отражает значение зависимой переменной *z*. Независимые переменные *x* и *y* соответствуют горизонтальным осям, которые размещены внизу. На второй гистограмме высота серого прямоугольника определяется как сумма всех *z*-переменных, которые представлены черными прямоугольниками внутри. В данном примере имеется два черных квадрата высотой 1, следовательно, сумма равна 2. Третья гистограмма получается иерархическим наложением переменных. Для разных значений переменной *y* строятся диаграммы с одними и теми же значениями переменной *x*. Для каждого нового значения переменной *y* диаграммы строятся по тем же принципам, что и диаграмма, представленная на среднем рисунке. Полученные таким образом диаграммы объединяются в одну так, как это показано на третьей диаграмме. В примере иерархия осей выстраивается от атрибута *x* к *z*.

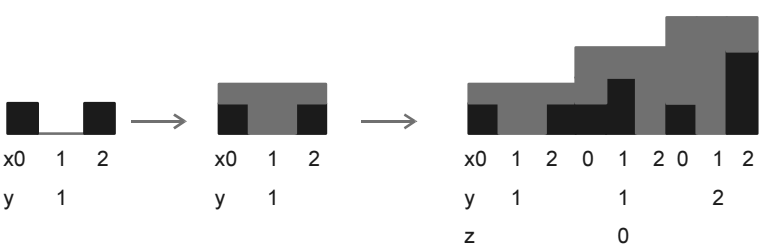



**Рис. 8.14.** Пример иерархических осей

Основная идея *метода наложения измерений* заключается во вставке одной координатной системы в другую. Иными словами, два атрибута формируют внешнюю систему координат, два других атрибута формируют другую систему координат, встроенную в предыдущую, и т. д. Этот процесс может быть повторен несколько раз.

Наглядность данного метода заключается в зависимости от распределения данных внешней системы координат. Поэтому измерения, которые используются для внешней системы координат, должны быть выбраны тщательно. Первыми нужно выбирать наиболее важные измерения.

На рис. 8.15 приведен пример визуализации методом наложения измерений, в котором географическая долгота и широта добычи нефти отображаются внешними *x* и *y* осями, а качество добываемой нефти и глубина — внутренними *x* и *y* осями.

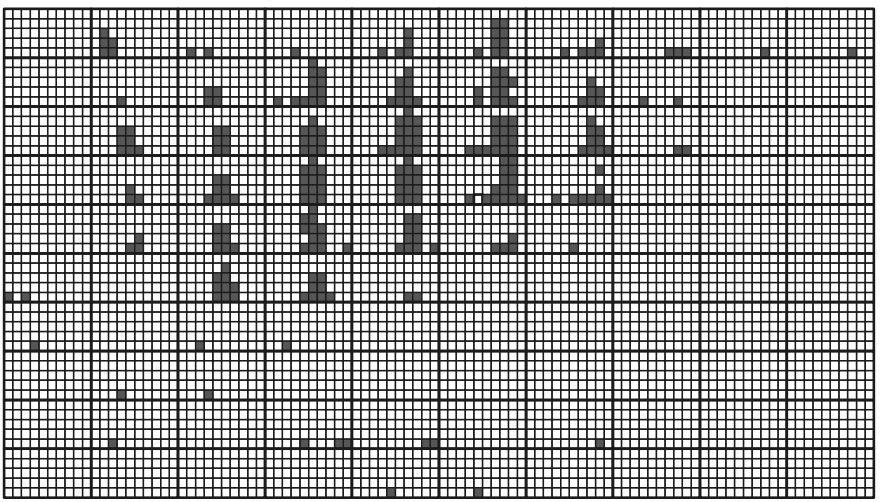


**Рис. 8.15.** Пример наложения измерений

Для визуализации данных используют два основных вида древовидных структур:

- древовидные карты;
- канонические деревья.

*Древовидные карты* иерархически делят экран, используя заполнение пространства (рис. 8.16). Этот метод использует разграниченные области для визуализации деревьев.

Следующие свойства всегда должны сохраняться для древовидных карт.

- Если узел *N*1 является предком узла *N*2, то ограниченный прямоугольник *N*1 целиком окружает прямоугольник *N*2.
- Ограниченные прямоугольники двух узлов пересекаются, если один узел является предком другого.
- Узлы занимают площадь строго пропорционально их весу.
- Вес узла больше или равен сумме весов его наследников.

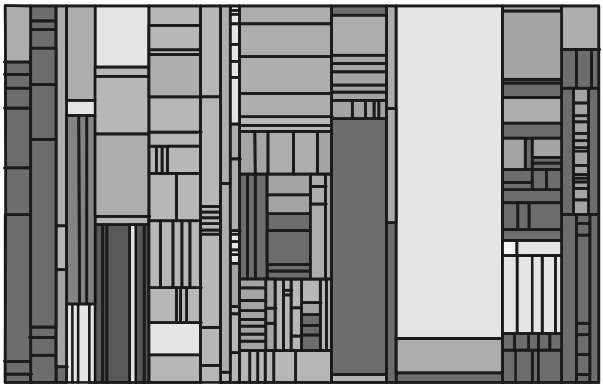


**Рис. 8.16.** Пример древовидной карты

Цвет используется, чтобы визуально представить тип содержимого узлов. Так, для этих целей могут использоваться оттенки, текстуры и яркость.

*Каноническое дерево* представляет собой древовидную структуру, которую можно интерактивно вращать и раскрывать новые ветви с данными (рис. 8.17).

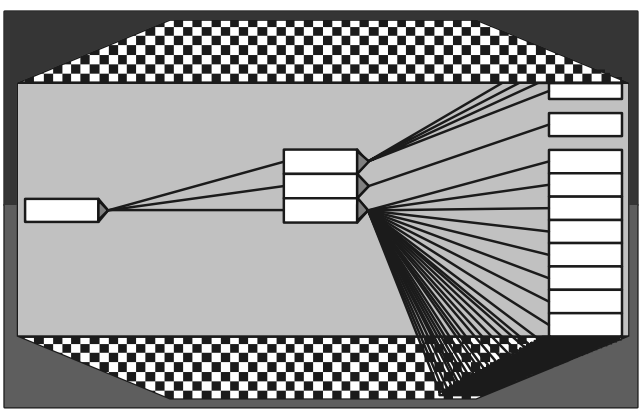


**Рис. 8.17.** Пример канонического дерева

# Выводы

По визуальному анализу данных можно сделать следующие выводы.

- Основной идеей визуального анализа данных является представление данных в некоторой визуальной форме, позволяющей человеку погрузиться в данные, работать с их визуальным представлением, понять их суть, сделать выводы и напрямую взаимодействовать с ними.

- Визуальный анализ данных обычно выполняется в три этапа: беглый анализ, увеличение и фильтрация, детализация по необходимости.
- Выделяют три основные характеристики средств визуализации: характер отображаемых данных, методы визуализации, возможности взаимодействия с визуальными образами.
- Выделяют следующие виды данных, с которыми могут работать средства визуализации: одномерные, двумерные и многомерные данные, тексты и гипертексты, иерархические и связанные данные, алгоритмы и программы.
- Выделяют следующие основные типы методов визуализации: стандартные 2D/3D-образы, геометрические преобразования, отображение иконок, ориентированные на пикселы методы, иерархические образы.
- Для анализа визуальных образов часто используют следующие возможности взаимодействия: динамическое проецирование, интерактивная фильтрация, масштабирование образов, интерактивное искажение, интерактивное комбинирование.
- Основная идея методов геометрических преобразований — визуализировать преобразования и проекции данных в декартовом и недекартовом геометрических пространствах.
- Подход, основанный на отображении иконок, предполагает каждому объекту данных ставить в соответствие некоторую иконку, при этом атрибуты объекта должны отображаться с помощью различных визуальных свойств иконок.
- Методы, ориентированные на пикселы, используют для представления каждого элемента данных цветные пикселы.
- Иерархические образы используются для отображения иерархий и отношений в данных.

ГЛАВА 9

# Анализ текстовой информации — Text Mining

## 9.1. Задача анализа текстов

### 9.1.1. Этапы анализа текстов

Анализ структурированной информации, хранящейся в базах данных, требует предварительной обработки: проектирования БД, ввод информации по определенным правилам, размещение ее в специальных структурах (например, реляционных таблицах) и т. п. Таким образом, непосредственно для анализа этой информации и получения из нее новых знаний необходимо затратить дополнительные усилия. При этом они не всегда связаны с анализом и не обязательно приводят к желаемому результату. Из-за этого КПД анализа структурированной информации снижается. Кроме того, не все виды данных можно структурировать без потери полезной информации. Например, текстовые документы практически невозможно преобразовать в табличное представление без потери семантики текста и отношений между сущностями. По этой причине такие документы хранятся в БД без преобразований, как текстовые поля (BLOB-поля). В то же время в тексте скрыто огромное количество информации, но ее неструктурированность не позволяет использовать алгоритмы Data Mining. Решением этой проблемы занимаются методы анализа неструктурированного текста. В западной литературе такой анализ называют Text Mining.

Методы анализа в неструктурированных текстах лежат на стыке нескольких областей: Data Mining, обработка естественных языков, поиск информации, извлечение информации и управление знаниями.

В работе [41] по аналогии с термином Data Mining *(см. гл. 4)* дано следующее определение:

*Обнаружение знаний в тексте — это нетривиальный процесс обнаружения действительно новых, потенциально полезных и понятных шаблонов в неструктурированных текстовых данных.*

Как видно, от определения Data Mining оно отличается только новым понятием "неструктурированные текстовые данные". Под такими знаниями понимается набор документов, представляющих собой логически объединенный текст без каких-либо ограничений на его структуру. Примерами таких документов являются: Web-страницы, электронная почта, нормативные документы и т. п. В общем случае такие документы могут быть сложными и большими и включать в себя не только текст, но и графическую информацию. Документы, использующие язык расширяемой разметки XML (eXtensible Markup Language), стандартный язык обобщенной разметки SGML (Standard Generalised Markup Language) и другие подобные соглашения по структуре формирования текста, принято называть полуструктурированными документами. Они также могут быть обработаны методами Text Mining.

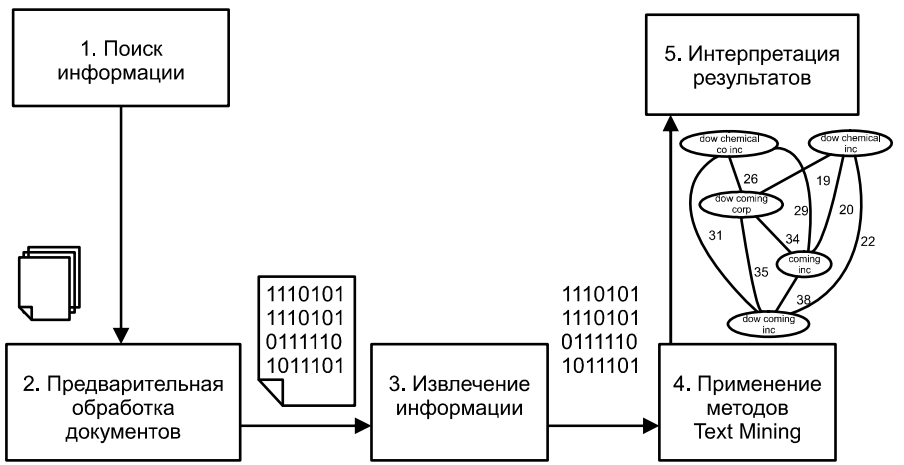



**Рис. 9.1.** Этапы Text Mining

Процесс анализа текстовых документов можно представить как последовательность нескольких шагов (рис. 9.1).

1. *Поиск информации.* На первом шаге необходимо идентифицировать, какие документы должны быть подвергнуты анализу, и обеспечить их доступность. Как правило, пользователи могут определить набор анализируемых документов самостоятельно — вручную, но при большом количестве документов необходимо использовать варианты автоматизированного отбора по заданным критериям.
2. *Предварительная обработка документов.* На этом шаге выполняются простейшие, но необходимые преобразования с документами для представления их в виде, с которым работают методы Text Mining. Целью таких преобразований является удаление лишних слов и придание тексту

более строгой формы. Подробнее методы предварительной обработки будут описаны в *разд. 9.1.2*.

3. *Извлечение информации.* Извлечение информации из выбранных документов предполагает выделение в них ключевых понятий, над которыми в дальнейшем будет выполняться анализ. Данный этап является очень важным и будет подробно описан в *разд. 9.1.3*.
4. *Применение методов Text Mining.* На данном шаге извлекаются шаблоны и отношения, имеющиеся в текстах. Данный шаг является основным в процессе анализа текстов, и практические задачи, решаемые на этом шаге, описываются в *разд. 9.1.4*.
5. *Интерпретация результатов.* Последний шаг в процессе обнаружения знаний предполагает интерпретацию полученных результатов. Как правило, интерпретация заключается или в представлении результатов на естественном языке, или в их визуализации в графическом виде.

Визуализация также может быть использована как средство анализа текста. Для этого извлекаются ключевые понятия, которые и представляются в графическом виде. Такой подход помогает пользователю быстро идентифицировать главные темы и понятия, а также определить их важность.

## 9.1.2. Предварительная обработка текста

Одной из главных проблем анализа текстов является большое количество слов в документе. Если каждое из этих слов подвергать анализу, то время поиска новых знаний резко возрастет и вряд ли будет удовлетворять требованиям пользователей. В то же время очевидно, что не все слова в тексте несут полезную информацию. Кроме того, в силу гибкости естественных языков формально различные слова (синонимы и т. п.) на самом деле означают одинаковые понятия. Таким образом, удаление неинформативных слов, а также приведение близких по смыслу слов к единой форме значительно сокращают время анализа текстов. Устранение описанных проблем выполняется на этапе предварительной обработки текста.

Обычно используют следующие приемы удаления неинформативных слов и повышения строгости текстов:

- удаление стоп-слов. Стоп-словами называются слова, которые являются вспомогательными и несут мало информации о содержании документа. Обычно заранее составляются списки таких слов, и в процессе предварительной обработки они удаляются из текста. Типичным примером таких слов являются вспомогательные слова и артикли, например: "так как", "кроме того" и т. п.;
- стемминг — морфологический поиск. Он заключается в преобразовании каждого слова к его нормальной форме. Нормальная форма исключает

склонение слова, множественную форму, особенности устной речи и т. п. Например, слова "сжатие" и "сжатый" должны быть преобразованы в нормальную форму слова "сжимать". Алгоритмы морфологического разбора учитывают языковые особенности и вследствие этого являются языковозависимыми алгоритмами;

- *N*-граммы — это альтернатива морфологическому разбору и удалению стоп-слов. *N*-грамма — это часть строки, состоящая из *N* символов. Например, слово "дата" может быть представлено 3-граммой "_да", "дат", "ата", "та_" или 4-граммой "_дат", "дата", "ата_", где символ подчеркивания заменяет предшествующий или замыкающий слово пробел. По сравнению со стеммингом или удалением стоп-слов, *N*-граммы менее чувствительны к грамматическим и типографическим ошибкам. Кроме того, *N*-граммы не требуют лингвистического представления слов, что делает данный прием более независимым от языка. Однако *N*-граммы, позволяя сделать текст более строгим, не решают проблему уменьшения количества неинформативных слов;
- приведение регистра. Этот прием заключается в преобразовании всех символов к верхнему или нижнему регистру. Например, все слова "текст", "Текст", "ТЕКСТ" приводятся к нижнему регистру "текст".

Наиболее эффективно совместное применение перечисленных методов.

### 9.1.3. Задачи Text Mining

В настоящее время в литературе описано много прикладных задач, решаемых с помощью анализа текстовых документов. Это и классические задачи Data Mining: классификация, кластеризация, и характерные только для текстовых документов задачи: автоматическое аннотирование, извлечение ключевых понятий и др.

*Классификация* (classification) — стандартная задача из области Data Mining. Ее целью является определение для каждого документа одной или нескольких заранее заданных категорий, к которым этот документ относится. Особенностью задачи классификации является предположение, что множество классифицируемых документов не содержит "мусора", т. е. каждый из документов соответствует какой-нибудь заданной категории.

Частным случаем задачи классификации является задача определения тематики документа [43].

Целью *кластеризации* (clustering) документов является автоматическое выявление групп семантически похожих документов среди заданного фиксированного множества. Отметим, что группы формируются только на основе попарной схожести описаний документов, и никакие характеристики этих групп не задаются заранее [43].

*Автоматическое аннотирование* (summarization) позволяет сократить текст, сохраняя его смысл. Решение этой задачи обычно регулируется пользователем при помощи определения количества извлекаемых предложений или процентом извлекаемого текста по отношению ко всему тексту. Результат включает в себя наиболее значимые предложения в тексте.

Первичной целью *извлечения ключевых понятий* (feature extraction) является идентификация фактов и отношений в тексте. В большинстве случаев такими понятиями являются имена существительные и нарицательные: имена и фамилии людей, названия организаций и др. Алгоритмы извлечения понятий могут использовать словари, чтобы идентифицировать некоторые термины и лингвистические шаблоны для определения других.

*Навигация по тексту* (text-base navigation) позволяет пользователям перемещаться по документам относительно тем и значимых терминов. Это выполняется за счет идентификации ключевых понятий и некоторых отношений между ними.

Анализ трендов позволяет идентифицировать тренды в наборах документов на какой-то период времени. Тренд может быть использован, например, для обнаружения изменений интересов компании от одного сегмента рынка к другому.

Поиск ассоциаций также является одной из основных задач Data Mining. Для ее решения в заданном наборе документов идентифицируются ассоциативные отношения между ключевыми понятиями.

Существует достаточно большое количество разновидностей перечисленных задач, а также методов их решения. Это еще раз подтверждает значимость анализа текстов. Далее в этой главе рассматриваются решения следующих задач: извлечение ключевых понятий, классификация, кластеризация и автоматическое аннотирование.

# 9.2. Извлечение ключевых понятий из текста

## 9.2.1. Общее описание процесса извлечения понятий из текста

Извлечение ключевых понятий из текста может рассматриваться и как отдельный этап анализа текста, и как определенная прикладная задача. В первом случае извлеченные из текста факты используются для решения различных задач анализа: классификации, кластеризации и др. Большинство методов Data Mining, адаптированные для анализа текстов, работают именно с такими отдельными понятиями, рассматривая их в качестве атрибутов данных.

В задаче извлечения ключевых понятий из текста интерес представляют некоторые сущности, события и отношения. При этом извлеченные понятия анализируются и используются для вывода новых. В данном разделе и будет описано решение такой задачи. При этом часть процесса решения может быть использована для выделения ключевых понятий при решении других задач анализа текста.

Извлечение ключевых понятий из текстовых документов можно рассматривать как фильтрацию больших объемов текста. Этот процесс включает в себя отбор документов из коллекции и пометку определенных термов в тексте. Существуют различные подходы к извлечению информации из текста. Примером может служить определение частых наборов слов и объединение их в ключевые понятия. Для определения частых наборов используется алгоритм Apriori, описанный в *разд. 6.3*.

Другим подходом является идентификация фактов в текстах и извлечение их характеристик [48]. Фактами являются некоторые события или отношения. Идентификация производится с помощью наборов образцов. Образцы представляют собой возможные лингвистические варианты фактов.

Такой подход позволяет представить найденные ключевые понятия, представленные событиями и отношениями, в виде структур, которые в том числе можно хранить в базах данных.

Процесс извлечения ключевых понятий с помощью шаблонов разбивается на две стадии: локальный анализ и анализ понятий (рис. 9.2). На первой стадии из текстовых документов извлекаются отдельные факты с помощью лексического анализа. Вторая стадия заключается в интеграции извлеченных фактов и/или в выводе новых фактов. В конце наиболее характерные факты преобразовываются в нужную выходную форму.

Сложность извлечения фактов с помощью образцов связана с тем, что на практике их нельзя представить в виде простой последовательности слов. В большинстве систем обработки естественных языков вначале идентифицируются различные уровни компонентов и отношений, а затем на их основе строятся образцы. Этот процесс обычно начинается с лексического анализа (определения частей речи и характеристик слов и фраз посредством морфологического анализа и поиска по словарю) и распознавания имен (идентификации имен и других лексических структур, таких как даты, денежные выражения и т. п.). За этим следует синтаксический разбор, целью которого является выявление групп существительных, глаголов и, если возможно, дополнительных структур. Затем применяются предметно-ориентированные образцы для идентификации интересующих фактов.

На стадии интеграции найденные в документах факты исследуются и комбинируются. Это выполняется с учетом отношений, которые определяются ме-

стоимениями или описанием одинаковых событий. Также на этой стадии делаются выводы из ранее установленных фактов.

Как уже отмечалось ранее, извлечение фактов выполняется при помощи сопоставления текста с набором регулярных выражений (образцов). Если выражение сопоставляется с текстовыми сегментами, то такие сегменты помечаются метками. При необходимости этим сегментам приписываются дополнительные свойства. Образцы организуются в наборы. Метки, ассоциированные с одним набором, могут ссылаться на другие наборы.

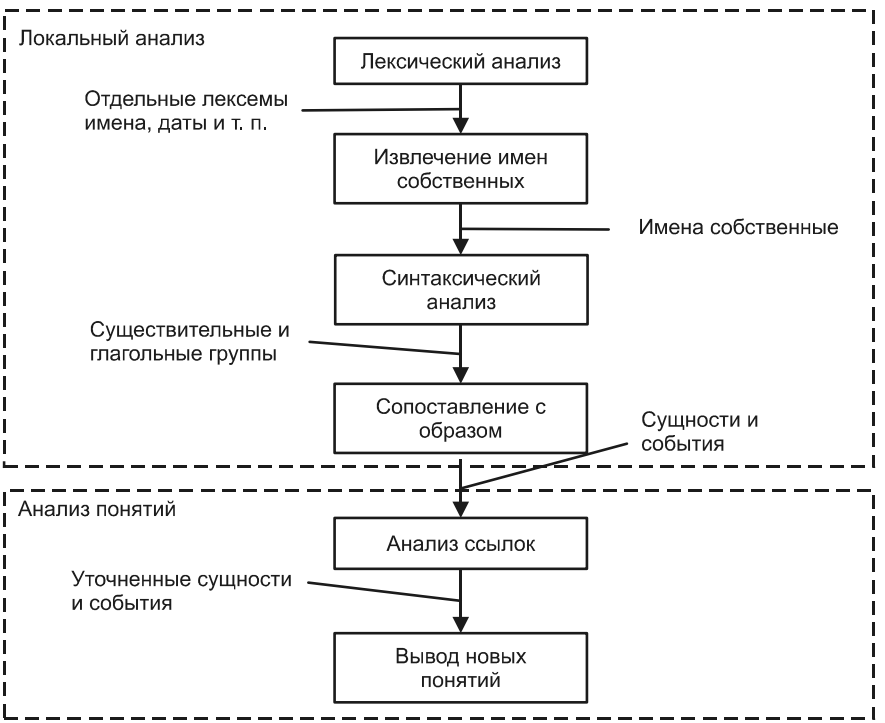



**Рис. 9.2.** Процесс извлечения ключевых понятий

Каждый образец имеет связанный с ним набор действий. Как правило, главное действие — это пометить текстовый сегмент новой меткой, но могут быть и другие действия. В каждый момент времени текстовому сегменту сопоставляется только один набор образцов. Каждый образец в наборе начинает сопоставляться с первого слова предложения. Если образец может быть сопоставлен более чем одному сегменту, то выбирается наиболее длинный сопоставленный сегмент. Если таких сегментов несколько, то выбирается первый. При сопоставлении выполняются действия, ассоциированные с этим образцом. Если не удалось сопоставить ни один образец, то сопоставление

повторяется, начиная со следующего слова в предложении. Если сегмент сопоставлен с образцом, то сопоставление повторяется, начиная со следующего слова после сегмента. Процесс продолжается до конца предложения.

Основной целью сопоставления с образцами является выделение в тексте сущностей, связей и событий. Все они могут быть преобразованы в некоторые структуры, которые могут анализироваться стандартными методами Data Mining.

## 9.2.2. Стадия локального анализа

Рассмотрим процесс выделения ключевых понятий на примере следующего текста:

*Петр Сергеевич Иванов покинул должность вице-президента известной фабрики ООО "Анкор". Его заменил Иван Андреевич Сидоров.*

На этапе лексического анализа текст делится на предложения и лексемы. Каждая лексема ищется в словаре для определения ее части речи и других свойств. Такой словарь готовится заранее экспертами в данной предметной области и должен включать, кроме специальных терминов, имена людей, названия городов, стран, префиксы компаний (такие как "ООО", "ЗАО", "АО" и т. п.) и др. В нашем примере на этом этапе должны быть идентифицированы следующие лексемы: "Петр", "Иван", "ООО". При этом "Петр" и "Иван" помечаются как имена, а "ООО" — как префикс фирмы.

На следующем этапе идентифицируются различные типы имен собственных и другие специальные формы, такие как даты, денежные выражения и т. п. Имена присутствуют в текстах различного вида. Определить их достаточно просто, но они являются важными ключевыми понятиями.

Имена идентифицируются с помощью образцов (регулярных выражений), которые строятся на основе частей речи, синтаксических и орфографических свойств (например, использование заглавных букв). Например, люди могут быть идентифицированы:

- предшествующими званиями: "мистер", "сударь", "господин", "товарищ" и т. п. (например, **мистер** Смит, **господин** Иванов, **товарищ** Сталин и т. д.);
- распространенными именами: "Иван", "Петр", "Елена" и т. п. (например, **Иван** Сидоров, **Елена** Премудрая, **Петр** Сергеевич Иванов и т. д.);
- предшествующими инициалами имени и отчества (например, **И. И.** Сидоров, **Е. А.** Иванова и т. д.).

Компании могут идентифицироваться с помощью лексем, обозначающих форму их организации "ООО", "ЗАО" и т. п.

В нашем примере можно идентифицировать три имени собственных:

- *Петр Сергеевич Иванов* с типом *"человек"*;
- *Иван Андреевич Сидоров* с типом *"человек"*;
- *ООО "Анкор"* с типом *"фирма"*.

В результате получим следующую структуру:

*[имя собственное тип: человек Петр Сергеевич Иванов] покинул должность вице-президента известной фабрики [имя собственное тип: фирма ООО "Анкор"]. Его заменил [имя собственное тип: человек Иван Андреевич Сидоров].*

При идентификации имен собственных также важно распознавать и альтернативное их написание (другие формы тех же имен). Например, "Петр Сергеевич Иванов", "П. С. Иванов", "Петр Иванов", "господин Иванов" должны быть идентифицированы как одно и то же лицо. Такое сопоставление различных написаний имен собственных может помочь в идентификации свойств понятия. Например, по выражению "Елена работает с 9:00 до 20:00" невозможно понять, Елена является человеком или фирмой с названием "Елена" (однозначно это сложно определить даже человеку). Однако если в тексте также встречается альтернативное написание "ООО "Елена", то понятию "Елена" можно присвоить тип "фирма".

Идентификация некоторых аспектов синтаксических структур упрощает последующие фазы извлечения фактов. С другой стороны, идентификация сложных синтаксических структур в предложении — трудная задача. В связи с этим различные методы анализа текста по-разному решают эту задачу. Некоторые из них опускают данный этап, а некоторые выполняют сложный разбор предложений. Однако большинство систем выполняют разбор последовательных фрагментов предложений. Они строят только такие структуры, которые могут быть точно определены или синтаксисом, или семантикой отдельного фрагмента предложения.

Примером такого подхода может служить построение структур для групп имен существительных (имя существительное плюс его модификации) и глагольных групп (глагол с его вспомогательными частями). Оба вида структур могут быть построены, используя только локальную синтаксическую информацию. Кроме того, этот подход позволяет строить большие структуры групп имен существительных (путем объединения нескольких групп), если имеется семантическая информация, подтверждающая корректность таких объединений. Все такие структуры строятся с использованием одних и тех же регулярных выражений.

Вначале помечаются все основные группы имен существительных меткой "сущ.". В нашем примере имеются следующие группы имен существитель-

ных: три имени собственных, местоимение и две больших группы. Далее помечаются глагольные группы меткой "гл.". В результате наш пример будет выглядеть следующим образом:

*[сущ. сущность: e1 Петр Сергеевич Иванов] [гл.: покинул] [сущ. сущность: e2 должность вице-президента] [сущ. сущность: e3 известной фабрики] [[сущ. сущность: e4 ООО "Анкор"]. [сущ. сущность: e5 Его] [гл.: заменил] [сущ. сущность: e6 Иван Андреевич Сидоров]*.

С каждой группой можно связать дополнительные свойства. Для глагольных групп такими свойствами являются информация о времени (прошедшее, настоящее, будущее) и залоге (активный или пассивный), а также корневая форма глагола. Для групп имен существительных это информация о корневой форме главного слова (например, имени собственном) и его числительность (единственное или множественное число). Кроме того, для каждой группы имен существительных создается сущность. В нашем примере их шесть:

- *e1* — тип: *человек*, имя: *"Петр Сергеевич Иванов"*;
- *e2* — тип: *должность*, значение: *"вице-президент"*;
- *e3* — тип: *фирма*;
- *e4* — тип: *фирма*, имя: *"ООО "Анкор"*;
- *e5* — тип: *человек*;
- *e6* — тип: *человек*, имя: *"Иван Андреевич Сидоров"*.

Для укрупнения групп имен существительных используют наборы образцов, присоединяющие правые модификаторы. Эти образцы обычно объединяют две группы имен существительных и, возможно, промежуточные слова в большую группу и модифицируют сущность, ассоциированную с главным существительным, чтобы соединить информацию из модификатора.

В нашем примере можно выделить два важных образца:

*описание фирмы*, *имя фирмы*

и конструкцию группы:

*должность фирмы*.

Во втором образце "должность" представляет собой элемент, который сопоставляется с сущностью типа "должность" (в нашем примере это сущность *e2*), а элемент "фирма" сопоставляется с сущностью типа "фирма" (*e3* и *e4*). Возможно использование некоторой иерархии семитических типов и сопоставление образцов с ее применением (например, "фирма" более общее понятие, чем "фабрика", поэтому сопоставление должно выполняться). В первом образце элемент "имя фирмы" определяет сущность типа "фирма", в которой главным словом является имя (*e4*); элемент "описание фирмы" определяет

группу типа "фирма", в котором главным словом является общее описание (*e3*). Эти образцы порождают следующие метки:

*[сущ. сущность: e1 Петр Сергеевич Иванов] [гл: покинул] [сущ. сущность: e2 должность вице-президента известной фирмы ООО "Анкор"]. [сущ. сущность: e5 Его] [гл: заменил] [сущ. сущность: e6 Иван Андреевич Сидоров]*.

Таким образом, список сущностей обновится следующим образом:

- e1 — тип: *человек*, имя: *"Петр Сергеевич Иванов"*;
- e2 — тип: *должность*, значение: *"вице-президент"* фирмы: *e3*;
- e3 — тип: *фирма*, имя: *"ООО "Анкор"*;
- e5 — тип: *человек*;
- e6 — тип: *человек*, имя: *"Иван Андреевич Сидоров"*.

## 9.2.3. Стадия интеграции и вывода понятий

Для извлечения событий и отношений используются образцы, которые получаются за счет расширения образцов, описанных ранее. Например, событие преемственности должности извлекается с помощью следующих образцов:

*человек покинул должность*

и

*человек заменяется человеком*.

В примере элементы шаблона: "человек" и "должность" сопоставляются с группами имен существительных. А элементы "покинул" и "заменяется" сопоставляются с активными и пассивными глагольными группами соответственно. В результате в тексте выделяются две структуры событий на основе ранее созданных сущностей:

*[событие: e7 Петр Сергеевич Иванов покинул должность вице-президента известной фирмы ООО "Анкор"]. [событие: e8 Его заменил Иван Андреевич Сидоров]*.

Список сущностей обновляется следующим образом:

- *e1* — тип: *человек*, имя: *"Петр Сергеевич Иванов"*;
- *e2* — тип: *должность*, значение: *"вице-президент"* фирмы: *e3*;
- *e3* — тип: *фирма*, имя: *"ООО "Анкор"*;
- *e5* — тип: *человек*;
- *e6* — тип: *человек*, имя: *"Иван Андреевич Сидоров"*;
- *e7* — тип: *покинул*, человек: *e1*, должность: *e2*;
- *e8* — тип: *заменил*, человек: *e6*, человек: *e5*.

Описанным образом могут быть получены основные ключевые понятия. По ним может выполняться анализ текстов методами Data Mining для решения задач классификации, кластеризации и др.

В результате локального анализа из текста извлекаются ключевые понятия: сущности и события. Для получения более структурированной информации выполняется анализ ссылок. Его целью является разрешение ссылок, представленных местоимениями и описываемыми группами имен существительных. В нашем примере таким местоимением является "Его" (сущность *e5*). Для разрешения этой ссылки будет выполняться поиск первой предшествующей сущности с типом "человек". В нашем примере такой сущностью является *e1*. В результате ссылки на *e5* должны быть заменены ссылками на *e1*. Таким образом, список сущностей и событий обновится следующим образом:

- *e1* — тип: *человек*, имя: *"Петр Сергеевич Иванов"*;
- *e2* — тип: *должность*, значение: *"вице-президент"* фирмы: *e3*;
- *e3* — тип: *фирма*, имя: *"ООО "Анкор"*;
- *e6* — тип: *человек*, имя: *"Иван Андреевич Сидоров"*;
- *e7* — тип: *покинул*, человек: *e1*, должность: *e2*;
- *e8* — тип: *заменил*, человек: *e6*, человек: *e1*.

При анализе ссылок также надо учитывать иерархию понятий (как в случае "фирма" и "фабрика").

Во многих ситуациях определенная информация о событии может распространяться на другие предложения. Используя механизмы вывода, можно получить новые факты. В нашем примере, строя выводы на смысле сказуемого "заменил", можно получить новый факт, что Иван Андреевич Сидоров тоже был вице-президентом. Такой вывод можно сделать на основе системы порождающих правил, таких как следующие:

покинул (*X*-человек, *Y*-должность) & заменил (*Z*-человек, *X*-человек) =>
вступил (Z-человек, Y-должность);

вступил (*X*-человек, *Y*-должность) & заменил (*X*-человек, *Z*-человек) =>
покинул (*Z*-человек, *Y*-должность).

Такие правила позволяют добавить еще одно событие:

- *e1* — тип: *человек*, имя: *"Петр Сергеевич Иванов"*;
- *e2* — тип: *должность*, значение: *"вице-президент"* фирмы: *e3*;
- *e3* — тип: *фирма*, имя: *"ООО "Анкор"*;
- *e6* — тип: *человек*, имя: *"Иван Андреевич Сидоров"*;

- *e7* — тип: *покинул*, человек: *e1*, должность:*e2*;
- *e8* — тип: *заменил*, человек: *e6*, человек: *e1*;
- *e9* — тип: *вступил*, человек: *e6*, человек: *e2*.

В результате описанной последовательности действий можно получить следующие извлеченные ключевые понятия, представленные в виде табл. 9.1.

**Таблица 9.1**

| | **Событие** | **Человек** | **Должность** | **Фирма** |
|---|---|---|---|---|
| 1 | Покинул | Петр Сергеевич Иванов | Вице-президент | ООО "Анкор" |
| 2 | Вступил | Иван Андреевич Сидоров | Вице-президент | ООО "Анкор" |

В описанном подходе не определялось время событий. Однако для многих методов это важно или для вывода в аналитический отчет, или для хронологии последовательности событий. В таких случаях информация о времени может быть получена из разных источников, включая абсолютные даты и время (например, "28 июля 2006 года"), относительные упоминания времени ("последняя неделя"), времена глаголов и знаний о последовательности вывода событий.

Извлеченные понятия должны быть преобразованы в единую форму. Это позволяет выполнять индексированный поиск и другие операции максимально правильно. Например, слова "изучающий" и "изучение" должны быть идентифицированы как одно слово "изучать".

# 9.3. Классификация текстовых документов

## 9.3.1. Описание задачи классификации текстов

Классификация текстовых документов, так же как и в случае классификации объектов *(см. гл. 5)*, заключается в отнесении документа к одному из заранее известных классов. Часто классификацию применительно к текстовым документам называют *категоризацией* или *рубрикацией*. Очевидно, что данные названия происходят от задачи систематизации документов по каталогам, категориям и рубрикам. При этом структура каталогов может быть как одноуровневой, так и многоуровневой (иерархической).

Формально задачу классификации текстовых документов описывают набором множеств. Множество документов представляется в виде:

$$D = \{d_1, ..., d_i, ..., d_n\}.$$

Категории документов представляются множеством:

$$C = \{c_r\}, \quad \text{где } r = 1, ..., m.$$

Иерархию категорий можно представить в виде множества пар, отражающих отношение вложенности между рубриками:

$$H = \{< c_j, c_p >, c_j, c_p \in C\}$$

(категория $c_p$ вложена в категорию $c_j$).

В задаче классификации требуется на основе этих данных построить процедуру, которая заключается в нахождении наиболее вероятной категории из множества $C$ для исследуемого документа $d_i$.

Большинство методов классификации текстов так или иначе основаны на предположении, что документы, относящиеся к одной категории, содержат одинаковые признаки (слова или словосочетания), и наличие или отсутствие таких признаков в документе говорит о его принадлежности или непринадлежности к той или иной теме.

Таким образом, для каждой категории должно быть множество признаков:

$$F(C) = \cup(c_r),$$

где $F(c_r) =< f_1, ..., f_k, ..., f_z >$.

Такое множество признаков часто называют *словарем*, т. к. оно состоит из лексем, которые включают слова и/или словосочетания, характеризующие категорию.

Подобно категориям каждый документ также имеет признаки, по которым его можно отнести с некоторой степенью вероятности к одной или нескольким категориям:

$$F(d_i) =< f_1^i, ..., f_l^i, ..., f_y^i >.$$

Множество признаков всех документов должно совпадать с множеством признаков категорий, т. е.:

$$F(C) = F(D) = \cup F(d_i).$$

Необходимо заметить, что данные наборы признаков являются отличительной чертой классификации текстовых документов от классификации объектов в Data Mining, которые характеризуются набором атрибутов.

Решение об отнесении документа $d_i$ к категории $c_r$ принимается на основании пересечения:

$$F(d_i) \cup F(c_r).$$

Задача методов классификации состоит в том, чтобы наилучшим образом выбрать такие признаки и сформулировать правила, на основе которых будет приниматься решение об отнесении документа к рубрике.

## 9.3.2. Методы классификации текстовых документов

Существует два противоположных подхода к формированию множества $F(C)$ и построению правил:

- машинное обучение — предполагается наличие обучающей выборки документов, по которому строится множество $F(C)$;
- экспертный метод — предполагает, что выделение признаков — множества $F(C)$ — и составление правил производится экспертами.

В случае машинного обучения анализируется статистика лингвистических шаблонов (таких как лексическая близость, повторяемость слов и т. п.) из документов обучающей выборки. В нее должны входить документы, относящиеся к каждой рубрике, чтобы создать набор признаков (статистическую сигнатуру) для каждой рубрики, который впоследствии будет использоваться для классификации новых документов. Достоинством данного подхода является отсутствие необходимости в словарях, которые сложно построить для больших предметных областей. Однако чтобы избежать неправильной классификации, требуется обеспечить хорошее представительство документов для каждой рубрики.

Во втором случае формирование словаря (множества $F(C)$) может быть выполнено на основе набора терминов предметной области и отношений между ними (основные термины, синонимы и родственные термины). Классификация может затем определить рубрику документа в соответствии с частотой, с которой появляются выделенные в тексте термины (ключевые понятия).

Возможна и комбинация двух описанных подходов, когда выделение признаков и составление правил выполняются автоматически на основе обучающей выборки, и в то же время правила строятся в таком виде, чтобы эксперту была понятна логика автоматической рубрикации, и у него была возможность вручную корректировать эти правила.

Для классификации текстовых документов успешно используются многие методы и алгоритмы классификации Data Mining: Naive Bayes, метод наименьших квадратов, C4.5, SVM и др. Некоторые из них подробно были описаны в *гл. 5*. Очевидно, что требуется модификация этих методов для работы с текстовой информацией. Как правило, адаптация алгоритмов связана с тем, что понятие независимой переменной связано не с атрибутами объекта, а с наличием в текстовом документе того или иного признака $f$. Рассмотрим модификацию таких алгоритмов на примере метода Naive Bayes, описанного в *разд. 5.3.2*.

**Метод Naive Bayes** предполагает вычисление вероятностей принадлежности текстового документа к каждой рубрике. Решение о принадлежности принимается по максимальной вероятности:

$$P(y = c_r \mid E) = P(x_1 = c_p^1 \mid y = c_r) \times P(x_2 = c_d^2 \mid y = c_r) \times \ldots \times$$
$$\times P(x_m = c_b^m \mid y = c_r) = P(y = c_r) / P(E).$$

Зависимая переменная $y$ указывает на принадлежность документа к категории $c_r$. Событие $E$ заключается в наличии в текстовом документе признаков (лемм), характеризующих категорию $c_r$. При этом независимой переменной $x_g$ является признак $f_g^i$ — наличие слова (леммы) из словаря $F(c_r)$ для категории $c_r$ в текстовом документе $d_i$, т. е.:

$$x_g = \begin{cases} 1, & \text{если } f_g^i \in F(d_i), \text{ где } f_g^i \in F(c_r); \\ 0, & \text{если } f_g^i \notin F(d_i), \text{ где } f_g^i \in F(c_r). \end{cases}$$

В остальном вычисление вероятности принадлежности документа к той или иной категории по методу Байеса выполняется так же, как это описано в *разд. 5.3.2*.

Аналогичную трактовку получают зависимая и независимая переменные и в других методах классификации при использовании их для текстовых документов.

Для классификации текстовых документов были разработаны и другие методы и разрабатываются новые. Примером такого метода является классификация, основанная на полнотекстовом поиске [49]. С помощью этого метода на основе обучающей выборки формируются запросы к полнотекстовой поисковой машине, соответствующие каждой из рубрик. Затем эти запросы выполняются для исследуемого документа, и выбирается та рубрика, запросы которой в наибольшей степени соответствуют исследуемому документу. Особенностью метода является то, что результат машинного обучения представляет собой набор запросов к поисковой системе и легко интерпретируется.

# 9.4. Методы кластеризации текстовых документов

## 9.4.1. Представление текстовых документов

Большинство алгоритмов кластеризации требуют, чтобы данные были представлены в виде модели векторного пространства (vector space model) [44]. Это наиболее широко используемая модель для информационного поиска.

Она концептуально проста и использует метафору для отражения семантического подобия как пространственной близости.

В этой модели каждый документ представляется в многомерном пространстве, в котором каждое измерение соответствует слову в наборе документов.

Эта модель представляет документы матрицей слов и документов:

$$M = |F| \times |D|,$$

где $F = \{f_1, ..., f_k, ..., f_z\}$; $D = \{d_1, ..., d_i, ..., d_n\}$, $d_i$ — вектор в $z$-мерном пространстве $R^z$.

Набор признаков $F$ конструируется при помощи исключения редких слов и слов с высокой частотой. Исключение слов означает, что слова рассматриваются только как признаки, если они встречаются бóльшее количество раз, чем обозначенный частый порог, или меньшее количество раз, чем обозначенный нечастый порог. Значения порогов определяются экспериментально.

Каждому признаку $f_k$ в документе $d_i$ ставится в соответствие его вес $\omega_{k,i}$, который обозначает важность этого признака для данного документа. Для вычисления веса могут использоваться разные подходы, например алгоритм TFIDF (Term Frequency Inverse Document Frequency). Идея этого подхода — гарантировать, что вес признака будет находиться в диапазоне от 0 до 1. При этом чем чаще слово появляется в тексте, тем его вес выше, и наоборот: чем частота меньше, тем вес меньше. Формула, по которой вычисляется вес, имеет следующий вид:

$$\omega_{k,i} = \frac{\left(1 + \log\left(N_{i,k}\right)\right) \cdot \log\left(|D| / N_k\right)}{\sqrt{\sum_{s \neq k} \left(\log\left(N_{i,s}\right) + 1\right)^2}},$$

где $N_{i,k}$ — количество появлений признака $f_k$ в документе $d_i$; $N_k$ — количество появлений признака $f_k$ во всех документах множества $D$; $|D|$ — количество документов (мощность множества $D$).

Необходимо отметить, что в знаменателе находится сумма по всем документам, кроме рассматриваемого. Таким образом, вес функции нормализуется относительно всех документов. Эта модель часто называется "мешок слов" (bag-of-words).

Кроме метода TFIDF для взвешивания термов часто используется подход TLTF (Term Length Term Frequency). Идея метода TLTF базируется на том, что слова, которые появляются часто, стремятся быть краткими. Такие слова не описывают основную тему документа, т. е. являются стоп-словами. Наоборот, слова, которые появляются редко, стремятся быть длинными.

Кластеры в данной модели представляются аналогично документам в виде векторов:

$$C = \{c_1, ..., c_j, ..., c_m\},$$

где $c_j$ — вектор в $z$-мерном пространстве $R^z$. Вектор $c_j$ часто является центром кластера (центроидом).

При этом целью кластеризации является группировка документов (представленных векторами) по кластерам в соответствии с близостью их к центрам. Близость документа и кластера, представленных пространственными векторами, вычисляется как угол между этими векторами:

$$\cos(\vec{d}_i, \vec{c}_j) = \frac{\vec{d}_i \cdot \vec{c}_j}{|\vec{d}_i| \cdot |\vec{c}_j|} = \frac{\sum^{|F|} d_{i,k} \cdot c_{j,k}}{\sqrt{\sum^{|F|} {d_{i,k}}^2} \cdot \sqrt{\sum^{|F|} {d_{j,k}}^2}}.$$

Все алгоритмы кластеризации основываются на измерениях похожести по различным критериям. Некоторые используют слова, часто появляющиеся вместе (лексическую близость), другие используют извлекаемые особенности (такие как имена людей и т. п.). Разница заключается также и в создаваемых кластерах. Выделяют три основных типа методов кластеризации документов:

- *иерархический* — создает дерево со всеми документами в корневом узле и одним документом в узле-листе. Промежуточные узлы содержат различные документы, которые становятся более и более специализированными по мере приближения к листьям дерева. Этот метод полезен, когда исследуют новую коллекцию документов и хотят получить общее представление о ней;
- *бинарный* — обеспечивает группировку и просмотр документальных кластеров по ссылкам подобия. В один кластер помещаются самые близкие по своим свойствам документы. В процессе кластеризации строится базис ссылок от документа к документу, основанный на весах и совместном употреблении определяемых ключевых слов;
- *нечеткий* — включает каждый документ во все кластеры, но при этом связывает с ним весовую функцию, определяющую степень принадлежности данного документа определенному кластеру.

## 9.4.2. Иерархические методы кластеризации текстов

Как описывалось в *главе 6*, методы иерархической кластеризации бывают:

- агломеративные — кластеризация выполняется, начиная с индивидуальных элементов, группируя их в кластеры (снизу вверх);

- дивизимные — кластеризация выполняется, начиная с единого кластера и разбивая его на несколько (сверху вниз).

Иерархическая агломеративная кластеризация (HAC — Hierarchical Agglomerative Clustering) изначально представляет каждый из $N$ документов отдельным кластером. В процессе кластеризации эти кластеры объединяются, и количество кластеров уменьшается до тех пор, пока один кластер не будет содержать все $N$ документов. Такой подход различается методами группировки отдельных кластеров:

- односвязный метод группирует ближайших членов;
- полносвязный — дальних членов;
- среднесвязный — ближайших к середине членов.

Результатами такой кластеризации является дентограмма.

Представителем дивизимной иерархической кластеризации текстовых документов является алгоритм дивизимного разделения по главному направлению (PDDP — Principal Direction Divisive Partitioning). Он строит бинарное дерево, в котором каждый узел содержит документы. PDDP начинает строить дерево с корневого кластера, который содержит все документы. Далее он рекурсивно делит каждый лист дерева на два дочерних узла, пока сохраняется критерий деления. Для сохранения балансировки бинарного дерева PDDP использует функцию разброса для определения необходимости разделения узла. Эта функция вычисляет, насколько близки элементы в кластере. Например, если среднеквадратичное расстояние кластера больше заданного порогового значения, то кластер (узел дерева) должен быть разделен. Матрица слов и документов используется для определения главного направления и разделения гиперпространства.

Например, пусть имеется матрица слов и предложений. Для того чтобы разделить матрицу на две подматрицы (узла), каждый документ проектируется на главное направление. Главным направлением матрицы является собственный вектор $e = \{e_1, ..., e_2, ..., e_T\}$ ковариационной матрицы $\sum = (\vec{d} - \vec{c})(\vec{d} - \vec{c})^T$.

Проекция документа $\vec{d}_i$ определяется следующим образом:

$$v = \vec{e} \cdot (\vec{d} - \vec{c}),$$

где $v$ — это значение, которое используется, чтобы определить разделение кластера; $\vec{c}$ — центроид матрицы.

Все документы, для которых $v \le 0$, группируются в левый узел, документы, для которых $v > 0$, помещаются в правый узел. Проекция может быть интерпретирована фактом существования гиперплоскости, которая делит набор многомерных векторов на две отдельные группы.

### 9.4.3. Бинарные методы кластеризации текстов

Интерактивная кластеризация обычно создает кластеры, оптимизируя целевую функцию, описанную локально (среди документов одного и того же кластера) или глобально (через все документы).

Типичным представителем интерактивных алгоритмов является алгоритм $k$-средних (подробно описанный в *главе 6*). Он интерактивно выполняет деление данных на $k$-кластеров, минимизируя расстояния между элементами кластеров и их центрами.

Для задачи кластеризации текстовых документов он адаптируется следующим образом. Имеется множество документов:

$$D = \{d_1, ..., d_i, ..., d_n\}, \quad d_i \in R^T.$$

Алгоритм $k$-средних создает $k$ декомпозиций так, чтобы если $\{\vec{c}_1, \vec{c}_2, ..., \vec{c}_k\}$ представляет собой $k$ центров, то минимизируется следующая целевая функция:

$$j = \arg\min_j \sum_{j=1}^{k} \sum_{\vec{d}_i \in D} \| \vec{d}_i - \vec{c}_j \|^2 .$$

## 9.5. Задача аннотирования текстов

### 9.5.1. Выполнение аннотирования текстов

Задача аннотирования документов является актуальной для любых хранилищ информации: от библиотек до интернет-порталов. Аннотирование требуется также и конкретному человеку, например, для быстрого ознакомления с интересующей его публикацией или с подборкой статей по одной тематике.

В настоящее время наиболее распространено ручное аннотирование, к достоинствам которого можно отнести, безусловно, высокое качество составления аннотации — ее "осмысленность". Типичные недостатки ручной системы аннотирования — высокие материальные затраты и присущая ей низкая скорость.

Хорошее аннотирование предполагает содержание в аннотации предложений, представляющих максимальное количество тем, представленных в документе, при минимальной избыточности.

Согласно статье [45], процесс аннотирования распадается на три этапа:

1. Анализ исходного текста.
2. Определение его характерных фрагментов.
3. Формирование соответствующего вывода.

Большинство современных работ концентрируются вокруг разработанной технологии реферирования одного документа.

Выделяют два основных подхода к автоматическому аннотированию текстовых документов:

- *Извлечение* — предполагает выделение наиболее важных фрагментов (чаще всего это предложения) из исходного текста и соединение их в аннотацию.
- *Обобщение* — предполагает использование предварительно разработанных грамматик естественных языков, тезаурусы, онтологические справочники и др., на основании которых выполняется переформулирование исходного текста и его обобщение.

В подходе, основанном на извлечении фрагментов методом сопоставления шаблонов, выделяют наиболее лексически и статистически значимые части. В результате аннотация в данном случае создается простым соединением выбранных фрагментов.

В большинстве методов, основанных на данном подходе, используются весовые коэффициенты, вычисляемые для каждого фрагмента. Вычисления выполняются в соответствии с такими характеристиками, как расположение фрагмента в тексте, частота появления, частота использования в ключевых предложениях, а также показатели статистической значимости. Общий вид формулы вычисления веса фрагмента текста $U$ выглядит следующим образом:

$$\text{Weight}(U) = \text{Location}(U) + \text{KeyPhrase}(U) + \text{StatTerm}(U) + \text{AddTerm}(U).$$

Весовой коэффициент расположения (Location) в данной модели зависит от того, где во всем тексте или в отдельно взятом параграфе появляется данный фрагмент — в начале, в середине или в конце, а также используется ли он в ключевых разделах, например, во вводной части или в заключении.

Ключевые фразы представляют собой лексические резюмирующие конструкции, такие как "в заключение", "в данной статье", "согласно результатам анализа" и т. д. Весовой коэффициент ключевой фразы (KeyPhrase) может зависеть также и от принятого в данной предметной области оценочного термина, например, "отличный" (наивысший коэффициент) или "малозначащий" (значительно меньший коэффициент).

Кроме того, при назначении весовых коэффициентов в этой модели учитывается показатель статистической важности (StatTerm). Статистическая важность вычисляется на основании данных, полученных в результате анализа автоматической индексации, при которой вычисляются весовые коэффициенты лексем (например, методами TFIDF или TLTF).

И наконец, эта модель предполагает просмотр терминов в фрагменте текста и определение его весового коэффициента в соответствии с дополнительным

наличием терминов (AddTerm) — появляются ли они также в заголовке, в колонтитуле, в первом параграфе и в пользовательском запросе. Выделение приоритетных терминов, наиболее точно отражающих интересы пользователя, — это один из путей настроить аннотацию на конкретного человека или группу.

В подходе обобщения для подготовки аннотации требуются мощные вычислительные ресурсы для систем обработки естественных языков (NLP — Natural Language Processing), в том числе грамматики и словари для синтаксического разбора и генерации естественно-языковых конструкций. Кроме того, для реализации этого метода нужны некие онтологические справочники, отражающие соображения здравого смысла, и понятия, ориентированные на предметную область, для принятия решений во время анализа и определения наиболее важной информации. Данный подход предполагает использование двух основных типов методов.

Первый тип опирается на традиционный лингвистический метод синтаксического разбора предложений. В этом методе применяется также семантическая информация для аннотирования деревьев разбора. Процедуры сравнения манипулируют непосредственно деревьями с целью удаления и перегруппировки частей, например, путем сокращения ветвей на основании некоторых структурных критериев, таких как скобки или встроенные условные или подчиненные предложения. После такой процедуры дерево разбора существенно упрощается, становясь, по существу, структурной "выжимкой" исходного текста.

Второй тип методов аннотирования опирается на понимание естественного языка. Синтаксический разбор также входит составной частью в такие методы анализа, но деревья разбора в этом случае не порождаются. Напротив, формируются концептуальные структуры, отражающие всю исходную информацию, которая аккумулируется в текстовой базе знаний. В качестве структур могут быть использованы формулы логики предикатов или такие представления, как семантическая сеть или набор фреймов. Примером может служить шаблон банковских транзакций (заранее определенное событие), в котором перечисляются организации и лица, принимающие в нем участие, дата, объем перечисляемых средств, тип транзакции и т. д.

Подход, основанный на извлечении фрагментов, легко настраивается для обработки больших объемов информации. Из-за того что работа таких методов основана на выборке отдельных фрагментов, предложений или фраз, текст аннотации, как правило, лишен связности. С другой стороны, такой подход выдает более сложные аннотации, которые нередко содержат информацию, дополняющую исходный текст. Так как он опирается на формальное представление информации в документе, то его можно настроить на достаточно высокую степень сжатия, например, для рассылки сообщений на мобильные устройства.

Подход, основанный на обобщении и предполагающий опору на знания, как правило, требует полноценных источников знаний. Это является серьезным препятствием для его широкого распространения. Поэтому разработчики средств автоматического аннотирования все больше склоняются к гибридным системам, а исследователям все более успешно удается объединять статистические методы и методы, основанные на знаниях.

## 9.5.2. Методы извлечения фрагментов для аннотации

Рассмотрим метод аннотирования документов, основанный на использовании карты текстовых отношений (TRM — Text Relationship Map). Идея метода заключается в представлении текста в виде графа [46]:

$$G = (P, E),$$

где $P = \{p_1, p_2, ..., p_k, ..., p_n\}$ — взвешенные векторы слов, соответствующие фрагментам документа. Вектор включает в себя веса составляющих его слов. Например, $k$ -й фрагмент будет представлен вектором:

$$\{\omega_{k,1}, \omega_{k,2}, ..., \omega_{k,i}, ..., \omega_{k,m}\},$$

где $\omega_{k,i}$ — вес слова, находящегося в позиции $i$ фрагмента $k$ ; $E$ — множество дуг между узлами графа:

$$E = \{(p_k, p_b),\ p_k, p_b \in V\}.$$

На рис. 9.3 изображен пример такой карты. Каждый узел на карте соответствует некоторому фрагменту текста (предложению, абзацу, разделу, параграфу) и представляется взвешенным вектором термов. Связи создаются между двумя узлами, если они имеют высокую меру подобия между параграфами, которая обычно вычисляется как скалярное произведение между векторами, представляющими эти фрагменты.

$$\mathrm{sim}(p_i, p_j) = \frac{\sum_{k=1}^{m} p_{i,k} \cdot p_{j,k}}{\sqrt{\sum_{k=1}^{|m|} p_{i,k}^2} \cdot \sqrt{\sum_{k=1}^{m} {p_{j,k}}^2}}.$$

Другими словами, если имеется связь между двумя узлами, то говорят, что соответствующие фрагменты "семантически близки". Количество входящих в узел дуг на карте соответствует важности фрагмента и служит причиной его извлечения в резюме. Например, на рис. 9.3 количество входящих дуг узла $P_5$ равно 5, т. к. в него входят дуги от узлов $P_1, P_2, P_3, P_4$ и $P_6$. Это значение мак-

симально по сравнению с другими узлами. Следовательно, узел $P_5$ своим содержанием может покрыть фрагменты, соответствующие связанным с ним узлам, и он должен быть помещен в аннотацию.

Основным недостатком данного подхода является то, что учитывается только один аспект важности фрагмента, а именно: его отношение с другими фрагментами документа. Здесь не рассматривается информативность слов, имеющихся внутри отдельного фрагмента. В результате в резюме могут быть выбраны фрагменты, тесно связанные с другими, но не характеризующие тематику документа (не имеющие внутри себя ключевых слов).

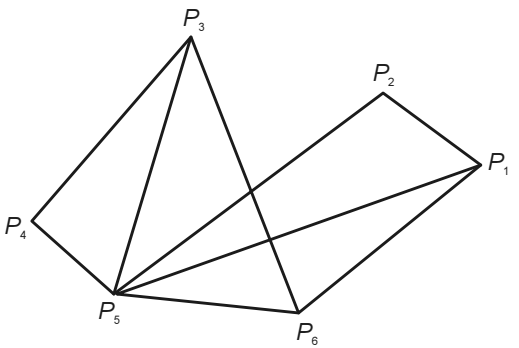



**Рис. 9.3.** Пример карты текстовых отношений

Для устранения этого недостатка в работе [47] предлагается использовать понятие локального и глобального свойства фрагмента, в качестве которого используются предложения документов. При этом в качестве локальных свойств рассматриваются кластеры слов внутри предложения, веса которых вычисляются методом TLTF. В качестве глобального свойства выступает отношение данного предложения со всеми остальными в тексте, которое определяется методом TRM. Комбинируя оба свойства, данный метод определяет степень значимости предложения и необходимость его включения в резюме.

Для вычисления кластеров слов в предложении используется не частота появления термов в тексте (как во многих методах), а более сложные правила. Если представить последовательность слов в предложении, как последовательность:

$$\beta = \{w_u, ..., w_v\},$$

то слова включаются в кластер, если выполняются следующие условия:

- первое $w_u$ и последнее $w_v$ слова в предложении значимые;
- значимые слова разделяются не более чем заранее определенным количеством незначимых слов.

Например, мы можем разделить последовательность слов в предложении следующим образом:

$$w_1\ [\mathbf{w_2}\ w_3\ \mathbf{w_4}]\ w_5\ w_6\ w_7\ w_8\ [\mathbf{w_9}\ w_{10}\ \mathbf{w_{11}}\ \mathbf{w_{12}}].$$

В этом случае предложение состоит из 12 слов. Полужирным шрифтом выделены значимые слова ($\mathbf{w_2}$, $\mathbf{w_4}$, $\mathbf{w_9}$, $\mathbf{w_{11}}$, $\mathbf{w_{12}}$). Кластеры заключены в квадратные скобки. Они сформированы согласно условию, что значимые слова должны быть разделены не более чем тремя незначимыми словами. Необходимо обратить внимание, что в предложении может быть несколько кластеров (в нашем примере их два). Наибольшее значение кластера определяет значимость предложения. Значение кластера в предложении $s_i$ вычисляется по формуле:

$$L_{s_i} = \arg\max_{\beta} \frac{ns(\beta, s_i)^2}{n(\beta, s_i)},$$

где $ns(\beta, s_i)$ — количество значимых слов в кластере; $n(\beta, s_i)$ — общее количество слов в кластере.

Как было сказано ранее, в качестве глобального свойства предложения используется его отношение с другими предложениями в документе. Оно вычисляется с помощью карты отношений в тексте (метод TRM).

Описанные локальные и глобальные свойства определяют различные аспекты значимости предложений. Локальное свойство определяет долю информации внутри предложения, а глобальное свойство больше определяет структурный аспект документа, оценивая информативность всего предложения. Для большей эффективности предлагается рассматривать оба аспекта в совокупности, объединяя их в единую оценку информативности предложения, которая может быть использована для заключения: выносить ли данное предложение в резюме или нет. Для вычисления комбинированной оценки используется формула:

$$F(s_i) = \lambda G' + (1-\lambda)L',$$

где:

- $G'$ — нормализованная глобальная связанность предложения, вычисляется по формуле:

$$G' = \frac{d_{s_i}}{d_{\max}},$$

  в которой $d_{\max}$ — максимальное количество ребер для одного узла на карте отношений в тексте, $d_{s_i}$ — количество ребер для узла соответствующего предложению $s_i$;

- $L'$ — нормализованное значение локальной кластеризации предложения $s_i$, вычисляется по формуле:

$$L' = \frac{L_{s_i}}{L_{\max}},$$

где $L_{\max}$ — максимальная локальная кластеризация во всем тексте;

- $\lambda$ — параметр, изменяющийся в зависимости от важности составляющих $G'$ или $L'$.

Таким образом, получается интегрированная оценка для всех предложений, на основании которой можно сделать выбор предложений в резюме.

# 9.6. Средства анализа текстовой информации[1]

## 9.6.1. Средства Oracle — Oracle Text [2]

Начиная с версии Oracle 7.3.3, средства текстового анализа являются неотъемлемой частью продуктов Oracle. В Oracle9i эти средства развились и получили новое название — Oracle Text — программный комплекс, интегрированный в СУБД, позволяющий эффективно работать с запросами, относящимися к неструктурированным текстам. При этом обработка текста сочетается с возможностями, которые предоставлены пользователю для работы с реляционными базами данных. В частности, при написании приложений для обработки текста стало возможно использование SQL. Данное средство входит в состав и последней версии Oracle 11g.

Система Oracle Text обеспечивает решение следующих задач анализа текстовой информации:

- поиск документов по их содержанию;
- классификацию документов;
- кластеризацию документов;
- извлечение ключевых понятий;
- автоматическое аннотирование;
- поиск в документах ассоциативных связей;

---

[1] Обзор подготовлен по материалам статьи Дмитрия Ландэ "Глубинный анализ текстов. Технология эффективного анализа текстовых данных".

[2] **http://technet.oracle.com/products/text/content.html**.

Основной задачей, на решение которой нацелены средства Oracle Text, является задача поиска документов по их содержанию — по словам или фразам, которые при необходимости комбинируются с использованием булевых операций. Результаты поиска ранжируются по значимости, с учетом частоты встречаемости слов запроса в найденных документах. Для повышения полноты поиска Oracle Text предоставляет ряд средств расширения поискового запроса, среди которых можно выделить три группы:

1. Расширение слов запроса всеми морфологическими формами, что реализуется привлечением знаний о морфологии языка.
2. Расширение слов запроса близкими по смыслу словами за счет подключения тезауруса — семантического словаря.
3. Расширение запроса словами, близкими по написанию и по звучанию — нечеткий поиск и поиск созвучных слов. Нечеткий поиск целесообразно применять при поиске слов с опечатками, а также в тех случаях, когда возникают сомнения в правильном написании фамилии, названия организации и т. п.

Все описанные средства могут использоваться совместно, что поддерживается языком запросов в сочетании с традиционным синтаксисом SQL и PL/SQL для поиска документов. Oracle Text предоставляет возможность работать с современными реляционными СУБД в контексте сложного многоцелевого поиска и анализа текстовых данных.

Возможности обработки текстовой информации на русском языке в Oracle Text достаточно ограничены. Для решения этой проблемы компанией "Гарант-Парк-Интернет" был разработан модуль Russian Context Optimizer (RCO), предназначенный для совместного использования с InterMedia Text (или Oracle Text). Помимо поддержки русскоязычной морфологии, RCO включает в себя средства нечеткого поиска, тематического анализа и реферирования документов.

## 9.6.2. Средства от IBM — Intelligent Miner for Text[1]

Продукт фирмы IBM Intelligent Miner for Text представляет собой набор отдельных утилит, запускаемых из командной строки или из скриптов независимо друг от друга. Система содержит следующие основные утилиты для решения задач анализа текстовой информации:

- утилита определения языка (Language Identification Tool) — автоматическое определение языка, на котором составлен документ;
- утилита классификации (Categorisation Tool) — автоматическое отнесение текста к некоторой категории (входной информацией на обучающей фазе

[1] **http://www-3.ibm.com/software/data/iminer/fortext/**.

работы этого инструмента может служить результат работы следующей утилиты — Clusterisation Tool);

- утилита кластеризации (Clusterisation Tool) — разбиение большого множества документов на группы по близости стиля, формы, различных частотных характеристик выявляемых ключевых слов;
- утилита извлечения ключевых понятий (Feature Extraction Tool) — выявление в документе ключевых слов (собственные имена, названия, сокращения) на основе анализа заданного заранее словаря;
- утилита автоматического аннотирования (Annotation Tool) — аннотации к исходным текстам.

IBM Intelligent Miner for Text объединяет мощную совокупность инструментов, базирующихся в основном на механизмах поиска информации (information retrieval), что является спецификой всего продукта. Система включает ряд базовых компонентов, которые имеют самостоятельное значение вне пределов технологии Text Mining:

- Text Search Engine — информационно-поисковая система;
- Web crawler — утилита сканирования Web-пространства;
- Net Question Solution — решение для поиска на локальном Web-сайте или на нескольких интранет/интернет-серверах;
- Java Sample GUI — набор интерфейсов Java Beans для администрирования и организации поиска на основе Text Search Engine.

Продукт IBM Intelligent Miner for Text включен в комплекс "Information Integrator for Content" для СУБД DB2 в качестве средства анализа информации.

## 9.6.3. Средства SAS Institute – Text Miner[1]

Американская компания SAS Institute выпустила систему SAS Text Miner для сравнения определенных грамматических и словесных рядов в письменной речи. Text Miner весьма универсальна, поскольку может работать с текстовыми документами различных форматов — в базах данных, файловых системах и даже в Web.

Text Miner обеспечивает логическую обработку текста в среде пакета SAS Enterprise Miner. Это позволяет пользователям обогащать процесс анализа данных, интегрируя неструктурированную текстовую информацию с существующими структурированными данными, такими как возраст, доход и характер покупательского спроса.

[1] **http://www.sas.com/technologies/analytics/datamining/textminer/**.

Пример успешного использования логических возможностей Text Miner демонстрирует компания Compaq Computer Corp., которая в настоящее время тестирует Text Miner, анализируя более 2,5 Гбайт текстовых документов, полученных по e-mail и собранных представителями компании. Ранее обработать такие данные было практически невозможно.

Программа Text Miner позволяет определять, насколько правдив тот или иной текстовый документ. Обнаружение лжи в документах производится путем анализа текста и выявления изменений стиля письма, которые могут возникать при попытке исказить или скрыть информацию. Для поиска таких изменений используется принцип, заключающийся в поиске аномалий и трендов среди записей баз данных без выяснения их смысла. При этом в Text Miner включен обширный набор документов различной степени правдивости, чья структура принимается в качестве шаблонов. Каждый документ, "прогоняемый" на детекторе лжи, анализируется и сравнивается с этими эталонами, после чего программа присваивает документу тот или иной индекс правдивости. Особенно полезной программа может стать в организациях, получающих большой объем электронной корреспонденции, а также в правоохранительных органах для анализа показаний наравне с детекторами лжи, чье действие основано на наблюдении за эмоциональным состоянием человека.

Интересен пример использования Text Miner в медицине. В одной из американских национальных здравоохранительных организаций было собрано свыше 10 тысяч врачебных записей о заболеваниях сердца, собранных из клиник по всей стране. Анализируя эти данные с помощью Text Miner, специалисты обнаружили некоторые административные нарушения в отчетности, а также смогли определить взаимосвязь между сердечно-сосудистыми заболеваниями и другими недугами, которые не были определены традиционными методами.

Вместе с тем, компания SAS Institute отмечает, что выпустит свой продукт Text Miner в основном для привлечения внимания бизнес-интеллигенции.

## 9.6.4. Средства Мегапьютер Интеллидженс — TextAnalyst[1]

Российская компания Мегапьютер Интеллидженс, известная своей системой PolyAnalyst класса Data Mining, разработала также систему TextAnalyst. Она решает следующие задачи Text Mining:

- создание семантической сети большого текста;
- автоматическое аннотирование текста;

[1] **http://www.megaputer.com/products/ta/index.php3**.

- поиск по тексту;
- классификацию документов;
- кластеризацию текстов.

Система TextAnalyst рассматривает технологию Text Mining в качестве отдельного математического аппарата, который разработчики программного обеспечения могут встраивать в свои продукты, не опираясь на платформы информационно-поисковых систем или СУБД. Основная платформа для применения системы — Microsoft Windows 9*x*/2000/NT. Существует плагин TextAnalyst для браузера Microsoft Internet Explorer.

# Выводы

По результатам данной главы можно сделать следующие выводы.

- Обнаружение знаний в тексте — это нетривиальный процесс обнаружения действительно новых, потенциально полезных и понятных шаблонов в неструктурированных текстовых данных.
- Процесс анализа текстовых документов можно представить как последовательность нескольких шагов: поиск информации, предварительная обработка документов, извлечение информации, применение методов Text Mining, интерпретация результатов.
- Обычно используют следующие приемы удаления неинформативных слов и повышения строгости текстов: удаление стоп-слов, стемминг, *N*-граммы, приведение регистра.
- Задачами анализа текстовой информации являются: классификация, кластеризация, автоматическое аннотирование, извлечение ключевых понятий, навигация по тексту, анализ трендов, поиск ассоциаций и др.
- Извлечение ключевых понятий из текстов может рассматриваться и как отдельная прикладная задача, и как отдельный этап анализа текстов. В последнем случае извлеченные из текста факты используются для решения различных задач анализа.
- Процесс извлечения ключевых понятий с помощью шаблонов выполняется в две стадии: на первой из текстовых документов извлекаются отдельные факты с помощью лексического анализа, на второй стадии выполняется интеграция извлеченных фактов и/или вывод новых фактов.
- Большинство методов классификации текстов так или иначе основаны на предположении, что документы, относящиеся к одной категории, содержат одинаковые признаки (слова или словосочетания), и наличие или от-

сутствие таких признаков в документе говорит о его принадлежности или непринадлежности к той или иной теме.

- Большинство алгоритмов кластеризации требуют, чтобы данные были представлены в виде модели векторного пространства, которая широко применяется для информационного поиска и использует метафору для отражения семантического подобия как пространственной близости.
- Выделяют два основных подхода к автоматическому аннотированию текстовых документов: извлечение (выделение наиболее важных фрагментов) и обобщение (использование предварительно собранных знаний).

ГЛАВА 10

# Стандарты Data Mining

## 10.1. Кратко о стандартах

Стандарты затрагивают три основных аспекта Data Mining. Во-первых, унификацию интерфейсов, посредством которых любое приложение может получить доступ к функциональности Data Mining. Здесь сложилось два направления. Это стандартизация интерфейсов для объектных языков программирования (CWM Data Mining, JDM, OLE DB for Data Mining) и попытки разработки надстройки для языка SQL, которая позволяла бы обращаться к инструментарию Data Mining, встроенному непосредственно в реляционную базу данных (SQL/MM, OLE DB for Data Mining).

Второй аспект стандартизации — это выработка единого соглашения по хранению и передаче моделей Data Mining. Нетрудно догадаться, что основой для подобного стандарта является язык XML. Сам стандарт носит название PMML (Predicted Model Markup Language). И наконец, существует стандарт CRISP, который дает рекомендации по организации процесса Data Mining в целом.

Отношения между стандартами можно представить в виде, изображенном на рис. 10.1.

## 10.2. Стандарт CWM

### 10.2.1. Назначение стандарта CWM

Стандарт CWM (Common Warehouse Metamodel) — это стандарт, разработанный консорциумом OMG для обмена метаданными между различными программными продуктами и репозиториями, участвующими в создании корпоративных СППР. Он основан на открытых объектно-ориентированных

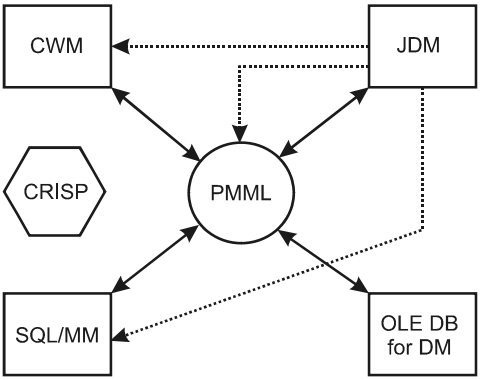



**Рис. 10.1.** Отношения между основными стандартами Data Mining

технологиях и стандартах, использует UML (Unified Modeling Language) в качестве языка моделирования, XML и XMI (XML Metadata Interchange) для обмена метаданными и язык программирования Java для реализации моделей и спецификаций.

Центральное место в технологии хранилищ данных и аналитических систем занимают вопросы управления метаданными, среди которых одной из наиболее сложных является проблема обмена данными между различными базами данных, репозиториями и продуктами. Прежде всего это связано с тем, что в любой СППР одновременно участвуют различные компоненты: базы данных, играющие роль информационных источников, хранилища и витрины, средства сбора данных, их согласования, преобразования и загрузки в целевые базы данных (ETL-средства), а также аналитические средства, поддерживающие различные технологии анализа, включая отчеты, нерегламентированные запросы, многомерный анализ (OLAP), извлечение знаний (Data Mining). Каждый из этих компонентов имеет свои метаданные, хранящиеся в соответствующем репозитории или словаре данных в специальных форматах. Проблема состоит в том, что все эти разнородные по структуре и синтаксису метаданные семантически взаимосвязаны, т. е. для согласованной и корректной работы системы в целом их необходимо передавать от одних средств другим, совместно использовать, устранять несоответствия, противоречия и т. д. Чтобы решить эту проблему, необходимы общие и достаточно универсальные стандарты для представления всевозможных метаданных, используемых в области хранилищ данных и аналитических систем.

Проект по выработке такого стандарта был организован консорциумом Object Management Group (OMG). Эта организация занимается разработкой стандартов на основе объектно-ориентированных подходов, в ее деятельности участ-

вуют более 500 различных компаний. Именно OMG был разработан и принят стандарт CORBA, существенно повлиявший на технологию распределенных вычислений и развитие компонентного подхода. Начиная с 1995 г. группа OMG активно работает в области моделирования метаданных. В 1997 г. консорциум принял и опубликовал стандарты UML (Unified Modeling Language) и MOF (Meta Object Facility), в 1999 г. — XMI (XML Metadata Interchange). В 1998 г. OMG начинает проект по созданию нового стандарта для обмена метаданными в хранилищах данных. В рабочую группу вошли представители нескольких компаний, ведущую роль среди которых играли специалисты из IBM, Oracle, Unisys, NCR, Hyperion. В это время подобная деятельность уже велась в рамках конкурирующей организации Meta Data Coalition (MDC), которая предложила свой стандарт Open Information Model (OIM). Окончательные спецификации для CWM были представлены рабочей группой в январе 2000 г. и приняты OMG в июле того же года, после чего в сентябре MDC объявила о прекращении независимой деятельности и слиянии с OMG для продолжения работ по усовершенствованию CWM и интеграции в него некоторых элементов OIM. В результате в настоящее время существует единый официально признанный стандарт CWM 1.1.

## 10.2.2. Структура и состав CWM

В основе CWM лежит модельно-ориентированный подход к обмену метаданными, согласно которому объектные модели, представляющие специфические для конкретного продукта метаданные, строятся в соответствии с синтаксическими и семантическими спецификациями некоторой общей метамодели. Это означает наличие общей системы фундаментальных понятий данной области, с помощью которых любой продукт должен "понимать" широкий спектр моделей, описывающих конкретные экземпляры метаданных.

CWM имеет модульную структуру, что позволяет минимизировать зависимости между различными компонентами, уменьшить сложность и повысить наглядность модели. Под модулем в данном случае понимается отдельная метамодель (или средство моделирования), предназначенная для представления определенного типа метаданных хранилища. Например, для представления метаданных процессов преобразований и загрузки используется метамодель "Преобразование", для спецификации особенностей многомерного анализа — метамодель "OLAP" и т. д. Каждая метамодель реализована в виде пакета, содержащего набор описанных на UML базовых классов. В CWM максимально используются существующие классы UML, и только в особых случаях определяются их специфические расширения.

Все пакеты структурированы и распределены по четырем слоям (рис. 10.2).

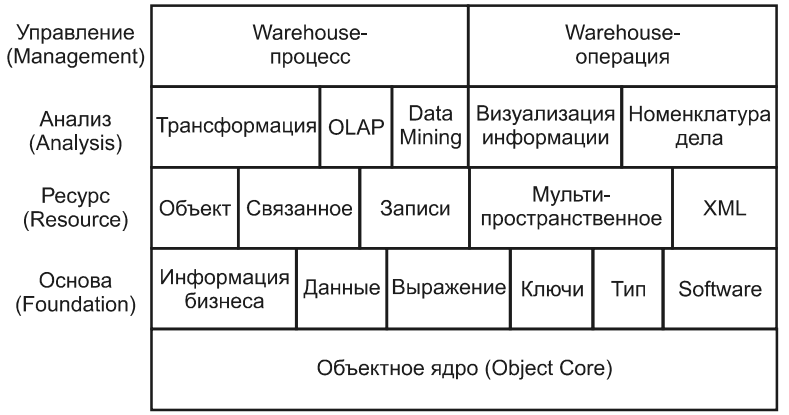



**Рис. 10.2.** Структура и состав CWM

Объектное ядро (ObjectCore) включает четыре пакета:

- Core (ядро) — содержит классы и ассоциации, которые формируют ядро объектной модели CWM и используются всеми другими пакетами, включая пакеты ObjectModel;
- Behavior (поведение) — содержит классы и ассоциации, которые описывают поведение CWM-объектов и обеспечивают основу для описания вызовов, определенных поведением;
- Relationships (отношения) — содержит классы и ассоциации, которые описывают отношения между CWM-объектами;
- Instance (экземпляр) — содержит классы и ассоциации, которые представляют классификаторы CWM.

Самый нижний слой метамодели CWM — Foundation (основа) — состоит из пакетов, которые поддерживают спецификацию базовых структурных элементов, таких как выражения, типы данных, типы отображений и др. Все они совместно используются пакетами верхних уровней. В него входят следующие пакеты:

- Business Information (бизнес-информация) — содержит классы и ассоциации, которые представляют бизнес-информацию об элементах модели;
- Data Types (типы данных) — содержит классы и ассоциации, которые представляют конструкторы, используемые при необходимости для создания специфичных типов данных;
- Expressions (выражения) — содержит классы и ассоциации, которые представляют деревья выражений;

- Keys and Indexes (ключи и индексы) — содержит классы и ассоциации, которые представляют ключи и индексы;
- Software Deployment (размещение программ) — содержит классы и ассоциации, которые описывают способ размещения программного обеспечения в хранилище данных;
- Type Mapping (отображение типов) — содержит классы и ассоциации, которые представляют отображение типов данных между разными системами.

Второй слой — Resource (ресурс) — содержит пакеты, используемые для описания информационных источников и целевых баз данных:

- Relational (реляционный) — содержит классы и ассоциации, которые представляют метаданные реляционных источников данных;
- Record (запись) — содержит классы и ассоциации, которые представляют отдельные записи источников данных;
- Multidimensional (многомерный) — содержит классы и ассоциации, которые представляют метаданные многомерных источников данных;
- XML — содержит классы и ассоциации, которые описывают метаданные источников данных, представленных в формате XML.

Третий слой называется Analysis (анализ) и содержит средства моделирования процессов или служб информационного анализа, включая визуализацию и распространение данных, многомерный анализ, извлечение знаний (Data Mining) и др. Он содержит следующие пакеты:

- Transformation (преобразование) — содержит классы и ассоциации, которые представляют инструментарий преобразования данных;
- OLAP — содержит классы и ассоциации, которые представляют метаданные инструментов оперативного анализа данных;
- Data Mining — содержит классы и ассоциации, которые представляют метаданные инструментов Data Mining;
- Information Visualization (информационная визуализация) — содержит классы и ассоциации, которые представляют метаданные инструментов визуализации информации;
- Business Nomenclature (бизнес-номенклатура) — содержит классы и ассоциации, которые представляют метаданные таксономии и глоссарии бизнеса.

И наконец, четвертый слой — Management (управление) — состоит из пакетов, относящихся к особенностям функционирования хранилища. Эти средства позволяют моделировать процедуры по управлению хранилищем, уста-

навливать регламент их выполнения, специфицировать процессы контроля и протоколирования для загрузки информации и произведенных корректировок данных хранилища. В его состав входят два пакета:

- Warehouse Process (процессы хранилища данных) — содержит классы и ассоциации, которые представляют метаданные процессов хранилищ данных;
- Warehouse Operation (операции хранилища данных) — содержит классы и ассоциации, которые представляют метаданные результатов операций хранилищ данных.

## 10.2.3. Пакет Data Mining

Для того чтобы лучше понять, что собой представляют пакеты CWM, рассмотрим более подробно классы и ассоциации из пакета Data Mining. Данный пакет разделен на три концептуальные области:

- Model — описание метаданных моделей, получаемых в результате работы методов Data Mining;
- Settings — описание метаданных настроек процесса построения моделей;
- Attributes — описание метаданных для атрибутов данных.

**Метамодель Model** — состоит из общего представления моделей Data Mining. Она представляет собой структуру, описывающую результат работы алгоритмов Data Mining (например, дерево решений, правила и т. п.).

В данную метамодель включены следующие классы (рис. 10.3):

- `MiningModel` — представляет модель Mining;
- `MiningSettings` — описывает настройки процесса конструирования модели;
- `ApplicationInputSpecification` — определяет набор входных атрибутов для модели;
- `MiningModelResult` — представляет результат проверки или применения сгенерированной модели.

Класс `SupervisedMiningModel` наследуется от класса `MiningModel` и используется в задачах supervised (классификации и регрессии), поэтому он нуждается в определении зависимой переменной — `target`.

Атрибут `function` класса `MiningModel` определяет вид функции, выполняемой моделью Data Mining (например, ассоциативные правила), а атрибут `algorithm` определяет алгоритм, породивший модель (например, Naive Bayes).

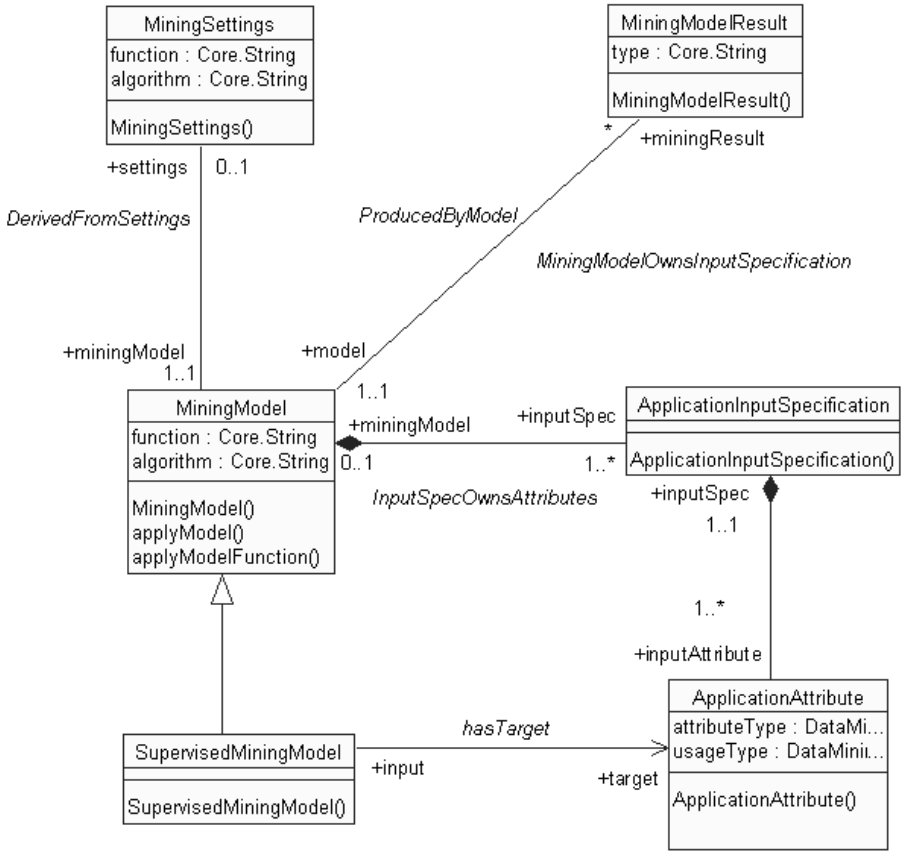



**Рис. 10.3.** Диаграмма классов метамодели Model

**Метамодель Settings** — конкретизирует настройки процесса построения моделей и их отношения с атрибутами входной спецификации. Данная метамодель включает в себя четыре подкласса класса `MiningSettings`, представляющих настройки для использования при решении конкретных задач (рис. 10.4):

- `StatisticsSettings` — для задач статистики;
- `ClusteringSettings` — для задач кластеризации;
- `SupervisedMiningSettings` — для задач с учителем. Он имеет два подкласса:
  - `ClassificationSettings` — для задачи классификации;
  - `RegressionSettings` — для задачи регрессии;
- `AssociationRulesSettings` — для задач поиска ассоциативных правил.

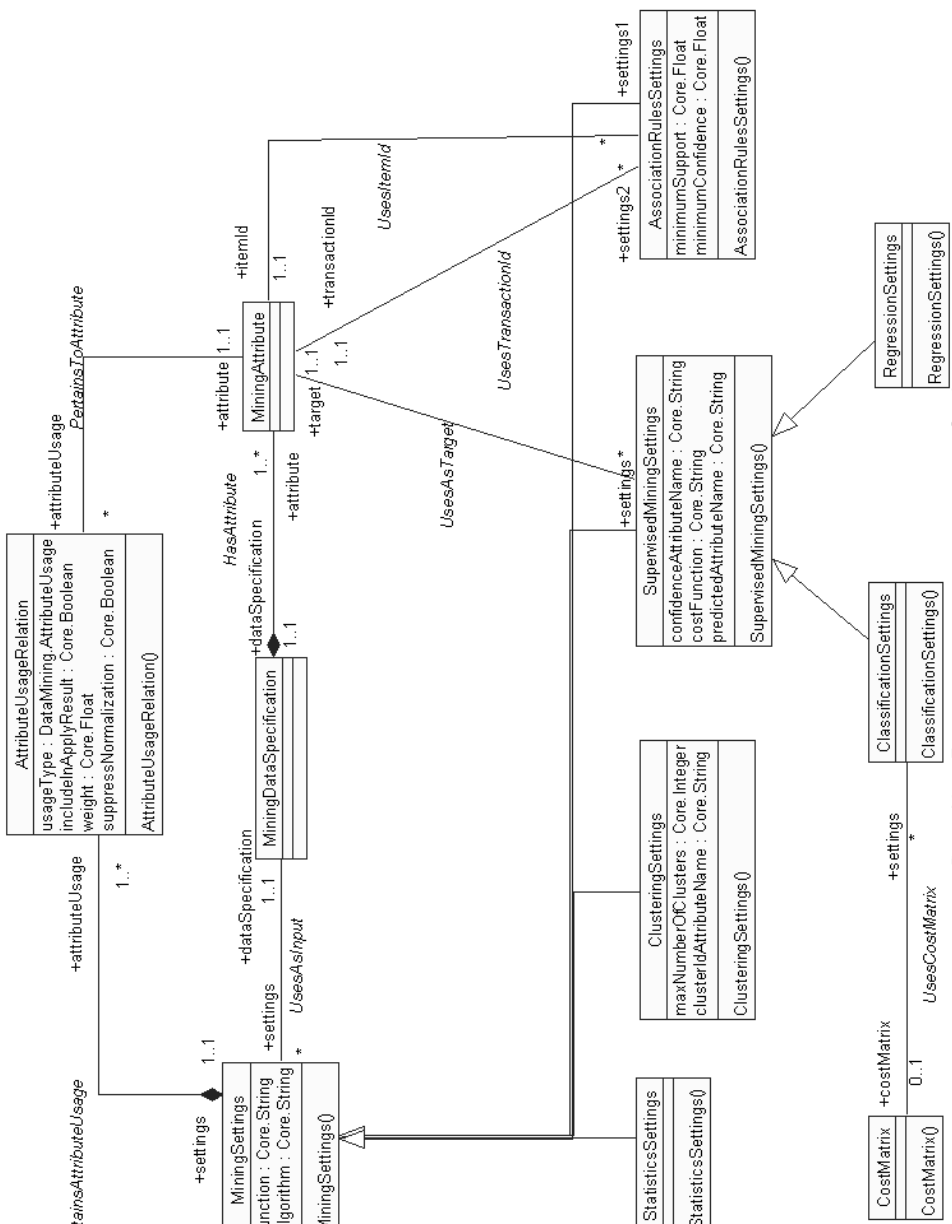



**Рис. 10.4.** Диаграмма классов метамодели Settings

Класс `CostMatrix` используется для представления стоимости ошибок классификации.

Класс `AttributeUsageRelation` состоит из атрибутов, описываемых классом `MiningAttributes` и используемых классом `MiningSettings`. Применяются также ассоциации, чтобы ясно описать требования, накладываемые на атрибуты определенными подклассами настроек (например, чтобы указать, кто из них является зависимой переменной, кто идентификатором объектов и т. п.).

**Метамодель Attributes** — описывает два подкласса класса `MiningAttribute` для разных типов атрибутов (рис. 10.5):

- `NumericAttribute` — для числовых атрибутов;
- `CategoricalAttribute` — для категориальных атрибутов. Данный класс имеет подкласс `OrdinalAttribute`, который используется для упорядоченных категориальных значений.

Класс `CategoryHierarchy` представляет любую иерархию, с которой может быть связан класс `CategoricalAttribute`.

> ***Примечание***
> Необходимо заметить, что `MiningAttribute` наследуется от класса `Attribute` из пакета Core.

Таким образом, в пакете Data Mining стандарта CWM описаны все основные метаданные, необходимые для реализации соответствующих методов в СППР. Описанные классы могут быть расширены в зависимости от конкретных задач, но их использование гарантирует, что такая реализация будет совместима с другими системами, поддерживающими стандарт CWM.

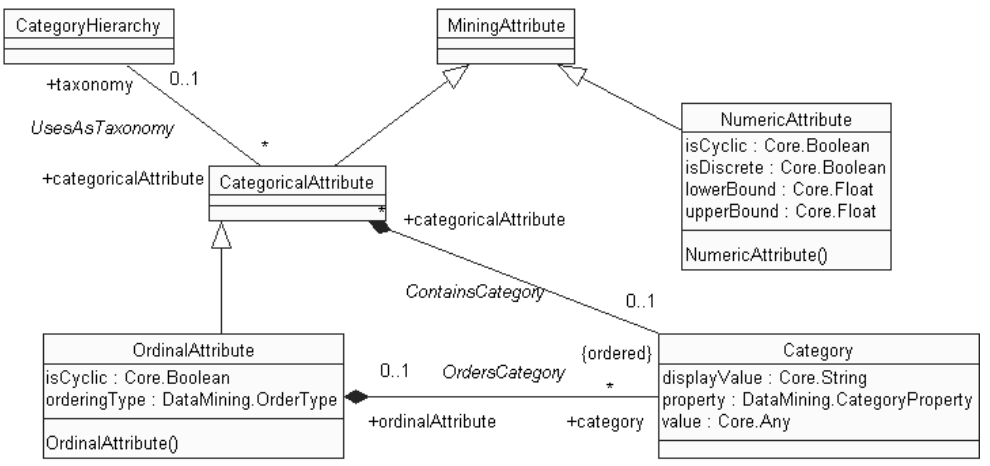



**Рис. 10.5.** Диаграмма классов метамодели Attributes

# 10.3. Стандарт CRISP

## 10.3.1. Появление стандарта CRISP

С ростом интереса к Data Mining возрастала и необходимость в разработке методологии создания таких систем. Эти потребности призван удовлетворить стандарт CRISP-DM (CRoss-Industry Standard Process for Data Mining) — непатентованная, документированная и свободно доступная модель, описывающая основные фазы, выполнение которых позволяет организациям получать максимальную выгоду от использования методов Data Mining.

Разработка CRISP была начата в 1996 г. компаниями Daimler-Benz (теперь DaimlerChrysler), Integral Solutions Ltd. (ISL), NCR и OHRA. Годом позже сформировался консорциум, целью которого стала разработка независимого от индустрии, прикладной области и используемых инструментов стандарта CRISP-DM. Консорциум привлек к решению этой задачи профессионалов широкого спектра, имеющих интерес к Data Mining (разработчиков хранилищ данных, консультантов по менеджменту и др.). Консорциум получил название CRISP-DM Special Interest Group, или кратко — SIG. В настоящее время он объединяет компании, специализирующиеся на анализе данных: NCR, SPSS, DaimlerChrysler и OHRA.

По истечении нескольких лет CRISP-DM SIG утвердил модель процесса разработки приложений Data Mining. Данный стандарт был опробован на проектах компаний Daimler-Benz и OHRA. В 2000 г. состоялась презентация первой версии стандарта CRISP-DM 1.0. В настоящее время ведется работа над версией 2.0 данного стандарта.

## 10.3.2. Структура стандарта CRISP

Стандарт CRISP-DM описывается в терминах иерархической модели процесса (рис. 10.6). Модель состоит из набора задач, описанных на четырех уровнях абстракции (от более общего к более конкретному): фазы, общие задачи, специализированные задачи и примеры процессов.

На верхнем уровне процесса Data Mining выделяется несколько фаз разработки. Каждая из них включает в себя несколько общих задач, относящихся ко второму уровню иерархии. Второй уровень называется общим ввиду того, что задачи, составляющие его, не учитывают особенностей прикладной области, для которой они решаются. Предполагается, что они являются законченными и неизменными. Законченность означает покрытие как всего процесса, так и возможных приложений Data Mining. В свою очередь, неизменность означает, что модель должна быть актуальной и для неизвестных до сих пор методов Data Mining.

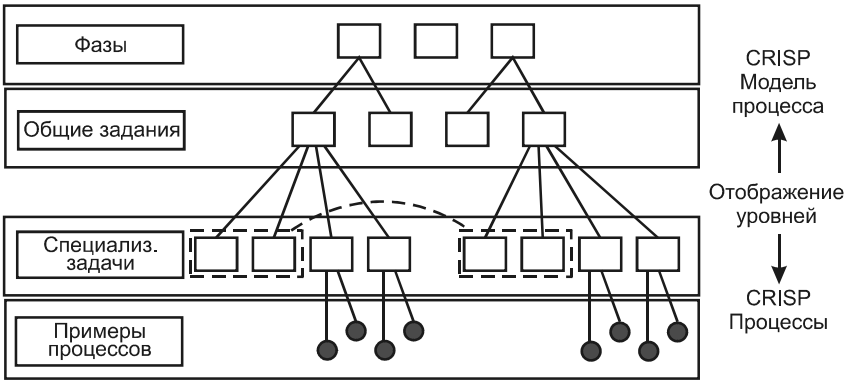



**Рис. 10.6.** Четыре уровня методологии CRISP

Третий уровень специализированных задач включает описание шагов, необходимых для адаптации общих задач к решению специализированных проблем. Например, общая задача очистки данных, описанная на втором уровне, на третьем уровне может быть представлена определенными задачами для конкретных ситуаций: задача очистки числовых данных, очистки категориальных данных и т. п.

На четвертом уровне представлены действия, решения и результаты, реально встречающиеся в Data Mining. Данный уровень организован в соответствии с задачами верхнего уровня, но в то же время представляет собой конкретные практические задачи.

Рассмотрим более подробно два верхних уровня модели: фазы проекта Data Mining и общие задачи каждой из фаз. CRISP-DM делит жизненный цикл проекта Data Mining на следующие шесть фаз:

- понимание бизнес-процессов (business understanding);
- понимание данных (data understanding);
- подготовка данных (data preparation);
- моделирование (modeling);
- оценка (evaluation);
- размещение (deployment).

На рис. 10.7 изображены перечисленные фазы и взаимоотношения между ними. Стрелками изображены более важные и частые зависимости между фазами. Внешние стрелки, имеющие циклическую природу, иллюстрируют спиралеобразный процесс разработки проектов Data Mining. Другими словами, после фазы размещения может возникнуть необходимость в новом переосмыслении бизнес-процессов и повторения всех шести фаз сначала.


**Рис. 10.7.** Жизненый цикл процесса Data Mining согласно методологии CRISP

## 10.3.3. Фазы и задачи стандарта CRISP

**Фаза понимания бизнес-процессов** — возможно, наиболее важная фаза в проекте Data Mining. Здесь должно быть уделено достаточно внимания целям проекта с точки зрения перспективности бизнеса, определения знаний в формулировке Data Mining проблемы и дальнейшей разработки первичного плана достижения целей. Чтобы понять, какие данные и как в дальнейшем они должны быть проанализированы, важным является полностью понять бизнес, для которого происходит поиск решения.

Эта фаза включает следующие задачи:

- определение бизнес-целей;
- определение ситуации;
- определение целей Data Mining;
- создание плана проекта.

Первой задачей является анализ истинных целей клиента. Это важный шаг, который должен гарантировать, что в результате разработанная система будет правильно решать именно те проблемы, которые интересуют пользователя в первую очередь. Чтобы достичь этого, необходимо обнаружить первоначальные бизнес-цели и правильно сформулировать соответствующие вопросы.

Хороший анализ позволяет определить меры успеха. Он может измеряться снижением потерь пользователя на 10 % или просто улучшением понимания данных. При анализе необходимо опасаться постановки недостижимых целей. Также должна быть уверенность, что каждый критерий успеха имеет отношение как минимум к одной определенной цели бизнеса.

При решении второй задачи выполняется первоначальная оценка ресурсов (от персонала до программного обеспечения), необходимых для завершения проекта. Важно удостовериться, что данные, подвергаемые анализу, относятся к решаемым первичным целям бизнеса. Необходимо составить список допущений для проекта, а также составить список рисков, которые могут возникнуть при реализации проекта, и предложения по их устранению, бизнес-глоссарий и список терминов Data Mining. В это же время можно определить приблизительную выгоду в денежном выражении от данного проекта.

Следующей задачей является формулировка целей Data Mining в терминах бизнеса, например, "предсказать на основании информации о закупках за последние три года и демографической информации, какой объем товара потребитель будет покупать". Успех достижения данных целей также должен быть описан в этих терминах, например, успехом является достижение определенного уровня точности предсказания. Если бизнес-цели не могут быть эффективно переведены в цели Data Mining, то это может быть поводом для пересмотра решаемых проблем.

Последняя задача, решаемая в этой фазе, — составление плана проекта, который последовательно описывает намерения для достижения целей Data Mining, включая набросок конкретных шагов, интервалы времени, первоначальную оценку потенциальных рисков, необходимый инструментарий и методы для поддержания проекта в рабочем состоянии. Общепринято, что 50...70 % времени и усилий при разработке проекта Data Mining требуется на фазу подготовки данных; 20...30 % — на фазу понимания данных; 10...20 % — на каждую из фаз моделирования, оценки и понимания бизнеса и 5...10 % — на фазу размещения.

**Фаза понимания данных** — начинается с первоначального сбора данных. Затем происходит более близкое знакомство с ними с целью идентифицировать проблемы качества данных, исследовать их суть и выявить интересные поднаборы для формирования гипотез о скрытых знаниях. В этой фазе выполняются четыре общие задачи:

- первичный сбор данных;
- описание данных;
- изучение данных;
- проверка качества данных.

При решении первой задачи данные загружаются и при необходимости интегрируются в единое ХД. В результате должен быть создан отчет о проблемах, возникших в процессе работы с ХД, и способах их решения, чтобы избежать повторения в будущем. Например, данные могут собираться из различных источников, отдельные из которых имеют большое время задержки. Информация об этом и правильное планирование загрузки может помочь оптимизировать время в будущем.

При решении задачи описания данных выполняется грубое исследование свойств полученных метаданных, по результатам составляется отчет, куда включается информация о формате данных, их качестве, количестве записей и полей в каждой таблице, идентификаторов полей и другие поверхностные свойства данных. Главный вопрос, на который должен быть получен ответ: удовлетворяют ли данные предъявляемым к ним требованиям?

При решении третьей задачи уточняются вопросы к данным, которые могут быть адресованы с использованием запросов, визуализации и отчетов. На этом шаге должен быть создан отчет исследования данных, который описывает первые найденные решения, первоначальные гипотезы и потенциальные коллизии, которые могут возникнуть в оставшейся части проекта.

Последней задачей во второй фазе является проверка данных, в результате которой необходимо ответить на вопросы — являются ли данные полными, часто ли встречаются пропущенные значения, особенно если данные были собраны из разных источников через длинные периоды времени? Должны быть проверены некоторые общие элементы и связанные с ними вопросы: пропущенные атрибуты и поля; все ли возможные значения представлены; достоверность значений; орфография значений; имеют ли атрибуты с разными значениями сходный смысл (например, мало жира, малокалорийный). Необходимо проверить также значения, которые противоречат здравому смыслу (например, подросток с высоким уровнем дохода).

**Фаза подготовки данных** — третья фаза, включающая в себя все действия, связанные с окончательным формированием набора данных для анализа. При этом выполняются пять задач:

- ❑ выбор данных;
- ❑ очистка данных;
- ❑ конструирование данных;
- ❑ интеграция данных;
- ❑ форматирование данных.

При выборе данных, которые будут использованы для анализа, опираются на несколько критериев: существенность для достижения целей Data Mining, качество и технические ограничения, накладываемые на них (такие как огра-

ничения на объем или типы данных). Частью данного процесса должно являться объяснение, почему определенные данные были включены или исключены.

*Очистка данных* представляет собой выбор "чистых" данных или использование специальных методов, таких как оценка пропущенных данных путем моделирования анализа.

После очистки должны быть совершены *подготовительные операции*, такие как добавление новых записей или порождение вторичных атрибутов. Например, новая запись должна соответствовать пустым покупкам для потребителей, не совершавших покупки в течение последнего года. Вторичные атрибуты отличаются от новых, которые конструируются на основании уже существующих (например, площадь = длина × ширина). Вторичные атрибуты должны добавляться, только если они облегчают процесс моделирования или необходимы для применения определенного метода Data Mining, не уменьшая количество входных атрибутов. Например, возможно, атрибут "доход главы" лучше (легче) использовать, чем "доход семейства". Другой тип вторичных атрибутов — одноатрибутная трансформация, обычно выполняемая для применения инструментов моделирования. Трансформация может понадобиться, чтобы преобразовать, например, категориальные поля в числовые.

Интеграция данных включает в себя комбинирование информации из множества таблиц для создания новых записей или значений, например, может связать одну или более таблиц, содержащих информацию об одном объекте. При решении этой задачи также выполняется агрегация. Агрегация предполагает операцию вычисления нового значения путем суммирования информации из множества записей и/или таблиц.

В некоторых случаях приходится решать задачи форматирования данных. Форматирование может быть как простое (удаление лишних пробелов из строки), так и более сложное (реорганизация информации). Иногда форматирование необходимо для использования подходящего инструмента моделирования. В других случаях форматирование нужно для формулирования необходимых вопросов Data Mining.

**Фаза моделирования** — предназначена для выбора оптимального метода построения моделей и настройки его параметров для получения оптимальных решений. Обычно для одних и тех же проблем Data Mining существуют несколько методов решения. Некоторые из них накладывают определенные требования на данные, поэтому может понадобиться возврат на предыдущую фазу. В данной фазе решаются следующие задачи:

- выбор метода моделирования;
- генерация тестового проекта;

- создание моделей;
- оценка моделей.

Результатом решения первой задачи является выбор одного или более методов моделирования, таких как деревья решений с помощью алгоритма C4.5 или посредством нейронных сетей.

Построив модель, аналитик должен проверить ее качество и правильность. В задачах Data Mining с учителем, таких как классификация, можно просто использовать степень ошибки как качественную меру для модели Data Mining. Поэтому необходимо разделять обучающий набор данных и тестовый набор, построить модель на обучающем наборе, а проверить на тестовом. Для проверки и оценки необходимо кроме тестового набора спроектировать и процедуру тестирования.

После проектирования тестов запускается инструмент моделирования на подготовленном наборе данных для создания одной или более моделей.

*Оценка построенной модели* выполняется в соответствии с ее областью знаний, критерием успеха и тестовым проектом. На данной фазе делается вывод об успехе применения методов Data Mining с технической точки зрения, но результаты не интерпретируются в контексте бизнеса.

**Фаза оценки** — призвана более основательно оценить модель до процесса ее окончательного размещения, чтобы убедиться в достижимости поставленных бизнес-целей. В конце этой фазы руководитель проекта должен решить, как дальше использовать результаты Data Mining. Эта фаза включает следующие задачи:

- оценка результатов;
- пересмотр процесса;
- определение дальнейших действий.

Предыдущая оценка имела дело с такими факторами, как правильность и всеобщность модели. На этой фазе оценивается, в какой степени модель решает бизнес-цели проекта, и определяется, имеются ли какие-нибудь бизнес-причины, по которым эта модель может быть неверной. Кроме того, при решении данной задачи, если позволяет время и бюджет проекта, построенные модели проверяются на реальных данных.

На этой фазе наиболее удобно пересмотреть обязательства Data Mining, чтобы выявить факторы или задачи, которые могли быть не учтены или пропущены. Такой пересмотр гарантирует качество результатов. При этом необходимо ответить на следующие вопросы: корректно ли построена модель, использованы ли только те атрибуты, которые доступны для будущего размещения?

В конце этой фазы руководитель проекта должен решить, заканчивать ли проект и переходить к фазе размещения или инициировать новую итерацию проекта.

**Фаза размещения** — служит для организации знаний и их представления в удобном для пользователя виде. В зависимости от требований фаза размещения может быть как простой (обычная генерация отчетов), так и сложной (процессы Data Mining через все предприятие). На этой фазе решаются следующие задачи:

- планирование размещения;
- планирование наблюдения и сохранения;
- производство конечных отчетов.

Чтобы упорядочить размещение Data Mining результатов в бизнесе, необходимо спланировать развитие результатов и разработку стратегии для размещения.

Наблюдение и размещение важно, если результаты Data Mining используются ежедневно. Тщательная подготовка стратегии сохранения позволит избежать некорректного их использования.

В конце проекта руководитель и его команда должны составить конечный отчет. В зависимости от плана размещения этот отчет может только суммировать опыт разработки проекта или может быть конечным и всеобъемлющим представлением результатов. Этот отчет включает все предыдущие промежуточные и итоговые результаты. Также здесь можно описать выводы по проекту.

В заключение необходимо пересмотреть проект, чтобы оценить все удачи и неудачи, потенциально важные для учета в будущих проектах. Эта задача включает сбор опыта, накопленного в течение работы над проектом, и может заключаться в интервьюировании участников проекта. Этот документ должен включать описание подводных камней, обманчивых подходов или советы для выбора лучших методов Data Mining в подобных ситуациях. В идеале опытная документация также покрывает индивидуальные отчеты, написанные членами проекта в течение фаз и задач проекта.

## 10.4. Стандарт PMML

Стандарт PMML (Predicted Model Markup Language) предназначен для обмена построенными mining-моделями между системами Data Mining. Данный стандарт описывает форму представления моделей в виде XML-документа. PMML. Сейчас опубликована версия 3.0. Можно смело утверждать, что в на-

стоящее время он имеет наибольшее практическое значение из всех стандартов Data Mining. Он активно используется разработчиками, т. к. позволяет системам обмениваться знаниями (моделями) в унифицированном виде.

Рассмотрим более подробно вид представления mining-моделей в соответствии со стандартом PMML 3.0.

Структура моделей описывается с помощью DTD-файла, который называется PMML DTD. Корневым элементом PMML-документа является тег `<PMML>`. Общая структура выглядит следующим образом:

```
<?xml version="1.0"?>
<PMML version="3.0"
  xmlns="http://www.dmg.org/PMML-3_0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

  <Header copyright="Example.com"/>
  <DataDictionary> ... </DataDictionary>

  ... a model ...

</PMML>
```

Тег `DOCTYPE` в PMML-документе не обязателен.

Структура тега `PMML` имеет следующий вид:

```
<xs:element name="PMML">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Header"/>
        <xs:element ref="MiningBuildTask" minOccurs="0"/>
        <xs:element ref="DataDictionary"/>
        <xs:element ref="TransformationDictionary" minOccurs="0"/>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:choice>
            <xs:element ref="AssociationModel"/>
            <xs:element ref="ClusteringModel"/>
            <xs:element ref="GeneralRegressionModel"/>
            <xs:element ref="MiningModel"/>
            <xs:element ref="NaiveBayesModel"/>
            <xs:element ref="NeuralNetwork"/>
            <xs:element ref="RegressionModel"/>
            <xs:element ref="RuleSetModel"/>
            <xs:element ref="SequenceModel"/>
            <xs:element ref="SupportVectorMachineModel"/>
```

```
            <xs:element ref="TextModel"/>
            <xs:element ref="TreeModel"/>
          </xs:choice>
        </xs:sequence>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
```

Первым элементом PMML-документа является заголовок — `Header`. Он содержит информацию о приложении, сформировавшем описываемую модель: название, версию, описание и т. п. Структура элемента имеет следующий вид:

```
<xs:element name="Header">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element minOccurs="0" ref="Application" />
      <xs:element minOccurs="0"maxOccurs="unbounded" ref="Annotation"/>
      <xs:element minOccurs="0" ref="Timestamp" />
    </xs:sequence>
    <xs:attribute name="copyright" type="xs:string" use="required" />
    <xs:attribute name="description" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Элемент `MiningBuildTask` может содержать любые XML-значения, описывающие конфигурацию обучающего запуска, во время которого была построена описываемая в документе модель. Такая информация может быть полезна потребителю, но необязательна. В стандарте PMML 3.0 не описывается структура данного элемента, она имеет произвольный формат, согласованный между получателем и отправителем.

Поля `DataDictionary` и `TransformationDictionary` используют вместе, чтобы идентифицировать уникальные имена. Другие элементы в моделях могут обращаться к этим полям по имени.

Элемент `DataDictionary` содержит описание для полей, используемых в mining-моделях. В нем определяются типы и пределы значений. Структура элемента имеет следующий вид:

```
<xs:element name="DataDictionary">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="DataField" maxOccurs="unbounded"  />
        <xs:element ref="Taxonomy" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="numberOfFields" type="xs:nonNegativeInteger" />
    </xs:complexType>
  </xs:element>
```

В зависимости от операций, которые могут выполняться над описываемыми полями, их типы разделяются на три вида:

- категориальные (categorical) — выполняется операция сравнения на равенство;
- упорядоченные (ordinal) — могут выполняться операции сравнения на больше или меньше;
- непрерывные (continuous) — могут выполняться арифметические операции.

Элемент `isCyclic` принимает значение 1, если поле является циклическим, т. е. расстояние вычисляется как сумма максимального и минимального значения.

Элемент `taxonomy` описывает иерархические поля.

Элемент `TransformationDictionary` используется для описания процесса преобразования данных. Дело в том, что для построения моделей часто используется преобразование пользовательских типов данных в типы, удобные для применения методов Data Mining. Например, нейронные сети обычно работают с числами в пределах от 0 до 1, поэтому числовые входные данные преобразуются к значениям от 0 до 1, а категориальные заменяют серией индикаторов 0/1. В то же время применение метода Naive Bayes требует преобразования всех данных в категориальные типы.

PMML описывает разные виды простых преобразований данных:

- нормализация (normalization) — отображает непрерывные или дискретные значения в числовые;
- дискретизация (discretization) — отображает непрерывные значения в дискретные;
- отображения (value mapping) — отображает одни дискретные значения в другие;
- агрегация (aggregation) — суммирует или собирает группы значений.

Трансформация в PMML не покрывает полный набор функций, которые используются для сбора и подготовки данных для Data Mining. Структура элемента `TransformationDictionary` выглядит следующим образом:

```
<xs:element name="TransformationDictionary">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="DefineFunction"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="DerivedField" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

В PMML-документе могут описываться несколько моделей. Кроме того, список моделей в PMML-документе может быть пуст. Такой документ можно использовать не для передачи моделей, а для других целей, например для передачи первоначальных метаданных до построения модели.

Вторая версия PMML поддерживает следующие типы моделей.

- Ассоциативные правила (AssociationModel) — данная модель представляет правила, где некоторый набор элементов (условная часть правила) ассоциируется с другим набором элементов (заключительная часть). Например, правила могут выражать зависимость частоты покупки некоторого набора продуктов от частоты покупки другого набора продуктов.
- Кластеры (ClusteringModel) — в PMML различают два вида кластерных моделей: основанных на центрах и на расстояниях. Обе модели имеют один и тот же DTD-элемент — `ClusteringModel` как верхний уровень. Для моделей, основанных на центре, кластер описывается вектором координат центра. Для моделей, основанных на расстояниях, кластеры описываются их статистикой.
- Общая регрессионная модель (GeneralRegressionModel) — предназначена для поддержки множества регрессионных моделей.
- Результат метода Naive Bayes (NaiveBayesModel) — данная модель по существу описывает набор матриц. Для каждой независимой переменной описывается матрица, которая содержит частоту его значений относительно значений зависимой переменной.
- Нейронные сети (NeuralNetwork) — нейронные сети имеют один или более входных узлов и один или более нейронов. Некоторые выходы нейронов являются выходами сети. Сеть описывается нейронами и их соединениями (иначе называемыми весами). Все нейроны организуются в уровни, последовательность уровней описывает порядок выполнения вычислений. Модель не описывает процесс эволюции нейронной сети, представляя только конечный результат. Все входящие соединения для некоторого нейрона описываются в элементе `Neuron`. Каждое соединение `Con` хранит

ID начального узла и его вес. Коэффициент смешения веса может быть сохранен как атрибут элемента `Neuron`.

- Регрессия (RegressionModel) — функции регрессии используются, чтобы определить отношение между зависимой переменной и одной или несколькими независимыми переменными. Элемент `RegressionModel` описывает три типа регрессионных моделей: линейную (linear), полиномиальную (polynomial) и логическую (logistic).
- Набор правил (RulesSetModel) — модель включает в себя некоторое количество правил. Каждое правило содержит предикатную часть (условную часть) и значение. Предикатная часть может быть составной, объединенной операцией `and`. Также в него включена некоторая информация, получаемая в процессе обучения.
- Последовательность (SequenceModel) — данная модель состоит из последовательности объектов, идентифицируемых первичным ключом (Primary Key), отдельные результаты которых описываются вторичным ключом (Secondary Key). Каждый результат состоит из набора упорядоченных элементов. Поле "порядок" (Order Field) описывает порядок элементов в результате.
- SVM-модель (SupportVectorMashineModel) — позволяет описать вектор поддержки. Модель включает в себя описание функции ядра.
- Текст (TextModel) — предназначена для определения результатов методов Text Mining. Модель включает в себя шесть главных частей: атрибуты модели, словарь терминов, тело текстового документа, матрицу термов, текстовую модель нормализации и текстовую модель подобия.
- Деревья решений (TreeModel) — модели деревьев решений в PMML позволяют описать классификационные или предсказательные структуры. Каждый узел содержит логическое выражение, которое описывает правило выбора узла или ветви.

Некоторые типы PMML-моделей, например нейронные сети или регрессия, могут быть использованы для разных целей. Одни реализуют числовые предсказания, в то время как другие могут использоваться для классификации. Поэтому PMML описывает четыре разных mining-функции. Для этого каждая модель имеет атрибут `functionName`, который определяет функцию, выполняемую моделью (т. е. задачу Data Mining, для решения которой эта функция может быть использована).

Для уникальной идентификации модели внутри документа (т. к. моделей может быть несколько) используются имена, которые записываются в атрибуте `modelName`. Этот атрибут не обязателен. Пользователи, читающие модель, могут по своему выбору использовать его или нет.

Для описания алгоритма, с помощью которого была порождена модель, используется атрибут `algorithmName`. Он служит только для информативных целей.

В PMML версии 3.0 был также добавлен еще один тип MiningModel. Он позволяет строить модель на основе комбинирования более простых моделей. Возможны два способа комбинирования:

- последовательный — все модели записываются в последовательность, где результаты одной модели являются входной информацией для следующей модели;
- выборный — где из нескольких моделей по определенным правилам выбирается одна главная модель.

В текущей версии стандарта комбинирование возможно для деревьев решений и регрессионной модели. В следующих версиях планируется расширить данный список.

# 10.5. Другие стандарты Data Mining

## 10.5.1. Стандарт SQL/MM

В конце 1991—начале 1992 гг. разработчики систем текстового поиска, действуя под протекцией организации IEEE, реализовали спецификацию языка, названного SFQL (Structured Full-text Query Language). Целью SFQL было описать расширение к языку SQL, которое могло бы быть использовано в полнотекстовых документах.

После опубликования данной спецификации она была подвергнута критике со стороны организаций, занимающихся анализом данных. Наибольшую критику вызывало использование ключевых слов языка SFQL в контексте, отличном от общепринятого.

В конце 1992 г. на конференции в Токио было принято решение избавиться от конфликтов в расширении языка SQL, и одновременно комитет по стандартизации SQL разработал дополнение для объектно-ориентированного SQL. Здесь же был принят стандарт, который описывал библиотеки классов для объектных типов SQL (по одному для каждой категории комплексных данных).

Структурные типы, описанные в подобной библиотеке, были первыми классовыми типами SQL. Предложенный стандарт стал известен как SQL/MM (MM расшифровывалось как мультимедиа — MultiMedia). Предложенные категории данных включали полнотекстовые данные, пространственные данные (spatial), изображения и др.

Подобно SQL, новый стандарт SQL/MM также состоит из нескольких частей. Эти части не зависят друг от друга, за исключением первой части. Она является основой и носит характер руководства по использованию других частей.

Процессу Data Mining посвящена шестая часть данного стандарта SQL/MM Data Mining. Она пытается обеспечить стандартный интерфейс к алгоритмам Data Mining. Они могут представлять как верхний уровень любой объектно-реляционной системы базы данных, так и промежуточный уровень.

Данным стандартом поддерживаются четыре основные модели Data Mining:

- модель правил — позволяет находить шаблоны (правила) в отношениях между различными частями данных;
- кластерная модель — помогает группировать вместе записи данных, которые имеют общие характеристики, и идентифицировать более важные из этих характеристик;
- регрессионная модель — помогает аналитику предсказать значения новых числовых данных на основе известных;
- классификационная модель — подобна регрессионной модели, но ориентируется на предсказание не числовых, а категориальных данных (классов).

Модели поддерживаются посредством новых структурных пользовательских типов. Для каждой модели известен тип DM_*Model, где * заменяется на:

- Clas — для модели классификации;
- Rule — для модели правил;
- Clustering — для кластерной модели;
- Regression — для регрессионной модели.

Эти типы используются для описания модели, которая извлекается из данных. Модели параметризируются с использованием типов DM_*Settings (где * — это Clas, Rule, Clus или Reg). Они позволяют задавать различные параметры моделей (например, глубину деревьев решений).

После того как модель создана и обучена, она должна быть подвергнута процессу тестирования. Для этого выполняется построение экземпляров типа DM_MiningData, которые содержат тестовые данные типа DM_MiningMapping. Эти данные определяют различные колонки в реляционных таблицах, которые используются как исходные данные. Результатом тестирования модели является один или более экземпляров типа DM_*TestResult (где * — это только Clas или Reg). Когда модель запускается на реальных данных, результат будет получен в экземпляре типа DM_*Result (где * — это Clas, Clus или Reg, но не Rule).

В большинстве случаев необходимо также использовать экземпляры типа DM_*Task, чтобы управлять тестированием и запуском моделей.

## 10.5.2. Стандарт Microsoft Data Mining eXtensions (DMX)

Стандарт Microsoft Data Mining eXtensions (DMX) разработан компанией Microsoft. Он, подобно языку SQL/MM, применяет методы Data Mining к реляционным базам данных. Этот стандарт расширяет OLE DB фирмы Microsoft и включен в SQL Server 2005 Analysis Services.

Интерфейс Microsoft Data Mining eXtensions (DMX) предназначен для применения как в качестве интерфейса для исследования данных, так и в качестве средства управления пользовательским интерфейсом (UI). Решение, предложенное Microsoft в SQL Server 2005, позволяет задействовать в качестве расширений этого интерфейса несколько алгоритмов исследования данных. Кроме того, оно включает поддержку мастеров исследования данных, позволяющих пользователям пройти все этапы исследования. Расширение DMX позволяет подключиться к единой взаимосвязанной информационной системе — приложениям OLAP, пользовательским приложениям, системным службам (таким как Commerce Server) и множеству приложений и инструментальных средств производства независимых компаний.

Спецификации OLE DB for OLAP и DMX обеспечивают доступ к службам исследования данных, соответствующим мастерам и приложениям независимых производителей. Помимо спецификации DMX к числу ключевых элементов среды Data Mining относятся модель исследования данных DSO и процессор Data Mining Engine, который включает алгоритмы построения деревьев принятия решений, кластерного анализа, временных рядов и др. Analysis Services и любые процессы компаний, совместимые с DMX, могут подключаться к этой среде Data Mining, чтобы определять и создавать модели исследования данных, а также манипулировать ими. Если для функционирования продуктов независимых производителей необходим интерфейс DMX, то выполняемые этими продуктами функции могут быть опубликованы (экспортированы), чтобы дать возможность применять их всем, кто работает в этом окружении.

На системном уровне специалисты Microsoft расширили модель DSO таким образом, чтобы она поддерживала добавление нового типа объекта — DM-модели. Они создали серверные компоненты, которые обеспечили встроенные возможности применения как методов OLAP, так и DM. Эти возможности и определяют основные особенности Analysis Services. На стороне клиента появились средства, позволяющие клиентским частям OLAP и Data Mining Engine эксплуатировать работающую на сервере службу Analysis Services. Клиентская часть предоставляет полный доступ к обеим моделям, OLAP и DM, через спецификацию DMX.

Наконец, выпустив спецификацию DMX, корпорация Microsoft предоставила независимым компаниям возможность применять интерфейсы COM, входя-

щие в состав DMX, чтобы и для средств исследования данных обеспечить реализацию принципа plug-and-play. Эти возможности позволяют добавлять новые функции исследования данных в те информационные среды, которые удовлетворяют требованиям спецификации DMX. В настоящее время такого рода расширения обеспечивают несколько независимых компаний, выпускающих прикладные системы и инструментальные средства. В основном они входят в альянс производителей хранилищ данных Microsoft Data Warehousing Alliance. Три компании, выпускающие продукты для исследования данных, являются членами альянса: Angoss Software, DBMiner Technology и Megaputer Intelligence.

Специалисты Microsoft спроектировали свою концепцию хранилищ данных Data Warehousing Framework таким образом, чтобы она объединила потребности бизнес-интеллекта и систем поддержки принятия решений. Эта концепция предлагает для них единое основание, обеспечивающее высокий уровень быстродействия, гибкость и невысокую стоимость. Члены альянса Data Warehousing Alliance предлагают свои инструментальные и прикладные программные продукты для решения задач расширения, преобразования и загрузки данных (Data Extension, Transformation and Loading — ETL), проведения аналитических работ, формирования запросов и отчетов, а также для исследования данных.

## 10.5.3. Стандарт Java Data Mining

Целью стандарта Java Data Mining является разработка Java API для разработчиков в области Data Mining. Он объединяет усилие двух групп: JSR 74 и JSR 247.

Целью JSR 74 является стандарт JDMAPI (Java Data Mining API). Он будет представлять собой спецификацию API-функций для построения моделей Data Mining, извлечения знаний, используя эти модели, а также создание, хранение, доступ и сохранение данных и метаданных, поддерживающих результаты Data Mining и выбор трансформации данных.

Предлагается следующая структура пакетов JDMAPI:

- javax.datamining;
- javax.datamining.settings;
- javax.datamining.models;
- javax.datamining.transformations;
- javax.datamining.results.

Одним из требований, предъявляемых к JDMAPI, является совместимость со стандартами OMG CWM, SQL/MM for Data Mining и DMG's PMML, он дол-

жен поддерживать четыре концептуальные области, описанные в CWM: настройки, модели, трансформация и результаты.

К сожалению, работа над данным стандартом в последнее время практически не продвигается. Однако на его основе разработан стандарт Java Data Mining (JDM) группой JSR 247. Последняя версия стандарта 2.0 датируется декабрем 2006 года.

Данный стандарт включает в себя API для задач классификации, регрессии, кластеризации, поиска ассоциативных правил, решения временных рядов, извлечения ключевых понятий и определения аномалий. В нем поддерживаются такие алгоритмы, как деревья решений, нейронные сети, Naive Bayes, Support Vector Machine, K-Means, Apriori, неотрицательное матричное разложение на множители и ARIMA.

Стандарт JDM поддерживает общие операции, необходимые для извлечения знаний: построение, проверка и применение моделей. JDM также поддерживает создание, персистентность, доступность и сохранение метаданных, используемых в операциях Data Mining. JDM 2.0 включает расширение для Text Mining, статистики трансформации интегрированной с процессом извлечения.

JDM определяет следующие пакеты для Data Mining:

- javax.datamining;
- javax.datamining.algorithm.arima;
- javax.datamining.algorithm.feedforwardneuralnet;
- javax.datamining.algorithm.kmeans;
- javax.datamining.algorithm.naivebayes;
- javax.datamining.algorithm.nmf;
- javax.datamining.algorithm.svm;
- javax.datamining.algorithm.svm.classification;
- javax.datamining.algorithm.svm.regression;
- javax.datamining.algorithm.tree;
- javax.datamining.anomalydetection;
- javax.datamining.association;
- javax.datamining.attributeimportance;
- javax.datamining.base;
- javax.datamining.clustering;
- javax.datamining.command;
- javax.datamining.data;

- javax.datamining.featureextraction;
- javax.datamining.modeldetail.arima;
- javax.datamining.modeldetail.feedforwardneuralnet;
- javax.datamining.modeldetail.naivebayes;
- javax.datamining.modeldetail.nmf;
- javax.datamining.modeldetail.svm;
- javax.datamining.modeldetail.tree;
- javax.datamining.resource;
- javax.datamining.rule;
- javax.datamining.statistics;
- javax.datamining.supervised;
- javax.datamining.supervised.classification;
- javax.datamining.supervised.multitarget;
- javax.datamining.supervised.regression;
- javax.datamining.task;
- javax.datamining.task.apply;
- javax.datamining.timeseries;
- javax.datamining.transformations.

Практическая реализация этой спецификации не требует поддерживать все сервисы и интерфейсы, описанные в JDM. Однако JDM предоставляет механизм для развития поддерживаемых интерфейсов и возможностей.

# Выводы

Из материала, изложенного в данной главе, можно сделать следующие выводы.

- Основными стандартами в области Data Mining являются: CWM Data Mining от OMG, CRISP, PMML, OLE DB for Data Mining корпорации Microsoft, SQL/MM, OLE DB for Data Mining и JDM.
- CWM — стандарт, разработанный консорциумом OMG для обмена метаданными между различными программными продуктами и репозиториями, участвующими в создании корпоративных СППР.
- CWM имеет модульную структуру, разбитую на четыре основных уровня: Foundation поддерживает спецификацию базовых структурных элементов; Resource описывает информационные источники; Analysis описывает

средства анализа, включая многомерный анализ и Data Mining; Management описывает особенности функционирования ХД.

- Пакет Data Mining стандарта CWM разделен на три концептуальные области: Model — описание метаданных моделей, получаемых в результате работы методов Data Mining; Settings — описание метаданных настроек процесса построения моделей; Attributes — описание метаданных для атрибутов данных.
- CRISP-DM — непатентованная, документированная и свободно доступная модель, описывающая основные фазы, выполнение которых позволяет организациям получать максимальную выгоду от использования методов Data Mining.
- Стандарт CRISP-DM описывается в терминах иерархической модели процесса. Она состоит из набора задач, описанных на четырех уровнях абстракции (от более общего к более конкретному): фазы, общие задачи, специализированные задачи и примеры процессов.
- CRISP-DM делит жизненный цикл проекта Data Mining на следующие шесть фаз: понимание бизнес-процессов (business understanding), понимание данных (data understanding), подготовка данных (data preparation), моделирование (modeling), оценка (evaluation), размещение (deployment).
- Стандарт PMML предназначен для обмена между системами Data Mining, построенными моделями Data Mining. Данный стандарт описывает форму представления моделей в виде XML-документа.
- Вторая версия PMML поддерживает следующие типы моделей: ассоциативные правила (AssociationModel), деревья решений (TreeModel), кластеры (ClusteringModel), регрессия (RegressionModel), нейронные сети (NeuralNetwork), результат метода Naive Bayes (NaiveBayesModel), последовательность (SequenceModel).
- SQL/MM Data Mining обеспечивает стандартный интерфейс к алгоритмам Data Mining. Он может представлять собой как верхний уровень любой объектно-реляционной системы базы данных, так и промежуточный уровень.
- Стандарт OLE DB для Data Mining разработан компанией Microsoft. Он, подобно языку SQL/MM, применяет методы Data Mining к реляционным базам данных. Этот стандарт расширяет OLE DB корпорации Microsoft и включен в SQL Server 2000 Analysis Services.
- Стандарт JDM представляет собой спецификацию API-функций для построения моделей Data Mining, извлечения знаний, их использование, а также создание, хранение, доступа и сохранение данных и метаданных, поддерживающих результаты Data Mining и выбор трансформации данных.

# ГЛАВА 11

# Библиотека Xelopes

## 11.1. Архитектура библиотеки

Xelopes — свободно распространяемая библиотека, разработанная немецкой компанией Prudsys в тесном сотрудничестве со специалистами российской фирмы ZSoft. Архитектура библиотеки соответствует стандарту MDA (Model Driven Architecture) от консорциума OMG, что позволило реализовать ее на языках Java, C++ и C#. На рис. 11.1 представлены основные уровни MDA, отражающие его идеологию.

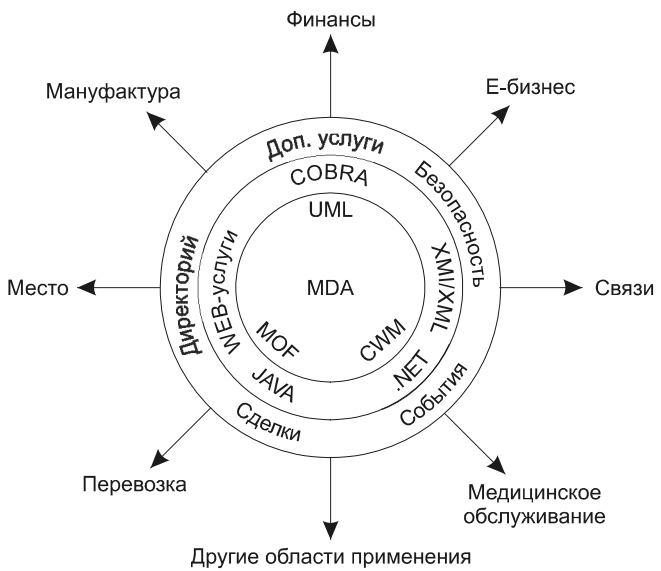



**Рис. 11.1.** Уровни стандарта MDA

Основная идея стандарта MDA заключается в разработке базовой платформонезависимой модели (Platform-Independent Model — PIM), которая отображается в одну или более платформозависимых моделей (Platform-Specific Models — PSM). В данном случае под платформой понимается реализация на конкретном языке программирования. PIM-модели описываются с использованием языка UML, в то время как PSM — это реализации данных моделей на определенном языке (для библиотеки Xelopes существует три реализации на языках Java, C++ и C# — рис. 11.2).

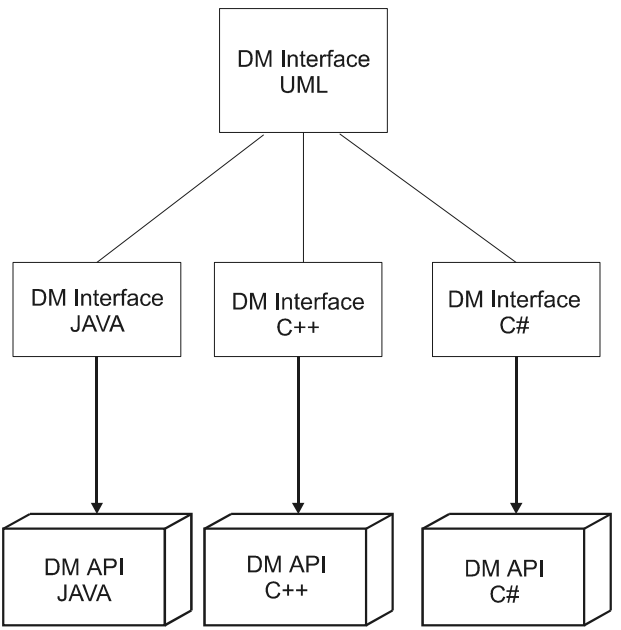



**Рис. 11.2.** Схема реализации ядра библиотеки Xelopes на трех языках

Xelopes соответствует CWM-стандарту, поэтому в основу PIM-библиотеки легли UML-диаграммы пакета Data Mining этого стандарта *(подробно они были описаны в гл. 8)*. В частности, были использованы следующие диаграммы:

- Model-диаграмма, представляющая Data Mining-модель в целом;
- Settings-диаграмма, представляющая настройки процесса Data Mining;
- Attribute-диаграмма, представляющая описание атрибутов Data Mining.

В Xelopes эти диаграммы расширены новыми классами, методами и интерфейсами. Особенно это относится к формированию моделей в формате PMML.

Библиотека Xelopes предназначена для использования во всем процессе Data Mining, поэтому дополнительно были добавлены четыре UML-диаграммы, представляющие следующие концептуальные области:

- DataAccess — диаграмма, описывающая доступ к данным;
- Algoritms-диаграмма, описывающая процесс создания Data Mining-моделей, т. е. алгоритмы Data Mining;
- Automation-диаграмма, описывающая автоматический выбор параметров Data Mining-алгоритмов;
- Transformation-диаграмма, описывающая процесс преобразования данных для моделирования Data Mining.

Перечисленные семь диаграмм представляют собой завершенное UML-описание PIM-ядра Xelopes. Необходимо заметить, что начиная с версии 1.1 все классы из пакетов Model, Settings, Attribute, DataAccess, Algorithms, Transformation и Automation основываются на классах стандарта CWM, следовательно, эта библиотека полностью совместима с данным стандартом, что очень важно для ее интеграции в другие приложения анализа данных.

На основании PIM созданы три основные PSM, реализованные на языках Java, C++ и C#. Не менее важно и то, что в Xelopes были внедрены адаптеры для популярной библиотеки Weka и OLE DB для Data Mining.

Таким образом, можно выделить следующие основные достоинства библиотеки Xelopes:

- свободно распространяемая — библиотека доступна для свободного использования. Кроме того, разработчики могут самостоятельно добавлять в нее новые алгоритмы;
- независимость от платформы — так как базовое ядро сформировано в UML, то могут быть получены реализации практически на любом объектно-ориентированном языке;
- независимость от исходных данных — одной из главных идей Xelopes является абстракция — матрица данных. Это необходимо в связи с тем, что каждый алгоритм Data Mining работает в декартовом пространстве переменных, характеризующих исследуемые данные;
- поддержка всех стандартов Data Mining — библиотека Xelopes поддерживает все основные стандарты в области Data Mining: CWM, PMML, OLE DB for Data Mining, SQL/MM и JDM. Также в библиотеке реализованы адаптеры к популярной библиотеке Weka.

Далее будет описана платформонезависимая модель (PIM) Xelopes версии 1.1. Все классы Xelopes расширяют пакет CWM Core. Это делается либо косвенно, расширением классов из пакетов Data Mining и Transformation, либо прямо, когда классы Xelopes расширяют элементы CWM Core.

# 11.2. Диаграмма Model

## 11.2.1. Классы модели для Xelopes

Как уже отмечалось, классы в библиотеке Xelopes, описывающие модели, основываются на элементах пакета Model из стандарта CWM. В библиотеке они дополняются для решения каждого вида задач. Во-первых, аналогично классу `SupervisedMiningModel` описываются расширения класса `MiningModel` — для других моделей, получаемых в результате методов Data Mining, а именно (рис. 11.3):

- `StatisticsMiningModel` — статистическая модель;
- `AssociationRulesMiningModel` — модель ассоциативных правил;
- `SequentialMiningModel` — модель сиквенциального анализа;
- `CustomerSequentialMiningModel` — расширение `SequentialMiningModel` для сиквенциального анализа рыночных корзин;
- `ClusteringMiningModel` — кластерная модель;
- `ClassificationMiningModel` — расширение `SupervisedMiningModel` для задачи классификации;
- `RegressionMiningModel` — расширение `SupervisedMiningModel` для задачи регрессии;
- `DecisionTreeMiningModel` — расширение `ClassificationMiningModel` для алгоритмов построения деревьев решений;
- `SupportVectorMiningModel` — расширение `RegressionMiningModel` для метода Support Vector Machines;
- `SparseGridsMiningModel` — расширение `RegressionMiningModel` для методов Sparse Grid;
- `TimeSeriesMiningModel` — модель предсказания многомерных временных рядов.

Класс `ClusteringMiningModel` имеет массив объектов типа `Cluster`, которые описывают полученные кластеры. Класс `Cluster` определяется очень абстрактно. Он описывается одним или более центральными векторами и набором векторов из обучающей выборки, которые принадлежат этому кластеру.

Класс `CDBasedClusteringMiningModel` расширяет `ClusteringMiningModel` для кластеров, основанных на центрах и расстояниях, поскольку они используются в том числе и для формирования PMML-моделей. Класс `ClusteringMiningModel` реализует интерфейс `applyModelFunction` для связывания новых векторов с одним из существующих кластеров.

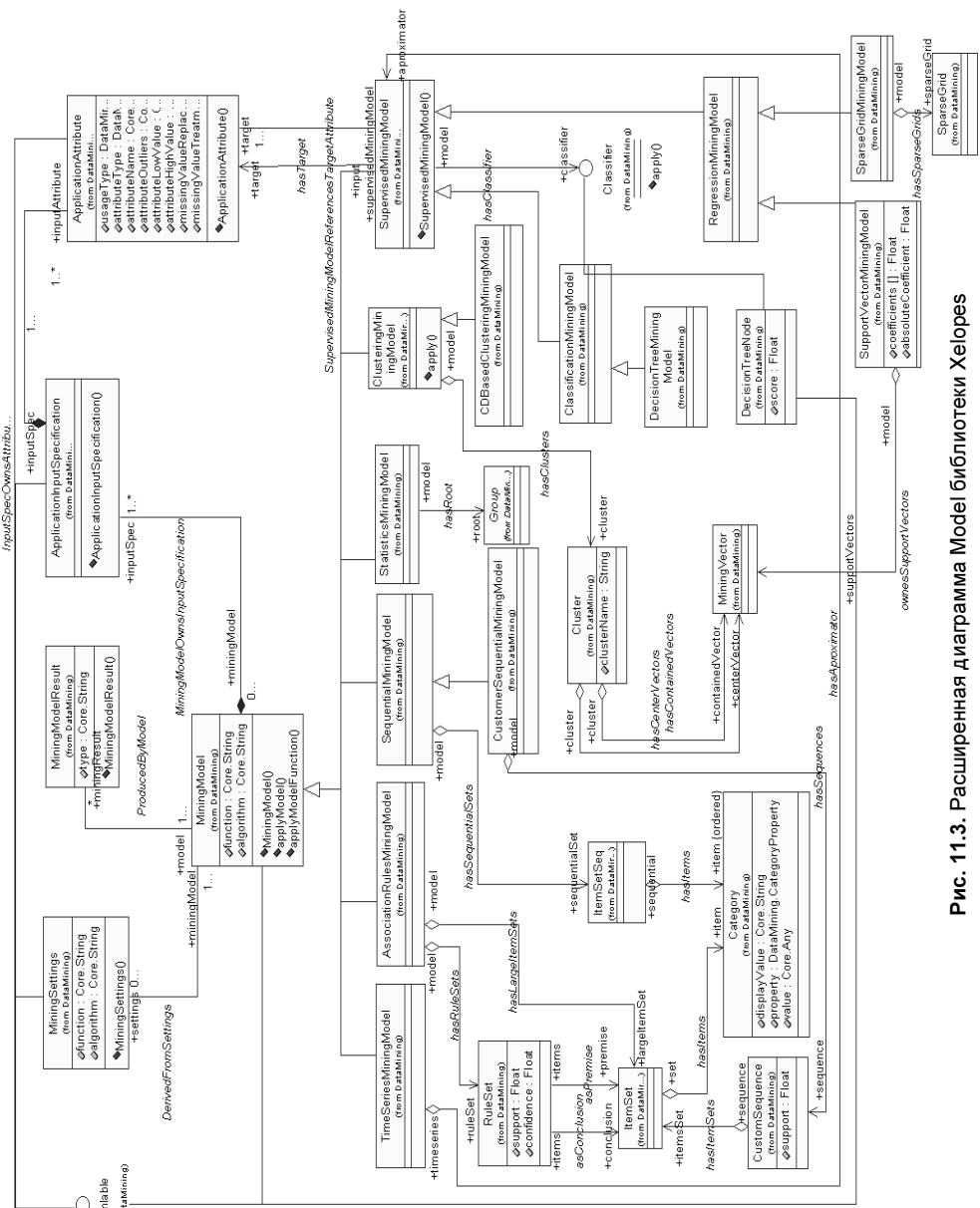



**Рис. 11.3.** Расширенная диаграмма Model библиотеки Xelopes

Класс SupervisedMiningModel представляет собой классификатор и реализует интерфейс Classifier. Этот интерфейс описывает единственный метод apply для классификации векторов.

Класс DecisionTreeMiningModel использует для представления узлов в дереве класс DecisionTreeNode, который расширяет MiningTreeNode, являющийся общим описанием узла дерева. MiningTreeNode определяет одного родителя для каждого узла (тогда как MiningHierarchyNodes допускает множество родителей для узла), описывая таким образом иерархию дерева с одним корнем (т. е. узлом без родителя). Переменная-классификатор DecisionTreeMiningModel имеет тип DecisionTreeNode, который в свою очередь реализует интерфейс Classifier. Необходимо заметить, что DecisionTreeMiningModel может также представлять деревья нелинейных решений, где его узлы DecisionTreeNode могут содержать функцию классификации, построенную методами Support Vector Machine или Sparse Grids. Эти две модели представлены классами SupportVectorMiningModel и SparseGridsMiningModel, которые могут также существовать вне DecisionTreeMiningModel.

Интерфейс Pmmlable служит признаком того, что класс, который его реализует, может быть преобразован в формат PMML (элемент или документ) или, наоборот, построен из документа PMML. Любой класс Pmmlable содержит метод для преобразования в PMML и метод для извлечения из PMML. Поскольку Xelopes полностью совместим со стандартом PMML, класс MiningModel и все его расширения реализуют интерфейс PMML.

Кроме формата PMML модель может быть записана в текстовый файл, а также в HTML-файл. Для этого используются методы writePlainText и toHtmlString соответственно.

## 11.2.2. Методы пакета Model

Как уже отмечалось, в класс MiningModel были добавлены два новых метода — applyModelFunction и applyModel, чтобы применить модель Mining к новым данным. Любая модель должна реализовывать эти два метода или вызывать соответствующие исключения. Первому методу applyModelFunction в качестве аргумента передается новый вектор, и метод возвращает вещественное значение в качестве результата. Это помогает представить данный метод как функцию в *m*-мерном пространстве Mining-атрибутов. Для классификации и регрессии это очевидно, но для кластерных и даже ассоциативных и последовательных моделей это тоже возможно. Тем не менее обычно существуют более удобные способы применения модели. Они реализуются через метод applyModel, который определен в более общем виде. В качестве аргумента он получает объект MiningMatrixElement и возвращает объект MiningMatrixElement как результат. Интерфейс MiningMatrixElement реализо-

ван всеми основными элементами данных Xelopes, такими как `MiningAttribute`, `MiningVector`, `MiningInputStream`, `ItemSet` и `RuleSet`.

## 11.2.3. Преобразование моделей

Одна из главных проблем использования моделей — это ее преобразование. В Xelopes для этих целей используются классы пакета Transformation, который описан далее. Модель поддерживает два типа преобразований: внутреннее и внешнее.

Внешние преобразования производятся перед тем, как сгенерирована модель и записываются в `MiningDataSpecification` (meta data), принадлежащих `MiningSettings`, которые, в свою очередь, связаны с моделью. Когда вызывается метод записи в PMML-формат, проверяются настройки на наличие преобразований. Если они были выполнены, то в тег `DataDictionary` вначале записывается первоначальная спецификация Mining data, а затем в тег `TransformationDictionary` модели PMML записываются преобразования. При считывании модели из формата PMML, наоборот, проверяется наличие тега `TransformationDictionary`, и если он существует, преобразования применяются к метаданным, полученным из тега `DataDictionary`. Выполненные преобразования сохраняются в Mining-настройках модели.

Внутренние преобразования специфичны для каждой модели и применяются в алгоритме, создающем модель (метод `buildModel`). Методы `applyModel` моделей также используют внутренние преобразования перед применением модели к новым данным. Внутренние преобразования хранятся в переменной `MiningTransform` модели и должны реализовывать основной интерфейс `MiningTransformer`. В данной реализации `MiningTransform` содержит процесс Mining-трансформации, который состоит из двух шагов: обработки сильно выделяющихся значений (требуется в методах `applyModel`) и обработки пропущенных значений, которые очень специфичны для методов Data Mining. (Например, деревья решений могут обрабатывать пропущенные значения, тогда как такие методы, как Sparse Grids и Support Vector Machines требуют явной замены значений.) Первый шаг преобразования (`MiningTransformationStep`) включает обработку сильно выделяющихся значений, а второй шаг — обработку пропущенных значений. Следовательно, если переменная `miningTransform` не пуста, то записываются оба преобразования в раздел MiningSchema PMML-документа (который содержит атрибуты для этих преобразований). И наоборот, если при чтении обнаруживается обработка выделяющихся или пропущенных значений в MiningSchema PMML-документе, то создается соответствующий объект `MiningTransformationActivity` и присваивается переменной `miningTransform`.

# 11.3. Диаграмма Settings

## 11.3.1. Классы пакета Settings

Как и в случае с Model, классы пакета Settings библиотеки Xelopes расширяют классы пакета Settings стандарта CWM. Расширенная диаграмма представлена на рис. 11.4. В Xelopes добавлены следующие новые значения для атрибута `function` класса `MiningSettings`:

- `Sequential`;
- `CustomerSequential`;
- `TimeSeriesPrediction`.

Были также добавлены новые значения для атрибута `algorithm`:

- `sequenceAnalysis`;
- `sequentialBasketAnalysis`;
- `supportVectorMachine`;
- `sparseGrids`;
- `nonlinearDecisionTree`;
- `multidimensionalTimeSeriesAlgorithm`.

Были добавлены шесть новых подклассов:

- `SequentialSettings` — настройки для сиквенциального анализа, включают те же атрибуты, что и `AssociationRulesSettings`, но без атрибута конфиденциальности, и новый `itemIndex` с соответствующей привязкой к `MiningAttributes`, определяющий порядок элементов в транзакциях;
- `CustomerSequentialSettings` — настройки для сиквенциального анализа рыночных корзин, атрибуты, как и в `AssociationRulesSettings`, без атрибута конфиденциальности, и новый `customerId` с соответствующей привязкой к `MiningAttributes`, определяющий клиентов транзакций; `transactionPosition` играет роль `transactionId`, но также определяет порядок транзакций для каждого клиента;
- `DecisionTreeSettings` — расширяет `ClassificationSettings` для деревьев решений;
- `SupportVectorSettings` — расширяет `RegressionSettings` для метода Support Vector Machines;
- `SparseGridsSettings` — расширяет `RegressionSettings` для метода Sparse Grid;
- `TimeSeriesMiningSettings` — настройки для предсказания временной серии, атрибуты `embeddingDimension` для вложенного измерения, `stepSize` для

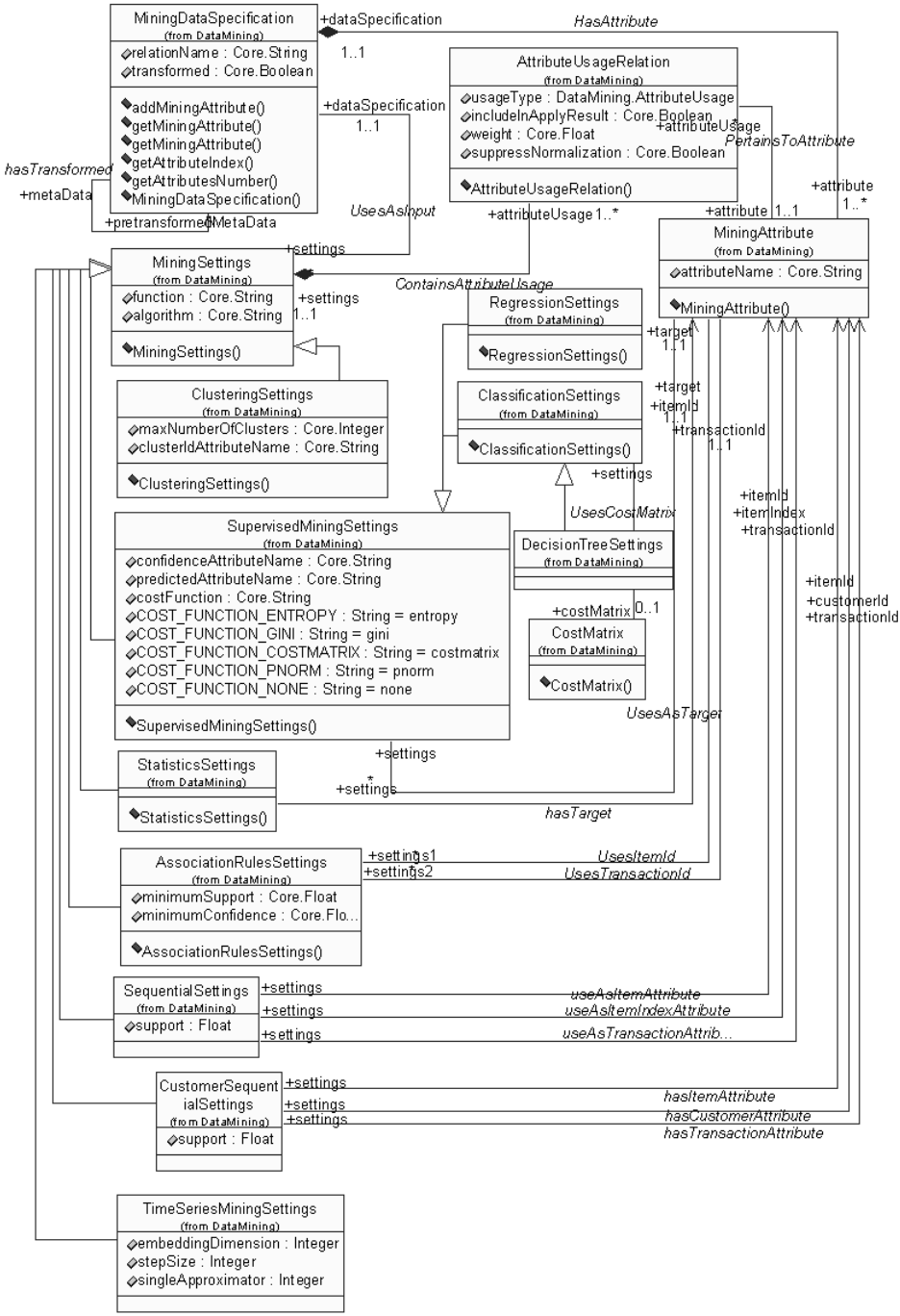



**Рис. 11.4.** Расширенная диаграмма пакета Settings библиотеки Xelopes

размера временного шага, singleApproximator для использования модели одиночной или множественной регрессии для предсказаний.

### 11.3.2. Методы пакета Settings

Класс MiningSettings и его подклассы содержат метод verifySettings, который проверяет настройки с точки зрения полноты и корректности. Кроме того, он содержит ссылку на MiningModel, которая была построена с использованием этих настроек.

Класс MiningDataSpecification содержит метаинформацию матрицы данных, которая главным образом является массивом Mining-атрибутов. Он реализует интерфейс Pmmlable для представления словаря данных в PMML-формате. В него также включены некоторые методы для работы с атрибутами Data Mining.

Как уже отмечалось, в классе MiningDataSpecification отслеживаются внешние преобразования, выполняемые с моделью. Если преобразования были применены, то MiningDataSpecification содержит описание всех этих преобразований в переменной miningTransformationActivity, а оригинал MiningDataSpecification (до преобразований) — в переменной pretransfomed-MetaData. На факт совершения преобразований указывает булева переменная transformed.

С тех пор как MiningDataSpecification содержит метаинформацию матрицы данных, на него обычно ссылаются просто как к метаданным. Это один из самых важных классов в библиотеке Xelopes.

## 11.4. Диаграмма Attribute

### 11.4.1. Классы пакета Attribute

Диаграмма Attribute в Xelopes (рис. 11.5) также расширяет классы одноименного пакета стандарта CWM. Во-первых, она реализует интерфейс Pmmlable для представления атрибутов в формате PMML. Для CategoricalAttributes замена Key <=> Category (ключ <=> категория) используется для отображения категорий как вещественных (или целочисленных) значений, и наоборот. Метод getKey возвращает внутреннее целочисленное значение, используемое для данной категории (Category), или его имя, в то время как getCategory возвращает категорию (Category) ключа. Атрибуты категорий XELOPES (начиная с версии 1.1) содержат булеву переменную unboundedCategories, которая может быть использована для отображения того, что число категорий не фиксируется изначально. Для NumericAttribute был также введен тип времени, который является важным.

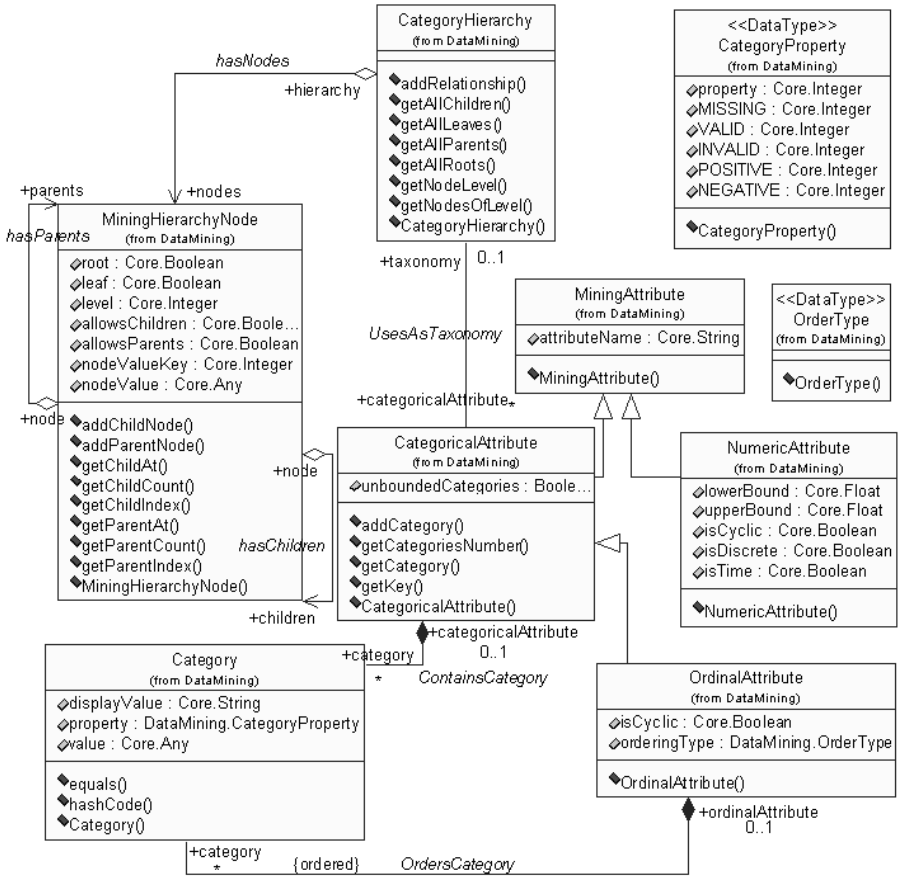



**Рис. 11.5.** Расширенная диаграмма Attribute Xelopes

## 11.4.2. Иерархические атрибуты

Класс `CategoryHierarchy` позволяет определять иерархию для атрибутов категорий. Он хранит иерархические отношения атрибутов, используя узлы `MiningHierarchyNode` — общее описание узла иерархии. Класс `MiningTaxonomyNode` допускает множество родителей для каждого узла (в отличие от `MiningTreeNode` с единственным родителем для каждого узла), определяя таким образом DAG (Directed Acyclic Graph — Ориентированный Нециклический Граф), который может иметь несколько корней.

Необходимо обратить внимание на следующее. Во-первых, из названия понятно, что любой DAG должен быть без циклов, т. е. ни один родитель узла не может быть сыном его сына. В противном случае никакой иерархии не будет существовать. Во-вторых, для некоторых DAG можно определить уров-

ни. Такие графы называются многоуровневыми DAG (level-based DAGs). Другими словами, DAG является многоуровневым, если для любого узла любой действительный граф до корня имеет одну и ту же длину. Это иллюстрирует рис. 11.6. В примере в правой части рисунка представление уровней, очевидно, не имеет смысла.

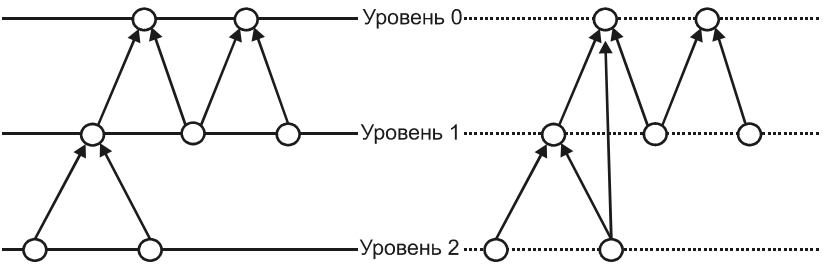



**Рис. 11.6.** Примеры многоуровневых (слева) и немногоуровневых (справа) DAG

Метод `addRelationship` класса `CategoryHierarchy` добавляет ребро (родительская категория — дочерняя категория) в DAG. Для любого заданного узла DAG методы `getAllParents` и `getAllChilds` возвращают все его родительские и дочерние категории соответственно. Методы `getAllRoots` и `getAllLeafs` возвращают все категории корня и листьев. И наконец, включены некоторые методы для обработки уровней (если DAG является многоуровневым), например, `getNodesOfLevel` возвращает все категории заданного уровня. Этот небольшой набор методов позволяет выполнять простую и гибкую обработку иерархических категориальных атрибутов.

# 11.5. Диаграмма Algorithms

Диаграмма Algorithms содержит все классы, которые необходимы для выполнения алгоритмов Data Mining.

## 11.5.1. Общая концепция

По существу, взаимодействие с любым алгоритмом Data Mining описывается четырьмя факторами:

- *Input (ввод)* — матрица данных, используемая алгоритмом Data Mining;
- *Output (вывод)* — Mining-модель, создаваемая алгоритмом Data Mining;
- *Settings (настройки)* — настройки алгоритма Data Mining;
- *Callback (обратный вызов)* — обработка событий алгоритма Data Mining.

Отношения между ними и соответствующие им классы показаны на рис. 11.7.

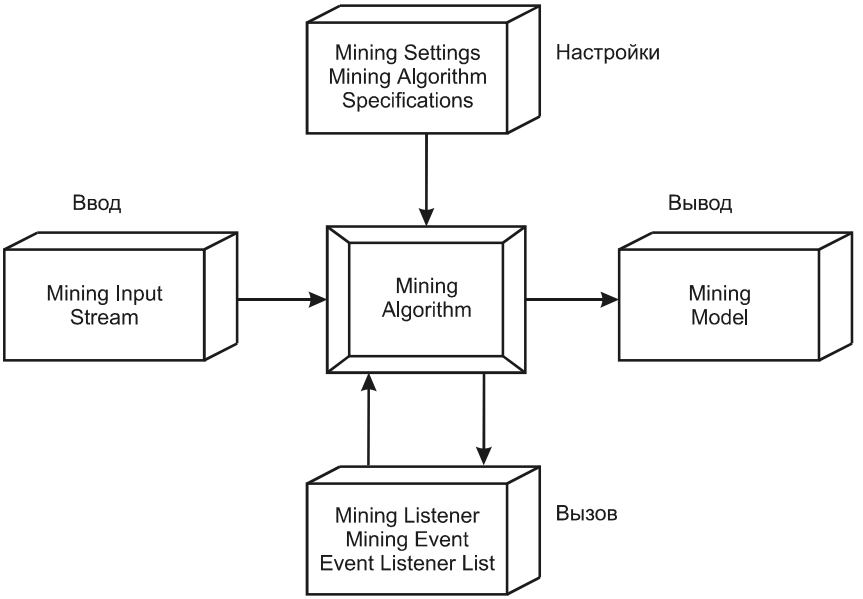



**Рис. 11.7.** Главные интерфейсы алгоритма Data Mining в Xelopes

## 11.5.2. Класс *MiningAlgorithm*

Центральным классом является `MiningAlgorithm`, который описывает Mining-алгоритм сам по себе. Алгоритм запускается с использованием защищенного (protected) метода `runAlgorithm`. С помощью метода `buildModel`, который внутри себя также запускает алгоритм, может быть создан объект класса `MiningModel` для представления и возвращения модели. Метод `buildModelWithAutomation` генерирует Mining-модель, используя средства автоматической настройки параметров, что позволяет строить Mining-модели полностью автоматически. Структура автоматизации будет описана далее. Метод `buildModelWithAutomation`, в свою очередь, запускает метод `buildModel` внутри себя (обычно несколько раз).

Класс `MiningAlgorithm` получает доступ к данным из `MiningInputStream` и возвращает результат как `MiningModel`. Основные настройки Mining берутся из `MiningSettings`, в то время как специфические настройки алгоритмов берутся из `MiningAlgorithmSpecification`. Используя подход "слушателя событий" (event listener), сходный с Java Swing, обработку событий можно реализовать очень гибко (далее эта концепция будет описана более детально). Вся эта информация хранится в переменных `miningInputStream`, `miningModel`, `miningSettings`, `miningAlgorithmSpecification` и `listenerList`.

Класс MiningSettings был описан ранее. Он содержит, например, метаинформацию о Mining-данных (используя класс MiningDataSpecification), которая доступна из MiningAlgorithm через переменную metaData. Метаинформация о применении модели (класс ApplicationInputSpecification) может быть получена через MiningDataSpecification, она содержится в MiningAlgorithm в переменной applicationInputSpecification. Еще раз остановимся на разнице между MiningDataSpecification и ApplicationInputSpecification. Каждая Mining-модель содержит ApplicationInputSpecification, которая перечисляет атрибуты (более конкретно, ApplicationAttributes), используемые в этой модели. Это подмножество атрибутов, как определено в MiningDataSpecification. В то время как ApplicationInputSpecification содержит информацию, специфичную для определенной модели, MiningDataSpecification содержит определения данных, которые не меняются от модели к модели. Основное назначение ApplicationInputSpecification в том, чтобы перечислять атрибуты, которые пользователь должен предоставить для построения модели.

## 11.5.3. Расширение класса *MiningAlgorithm*

По аналогии с диаграммами модели и настроек, MiningAlgorithm является суперклассом для основных типов алгоритмов Data Mining (рис. 11.8):

- StatisticsAlgorithm — статистические методы;
- AssociationRulesAlgorithm — алгоритмы построения ассоциативных правил;
- SequentialAlgorithm — алгоритмы сиквенциального анализа;
- CustomerSequentialAlgorithm — алгоритмы сиквенциального анализа рыночных корзин;
- ClusteringAlgorithm — кластерные алгоритмы;
- SupervisedMiningAlgorithms — supervised Mining-алгоритмы;
- RegressionAlgorithm — расширение SupervisedMiningAlgorithm для регрессии;
- DecisionTreeMiningAlgorithm — расширение ClassificationMiningAlgorithm для алгоритмов построения деревьев решений;
- SupportVectorAlgorithm — расширение RegressionAlgorithm для алгоритмов SVM;
- SparseGridsAlgorithm — расширение RegressionAlgorithm для алгоритмов SG;
- TimeSeriesMiningAlgorithm — алгоритмы предсказания временных серий.

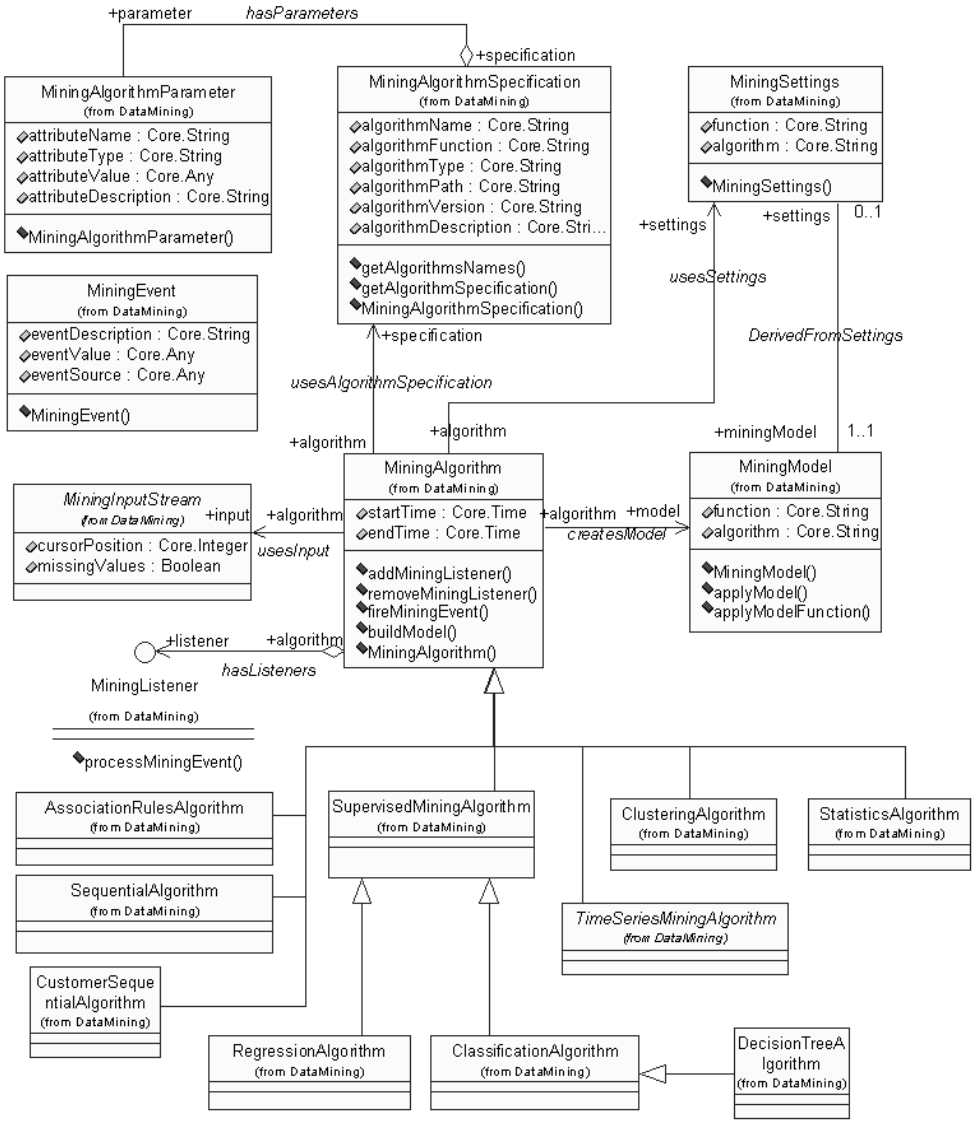



**Рис. 11.8.** Диаграмма Algorithms Xelopes

Эти классы включают специфические методы для получения результатов алгоритма. Например, `AssociationRulesAlgorithms` содержит методы `getAssociationRules` и `getLargeItemSets` для получения ассоциативных правил и частых наборов элементов в определенных форматах. Еще одним примером является метод `getClassifier` в `SupervisedMiningAlgorithm`, который возвращает классификатор как класс `Classifier`.

### 11.5.4. Дополнительные классы

Класс `MiningSettings` и его подклассы содержат основные параметры алгоритмов, т. е. те параметры, которые требуются для всех алгоритмов, решающих одни и те же задачи (для одинаковых значений атрибута `function`). Например, `AssociationRulesSettings` содержит параметры минимальной поддержки и доверия, которые требуются всем алгоритмам, выполняющим построение ассоциативных правил.

Кроме основных параметров для каждого алгоритма могут потребоваться специфичные настройки. Они описываются в классе `MiningAlgorithmSpecification`. Кроме них в данном классе описывается задача, для решения которой будет построена модель (поле `algoritmFunction`) и тип (`algoritmType`), имя (`algoritmName`), путь к классу (`algoritmPath`), реализующему данный алгоритм (например, в случае реализации на Java это имя класса с указанием пакета, в котором он находится), и версия (`algoritmVersion`).

Специфичные для алгоритма настройки описываются в классе `MiningAlgorithmParameter`, на который ссылается переменная `parameter`. Каждая такая настройка имеет имя, тип, значение и описание. Все настройки для алгоритмов описываются в конфигурационном файле algorithms.xml.

### 11.5.5. Слушатели

Как уже упоминалось, механизм обработки событий в библиотеке Xelopes использует концепцию слушателей, подобную Java Swing.

Любой алгоритм для обработки обратных вызовов должен реализовывать интерфейс `MiningListener`, включающий только один метод `processMiningEvent`, который вызывается, когда происходит Mining-событие. С другой стороны, алгоритмы являются подклассами класса `MiningAlgorithm`, который внутри содержит список слушателей, являющихся объектами класса `EventListenerList`. Используя методы `addMiningListener` и `removeMiningListener`, можно добавлять или удалять слушателя `MiningListeners`. Метод `fireMiningEvent` уведомляет всех добавленных слушателей и вызывает `processMiningEvent` для выполнения соответствующей реакции.

## 11.6. Диаграмма DataAccess

Диаграмма DataAccess (рис. 11.9) содержит все классы, необходимые для доступа к данным из алгоритмов.

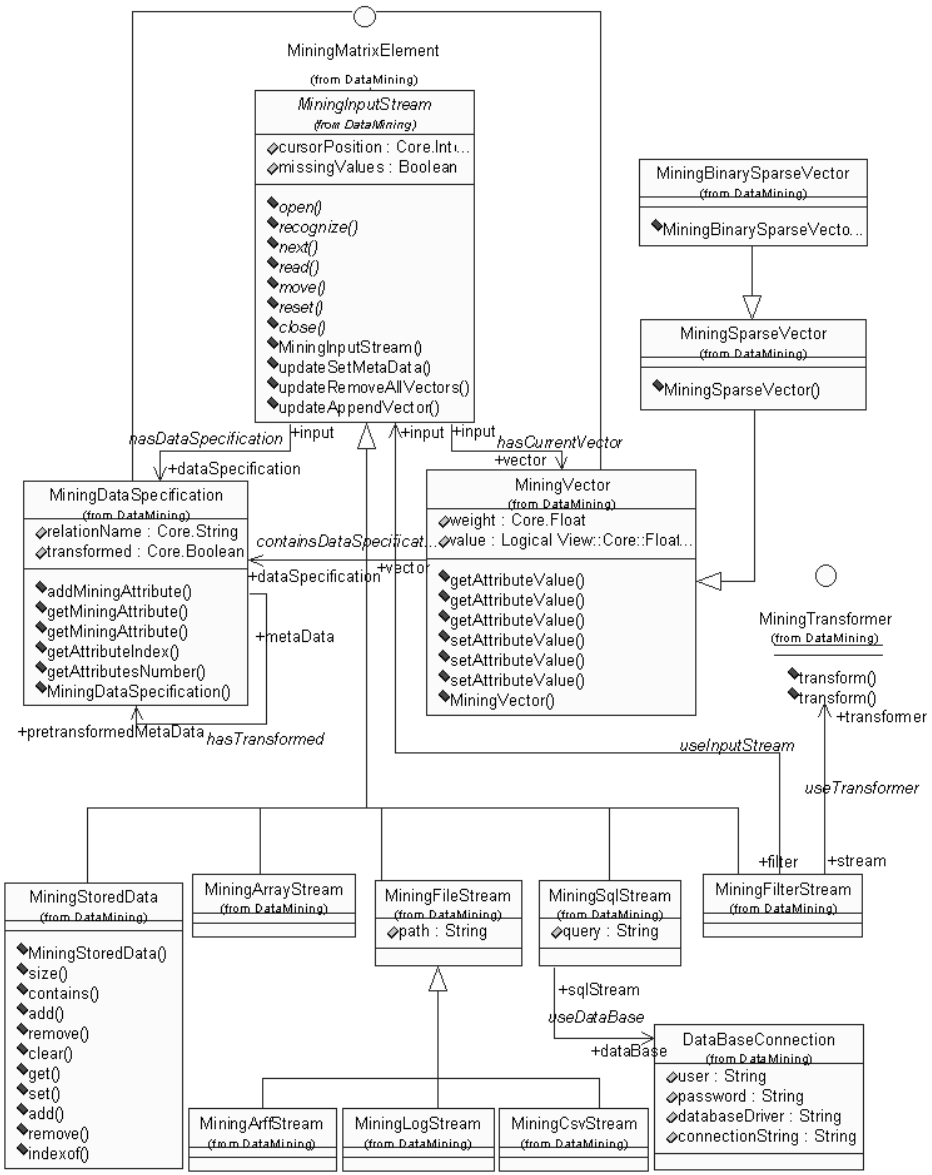



**Рис. 11.9.** Диаграмма классов пакета DataAccess

## 11.6.1. Общая концепция

Необходимость реализации собственного пакета доступа к данным связана с интерпретацией алгоритмами Data Mining входных данных как единой матрицы данных (пусть даже и сильно разреженной). Они "не умеют" работать

с несколькими реляционными таблицами или с текстовыми файлами. В связи с этим невозможно использовать стандартные классы, описанные в CWM, для работы хранилищ данных.

## 11.6.2. Класс *MiningInputStream*

Матрица данных, используемая алгоритмами Data Mining, моделируется абстрактным классом `MiningInputStream`, содержащим гибкие механизмы доступа к данным. Поток открывается вызовом метода `open` и закрывается методом `close`. Класс `MiningInputStream` содержит метаинформацию об атрибутах данных в переменной `metaData`. При создании объекта входного Mining-потока метаданные могут или явно передаваться через конструктор, или могут быть добыты с помощью метода `recognize` (если он реализован) входного потока.

Кроме того, класс `MiningInputStream` содержит градуированный спектр методов доступа к данным, зависящих от их реализации. В большинстве простейших случаев матрица данных может быть пройдена только с использованием курсора методом `next`. Если поддерживается метод `reset`, то курсор может быть установлен в начальное положение. Такой тип доступа поддерживается файлами и базами данных. В более удобных случаях курсор может быть перемещен произвольно с использованием метода `move`. Более удобен прямой доступ к массиву или матрице данных, если матрица размещается в памяти (например, класс `MiningStoredData`). Метод `read` возвращает вектор данных как `MiningVector`. Начиная с версии 1.1 существуют также обновленные методы для манипуляции данными из входного Mining-потока. Входной Mining-поток поддерживает такие методы обновления, называемые Updateble-потоки.

## 11.6.3. Классы Mining-векторов

Класс `MiningVector` и его подклассы позволяют удобно обрабатывать Mining-векторы. Такой вектор содержит переменную класса `MiningDataSpecification`, описывающую метаданные, и массив `value`, содержащий значения вектора. Посредством методов `get` и `set` можно получить эти значения, а также их изменить. Если вектор разряжен, т. е. были сохранены только ненулевые элементы с их индексами колонок, то должен быть использован класс `MiningSparseVector`, который расширяет `MiningVector`. Если вектор содержит только одноэлементные ненулевые значения, может быть использован класс `MiningBinarySparseVector`, который расширяет класс `MiningSparseVector`.

## 11.6.4. Классы, расширяющие класс *MiningInputStream*

На практике матрица может быть реализована разными способами. В простейшем случае все данные и их метаданные могут размещаться в памяти.

Такой подход может быть реализован посредством классов `MiningArrayStream` и `MiningStoredData`, расширяющих класс `MiningInputStream`. Оба потока могут принимать в качестве параметров конструктора любые Mining-потоки, которые должны быть размещены в памяти. Основное достоинство такого способа представления данных — возможность доступа к любой ячейке матрицы.

Класс `MiningFileStream` также расширяет класс `MiningInputStream` и предоставляет возможность обработки данных из файла. Он имеет метод `reset`, который позволяет "переоткрывать" файловый поток. С помощью метода `move` можно получить последовательный доступ к данным из файла. От класса `MiningFileStream` наследуется класс `MiningArffStream`, предоставляющий доступ к данным из файла, записанным в формате ARFF. От класса `MiningFileStream` наследуются также классы `MiningCsvStream` и `MiningLogStream`. Первый из них позволяет читать файлы, использующие в качестве разделителя запятые. Второй читает данные из протоколов работы Web-серверов формата CSV.

Для чтения данных из баз данных должен быть использован класс `MiningSqlStream`. Класс `MiningSqlSource` позволяет определить спецификацию базы данных, включая имя, адрес и т. п.

# 11.7. Диаграмма Transformation

Пакет Transformation библиотеки Xelopes содержит классы, преобразующие исходные данные в соответствии с требованиями применяемых к ним алгоритмов. К таким преобразованиям относятся замена числовых значений на категориальные и, наоборот, обработка пропущенных значений и т. п. Пакет является достаточно сложным, однако предоставляет богатые возможности по преобразованию. В простейших случаях нет необходимости в использовании этих классов, поэтому в данной главе остановимся только на общей концепции (для получения более подробной информации можно обратиться к документации по библиотеке Xelopes).

Прежде всего необходимо заметить, что пакет Transformation присутствует и в стандарте CWM (рис. 11.10). Он также описывает классы, необходимые для выполнения преобразований.

Классы из пакета Transformation в библиотеке Xelopes наследуются не напрямую от элементов стандарта CWM, а, реализуя стратегию пошагового преобразования, представленную на рис. 11.11, используют их.

Стратегия пошагового преобразования предполагает, что любое преобразование исходных данных можно представить как композицию $n$ последовательных шагов $t_1$, $t_2$, ..., $t_n$. На первом шаге преобразования выполняются

над исходными данными, включая и их метаданные. Преобразование метаданных должно выполняться на любом шаге. На каждом шаге решается конкретная задача по преобразованию данных.

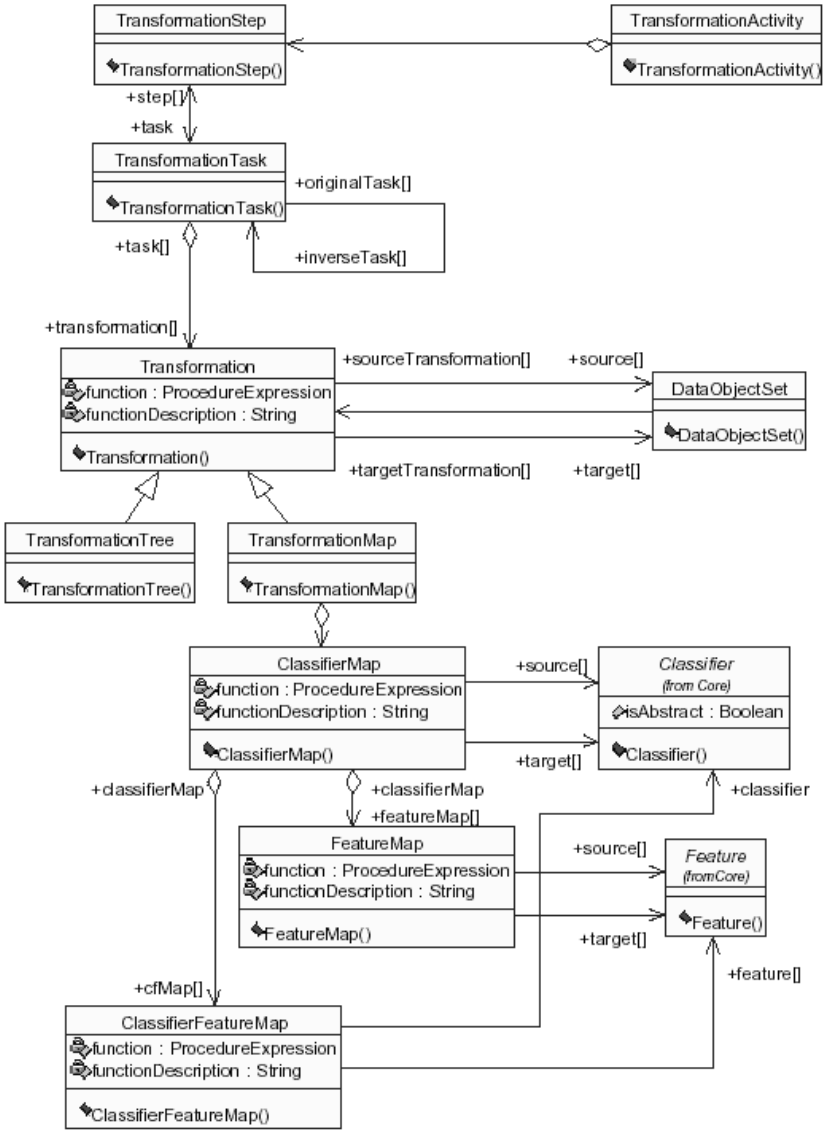



**Рис. 11.10.** Диаграмма классов пакета Transformation стандарта CWM

В пакете Transformation (рис. 11.12) библиотеки Xelopes для представления шагов и задач преобразования существуют классы `MiningTransformationStep`

и `MiningTransformationTask`. Все шаги, выполняемые по преобразованию, объединяются в классе `TransformationActivity`, представляющем собой всю деятельность, связанную с преобразованием.

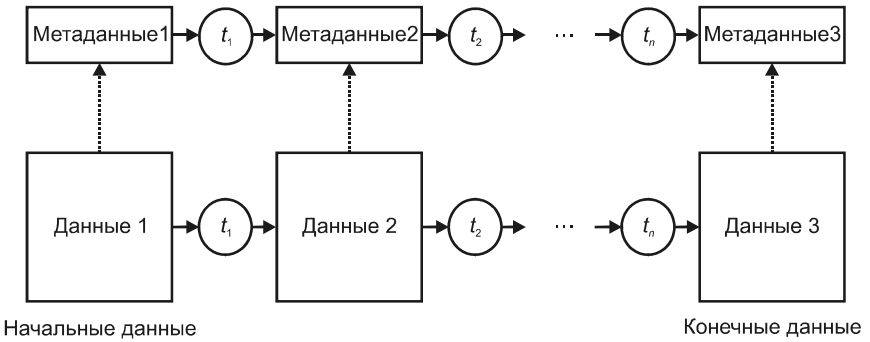



**Рис. 11.11.** Концепция пошагового преобразования данных

# 11.8. Примеры использования библиотеки Xelopes

## 11.8.1. Общая концепция

Основной подход решения задач Data Mining с помощью библиотеки Xelopes не зависит от вида или используемого метода и включает в себя выполнение следующих шагов:

1. Загрузка исходных данных.
2. Настройка процесса построения моделей:
   - общих для модели;
   - специфичных для алгоритма.
3. Инициализация алгоритма.
4. Построение модели.
5. Применение построенной модели для задач с учителем.

Из перечисленных шагов только второй и третий зависят от решаемой задачи и используемого алгоритма. Остальные шаги практически остаются без изменений. Рассмотрим их более подробно.

Для загрузки исходных данных создается экземпляр класса исходных данных — `MiningInputStream`, например, из файла формата ARFF.

```
MiningInputStream inputData = new MiningArffStream("data.arff");
```

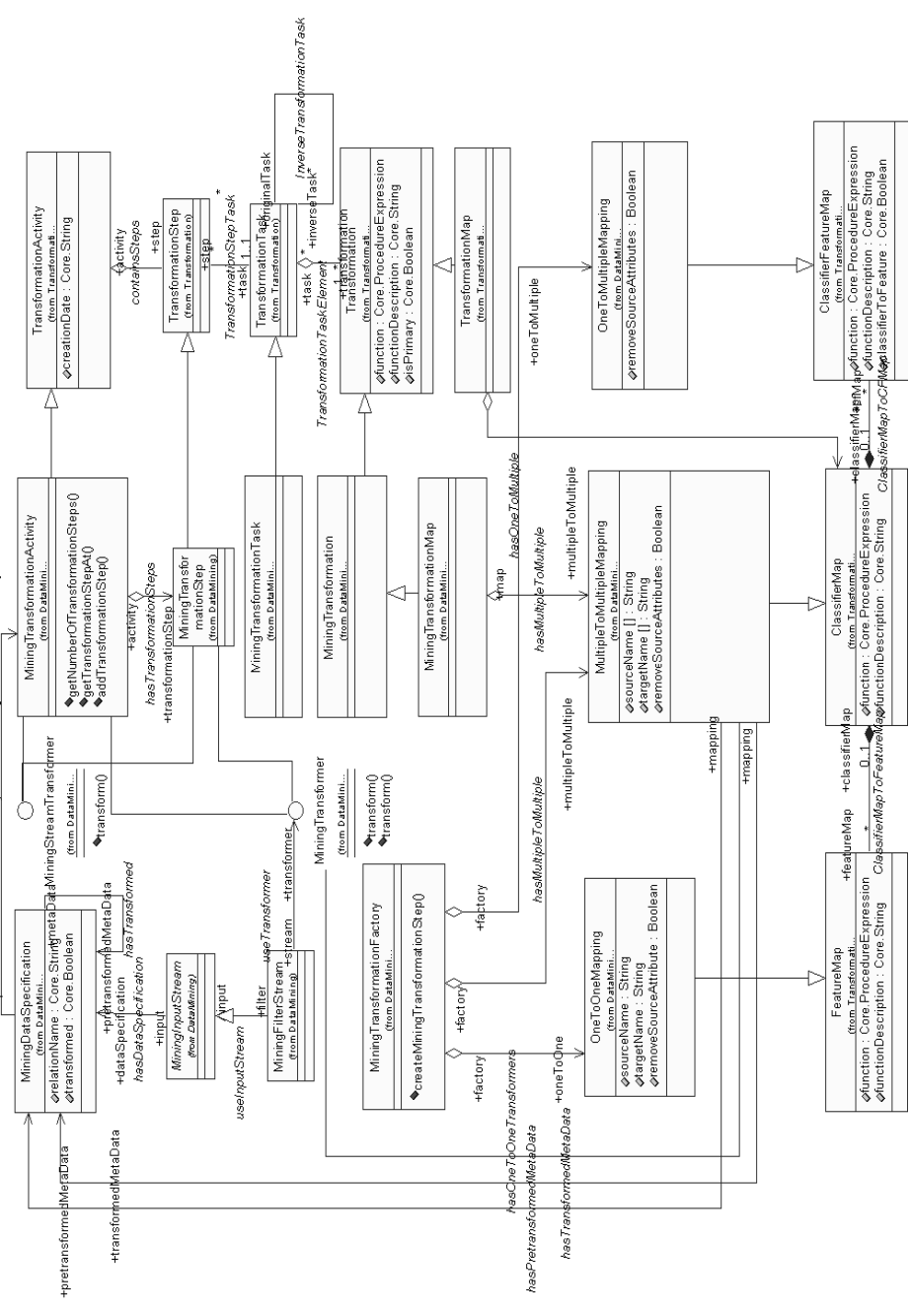



**Рис. 11.12.** Диаграмма классов пакета Transformation библиотеки Xelopes

Затем выделяются метаданные загруженных данных.

```
MiningDataSpecification metaData = inputData.getMetaData();
```

Для настройки процесса построения модели создаются экземпляры классов `MiningSettings` и `MiningAlgorithmSpecification`. У данных экземпляров устанавливаются необходимые параметры. Создание конкретных экземпляров классов зависит от используемого метода и решаемой задачи.

Сделанные настройки должны быть проверены с помощью метода: `verifySettings()`:

```
miningSettings.verifySettings();
```

Необходимо заметить, что настройки, специфичные для алгоритма, могут быть выполнены как непосредственно в коде, так и загружены из конфигурационного файла algorithms.xml, обозначенные по имени алгоритма.

> ***Внимание!***
> Создание экземпляра класса `MiningAlgorithm` выполняется динамической загрузкой класса в соответствии с указанным в специфических параметрах именем класса.

```
String className = miningAlgorithmSpecification.getClassname();
Class algorithmClass = = Class.forName( className);
Object alg = algorithmClass.newInstance();
MiningAlgorithm algorithm = (MiningAlgorithm)alg;
```

Созданному экземпляру алгоритма передаются исходные данные и настройки.

```
algorithm.setMiningInputStream( inputData);
algorithm.setMiningSettings( miningSettings);
algorithm.setMiningAlgorithmSpecification(
miningAlgorithmSpecification);
```

Создание модели выполняется вызовом метода `buildModel()`. Он возвращает модель, построенную в виде экземпляра класса `MiningModel`.

```
MiningModel model = algorithm.buildModel();
```

Любая построенная модель может быть сохранена в формате PMML.

```
FileWriter writer = new FileWriter("example.xml");
model.writePmml(writer);
```

Или в текстовом формате.

```
writer = new FileWriter("example.txt");
model.writePlainText(writer);
```

Если построенная модель является экземпляром класса `SupervisedMiningModel`, то она может быть применена к новым данным с целью определения значения зависимой переменной.

```
MiningVector vector = inputData.read();
double predicted = model.applyModelFunction(vector);
```

Далее рассмотрим примеры решения задач поиска ассоциативных правил, кластеризации и классификации.

## 11.8.2. Решение задачи поиска ассоциативных правил

Решим задачу поиска ассоциативных правил для примера, рассмотренного в *гл. 6*. Данные, представленные в файле transact.arff формата ARFF, имеют следующий вид:

```
@relation 'transact'

@attribute transactId { 0, 1, 2, 3 }
@attribute itemId { 0, 1, 2, 3, 4, 5 }
@attribute itemName { Chewing-gum, Craker, Coke, Water, Beer, Nut }
@attribute itemPrice real

@data
0 1 Craker 12.00
0 3 Water 4.00
0 4 Beer 14.00
1 2 Coke 10.00
1 3 Water 4.00
1 5 Nut 15.00
2 5 Nut 15.00
2 2 Coke 10.00
2 1 Craker 12.00
2 2 Coke 10.00
2 3 Water 4.00
3 2 Coke 10.00
3 5 Nut 15.00
3 2 Coke 10.00
```

Код, реализующий поиск частых наборов и построение ассоциативных правил, приведен далее с подробными комментариями.

```
// Открыть файл "transact.arff" и загрузить из него данные
MiningArffStream arff = new MiningArffStream(
"data/arff/transact.arff");
// и их метаданные
MiningDataSpecification metaData = inputData.getMetaData();
```

```
// Получить атрибут, идентифицирующий транзакции
CategoricalAttribute transactId = (CategoricalAttribute)
metaData.getMiningAttribute( "transactId");

// Получить атрибут, идентифицирующий элементы
CategoricalAttribute itemId = (CategoricalAttribute)
metaData.getMiningAttribute( "itemId");

// Создать экземпляр для выполнения настроек построения модели
// ассоциативных правил
AssociationRulesSettings miningSettings = new AssociationRulesSettings();
// Передать в настройки метаданные
miningSettings.setDataSpecification( metaData);
// Передать в настройки атрибут, идентифицирующий транзакции
miningSettings.setTransactionId( transactId);
// Передать в настройки атрибут, идентифицирующий элементы
miningSettings.setItemId( itemId);
// Установить минимальную поддержку
miningSettings.setMinimumSupport( 0.5);
// Установить минимальную степень доверия
miningSettings.setMinimumConfidence( 0.30);

// Создать экземпляр для выполнения настроек, специфичных для
// алгоритма Apriori, и загрузить их из конфигурационного файла
// algorithms.xml по имени 'AprioriSimple'
MiningAlgorithmSpecification miningAlgorithmSpecification =
MiningAlgorithmSpecification.getMiningAlgorithmSpecification(
AprioriSimple");
// Получить имя класса алгоритма
String className = miningAlgorithmSpecification.getClassname();

// Создать экземпляр алгоритма
AssociationRulesAlgorithm algorithm = (AssociationRulesAlgorithm)
initAlgorithm(className);

// Передать алгоритму исходные данные и настройки
algorithm.setMiningInputStream( inputData);
algorithm.setMiningSettings( miningSettings);
algorithm.setMiningAlgorithmSpecification(
miningAlgorithmSpecification);

// Построить модель
MiningModel model = algorithm.buildModel();

// Вывести полученный результат
showRules((AssociationRulesMiningModel) model);
```

В результате работы приведенного кода будут получены следующие частые наборы в соответствии с выполненными настройками:

```
0: 1 Supp = 50.0
1: 1 3 Supp = 50.0
2: 2 Supp = 75.0
3: 2 3 Supp = 50.0
4: 2 3 5 Supp = 50.0
5: 2 5 Supp = 75.0
6: 3 Supp = 75.0
7: 3 5 Supp = 50.0
8: 5 Supp = 75.0
```

А также следующие, построенные для них ассоциативные правила:

```
0: 3 => 1 Supp = 50.0, Conf = 66.67
1: 1 => 3 Supp = 50.0, Conf = 100.0
2: 3 => 2 Supp = 50.0, Conf = 66.67
3: 2 => 3 Supp = 50.0, Conf = 66.67
4: 3 5 => 2 Supp = 50.0, Conf = 100.0
5: 5 => 2 3 Supp = 50.0, Conf = 66.67
6: 3 => 2 5 Supp = 50.0, Conf = 66.67
7: 2 5 => 3 Supp = 50.0, Conf = 66.67
8: 2 => 3 5 Supp = 50.0, Conf = 66.67
9: 2 3 => 5 Supp = 50.0, Conf = 100.0
10: 5 => 2 Supp = 75.0, Conf = 100.0
11: 2 => 5 Supp = 75.0, Conf = 100.0
12: 5 => 3 Supp = 50.0, Conf = 66.67
13: 3 => 5 Supp = 50.0, Conf = 66.67
```

## 11.8.3. Решение задачи кластеризации

Рассмотрим решение задачи кластеризации на примере сегментации потребителей. Исходные данные для этой задачи приведены в табл. 11.1 и представляют собой информацию о клиентах туристического агентства.

**Таблица 11.1**

| Пол | Возраст | Кол-во поездок | Любимая страна | Потраченная сумма |
|---|---|---|---|---|
| f | 33 | 3 | Spain | 10 500 |
| f | 28 | 1 | Turkey | 645 |
| m | 16 | 1 | Poland | 433 |
| m | 34 | 2 | USA | 15 230 |

**Таблица 11.1** *(окончание)*

| Пол | Возраст | Кол-во поездок | Любимая страна | Потраченная сумма |
|---|---|---|---|---|
| f | 52 | 12 | Spain | 12 450 |
| m | 19 | 1 | France | 1426 |
| f | 45 | 5 | Russia | 4900 |
| m | 72 | 7 | Germany | 8560 |
| m | 21 | 4 | Canada | 17 870 |
| m | 49 | 4 | Spain | 5400 |

Данные в формате ARFF имеют следующий вид:

```
@relation 'travel'

@attribute sex { m, f }
@attribute age real
@attribute numb_journeys real
@attribute favor_country { Spain, Turkey, Poland, USA, France, Russia,
Germany, Canada }
@attribute money_spent real

@data
f 33 3 Spain 10500
f 28 1 Turkey 645
m 16 1 Poland 433
m 34 2 USA 15230
f 52 12 Spain 12450
m 19 1 France 1426
f 45 5 Russia 4900
m 72 7 Germany 8560
m 23 4 Canada 17870
m 49 4 Spain 5400
```

Код с комментариями, решающий для приведенных данных задачу кластеризации, приведен далее.

```
// Загрузить исходные данные и получить их метаданные:
MiningArffStream arff = new MiningArffStream(
"data/arff/travel.arff");
MiningDataSpecification metaData = inputData.getMetaData();
// Создать экземпляр настроек и передать метаданные
ClusteringSettings miningSettings = new ClusteringSettings();
miningSettings.setDataSpecification( metaData);
```

```
// Создать экземпляр для выполнения настроек, специфичных для
// алгоритма k-средних, и загрузить их из конфигурационного файла
// algorithms.xml по имени 'KMeans'
MiningAlgorithmSpecification miningAlgorithmSpecification =
MiningAlgorithmSpecification.getMiningAlgorithmSpecification(
"KMeans");
// Установить количество кластеров, на которые необходимо
// разбить данные
setNumberOfClusters(miningAlgorithmSpecification, 3);

// Создать экземпляр алгоритма
String className = miningAlgorithmSpecification.getClassname();
ClusteringAlgorithm algorithm = (Clustering)initAlgorithm(className);

// Передать ему исходные данные и настройки
algorithm.setMiningInputStream( inputData);
algorithm.setMiningSettings( miningSettings);
algorithm.setMiningAlgorithmSpecification(
miningAlgorithmSpecification);

// Построить модель
MiningModel model = algorithm.buildModel();
```

В результате данные были разбиты на три сегмента:

- клиенты среднего возраста, предпочитающие поездки в южные страны;
- пожилые клиенты, которые посещают европейские страны;
- молодые клиенты, мало путешествующие.

## 11.8.4. Решение задачи классификации

Решим задачу классификации данных (табл. 11.2), представляющих собой информацию о клиентах телекоммуникационной компании. В результате классификации необходимо будет построить модель, которая позволит определить, покинет ли клиент компанию после двух лет сотрудничества.

**Таблица 11.2**

| Пол | Возраст | Текущий тариф | Израсходованные единицы | Покинул |
|---|---|---|---|---|
| f | 23 | Normal | 345 | no |
| m | 18 | Power | 9455 | no |
| m | 36 | Power | 456 | no |
| m | 34 | Normal | 3854 | yes |

**Таблица 11.2** *(окончание)*

| Пол | Возраст | Текущий тариф | Израсходованные единицы | Покинул |
|---|---|---|---|---|
| f | 52 | Economy | 2445 | no |
| f | 19 | Economy | 14 326 | no |
| f | 45 | Normal | 347 | no |
| m | 42 | Economy | 5698 | yes |
| m | 21 | Power | 267 | no |
| m | 48 | Normal | 4711 | yes |

Данные в формате ARFF имеют следующий вид:

```
@relation 'cancelling'

@attribute sex { f, m }
@attribute age { below20, 20to30, 31to40, 41to50, 51to60, above61 }
@attribute curent_tariff { normal, power, economy }
@attribute consumed_units { below500,501to1000,1001to10000,above10001 }
@attribute canceller { yes, no }

@data
f 20to30 normal below500 no
m below20 power 500to1000 no
m 31to40 power below500 no
m 31to40 normal 1001to10000 yes
f 51to60 economy 1001to10000 no
f below20 economy above10000 no
f 41to50 normal below500 no
m 41to50 economy 5000to10000 yes
m 20to30 power below500 no
m 41to50 normal 501to1000 yes
```

Код с комментариями, строящий для приведенных данных дерево решений, приведен далее.

```
// Загрузить исходные данные и получить их метаданные
MiningArffStream arff = new MiningArffStream(
"data/arff/cancelling.arff");
MiningDataSpecification metaData = inputData.getMetaData();

// Получить атрибут, представляющий зависимую переменную
MiningAttribute targetAttribute = (MiningAttribute)
metaData.getMiningAttribute( "canceller");
```

```
// Создать экземпляр настроек и передать метаданные
SupervisedMiningSettings miningSettings = new
SupervisedMiningSettings();
miningSettings.setDataSpecification( metaData);

// Передать атрибут, представляющий зависимую переменную
miningSettings.setTarget( targetAttribute);

// Создать экземпляр для выполнения настроек, специфичных для
// алгоритма ID3, и загрузить их из конфигурационного файла
// algorithms.xml по имени 'DecisionTree (Id3)'
MiningAlgorithmSpecification miningAlgorithmSpecification =
MiningAlgorithmSpecification.getMiningAlgorithmSpecification(
"DecisionTree (Id3)");
// Создать экземпляр алгоритма, строящего дерево решений
String className = miningAlgorithmSpecification.getClassname();
DecisionTreeAlgorithm algorithm = (DecisionTreeAlgorithm)
initAlgorithm(className);

// Передать алгоритму исходные данные и настройки
algorithm.setMiningInputStream( inputData);
algorithm.setMiningSettings( miningSettings);
algorithm.setMiningAlgorithmSpecification(
miningAlgorithmSpecification);

// Построить модель
MiningModel model = algorithm.buildModel();

// Показать дерево
showTree((DecisionTreeMiningModel) model);
```

В результате будет построено дерево, представленное на рис. 11.13.

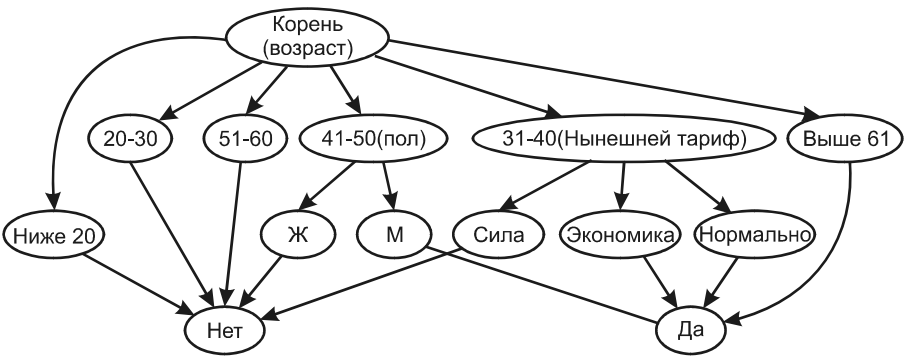



**Рис. 11.13.** Дерево решений, построенное алгоритмом ID3

Решим задачу для новых данных, представленных в табл. 11.3.

**Таблица 11.3**

| Пол | Возраст | Текущий тариф | Израсходованные единицы |
|---|---|---|---|
| f | 34 | Economy | 155 |
| m | 56 | Power | 2398 |
| m | 18 | Power | 253 |
| f | 39 | Normal | 7544 |

Результат классификации с помощью построенного дерева решений со степенью ошибки классификации 50 % будет следующий:

```
f 34 economy 155 => yes
m 56 power 2398 => no
m 18 power 253 => yes
f 39 normal 7544 => no
```

# Выводы

Кратко отметим материал, изложенный в этой главе.

- Библиотека Xelopes — свободно распространяемая библиотека, обеспечивающая универсальную основу для стандартного доступа к алгоритмам Data Mining.
- Достоинствами библиотеки Xelopes являются: независимость от платформы; независимость от исходных данных; поддержка всех основных стандартов Data Mining; реализация на языках Java, C++ и C#.
- Классы пакета Model расширяют классы из одноименного пакета стандарта CWM и описывают различные модели. Например, статистическая модель, модель ассоциативных правил, деревья решений, кластерная модель, модель сиквенциального анализа и др.
- Классы пакета Settings расширяют классы из одноименного пакета стандарта CWM и описывают классы настроек. Например, настройки для создания статистической модели, модели ассоциативных правил, деревьев решений, кластерной модели, модели сиквенциального анализа и др.
- Классы пакета Attribute расширяют классы из одноименного пакета стандарта CWM и описывают различные типы атрибутов данных: численные, категориальные, категориально-иерархические и др.

- Пакет Algorithms включает классы, реализующие алгоритмы построения различных моделей. Например, статистической модели, модели ассоциативных правил, деревья решений, кластерной модели, модели сиквенциального анализа и др.
- Пакет DataAccess включает классы, реализующие унифицированный доступ к различным источникам информации. Например, таким как файлы формата ARFF, файлы протоколов Web-серверов, базы данных и др.
- Пакет Transformation содержит классы для реализации пошаговой концепции преобразования данных. Они опираются на стандарт CWM.
- Решение задач Data Mining с помощью библиотеки Xelopes реализуется в несколько этапов: загрузка данных, выполнение настроек, инициализация алгоритма, построение модели. При решении задач с учителем построенная модель может применяться к новым данным.

# ГЛАВА 12

# Распределенный анализ данных

## 12.1. Системы мобильных агентов

### 12.1.1. Основные понятия

Исследования в области агентных технологий начали проводиться в 70-х гг. и были связаны с работами по искусственному интеллекту. Выделяют два основных этапа в истории исследования агентов. Первый этап начался около 1977 г. и был связан с исследованиями в области распределенного искусственного интеллекта. Второй этап начался в начале 90-х гг. В это время основное внимание было сосредоточено на агентах, выполняющих некоторые действия от имени человека.

В настоящее время вопрос об определении термина "агент" находится в стадии интенсивного обсуждения. В работе [50] дается следующее обобщенное определение:

*Агентом называется система, находящаяся в некоторой среде и являющаяся ее частью. Агент воздействует на среду при выполнении собственных задач и сам подвергается воздействию с ее стороны. Таким образом, изменения, произведенные агентом в среде, отражаются на нем самом в будущем*.

Агенты могут обладать следующими свойствами [51]:

- автономность (autonomous) — способность выполнять поставленную задачу без необходимости вмешательства извне;
- реактивность (reactivity) — способность воспринимать изменение среды и предпринимать ответные действия;
- целенаправленность (goal oriented) — способность выполнять поставленные перед ним задачи;
- устойчивость (persistent) — способность восстанавливать свое состояние после аварийного завершения;

- общительность (communicative) — возможность взаимодействовать с другими элементами среды;
- адаптивность (adaptive) — способность изменять свое поведение в зависимости от накопленного опыта и текущей обстановки;
- мобильность (mobility) — возможность перемещаться в среде;
- гибкость (flexible) — возможность изменять собственное поведение.

Совокупность нескольких агентов, работающих совместно, а следовательно, обладающих свойством общительности, называют *многоагентными системами*. Для работы многоагентная система должна включать в себя вспомогательные службы (например, поисковую службу). Для работы агентов, обладающих свойствами мобильности (мобильных агентов — МА), на узлах устанавливаются программные компоненты — платформы, обеспечивающие среду выполнения и интерфейс между агентами и компьютером. МА в процессе своего выполнения перемещаются между узлами сети, взаимодействуют с их ресурсами и другими агентами, что требует решения дополнительных проблем в организации систем мобильных агентов (СМА). СМА — это распределенное приложение, обеспечивающее работу МА.

## 12.1.2. Стандарты многоагентных систем

В настоящее время существуют десятки систем, использующих МА. Для их совместимости разрабатываются спецификации и стандарты. В настоящее время существует два основных стандарта, разработанные в ассоциациях OMG (Object Management Group) и FIPA (Foundation for Intelligent Physical Agents).

Ассоциация OMG разработала стандарт MASIF (Mobile Agent System Interoperability Facilities) [51]. В нем она уделила внимание стандартизации следующих проблем технологии МА:

- *управление агентами*. Унифицируется синтаксис и правила выполнения операций управления жизненным циклом МА: создание агентов, приостановка и возобновление их выполнения, перемещение агента и завершение его работы;
- *идентификация агентов*. Унифицируется синтаксис имен агентов. Введение подобного синтаксиса позволяет идентифицировать агента в системе;
- *типизация и адресация агентской платформы*. Унифицируется синтаксис описания типа и адреса агентской платформы. Это необходимо для получения агентом информации о типе платформы, предоставляемой ее сервисами, а также для идентификации локальной и удаленной платформ друг друга.

Согласно стандарту MASIF [52], СМА делится на регионы (рис. 12.1). Регион объединяет платформы, обладающие общими полномочиями. Также он включает в себя реестр, содержащий информацию о платформах и агентах, находящихся в этом регионе. Для совместимости со стандартом MASIF он должен реализовывать интерфейс MAFFinder. В нем определены операции регистрации создания, удаления и перемещения агентов, мест и платформ.

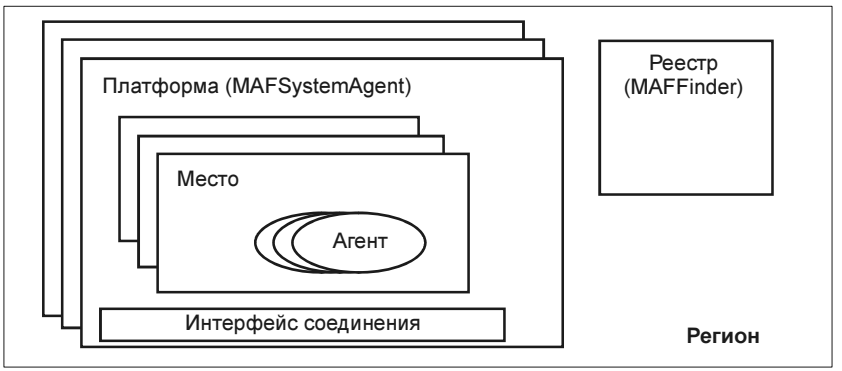



**Рис. 12.1.** Организация СМА согласно стандарту MASIF

Платформа (в стандарте MASIF называется "агентная система") может создавать, интерпретировать, запускать, перемещать и уничтожать МА. Для совместимости со стандартом MASIF она должна реализовывать интерфейс MAFAgentSystem. В нем определены операции приема/передачи, создания/уничтожения, прерывания/возобновления агентов. Платформа идентифицируется именем и адресом.

В состав платформы входят как минимум одно место и интерфейс соединения. Место обеспечивает среду выполнения агентов на компьютере. В нем могут одновременно находиться несколько агентов. Интерфейс соединения реализует службу связи, службу имен и службу безопасности.

MASIF поддерживает некоторые сервисы CORBA, но не требует обязательного их использования. Например, служба имен (Naming Service) поддерживает имена CORBA объектов и может использоваться объектами MAFAgentSystem и MAFFinder. Для поддержки жизненного цикла агента может использоваться сервис LifeCycle. Для сериализации и десериализации агентов возможно использование сервиса Externalization. Сервис Security Service может быть использован для авторизации агентов и управления их доступом к сетевым ресурсам.

FIPA — некоммерческая организация, созданная в 1996 г. Ее главной задачей является разработка спецификаций, определяющих взаимодействие агентов [53].

В них рассматриваются следующие основные темы:

- *управление агентами*. Унифицируется архитектура платформ агентов, которые включают в себя службы маршрутизации сообщений и управления жизненным циклом агентов;
- *язык взаимодействия агентов (ACL)*. Описывается синтаксис языка, предназначенного для взаимодействия агентов разных систем;
- *взаимодействие с неагентскими программами*. Унифицируются способы взаимодействия агентов с программным обеспечением, которое не использует агентную технологию, через "оболочки", включающие специфицированную онтологию и динамический механизм регистрации, и способы взаимодействия с человеком;
- *управление безопасностью агентов*. Выделяются ключевые угрозы безопасности в управлении агентом и описываются возможные средства защиты;
- *управление МА*. Унифицируются операции ("передать" и "запустить") агентской платформы, необходимые для управления МА;
- *служба онтологии*. Описывается служба, обеспечивающая правильное понимание запросов, сообщений, терминов в контексте области применения (коммерция, спорт, медицина и т. д.);
- *области применения*. Рассматриваются приложения, использующие агентскую технологию: сетевой помощник, аудио/видеоконференции, помощник планирования путешествий. С их помощью производится тестирование разработанных спецификаций.

Согласно спецификациям FIPA, CMA состоит из платформ, в состав которых входят следующие компоненты (рис. 12.2):

- система управления агентом (AMS — Agent Management System) управляет созданием, удалением, деактивацией, возобновлением, аутоинтефикацией и миграцией агентов, находящихся на платформе. Она обеспечивает сервис "белых страниц", который хранит текущее местонахождение МА;
- служба каталога (DF — Directory Facilitator) реализует сервис "желтых страниц", который хранит описание агентов;
- менеджер безопасности платформы агентов (APSM — Agent Platform Security Manager) отвечает за осуществление политики безопасности на транспортном уровне и проверку выполнения операций управления агентами;
- канал связи агентов (ACC — Agent Communication Channel) использует информацию, предоставляемую системой управления агентов для маршрутизации сообщений между агентами.

Для достижения поставленных целей ассоциации FIPA и OMG объединяют свои усилия. Первым результатом совместного сотрудничества является раз-

работка расширения языка UML для агентов — Agent UML (AUML). Некоторые элементы языка AUML предполагается включить во вторую версию языка UML.

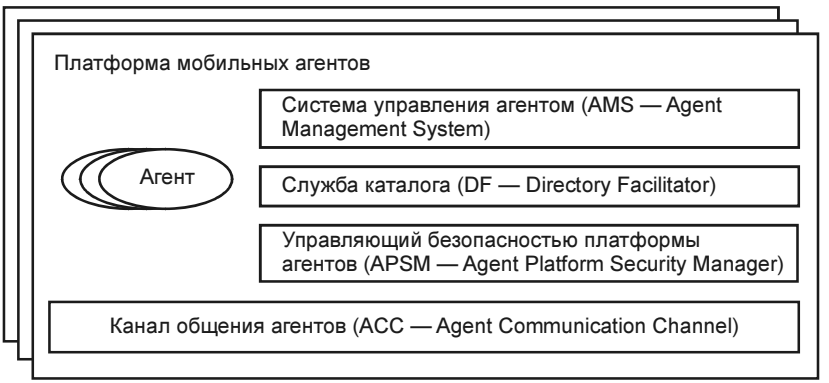



**Рис. 12.2.** Организация СМА согласно стандарту FIPA

## 12.1.3. Системы мобильных агентов

В настоящее время существует несколько десятков СМА. Многие из них (Gypsy, JADE, Ajanta, JATLite и др.) разработаны в университетах с целью исследования этой технологии. Некоторые системы (такие как ASDK, JAFMAS и др.) существуют на уровне библиотек, предоставляя программисту только базовые классы для реализации основных компонентов: агентов, платформ, механизмов взаимодействия и безопасности. На их основе разрабатываются самостоятельные системы, например MagNet и E-Commercia.

В последнее время появляются и коммерческие системы, такие как Gossip фирмы Tryllian, Bee-gent и Plangent корпорации Toshiba. К сожалению, техническая документация для них недоступна.

Некоторые проекты, проводимые в рамках исследования технологии МА (ARA в университете Кайзерслаутер, Mole в университете Штутгарта, Odyssey компании General Magic и др), в настоящее время закрыты. Существует ряд систем, использующих МА только для решения собственных задач. К такому типу систем относится система Voyager фирмы ObjectSpace.

## 12.1.4. Система мобильных агентов JADE

Система JADE (Java Agent Development Framework) была разработана в Италии фирмой CSELT S.p.A. совместно с университетом города Парма. Система реализована на Java и совместима со стандартом FIPA.

Среду выполнения агентов на узле обеспечивают контейнеры, которые входят в состав распределенной платформы (рис. 12.3). Контейнеры реализованы как RMI-объекты, поэтому для получения информации о действующих в системе контейнерах используется RMI-реестр.

Главной особенностью рассматриваемой системы является централизованная система управления агентами AMS. Система управления агентами предназначена для создания, удаления, деактивации, возобновления и перемещения МА на платформе. Она реализует сервис "белой страницы" для всех агентов, находящихся на ней. Сервис хранит информацию о текущем местонахождении агентов и аналогичен региональному реестру в системе Grasshopper.

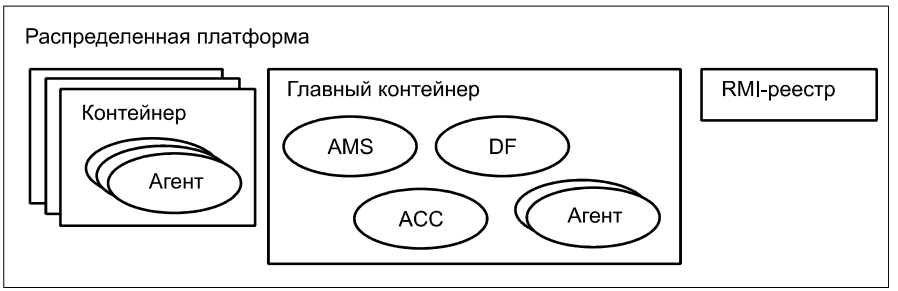



**Рис. 12.3.** Платформа агентов в JADE

В соответствии со стандартом FIPA, платформа системы JADE включает в себя службу каталога DF и канал связи агентов ACC. Служба каталога реализует сервис "желтой страницы" и хранит информацию об агентах, действующих в системе. Канал общения агентов управляет маршрутизацией сообщений между агентами, используя информацию от системы управления агентами.

Все описанные ранее компоненты реализованы как стационарные агенты и размещаются на главном контейнере.

Агент может взаимодействовать с другими агентами посредством асинхронной передачи ACL-сообщений. При отправке сообщения указывается ID получателя. Платформа (точнее ACC) сама определяет текущее местонахождение агентов и более подходящий механизм транспортировки. Все получаемые сообщения ставятся в очередь, доступ к которой реализуется агентской платформой. Следовательно, сообщения будут доставлены агенту, даже если он в момент получения запроса находится в состоянии ожидания. Если в процессе взаимодействия один из агентов перемещается, то связь между ними теряется.

Система JADE не предоставляет возможности поиска агента, переместившегося на другую распределенную платформу, следовательно, область взаимо-

действия агентов ограничена одной платформой. Централизованный сервис "белых страниц" предоставляет возможность агентам получать информацию о состоянии системы в рамках одной платформы.

# 12.2. Использование мобильных агентов для анализа данных

## 12.2.1. Проблемы распределенного анализа данных

Ценность и качество получаемых в результате анализа знаний зависит от объемов анализируемых данных. Чем бóльший объем подвергается анализу, тем бóльшее количество закономерностей лучшего качества могут быть извлечены. В настоящее время большие объемы данных хранятся распределенно (т. е. на разных вычислительных машинах, территориально удаленных друг от друга и объединенных вычислительной сетью, например, интранет). Кроме того, распределение данных может быть вызвано природой источников информации (например, датчики на транспортной магистрали). В связи с этим возникает необходимость в адаптации алгоритмов и методов интеллектуального анализа для распределенных данных.

Наиболее подходящим способом адаптации представляется агентная технология и, в частности, системы мобильных агентов (СМА). Агентная технология объединяет в себе опыт работ по искусственному интеллекту и созданию распределенных систем.

## 12.2.2. Агенты-аналитики

Анализ данных в большинстве случаев не ограничивается применением одного алгоритма или решением одной задачи. Как правило, аналитик последовательно использует несколько аналитических средств с целью получения новых знаний. Такими средствами являются методы и алгоритмы анализа данных. При этом знания, полученные в результате применения одного алгоритма, являются предпосылкой применения следующего алгоритма. На основе уже имеющихся у аналитика знаний производится выбор метода анализа, а также способ и условия его применения. В результате применения метода знания аналитика обогащаются, следовательно, можно повторить итерацию на тех же или на новых данных. При этом выбор средства и/или способа его применения может быть другим. Изменение условий анализа должно привести к получению новых знаний. Описанный цикл выполняется до тех пор, пока в результате очередной итерации будут получены уже неновые знания.

Полученные знания могут быть применены на практике. В результате такой практической деятельности вносятся изменения в среду, из которой поступают данные.

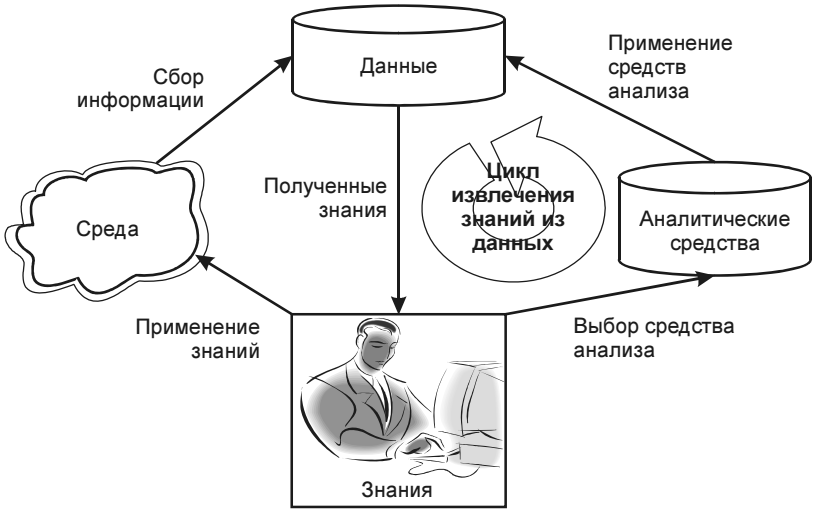



**Рис. 12.4.** Аналитический цикл по извлечению знаний из данных

Описанный процесс можно представить так, как это изображено на рис. 12.4.

Основная идея использования агентной технологии в области интеллектуального анализа данных (ИАД) заключается в том, чтобы инкапсулировать в агенте цикл извлечения знаний из данных. В этом случае агент должен выступать в роли аналитика. В его обязанности будут входить следующие задачи:

- накопление знаний;
- выбор средства анализа и способа его применения на основе имеющихся знаний;
- применение выбранного средства к данным.

Агент должен решать эти задачи самостоятельно или с помощью своего владельца. Для решения этих задач агент должен обладать следующими свойствами:

- целенаправленность — целью агента является извлечение максимального количества знаний из данных;
- общительность — агент должен иметь возможность обмениваться своими знаниями с другими агентами для более продуктивного анализа;
- адаптивность — агент должен использовать полученные в результате анализа и общения с другими агентами знания для новых исследований и применения их к данным;
- мобильность — для работы с данными, находящимися на разных узлах, объединенных сетью.

Остальные свойства не являются обязательными для агента, выполняющего ИАД.

### 12.2.3. Варианты анализа распределенных данных

Многоагентная система характеризуется наличием в ней нескольких агентов и вспомогательных служб. В частности, инфраструктура многоагентной системы должна обеспечивать следующие свойства агентов: *общительность* и *мобильность*. Рассмотрим, как могут быть использованы эти свойства в рамках решения задачи анализа данных.

Общительность позволяет агентам обмениваться между собой сообщениями. Это свойство можно использовать для обмена знаниями между агентами. Модель, полученная в результате применения аналитического средства, может быть передана другому агенту. В результате принимающий агент перейдет в другое состояние, не применяя данное аналитическое средство к тем же данным.

Необходимо учитывать, что такой обмен знаниями может выполняться только между агентами, выполняющими анализ над подобными (одинаковыми, однородными и т. п.) данными. В противном случае такой обмен знаниями может привести к дезинформированию агентов.

Общение и обмен знаниями позволяет выполнять анализ параллельно несколькими агентами. Наиболее эффективен такой подход при анализе данных, хранящихся распределенно. В этом случае однородные данные хранятся на нескольких узлах, объединенных сетью. Распределение необходимо, как правило, из-за большого объема данных и для повышения производительности работы с ними. Использование нескольких агентов для анализа таких данных может выглядеть так, как это представлено на рис. 12.5. На каждом узле анализ локальных данных выполняет отдельный агент. Результатами, полученными при анализе, он обменивается с агентами, выполняющими параллельно с ним анализ данных на других узлах.

При анализе распределенных данных эффективно также и другое свойство — мобильность. Она позволяет агенту, последовательно перемещаясь между узлами, выполнять анализ данных, взаимодействуя с ними локально (рис. 12.6). Мобильные агенты наиболее полезны, когда информация о доступных для анализа источниках данных появляется в процессе анализа.

Агенты могут одновременно обладать общительностью и мобильностью. В этом случае анализ распределенных данных может выполняться комбини-

рованным подходом, т. е. агенты могут и обмениваться знаниями, и при необходимости перемещаться на другие узлы.

Методы и алгоритмы Data Mining наиболее эффективны при анализе больших объемов данных. Такие объемы в современных условиях хранятся распределенно в локальных вычислительных сетях. В связи с этим применение многоагентной технологии (в частности таких свойств агентов, как общительность и мобильность) для анализа в распределенных хранилищах должно повысить производительность аналитических систем.

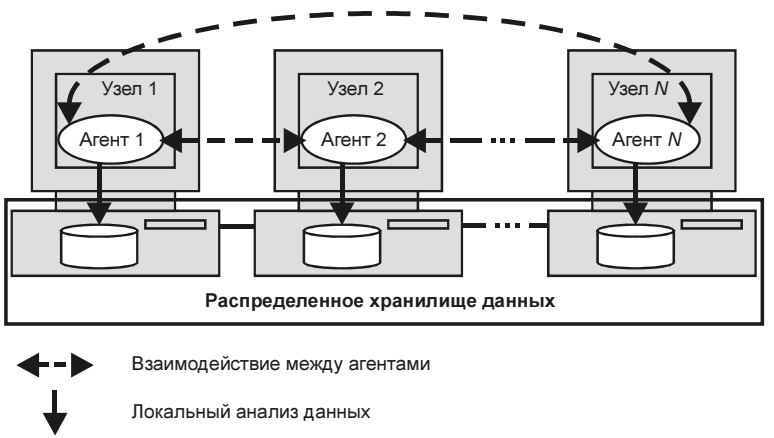



**Рис. 12.5.** Параллельный анализ данных в распределенном хранилище

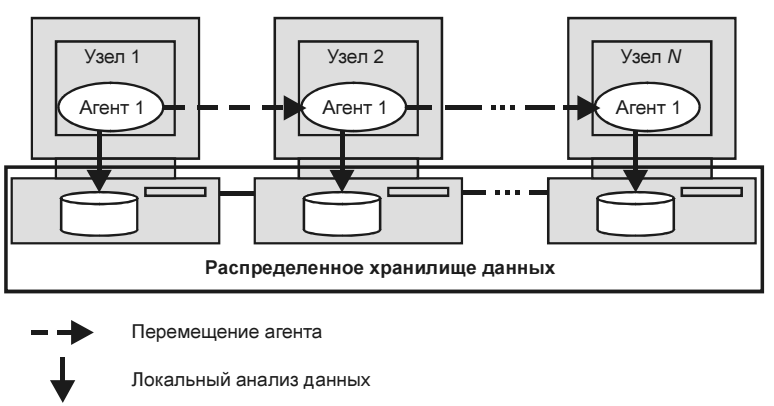



**Рис. 12.6.** Анализ данных в распределенном хранилище с помощью мобильного агента

# 12.3. Система анализа распределенных данных[1]

## 12.3.1. Общий подход к реализации системы

Система анализа распределенных данных представляет собой многоагентную систему. В процессе анализа существующих библиотек для разработки многоагентных систем и библиотек алгоритмов интеллектуального анализа данных были выбраны библиотеки JADE и Xelopes.

Полученная система включает в себя агентов, выполняющих следующие задачи:

- сбор информации о базе данных;
- сбор статистической информации о данных;
- решение одной задачи интеллектуального анализа данных;
- решение интегрированной задачи интеллектуального анализа данных.

В разработанной системе в качестве базовых классов мобильных и стационарных агентов были использованы агенты JADE.

Мобильные агенты используются в системе распределенного анализа непосредственно для применения алгоритмов анализа к данным, находящимся на разных машинах, объединенных локальной сетью.

Стационарные агенты, не обладая свойством мобильности, выполняют вспомогательные функции. К таким функциям можно отнести следующие:

- доступ к базе данных;
- настройка мобильных агентов;
- отображение результатов работы мобильных агентов;

и т. п.

Взаимодействие между агентами обоих видов осуществляется посредством языка взаимодействия агентов ACL, являющегося частью стандарта FIPA.

Мобильные агенты для выполнения анализа могут использовать две стратегии:

- последовательный анализ — агент последовательно перемещается между узлами сети и выполняет анализ на каждом;
- параллельный анализ — агент создает клонов, которые, перемещаясь каждый на свой узел, выполняют анализ и возвращают его результаты исходному агенту.

---

[1] Многоагентная система анализа данных была разработана в СПбГЭТУ "ЛЭТИ" в рамках проекта "Многоагентная технология интеллектуального анализа данных и извлечения знаний", выполненного по ведомственной научной программе "Развитие научного потенциала высшей школы", 2005.

Выбор стратегии анализа осуществляется пользователем при создании агента в диалоговом окне, представленном на рис. 12.7.

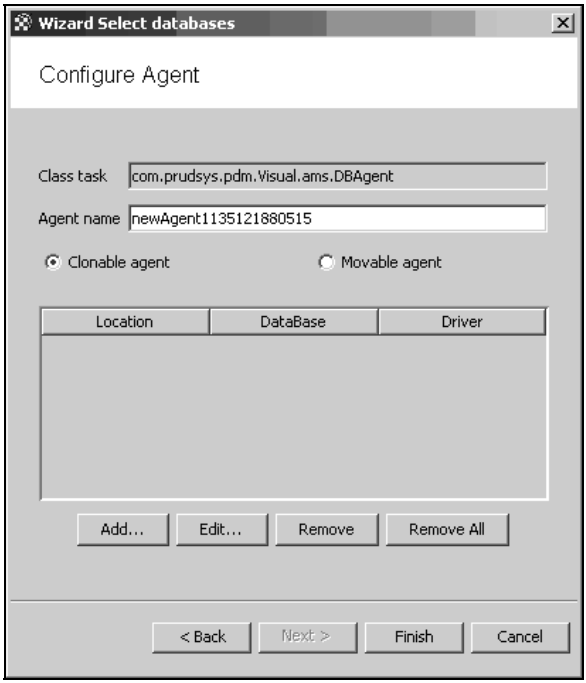



**Рис. 12.7.** Выбор стратегии анализа

Все агенты могут быть сохранены и затем восстановлены для дальнейшей работы.

## 12.3.2. Агент для сбора информации о базе данных

На начальном этапе анализа данных, если неизвестна даже структура баз данных, в которых хранится анализируемая информация, необходимо получить информацию о таблицах, полях таблиц и их типах. Для решения этой задачи в разработанной системе может быть использован агент сбора информации о базе данных.

Для создания и настройки такого вида агента необходимо выполнить следующую последовательность действий:

1. Выбрать в главном меню окна приложения пункт **File | New Agent** (Файл | Новый агент) (рис. 12.8).
2. В открывшемся диалоговом окне выбрать тип агента (DataBase Agent). Для этого в дереве типов (рис. 12.9) надо выделить лист дерева **Data base**,

после чего станет доступна кнопка **Next** (Далее) для перехода к следующему диалогу.


**Рис. 12.8.** Меню программы


**Рис. 12.9.** Выбор типа создаваемого агента

3. В следующем окне настройки можно ввести имя агента и выбрать контейнеры, которые он должен обойти (см. рис. 12.7). Имя агента должно быть длиннее 2 символов, поэтому если введенное имя меньшей длины, то переход на следующее окно не возможен (кнопки **Next** и **Finish** будут недоступны). Если доступен переход к следующему окну, то будет доступна кнопка **Next**, иначе кнопка **Finish** (Завершить). Кнопка **Back** (Назад) вернет вас к предыдущему окну выбора типа агента. Кнопка **Cancel** (Отмена) закроет мастер настроек.
4. Чтобы добавить новый контейнер, который агент должен посетить, необходимо нажать кнопку **Add** (Добавить) в окне, приведенном на рис. 12.7. В результате откроется диалоговое окно, приведенное на рис. 12.10, в котором можно настроить параметры добавляемого контейнера. В этом окне можно указать:
   - имя базы данных, чью структуру агент будет анализировать;
   - драйвер для доступа к базе данных.

Результаты работы агента могут быть представлены пользователю в виде дерева (рис. 12.11). Кроме того, агент может передать собранную информацию на языке ACL другим агентам.

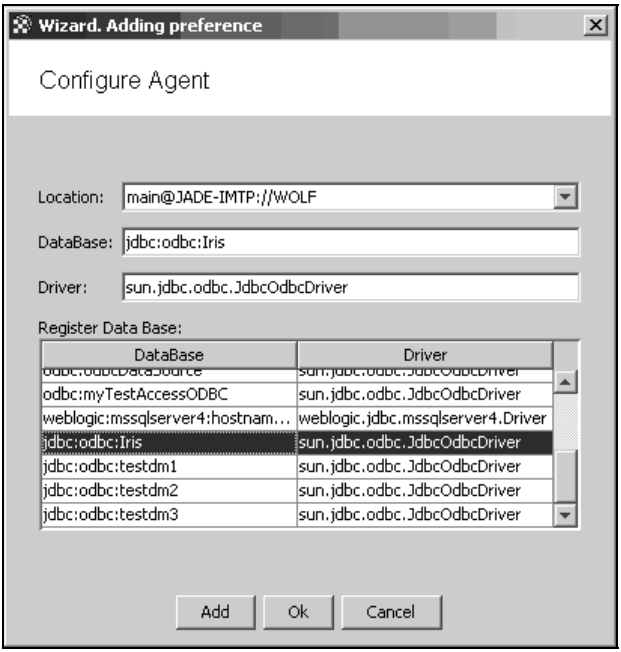



**Рис. 12.10.** Настройка задачи агента в зависимости от контейнера

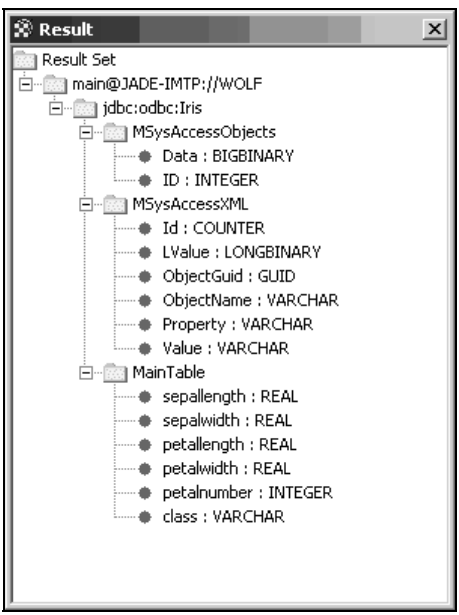



**Рис. 12.11.** Результат работы агента

## 12.3.3. Агент для сбора статистической информации о данных

Простейшим видом анализа является сбор статистической информации о данных, хранящихся в базе данных. Для такого анализа может быть использован агент для сбора статистической информации.

Для создания и настройки такого вида агента необходимо выполнить следующую последовательность действий:

1. Выбрать в главном меню окна приложения пункт **New Agent** (см. рис. 12.8).
2. Выбрать тип агента **Statistic** в дереве типов (см. рис. 12.9).
3. В диалоговом окне настройки выбрать контейнеры, по которым проходит агент (рис. 12.12). В этом окне можно пойти по пути, описанному в предыдущем разделе, или использовать информацию, которую собрал DataBase Agent.

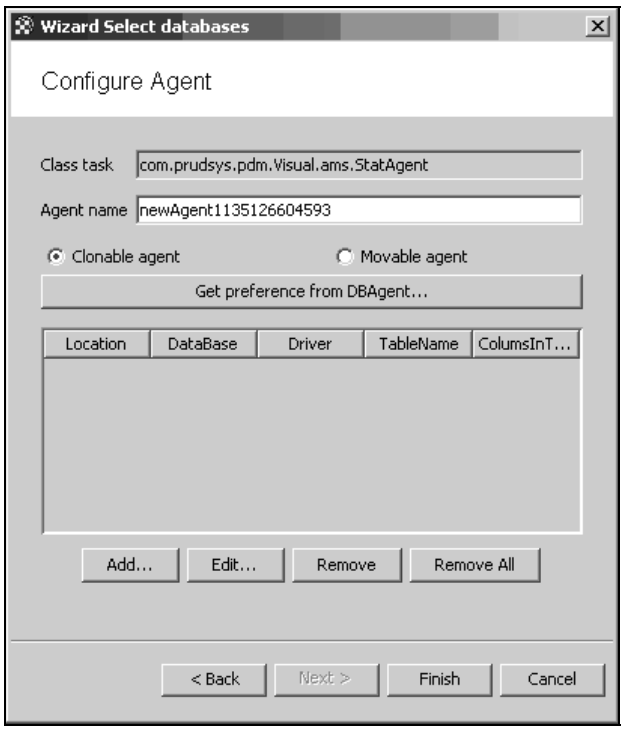



**Рис. 12.12.** Настройка задачи Statistic Agent

4. Если вы хотите использовать информацию DataBase Agent, то нажмите кнопку **Get preference from DBAgent** (Получить настройки от DBAgent).

Открывшееся окно (рис. 12.13) позволяет загрузить настройки с учетом собранной информации. В левой части окна располагается список загруженных агентов. Если необходимо загрузить информацию с агента, который сохранен в файл, то для этого предусмотрена кнопка **LoadAgent** (Загрузить агента), иначе можно загрузить агента через центр управления агентами.

После выбора желаемого агента в левой части диалога надо нажать кнопку **Show** (Показать). Выбранные поля в дереве (поле выбирается установкой флажка около него) будут автоматически добавлены для анализа на заданном контейнере при нажатии кнопки **Apply** (Применить).

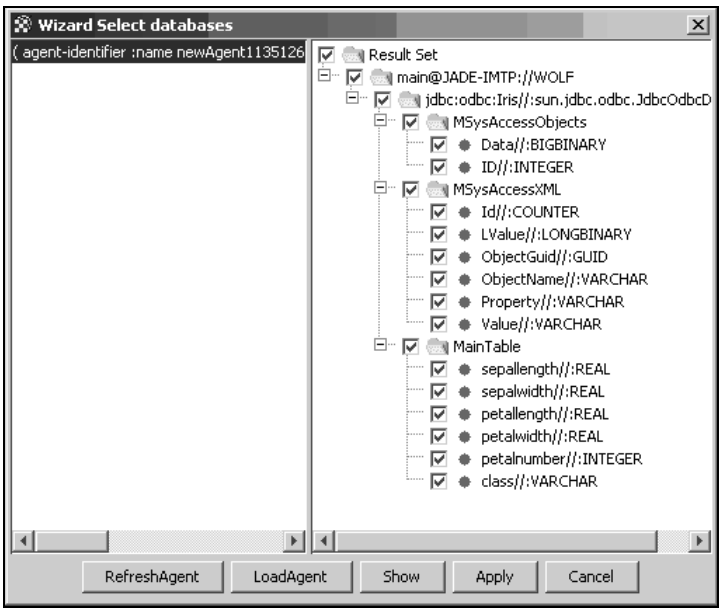



**Рис. 12.13.** Загрузка свойств на основе DataBase Agent

5. По приходу на новый контейнер агент будет использовать следующие настройки (рис. 12.14). Данное диалоговое окно появляется при нажатии кнопки **Add** в окне, приведенном на рис. 12.12.

   В данном диалоге можно указать:

   - имя базы данных, поля которой агент будет анализировать;
   - драйвер для доступа к базе данных;
   - имя таблицы;
   - исключаемые из обработки поля таблицы.

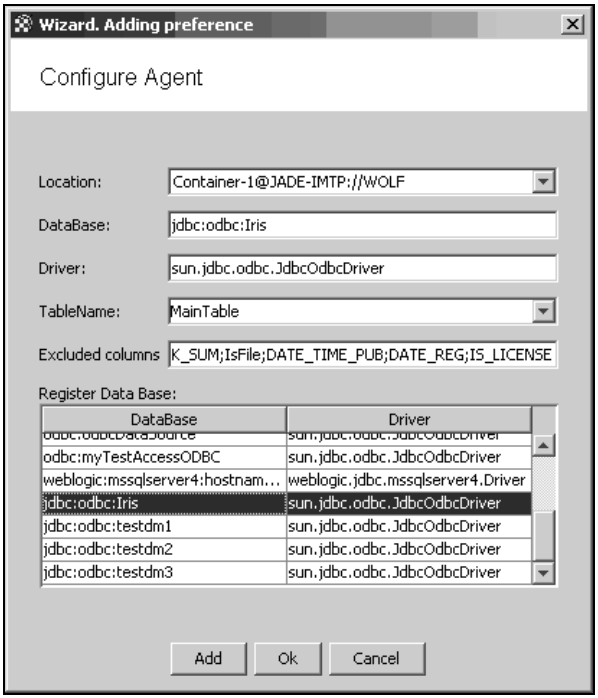



**Рис. 12.14.** Альтернативная загрузка свойств на основе DataBase Agent

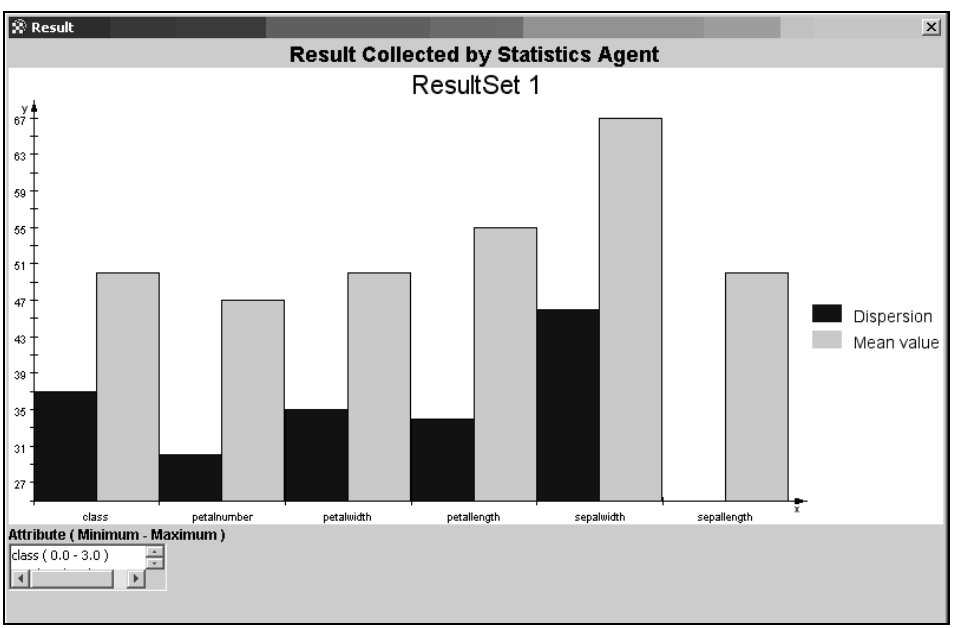



**Рис. 12.15.** Результат работы агента

Результаты работы агента могут быть представлены пользователю в виде диаграмм (рис. 12.15).

## 12.3.4. Агент для решения одной задачи интеллектуального анализа данных

Для более интеллектуального анализа данных в системе реализованы специальные агенты, решающие разные виды задач. Для их решения они могут использовать соответствующие алгоритмы интеллектуального анализа данных из библиотеки Xelopes.

Для создания и настройки такого вида агента необходимо выполнить следующую последовательность действий (на примере Clustering Agent):

1. Выбрать в главном меню окна приложения пункт **New Agent** (см. рис. 12.8).
2. Выбрать тип агента **Clustering Mining Agent** в дереве типов (см. рис. 12.9).
3. В окне настроек выбрать контейнеры, по которым проходит агент (см. рис. 12.7).
4. По приходу на новый контейнер агент будет использовать следующие настройки (рис. 12.16). Данное окно появляется при нажатии кнопки **Add** в окне, приведенном на рис. 12.7.

   В данном диалоге можно указать:

   - имя базы данных, поля которой агент будет анализировать;
   - драйвер для доступа к базе данных;
   - имя таблицы;
   - исключаемые из обработки поля таблицы;
   - каталог для файла с результатами работы агента в формате PMML;
   - результирующую таблицу.
5. После нажатия кнопки **Next** появляется окно настройки непосредственно самого алгоритма (рис. 12.17).

   На вкладке **Settings** (Настройки) устанавливаются следующие параметры модели алгоритма:

   - максимальное число кластеров;
   - параметры вычисления расстояния между векторами:
     - тип вычисления расстояния;
     - функция сравнения;
     - нормирование расстояний.

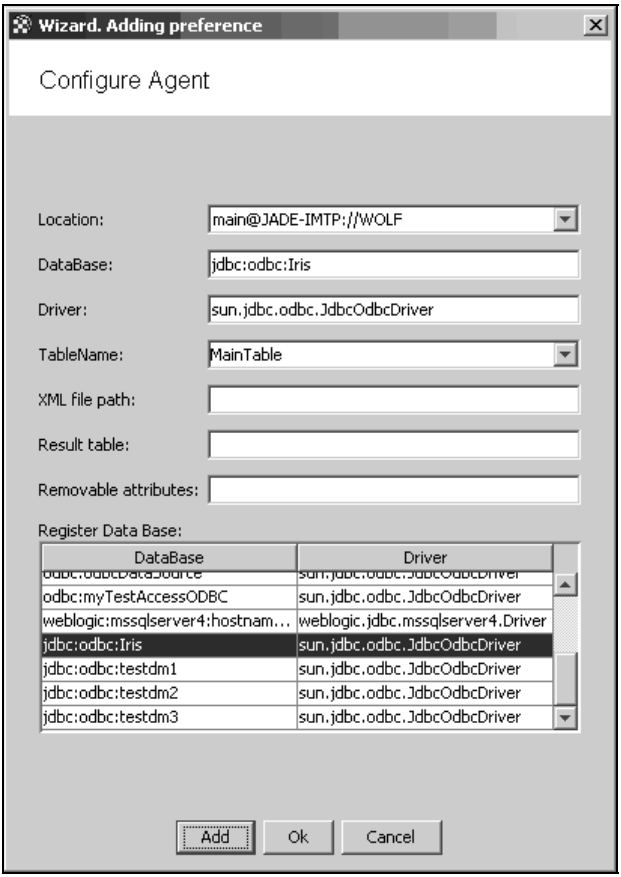



**Рис. 12.16.** Настройка задачи агента в зависимости от контейнера

6. На вкладке **Algorithm** (рис. 12.18) необходимо указать название алгоритма, например **CFuzzy**. Далее нужно установить параметры для выбранного алгоритма:

   - число кластеров;
   - максимальное число итераций алгоритма;
   - минимальное межкластерное различие для завершения алгоритма.

Результаты работы агента могут быть представлены пользователю в формате PMML (рис. 12.19).

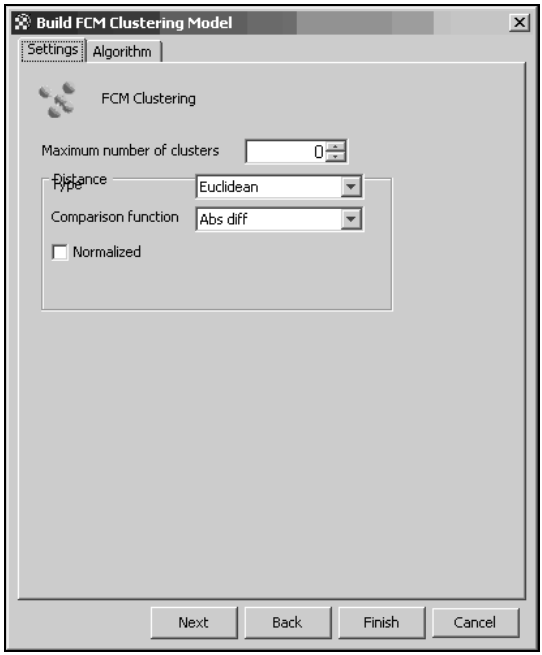



**Рис. 12.17.** Настройка алгоритма, вкладка **Settings**

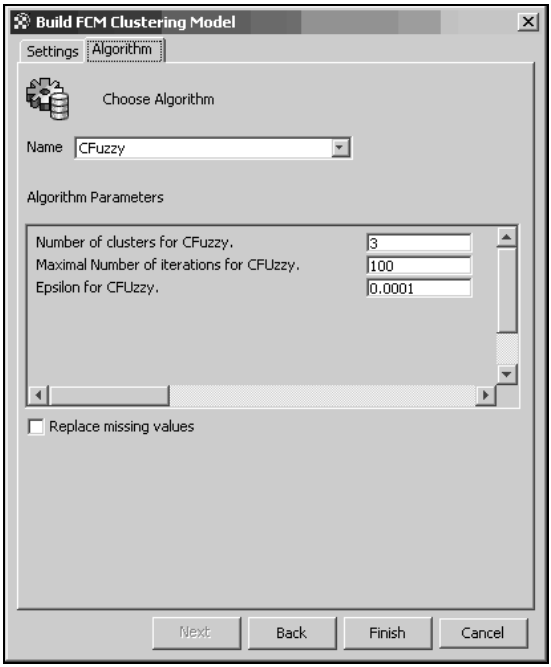



**Рис. 12.18.** Настройка алгоритма, вкладка **Algorithm**

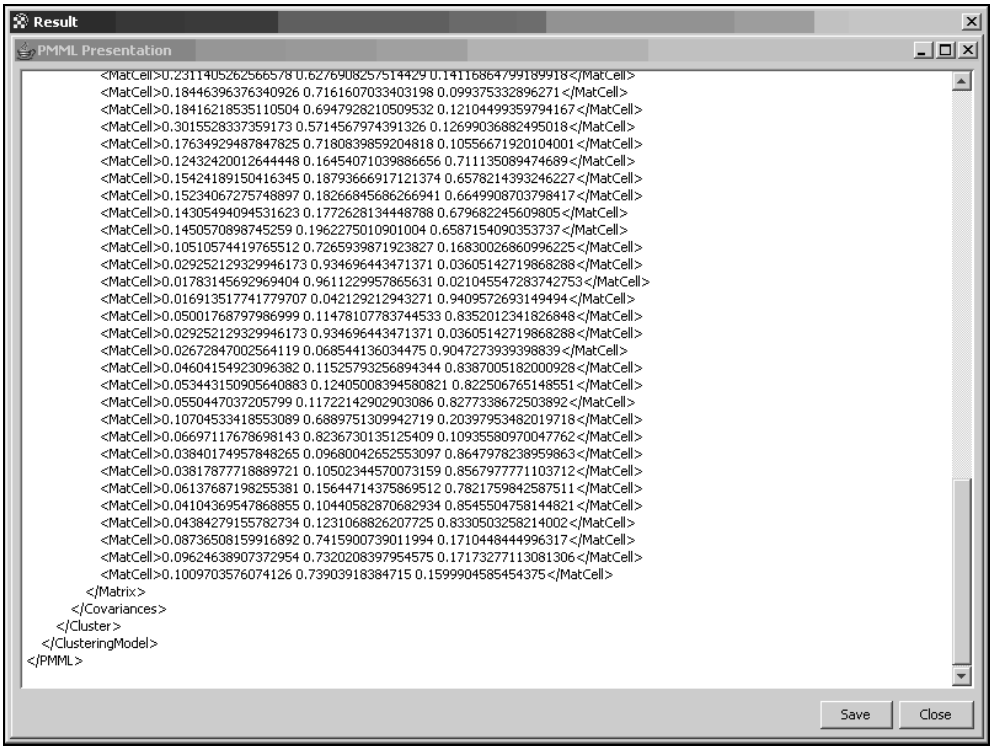



**Рис. 12.19.** Результат работы агента

## 12.3.5. Агент для решения интегрированной задачи интеллектуального анализа данных

Наиболее сложным видом анализа является анализ, в процессе которого решаются последовательно различные классы задач анализа данных. Примером такой последовательности являются решение задачи кластеризации, а затем применение алгоритмов для решения задачи классификации к найденным кластерам.

Такое последовательное решение задач интеллектуального анализа данных может выполняться с применением специального типа агента. Его настройка в части задания параметров для решения каждой задачи анализа выполняется так же, как это было описано в предыдущем разделе. При работе агента он может создать вспомогательных агентов, описанных в предыдущем разделе, для решения определенных задач анализа.

# Выводы

По результатам данной главы можно сделать следующие выводы.

- Агентом называется система, находящаяся в некоторой среде и являющаяся ее частью. Агент воздействует на среду при выполнении собственных задач, и сам подвергается воздействию с ее стороны. Таким образом, изменения, произведенные агентом в среде, отражаются на нем самом в будущем.
- Агент обладает следующими свойствами: автономность, реактивность, целенаправленность, устойчивость, общительность, адаптивность, мобильность, гибкость.
- В настоящее время существуют десятки систем, использующих МА (мобильный агент). Для их совместимости разрабатываются спецификации и стандарты. В настоящее время существует два основных стандарта, разработанных в ассоциациях OMG (Object Management Group) и FIPA (Foundation for Intelligent Physical Agents).
- Наиболее подходящим способом адаптации представляется агентная технология и, в частности, системы мобильных агентов (СМА). Она объединяет в себе опыт работ по искусственному интеллекту и созданию распределенных систем.
- Основная идея использования агентной технологии в области анализа данных заключается в том, чтобы инкапсулировать в агенте цикл извлечения знаний из данных. В этом случае агент должен выступать в роли аналитика.
- Мобильные агенты для выполнения анализа могут использовать две стратегии: последовательный анализ или параллельный анализ.
- Для сбора информации о базе данных реализован специальный вид агента, собирающий информацию о таблицах, полях и их типах, входящих в базу данных.
- Простейший вид анализа данных выполняет мобильный агент, собирающий статистическую информацию о данных.
- Для решения задач интеллектуального анализа данных реализованы разные типы мобильных агентов, которые могут использовать соответствующие алгоритмы анализа.
- Для решения комплексной задачи анализа данных реализован мобильный агент, который последовательно решает задачи интеллектуального анализа данных.

ГЛАВА 13

# Data Mining в реальном времени (Real-Time Data Mining)

## 13.1. Идея Data Mining в реальном времени

### 13.1.1. Адаптация системы к общей концепции

Большинство методов Data Mining, изучаемых в этой книге, являются *статическими*. Это означает, что с их помощью выполняется анализ исторических данных (т. е. данных, накопленных ранее и неизменяемых в процессе анализа), как для описания свойств данных *(см. разд. 4.4.2)*, так и для построения моделей предсказания на основе обучающей выборки *(см. разд. 4.4.1)*.

Методы Data Mining в реальном времени (или *Real-Time Analytics*), в основном, относятся к задаче предсказания. В отличие от статических методов они обучаются динамически и основаны на обратной связи от прогноза, полученного с помощью предсказательной модели (постоянном переобучении).

Возможный путь достичь этого — собрать все данные (постоянно добавлять к данным, подвергаемым анализу, новые данные) и применять алгоритмы Data Mining, начиная с начального (возможно незначительного) набора данных и заканчивая имеющимся в текущий момент времени полным набором данных. Такая процедура называется *накапливаемое обучение* (Batch Learning). Однако накапливаемое обучение часто не эффективно, т. к. требует запоминания больших объемов данных и приводит к длительному периоду обучения.

По этой причине чаще всего приложения реального времени для обновления существующих моделей Data Mining используют только новые данные. Такой подход является адаптивным, или пошаговым, и будет рассмотрен в данной главе.

## 13.1.2. Адаптивная добыча данных

Методы Data Mining в реальном времени стали чрезвычайно популярны в последние годы и, несомненно, занимают лидирующее положение. Однако сам по себе адаптивный подход не ограничивается только областью Data Mining, он находит более широкое применение во многих научных областях.

Примером использования адаптивного подхода является решение дифференциальных уравнений, особенно частных дифференциальных уравнений — одной из классических областей математики.

Из-за их сложности, большинство дифференциальных уравнений не может быть решено аналитически, поэтому часто для их решения используют методы аппроксимации. Самые популярные схемы аппроксимации — метод конечного элемента (Finite Element Method — FEM) и метод конечных разностей (Finite Difference Method — FDM). Методы конечного элемента аппроксимируют решением функций $f$ дифференциальных уравнений через базисы на решетках. Чем плотнее решетка, тем лучше аппроксимация; с другой стороны, более плотные решетки требуют более высокого уровня вычислений [54].

Конвергенция и сходимость скорости аппроксимации напрямую зависят от регулярности проблемы; особенно от геометрической формы области (условие Лифшица, унифицированное коническое условие и т. п.), так же, как и от специфических пограничных условий [54].

В большинстве приложений сделанные "классические" предположения регулярности приближены к действительным, и итерационные методы конечных элементов быстро сходятся в одну точку. Однако иногда мы имеем грубую ("ill-natured") специфичность и чрезвычайно сложные пограничные условия, которые трудно классифицировать. Это порождает много проблем. Являются ли границы искаженной заготовки действительно условием Лифшица? Является ли обратное давление действительно функцией Дирака или скорее пространством [54].

В результате, на практике такие проблемы часто приводят к медленной конвергенции с непригодным результатом. Для того чтобы преодолеть эти проблемы, применяются адаптивные методы конечных элементов, которые используют пошаговый подход, чтобы построить решетки и решить уравнения, появляющиеся в результате применения методов конечных элементов. Общая последовательность действий включает в себя следующие шаги:

1. Генерация начальной решетки.
2. Решение методами конечных элементов систем текущей решетки.
3. Вычисление ошибки с помощью оценки апостериорной ошибки.
4. Если ошибка меньше границы, то закончить вычисления.

5. Обновить решетку, используя реальную геометрию в областях с высоким значением ошибки.
6. Переход к шагу 2.

Подобная адаптивная процедура не только гарантирует быструю конвергенцию, но и обеспечивает группировку узлов вокруг оригинала (рис. 13.1). Более подробно вопрос изложен в [59].

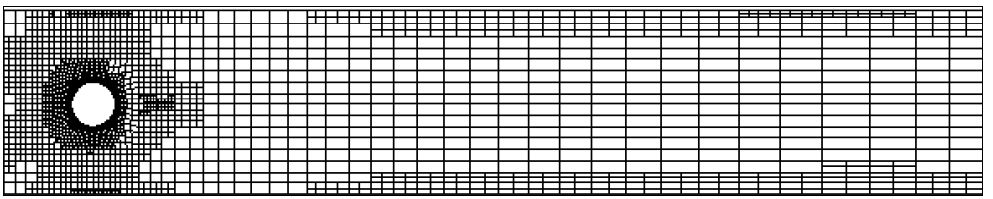


**Рис. 13.1.** Пример фигуры решетки адаптивного FEM-метода

Далее рассмотрим, как ранее указанный подход можно адаптировать к алгоритмам Data Mining:

1. Генерация начальной модели Data Mining (например, на основе имеющихся не обновляемых данных).
2. Основываясь на модели выбрать оптимизационное действие (рекомендация, почта и т. п.).
3. Анализ ответа и вычисление ошибки.
4. Если цель достигнута (например, вся продукция продана или нет новых данных), то завершить вычисления.
5. Обновить модель добычи данных, основываясь на ошибке ответа.
6. Переход к шагу 2.

Следует отметить, что в области Data Mining использование адаптивной процедуры имеет важное отличие. В Data Mining мы не только конструируем точную модель (как в методе конечного элемента), но и планируем достичь бизнес-цели!

Чтобы лучше понять разницу, рассмотрим пример использования адаптивного подхода Data Mining в call-центре компании, продвигающей новый товар. Агенты компании имеют список потенциальных клиентов с разными характеристиками (пол, адрес, возраст и др.) и стремятся обращаться только к тем клиентам, которые, вероятнее всего, закажут новый продукт непосредственно по телефону. Это типичная задача классификации *(см. гл. 5)*. Применение адаптивного подхода означало бы, что после каждого звонка (или ряда звонков) используются ответы клиентов для обновления модели классификации,

которая затем используется повторно для нового предсказания потенциальных клиентов.

В результате использования адаптивного подхода будет построена модель классификации, которая достаточно точно определяет клиента на основании его характеристик. Следует отметить, однако, здесь существенное противоречие с бизнес-целью: обзванивать только тех клиентов, которые действительно сделают заказ (не тратя деньги на звонки людям, которые заказ не сделают). С другой стороны, для того чтобы построить часть классификационной модели с низкой вероятностью заказа, необходимо также обзванивать людей, которые не делают заказы, что и противоречит бизнес-цели.

Первая задача называется *исследование* (exploring) (построение хорошей модели), вторая задача называется *эксплуатирование* (exploiting) (использование модели для достижения бизнес-цели). Правильная комбинация исследования и эксплуатации является важной темой в направлении Data Mining в реальном времени и обсуждается в *разд. 13.2.5*.

## 13.1.3. Статический Data Mining и Data Mining в реальном времени

Data Mining в реальном времени представляет динамический подход, тогда как классический Data Mining является более статическим. "Динамический" звучит лучше, чем "статический" и действительно имеет много преимуществ, поэтому может возникнуть вопрос: нуждаемся ли мы в классическом Data Mining? Ответ — конечно, да! Прежде всего, во многих областях может применяться только статический Data Mining, т. к. в них отсутствует автоматическое взаимодействие с клиентами. Например, оптимизация отгрузки правильных каталогов клиентам — одноразовая задача и трудная для автоматизации. Кроме того, в этом случае уходят недели, чтобы получить обратную связь от клиентов.

Во-вторых, эксперты в области Data Mining обычно достигают лучших результатов анализа, чем полностью автоматическое аналитическое программное обеспечение. Наконец, наиболее передовые методы Data Mining не поддерживают итерационное обучение. Даже простые деревья решений *(см. разд. 5.2.2)* трудно сделать итерационным, т. к. они являются нелинейными методами.

Однако появление адаптивных методов внесло изменение в Data Mining. До сих пор было кредо:

"Чем большие объемы доступных данных, тем лучше качество предсказания".

Хотя это все еще верно в статистическом смысле, однако для динамического Data Mining более верно новое понимание:

"Более важным, чем анализ просто исторических данных является процесс изучения через прямое взаимодействие с пользователем".

Действительно, намного более важным чем вопрос, купил ли определенный клиент молоко 5 лет назад и в какой комбинации с другими продуктами он это сделал, является то, как клиент реагирует в течение текущей сессии на предложение о молоке! Намного более важным чем вопрос, как шахматист играл два года назад, является то, как он это делает в текущей игре!

Конечно, лучший результат дает комбинация обоих подходов. Начальное изучение на исторических данных с использованием статического Data Mining, а затем применение адаптивного обучения через взаимодействие с пользователем. Необходимо заметить также, что тенденция IT-отделов предприятий к сервисно-ориентированным подходам (SOA, Web 2.0 и т. д) прямо способствует внедрению адаптивного подхода. Это связано с одним из достоинств сервисно-ориентированной архитектуры — независимость сервисов, которые интегрируются для выполнения бизнес-процессов. Такая независимость позволяет легко заменять модернизированный (адаптированный) сервис без изменения программного интерфейса, что в свою очередь способствует адаптации всей системы.

Подводя итог, можно выделить следующие преимущества применения адаптивного подхода в Data Mining:

- требуется меньше статистических предположений для алгоритмов Data Mining;
- модели быстро приспосабливаются к изменяющейся окружающей среде;
- системы могут регулярно собирать новые данные.

Однако существуют и некоторые неудобства:

- требуется обширная IT-инфраструктура, особенно для осуществления обратной связи;
- требуются более сложные математические инструменты (нестационарные процессы).

## 13.1.4. Применение Data Mining в реальном времени

Существует много областей применения Data Mining в реальном времени и их число непрерывно растет. Уже описывалось одно применение: оптимизация звонков в компаниях по продвижению товаров.

Другое применение — прием звонков в call-центрах. Если агент принимает звонок от клиента (например, запрос о продукции), агент может рекомендо-

вать дополнительно другие продукты клиенту. Это пример взаимной продажи. В зависимости от реакции клиентов (предложение принято или отклонено) модель обновляется в реальном времени и используется для рекомендаций следующим клиентам. Этот пример иллюстрирует применение *рекомендательной машины* (Recommendation Engine), которая является одной из самых успешных областей Data Mining в реальном времени и будет подробно рассмотрена в следующем разделе.

Другие примеры могут быть найдены, например, в финансовой сфере: автоматические решения относительно закупки или продажи, оптимизация цен и т. д.

# 13.2. Рекомендательные машины

Рекомендательные машины (их также называют рекомендательными системами) являются предшественниками приложений Data Mining в реальном времени. Они представляют клиенту продукты (книги, одежда, товары электроники и т. д.) или общее содержание (музыка, кинофильмы, изображения, добавляемые баннеры, связь и т. д.), которые вероятно его заинтересуют. Особенно широко рекомендательные машины используются в электронной коммерции. Диапазон их применения непрерывно растет, новые области включают стационарную розничную торговлю (PSA/PDA-устройства контроля, весы, терминалы и т. д.), справочные центры, системы самообслуживания и мобильные телефоны.

Все эти области характеризуются тем фактом, что они автоматически представляют (или посылают) рекомендации пользователям и принимают от него обратную связь (непосредственно как Web-магазины или как карточная система). Из-за быстрого роста рынка рекомендательных систем и из-за сложных алгоритмов, требующих обучения в реальном времени, рекомендательные системы стали одной из центральных тем исследования Data Mining.

## 13.2.1. Классификация рекомендательных машин

В настоящее время существует большое число различных подходов к построению рекомендательных машин, что затрудняет их полную характеристику. Необходимо отметить также, что многие из рекомендательных машин основаны на гибридных подходах.

В рекомендательных машинах используются следующие способы сбора данных:

- явный сбор данных — система просит пользователя оценивать пункты или заполнять опросные анкеты;

- неявный сбор данных — система наблюдает за действиями пользователя (щелчки, закупки и т. д.).

Очевидно, неявный сбор данных — самый важный.

Формирование рекомендаций может происходить на основе следующих подходов:

- на основе содержания — рекомендации генерируются на основе изучения содержания элементов (продуктов);
- на основе транзакции — рекомендации генерируются на основе пользовательского поведения.

Исторически первые рекомендательные системы использовали метод *совместного фильтрования* (CF, *см. разд. 13.2.3*), который основывается на транзакциях и является очень популярным на сегодняшний день. Из-за своей популярности совместное фильтрование иногда используется как синоним для всех рекомендательных систем. По аналогии подход на основе содержания иногда называют *фильтрованием на основе содержания*, тогда как подход на основе транзакций ссылается на метод *совместного фильтрования*.

## 13.2.2. Подход на основе содержания

Подход на основе содержания использует такие свойства элементов, как название, описание, цена с тем, чтобы вычислить степень сходства между ними. При этом обычно применяют кластеризацию *(см. гл. 7)* и методы Text Mining *(см. гл. 9)*. В данном случае, если пользователь выбирает некоторый элемент, ему рекомендуются элементы, подобные выбранному.

Успешными примерами использования рекомендательных систем, построенных на основе содержания, являются системы рекомендации музыки, которые рекомендуют песни, основанные на анализе звуковой волны музыки. Известный пример — поиск музыки и рекомендации на сайте MusicLens.

В общем случае рекомендации, построенные на основе содержания, имеют более низкую прогнозирующую способность, чем рекомендации на основе транзакций, которые будут рассмотрены дальше. Однако для новых элементов (которые не имеют никакой большой операционной истории и с которыми не связаны какие-либо транзакции) подобные системы часто бывают очень полезны. Наиболее полезны рекомендации, полученные на основе комбинации двух обозначенных ранее подходов.

## 13.2.3. Совместное фильтрование

Совместное фильтрование (Collaborative Filtering, CF) является одним из первых методов генерации рекомендаций. Первой системой, которая использо-

вала совместное фильтрование, была система Информационный Гобелен (Information Tapestry). Данный проект был разработан Xerox PARC в начале 90-х годов [55].

Совместное фильтрование является методом информационного поиска (IR), который использует набор явно или неявно собранных/полученных пользовательских предпочтений, как меру информационного качества/уместности. В более общем виде CF близок к методу ближайшего соседа.

**Классическая формулировка.** Введем некоторую нотацию. Количество пользователей (обозначенных индексом $u$) — $n$ и количество элементов (например, продукты, обозначенные индексом $i$) — $m$. Матрица $K$ размерностью $n \times m$, где каждая колонка представляет элемент, а каждый ряд — пользователя. В ячейках матрицы содержится величина $r$, отражающая интерес пользователя к элементу (щелчки, закупки, оценки и т. д.). Пример матрицы $K$ показан в табл. 13.1.

**Таблица 13.1.** Пример 4×3 матрицы интересов пользователя

| | Продукт 1 | Продукт 2 | Продукт 3 |
|---|---|---|---|
| Valentina | 3 | 0 | 2 |
| Boris | 0 | 0 | 1 |
| Katya | 2 | 1 | 0 |
| Alexander | 1 | 1 | 1 |

Обозначим набором $N(u,i)$ всех $u$ пользователей, которые проявили интерес к элементу $i$. Тогда предсказание для элемента $i$ вычисляется следующим образом:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in N(u,i)} s_{uv}\left(r_{vi} - b_{vi}\right)}{\sum\limits_{v \in N(u,i)} s_{uv}},$$

где $b_{ui}$ — базовая линия предсказания для $r_{ui}$ (как среднее значение по всем элементам $u$ пользователей).

$s_{uv}$ — мера подобия между пользователями $u$ и $v$. Простейшим выбором меры подобия может являться коэффициент корреляции Pearson и близко связанная мера косинуса.

В данном случае коэффициент Pearson и косинус определяются как

$$s^{(pearson)}{}_{uv} = \frac{< r_u - \bar{r}_u, r_v - \bar{r}_v >}{\left\| r_u - \bar{r}_u \right\| \left\| r_v - \bar{r}_v \right\|}, \quad s^{(\cos ine)}{}_{uv} = \frac{< r_u, r_v >}{\left\| r_u \right\| \left\| r_v \right\|}.$$

Раскроем смысл данной формулы. Для предсказания интереса $r_{ui}$ мы выбираем всех пользователей, которые также выбрали этот элемент $i$, и берем взвешенное среднее число их интересов к элементу $i$, где вес — это их подобие к пользователю $u$. Таким образом, если пользователь $w$ ближе к пользователю $u$, чем другой пользователь $v$, то интерес $r_{wi}$ будет больше, чем интерес $r_{vi}$.

Описанный метод CF дает хорошее предсказание на практике, т. к. принимает во внимание полную пользовательскую историю. Однако он имеет серьезный недостаток — плохую масштабируемость по памяти и по скорости. Фактически, необходимо хранить полную матрицу $K$ в памяти, и вычисление каждой рекомендации требует вычисления подобия всех пользователей $N(u,i)$. Соответственно увеличение числа пользователей и/или числа элементов приводит к возрастанию требований к памяти и замедляет работу алгоритма. По этим причинам, "классический" CF используется только при приемлемом размере матрицы интересов пользователей.

**Применение разложения по особым значениям.** Для того чтобы преодолеть проблемы масштабирования, а также для повышения качества прогнозирования, использование разложения по особым значениям становится все более и более популярными для CF.

*Разложение по особым значениям* (Singular Value Decomposition — SVD) является хорошо известным методом факторизации матрицы, который разбивает факторы матрицы $m \times n$ (для удобства используем транспонированную матрицу $X = K^T$) на три матрицы следующим образом:

$$X = USV^T.$$

Матрица $S$ является диагональной матрицей, содержащей сингулярные значения матрицы $X$. Имеется ровно $l$ сингулярных значений, где $l$ — это ранг $X$. Рангом матрицы является число линейно независимых строк и столбцов в матрице. Строго говоря, разложение по особым значениям является обобщением разложения по собственным значениям, из квадратичной в прямоугольную матрицу.

Интерпретация SVD для CF заключается в следующем: разложение разбивает матрицу интересов в матрицу $U$, которая связывает пользователей со свой-

ствами (например, виртуальные профили), диагональную матрицу $S$, которая содержит веса функций, и матрицу $V$, которая связывает элементы с профилями.

Таким образом, мы можем использовать это разложение, чтобы аппроксимировать исходную матрицу $m \times n$. Взяв $k$ первых сингулярных значений матрицы $S$, мы можем эффективно получить сжатое представление данных:

$$\hat{X} = U_k S_k V_k^T.$$

В большинстве практических приложений $k$ не превышает значение 100. Теперь, если новый пользователь заходит на сайт и проявляет интерес к некоторым его пунктам, он будет представлен своим вектором пользователя $w$, и будут выполнены следующие вычисления:

$$w_k = w^T U_k S_k^{-1},$$

в результате которых будет получен вектор свойств. Теперь мы можем выбрать ближайших пользователей в пространстве свойств и использовать их интересы в качестве предсказаний.

Одной из задач, возникающих при использовании алгоритмов, основанных на SVD, в рекомендательных системах, является высокая стоимость поиска разложения по особым значениям. Хотя это может быть вычислено в автономном режиме, поиск SVD для очень больших объемов данных вычисления может быть весьма трудоемкими. Для решения этой проблемы, ряд исследователей изучили дополнительные методы обновления имеющегося SVD без повторного ее вычисления с нуля. Эти методы позволяют использовать SVD в задаче обучения в реальном времени.

Подводя итог, необходимо сказать, что SVD является очень мощным методом, который не только снижает сложность вычисления, но и обеспечивает большее понимание поведения пользователей, автоматически извлекая "виртуальные профили".

**Поэлементное совместное фильтрование.** Вместо расчета сходства между пользователями, можно также рассчитывать сходство между элементами. Такой подход называется *поэлементной совместной фильтрацией.*

Пусть $N(u,i)$ будет набором $i$ всех элементов, которые были выбраны пользователем $u$. Тогда предсказание рассчитывается следующим образом:

$$\hat{r}_{ui} = c_{ui} + \frac{\sum_{j \in N(i,u)} s_{ij} \left( r_{uj} - c_{uj} \right)}{\sum_{j \in N(i,u)} s_{ij}},$$

где $c_{ui}$ — базовая величина для прогнозирования $r_{ui}$ (например, средняя величина по всем пользователям по пункту $i$), а $s_{ij}$ является мерой сходства между пунктами $i$ и $j$.

Для прогнозирования незамеченных интересов $r_{ui}$ мы выбираем все элементы, которые также были отобраны $u$ пользователям, и принимаем взвешенное среднее значение своих интересов к $i$, где веса — их сходство с элементом $i$. Таким образом, если пункт $j$ более тесно связан с элементом $i$, чем с другим элементом, то проценты $r_{uj}$ будут выше, чем весовые проценты $r_{uk}$.

## 13.2.4. Анализ рыночной корзины и секвенциальный анализ

В *гл. 6* были представлены алгоритмы поиска ассоциативных правил и секвенциального анализа. Результирующие правила имеют вид $X \Rightarrow Y$, где $X$ и $Y$ — наборы элементов. Например, правило в следующем виде

$$X = \{i_1, i_2\} \Rightarrow Y = \{i_3, i_4\}$$

означает, что пользователи, которые обычно выбирают элементы $i_1$ и $i_2$, обычно также выбирают $i_3$ и $i_4$.

Проблема заключается в том, что обычно имеется слишком много объектов (в интернет-магазинах количество объектов колеблется от нескольких тысяч до нескольких миллионов), поэтому вероятность нахождения правила для множества объектов в данном предположении очень мала, даже если у нас есть миллион правил. Например, если у нас есть 10 тысяч объектов, нам необходимо было бы 100 миллионов правил для покрытий всех предположений, содержащих 2 элемента $X = \{i_1, i_2\}$. Для трех наименований $X = \{i_1, i_2, i_3\}$ нам потребовалось бы более 1 триллиона правил и т. д.

Таким образом, для большей части приложений используются простые правила вида (с очень низким значением поддержки):

$$X = \{i_1\} \Rightarrow Y = \{i_2\}.$$

Данные правила очень легки для вычислений и хорошо работают в механизмах рекомендаций. Они позволяют также упростить алгоритм совместной фильтрации, который очень популярен для данной задачи.

## 13.2.5. Усиление обучения и агенты

Во всех подходах, которые были рассмотрены ранее, основная идея заключалась в том, чтобы предсказать следующий элемент так, как если бы пользова-

тель выбрал его сам, а затем рекомендовать этот или схожий элемент. Практика показала, что данная стратегия действительно эффективна.

Однако может возникнуть вопрос: действительно ли данная стратегия самая лучшая? Может быть, было бы лучше предложить совершенно другой элемент, такой, что ни сам пользователь, ни другие подобные пользователи никогда не видели. Кроме того, предсказание всегда делается только для одного шага. И поэтому опять возникает вопрос, может быть, было бы лучше прогнозировать полную последовательность следующих переходов пользователя и оптимизировать рекомендации для всех следующих шагов?

Эти вопросы изучения максимизации задачи бизнеса для всего множества шагов путем последовательных итераций относятся к области *усиления обучения* (Reinforcement Learning — RL). Усиление обучения может рассматриваться как адаптивная оптимизация последовательности задач Data Mining.

Усиление обучения рассматривается как набор состояний, в которых можно выполнять действия, и каждое действие переводит систему в новое состояние, получая некоторое значение — вознаграждение (цена перехода). Сумма всех вознаграждений должна быть максимизирована. Для примера рассмотрим механизм рекомендаций интернет-магазина. Каждому состоянию соответствует страница продукта, и каждое действие есть рекомендованный продукт. Вознаграждением является, например, цена продукта или просто единица (если мы хотим максимизировать время посещения магазина). Таким образом, мы хотим максимизировать доход клиента или деятельность, выраженную последовательностью переходов.

Усиление обучения является молодой наукой. Классические прикладные области RL включают в себя управление мобильными роботами или играми, такими как Blackjack или шахматами. В случае мобильных роботов состоянием является местонахождение робота, а действиями — например, новое направление и скорость. Вознаграждение равно –1 для каждого шага, и большому отрицательному числу, если он движется против барьера. Для игр состоянием является текущая позиция, а действием — новое движение, вознаграждение обычно равно 0 до конечного состояния, для которого вознаграждение равно +1 в случае победы, –1 в случае поражения и 0 в случае ничьи.

Далее мы представим короткий обзор RL. Для более детального ознакомления рекомендуем книгу Sutton и Barto [56].

**Формулировка задачи усиления обучения.** На рис. 13.2 показано взаимодействие в терминах агента и среды, которое используется в RL.

Агент получает новое состояние и вознаграждение из среды, принимает решение о соответствующем действии (обычно включая изучение) и воздействует на среду. Среда отвечает на действие, переводя агента в новое состояние

и передавая ему соответствующее вознаграждение и т. д. Далее будем различать эпизодические задачи, которые завершаются (например, игры, завершающиеся победой или поражением), и продолжающиеся задачи, не имеющие завершения (например, передвижение робота, который может перемешаться бесконечно).

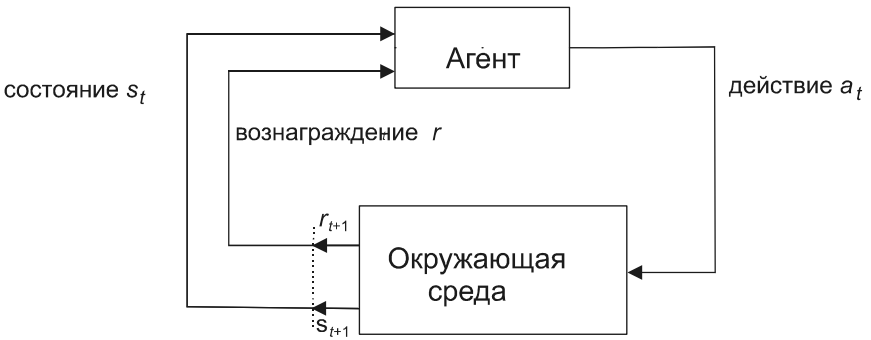



**Рис. 13.2.** Взаимодействие агента и среды в RL

Для большинства важных случаев усиление обучения основано на методах динамического программирования (ДП).

Уравнение Белмана для дискретного случая представлено далее [56]:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathrm{P}^{a}_{ss'} \left[ R^{a}_{ss'} + \gamma V^{\pi}(s') \right], \qquad (13.1)$$

где $s$ — состояние из множества всех состояний $S$;

$a$ — действие из множества действий $A(s)$, доступных в состоянии $s$;

$\pi(s,a)$ — требуемая стратегия, т. е. стохастический выбор действия $a$ в состоянии $s$;

$\mathrm{P}^{a}_{ss'}$ — вероятности перехода из состояния $s$ в состояние $s'$ при выполнении действия $a$;

$R^{a}_{ss'}$ — соответствующее вознаграждение перехода;

$V^{\pi}(s)$ — функция оценки состояния, которая связывает ожидаемое значение кумулятивного вознаграждения для каждого состояния $s$ с учетной ставкой $\gamma$ для нагрузки влияния последующих вознаграждений.

Кроме функции оценки состояния $V^{\pi}(s)$ в RL часто используется функция оценки действия $Q^{\pi}(s,a)$, которая определяет ожидаемое кумулятивное воз-

награждение для каждого состояния $s$ при выборе действия $a$. Связь между функциями оценки состояний и оценки действий выглядит так:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) Q^{\pi}(s,a), \quad Q^{\pi}(s,a) = \sum_{s'} \mathrm{P}^{a}_{ss'} \left[ R^{a}_{ss'} + \gamma V^{\pi}(s') \right],$$

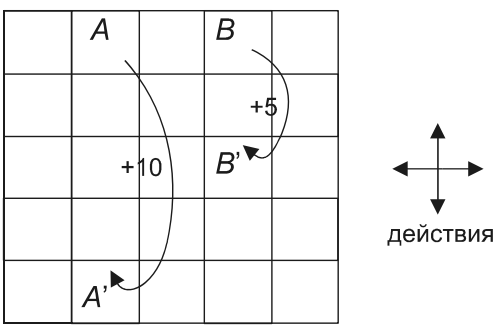
и их комбинация в соответствии с функцией оценки действия приводит к уравнению Белмана (13.1). Если заменить функцию оценки состояния, то получим уравнение Белмана для функции оценки действия:

$$Q^{\pi}(s,a) = \sum_{s'} \mathrm{P}^{a}_{ss'} \left[ R^{a}_{ss'} + \gamma \sum_{a'} \pi(s',a') Q^{\pi}(s',a') \right].$$

Для нахождения оптимальной функции оценки состояний $V(s)$ и оптимальной стратегии $\pi(s,a)$ нам необходимо решить уравнение Белмана (13.1).

**Пример.** Для иллюстрации уравнения Белмана, мы будем использовать пример Gridworld из [56], представленный на рис. 13.3 (*а*).

Ячейки сетки соответствуют состояниям среды. Из каждой ячейки возможно 4 действия (перехода): вверх, вниз, влево, вправо. Данные действия полностью определяют передвижение агента из заданной ячейки в соответствующем направлении по сетке. Действия, которые приводят агента за пределы сетки, не изменяют его местонахождение, при этом результат вознаграждения будет равен –1. Другие действия выполняются с вознаграждением 0, за исключением тех, которые выводят агента из особенных состояний $A$ и $B$. При переходе из состояния $A$, все четыре действия получают вознаграждение +10 и переводят агента в состояние $A'$. При переходе из состояния $B$, все действия получают вознаграждение +5 и переводят агента в состояние $B'$.




*а*

| | | | | |
|---|---|---|---|---|
| 3.3 | 8.8 | 4.4 | 5.3 | 1.5 |
| 1.5 | 3.0 | 2.3 | 1.9 | 0.5 |
| 0.1 | 0.7 | 0.7 | 0.4 | -0.4 |
| -1.0 | -0.4 | -0.4 | -0.6 | -1.2 |
| -1.9 | -1.3 | -1.2 | -1.4 | -2.0 |

*б*

**Рис. 13.3.** Пример сетки: *а* — ожидаемая динамика вознаграждений; *б* — функции оценки вознаграждения для равновероятных случайных стратегий

Предположим, что агент выбирает все четыре действия с равной вероятностью в каждом состоянии. Рис. 13.3 (*б*) показывает функцию значений $V^{\pi}(s)$ для данной стратегии в случае, где $\gamma = 0.9$.

Чтобы доказать, что функция оценки состояния, приведенная на рис. 13.3 (*б*), корректна, рассмотрим, например, верхнюю левую ячейку и вычислим функцию оценки состояния:

$$\begin{aligned} V^{\pi}(s_0) &= \sum_a \pi(s_0,a)\sum_{s'} \mathrm{P}^a_{s_0 s'}\left[R^a_{s_0 s'} + 0.9\cdot V^{\pi}(s')\right] \\ &= \sum_{i=1}^{4} \pi(s_0,a_i)\cdot 1\cdot \left[R^{a_i}_{s_0 s'} + 0.9\cdot V^{\pi}(s')\right]_{s'=s_0+a_i} \\ &= \frac{1}{4}\sum_{i=1}^{4} \left[R^{a_i}_{s_0 s'} + 0.9\cdot V^{\pi}(s')\right]_{s'=s_0+a_i} \\ &= \frac{1}{4}\left\{\left[-1+0.9\cdot 3.3\right]+\left[-1+0.9\cdot 3.3\right]+\left[0+0.9\cdot 8.8\right]+\left[0+0.9\cdot 1.5\right]\right\} \\ &= 3.3 \end{aligned}$$

Теперь расширим дискретный случай на непрерывный, рассмотренный в [57]. Пусть $s(t)$ есть состояние, а $a(t)$ — действие (или управление) в момент времени $t$

$$\frac{ds(t)}{dt} = g(s(t), a(t)),$$

где $g$ — *динамика состояния.* Для начального состояния $s_0$ выбор действия $a(t)$ приводит к единственной траектории $s(t)$. Далее $r(s,a)$ — функция вознаграждения. Для простоты будем считать $\pi(s)$ детерминированной стратегией, которая назначает единственное действие $a$ состоянию в момент времени $t$: $a(t) = \pi(s(t))$.

Непрерывное к уравнению Белмана (13.1) уравнение Гамильтон — Якоби — Белмана выглядит так:

$$V^{\pi}(s)\ln\gamma + \nabla V^{\pi}(s)\cdot g(s,\pi(s)) + r(s,\pi(s)) = 0.$$

**Пример.** Рассмотрим проблему контроля горного автомобиля (рис. 13.4), сформулированную в литературе. Данная проблема является двухразмерной (скорость и направление скорости), поэтому $s \in R^2$.

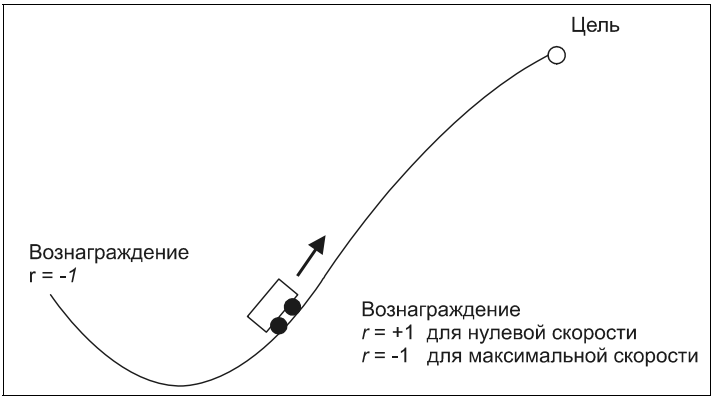



**Рис. 13.4.** Проблема контроля горного автомобиля

Автомобиль должен достигнуть вершины горы как можно быстрее и остановиться там. Автомобиль не может подниматься вверх без начальной скорости. Поэтому он должен получить ускорение перед началом движения. Также он должен быть осторожен, чтобы не попасть за левую границу. Функция вознаграждения $r(s,a)$ равна нулю везде, кроме границы, где функция равна –1 слева, и линейно изменяется от –1 до +1 в зависимости от ускорения автомобиля, когда он есть справа. В данном случае возможны два действия: максимально положительный или отрицательный толчок.

В оставшейся части данной главы будет дискретное уравнение Белмана (13.1). Непрерывный случай рассмотрен в [57].

**Общее решение задачи усиления обучения.** Чтобы решить задачу RL, рассмотрим оптимальную стратегию $\pi^*$ и соответствующую ей оптимальную функцию оценки состояний $V^*$. Значения функции определяют частичную упорядоченность над стратегиями. Стратегия $\pi$ определяется как лучшая или равная стратегии $\pi'$, $\pi \geq \pi'$ тогда и только тогда, когда $V^{\pi}(s) \geq V^{\pi'}(s), \forall s \in S$. Всегда существует, по крайней мере, одна стратегия, которая лучше или равна всем другим стратегиям. Это оптимальная стратегия $\pi^*$. Все самые лучшие стратегии одной и той же оптимальной функции оценки состояния $V^*$, определенной как:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \forall s \in S.$$

Как найти оптимальные функции оценки состояния и стратегии? В динамическом программировании для решения уравнения Белмана существует два базовых алгоритма: алгоритм итерации по стратегиям и по значениям.

Для RL, который охватывает не только задачи динамического программирования, идея итераций по стратегиям была обобщена *алгоритмом итерации*

(General Policy Iteration — GPI) *по стратегиям*, который проиллюстрирован на рис. 13.5.

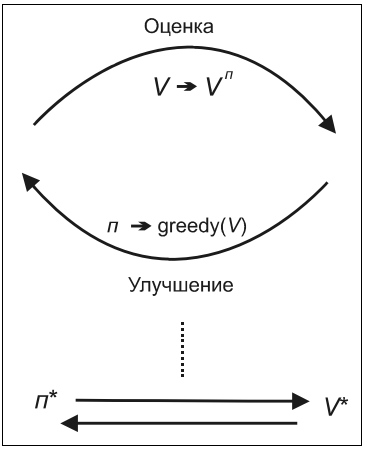



**Рис. 13.5.** Обобщенный алгоритм итерации по стратегиям, функции значений и стратегий

Отметим, что "жадная" стратегия — это стратегия, которая в каждом состоянии $s$ выбирает наилучшее действие в соответствии с $Q^{\pi}(s,a)$. Поэтому подход обобщенного алгоритма итераций по стратегиям заключается в следующем: стратегию всегда можно улучшить относительно функции значений *(совершенствование стратегии)*, а функцию значений — относительно функции значения стратегии *(оценка стратегии)*. Практически все методы RL хорошо описаны как GPI.

Совершенствование стратегий часто заключается просто в вычислении "жадной" стратегии. Алгоритм оценки стратегии основан на том факте, что функция оценки состояния $V^{\pi}$ предшествует неподвижной точке оператора Беллмана $T_{\pi}$:

$$(T_{\pi}\ V)(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathrm{P}^{a}_{ss'} \left[ R^{a}_{ss'} + \gamma V\ (s') \right],$$

и поэтому решение может быть получено итеративно (алгоритм итераций по значениям).

Часто функция оценки состояний слишком громоздка для табличного представления. В таких случаях используются методы аппроксимации. Соответствующий алгоритм *аппроксимации итерации по значениям* (Approximation Value Iteration — AVI) определяется как следующий итерационный метод [58]:

$$V_{n+1} = \mathrm{AT}_{\pi} V_n, \tag{13.2}$$

где A — оператор аппроксимации, обычно регрессионный оператор *(см. гл. 5)*. При этом для табличного представления $V$ не нужен оператор приближения A (то есть $A = I$).

Таким образом, из равенства (13.2) следует, что алгоритм регрессионного Data Mining используется в RL.

В этой связи мы подходим к приближенным алгоритмам итерации по стратегиям. Существует много типов алгоритмов итераций по стратегиям и значениям.

**Методы решения Reinforcement Learning.** В RL существует три основных типа решений задачи RL.

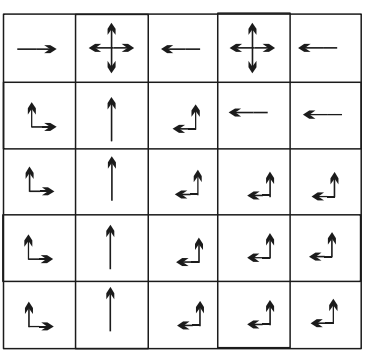
- Динамическое программирование (ДП).

  Если вероятности перехода $P^a_{ss'}$ и вознаграждений $R^a_{ss'}$ известны или могут быть оценены (например, с помощью Data Mining), тогда можно считать, что модель среды известна. В этом случае, используя методы итераций по стратегиям и значениям, мы можем явно определить оптимальные стратегии и решение функции оценки состояния (13.1).

  *Пример.* Для таблицы, приведенной на рис. 13.3, оптимальная функция значения представлена на рис. 13.6 (*а*), а на рис. 13.6 (*б*) показаны соответствующие оптимальные стратегии. Несколько стрелок в одной ячейке означает, что любое соответствующее действие является оптимальным.

| | | | | |
|---|---|---|---|---|
| 22.0 | 24.4 | 22.0 | 19.4 | 17.5 |
| 19.8 | 22.0 | 19.8 | 17.8 | 16.0 |
| 17.8 | 19.8 | 17.8 | 16.0 | 14.4 |
| 16.0 | 17.8 | 16.0 | 14.4 | 13.0 |
| 14.4 | 16.0 | 14.4 | 13.0 | 11.7 |

*а*



*б*

**Рис. 13.6.** Оптимальное решение для примера Сетки: *а* — оптимальная функция оценки состояния $V^*$; *б* — оптимальная стратегия $\pi^*$

- Методы Монте — Карло (МК).

  В отличие от случая с ДП, в данном случае информация о среде отсутствует, т. е. модель среды неизвестна. Методы Монте — Карло требуют

только данных, полученных опытным путем или путем имитационного моделирования в среде — примеры последовательности состояний, действий и вознаграждений.

Методы Монте—Карло являются способом решения проблемы RL путем последовательных усреднений с возвращением. Поэтому они являются полным обобщением классических методов Монте—Карло в случае нескольких шагов. Методы Монте—Карло очень просты и легки в применении. В основном они могут быть определены только для нерегулярных задач. Основным недостатком методов Монте—Карло является то, что для обновления стратегии и функции значений необходимо дождаться окончания выполнения этого эпизода.

- Обучение временного отличия Temporal-Difference Learning (TD).

  TD-обучение является комбинацией идей ДП и МК. Так же как и методы МК, методы TD могут получать знания прямо из сырых данных без модели среды. Так же как и методы ДП, TD обновляют оценки, опираясь частично на другие полученные оценки, не дожидаясь завершения. Главным образом, они улучшают свои стратегии постоянно в течение выполнения эпизодической задачи, и поэтому данный подход является более гибким, чем методы Монте—Карло и с точки зрения оценки, и с точки зрения управления.

  Методы TD очень популярны среди методов RL. Существует огромное количество методов TD, большинство которых очень просты для реализации. Самым важным является алгоритм TD($\lambda$) со следами преемственности, которые в равной степени используют подходы методов МК и TD.

Заметим, что существует огромное число алгоритмов, сочетающих алгоритмы DP, MC и TD.

Книга [56] полностью посвящена этой проблеме. Библиотека Xelopes *(см. гл. 11)* также реализует полный пакет алгоритмов RL.

**Применение к рекомендательным системам.** Очевидно, что усиление обучения хорошо укладывается в концепцию рекомендательных систем.

Вернемся к начальному примеру: каждому состоянию соответствует страница продукта, а каждому действию — рекомендуемый продукт. На рис. 13.7 показано взаимодействие рекомендательной системы и пользователя, применяющего эти рекомендации в интернет-магазине, в котором продаются товары. Самые лучшие проверенные рекомендации помечены звездочкой (*).

На первом и третьем шагах механизм рекомендации следует проверенным рекомендациям; на втором шаге проверяется новая рекомендация. Пользователь пропускает первую рекомендацию, но на втором и третьем шагах при-

нимает рекомендации к рассмотрению. Обратные стрелки указывают на обновление рекомендаций.

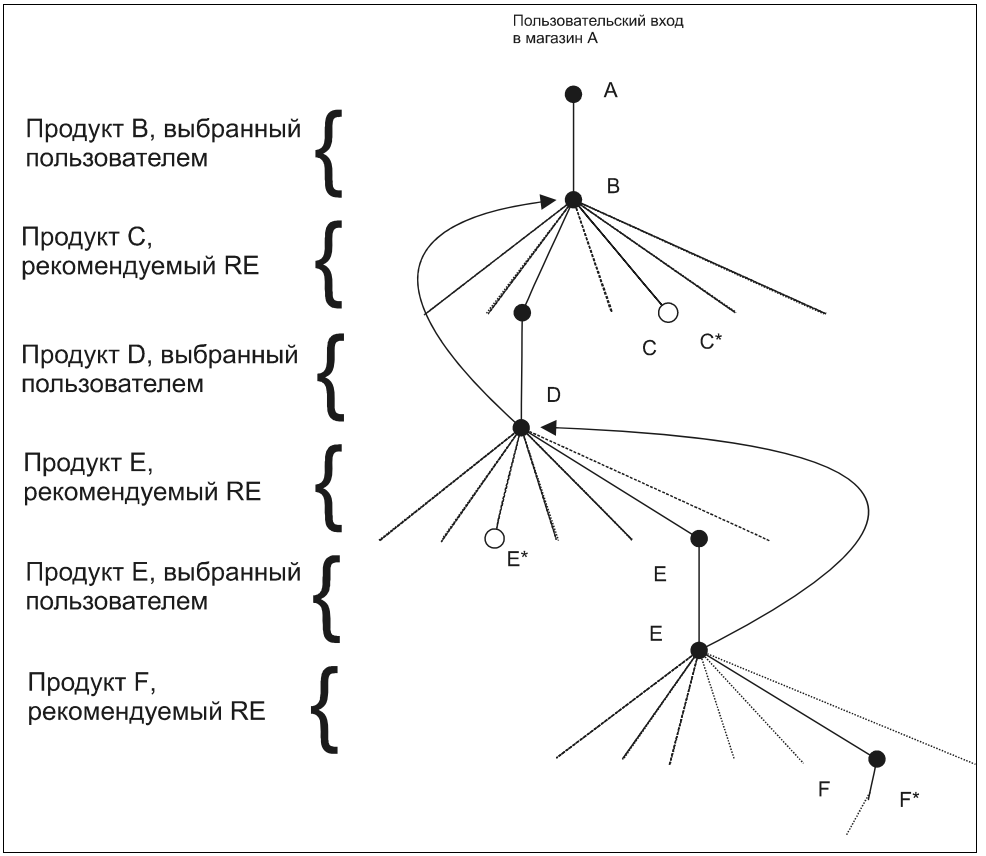



**Рис. 13.7.** Процесс усиления обучения рекомендательной машины

Безусловно, это очень простая иллюстрация, и для каждого шага можно добавить намного больше информации, например, информацию о пользователях, категориях, каналах и историю операций.

Несмотря на достоинства RL, на практике при применении возникает много серьезных проблем. Во-первых, RL выполняет изучение относительно медленно, так что для большинства элементов требуется большое число операций, чтобы получить улучшение. Следующая проблема заключается в разработке робастной приближенной схемы для последовательной задачи регрессии (13.2). Чтобы решить эти проблемы, в последние годы была представлена концепция иерархического усиленного обучения (Hierarchical Reinforcement Learning), которая вносит больше концептуальной структуры в RL.

# 13.3. Инструменты Data Mining в реальном времени

## 13.3.1. Инструмент Amazon.com — механизм рекомендаций

Amazon.com — пионер в технологии рекомендаций. Он помещает рекомендации различных типов во многих местах магазина. Примерами являются рекомендации продуктов, основанные на закупках ("Покупатели, купившие этот товар, купили также..."), щелчках мыши ("Покупатели, которые просмотрели этот пункт, смотрели также..."), смешанные ("Что в конечном счете приобрели покупатели, просмотревшие этот пункт?"), и индивидуальные рекомендации ("Рекомендация для Вас").

Изначально Amazon.com использовал систему совместной фильтрации (Collaborative Filtering) *BookMatcher*. Однако она не очень хорошо работала, т. к. требовала более 20 оценок, перед тем как выдать рекомендацию, и имела проблемы с вычислениями. Позднее использовался более простой, но грубый алгоритм совместной фильтрации *Item-to-Item*, который оказался очень удачным. Между тем Amazon.com использует более комплексные стратегии, включающие множество критериев, таких как закупки, щелчки мыши, пользовательские оценки и свойства книг.

## 13.3.2. Инструмент Prudsys — рекомендательная машина Prudsys

В этом разделе мы представляем механизм рекомендаций, основанный на усилении обучения — рекомендательная система Prudsys (Prudsys Recommendation Engine — Prudsys RE).

Prudsys RE — это инструмент для анализа и автоматизации рекомендаций продукта в точке продажи (point of sales — POS). Между тем, это один из самых удачных коммерческих механизмов рекомендаций, и он используется такими торговыми марками, как Coop, Quelle, Freemans, Karstadt, OTTO, MEXX, Metro, Baur и Home Shopping Europe. Кроме того, Prudsys предоставляет портал IREUS, где небольшие компании могут включить Prudsys RE в свои интернет-магазины в качестве платформы сервиса приложения (Application Service Platform — ASP) или SaaS-решения.

Prudsys RE разработан на единой основе — библиотеке Xelopes *(см. гл. 11)*, что позволило существенно расширить его возможности: инструмент не только предоставляет рекомендации, но также выполняет классификацию покупателей, оптимизацию цен, симуляцию и даже составление отчетов.

Таким образом, Prudsys RE можно считать вторым уровнем над Xelopes, главным образом, представляющим бизнес-логику. Основная архитектура Prudsys RE изображена на рис. 13.8.

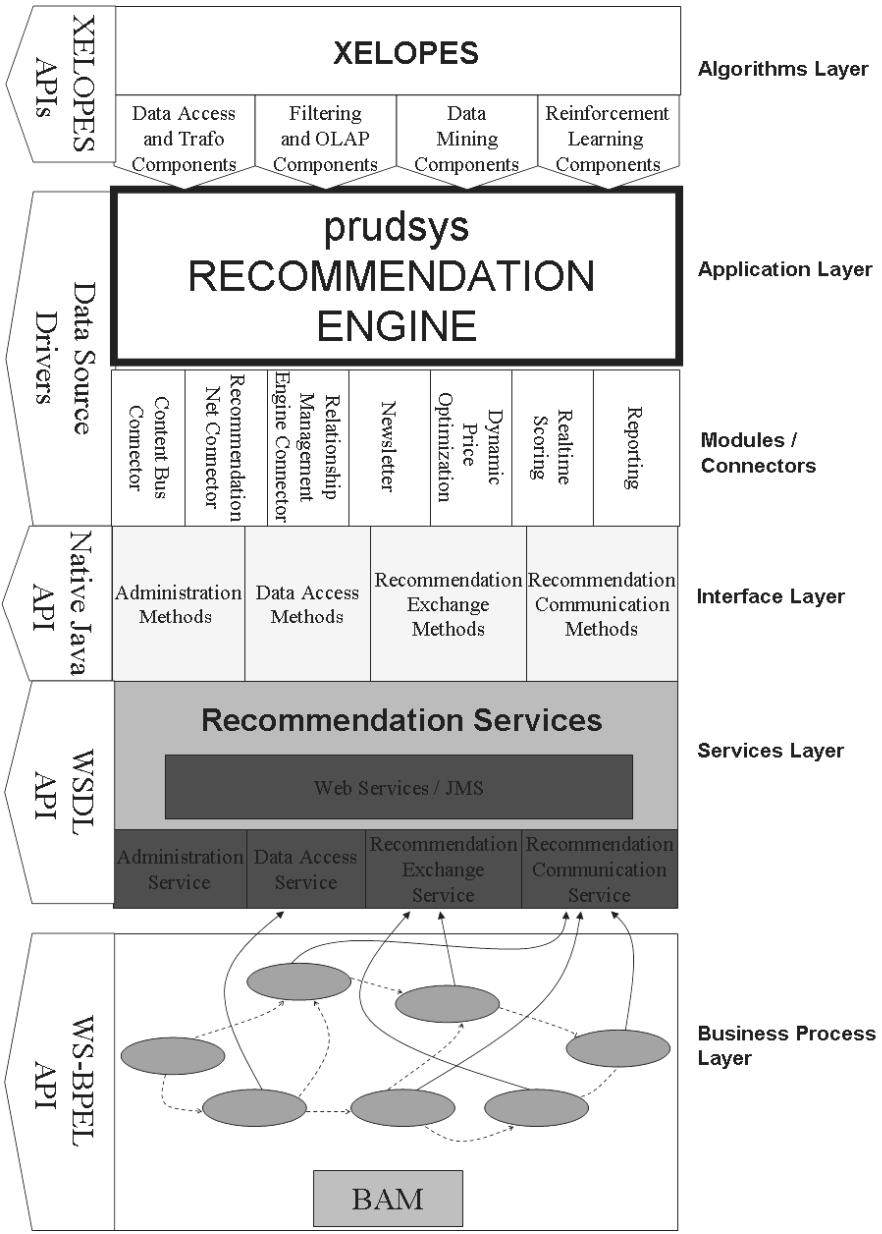



**Рис. 13.8.** Основная архитектура Prudsys RE

Prudsys RE предоставляет всесторонний интерфейс, реализованный через Java API. Он структурирован в четыре базовых блока, которые используются для создания сервисов на сервисном уровне. *Сервис администрирования* (Administration Service) служит для управления, конфигурирования и мониторинга Prudsys RE. *Сервис доступа к данным* (Data Access Service) используется для доступа к стационарным данным (размерным данным и данным по истории транзакций). *Сервис обмена рекомендациями* (Recommendation Exchange Service) позволяет получать модели и обмениваться моделями во время рабочего цикла главным образом между несколькими RE. *Сервис коммуникации рекомендаций* (Recommendation Communication Service) наиболее важен, он поддерживает online-коммуникацию.

Prudsys RE поддерживает как offline-, так и online-режимы. В режиме offline анализируются данные по истории транзакций, предоставленные сервисом доступа к данным (Data Access Service), генерируются модели и экспортируются как PMML *(см. разд. 10.4)* или правила (только для моделей баз правил) в файлы или базы данных. Модули действующих систем читают модель и применяют ее для рекомендаций. В режиме online после каждой транзакции вызывается метод коммуникации RE (через сервис коммуникации рекомендаций), он обновляет модель и возвращает рекомендации. В большинстве случаев online- и offline-режимы комбинируются: сначала используются исторические данные для генерации первичной модели в режиме offline, затем RE постоянно совершенствует модель через взаимодействие с пользователем в режиме online.

В завершение главы рассмотрим механизм рекомендаций. Prudsys RE может работать как автономная программа. Prudsys RE имеет интегрированный Web-сервер, таким образом возможна коммуникация в режиме online с помощью Web-сервисов. Кроме того, благодаря использованию Java API Prudsys RE может быть напрямую подключен в качестве библиотеки в приложения третьих лиц.

Это приводит к созданию весьма интересных типов приложений с RE: создается множество экземпляров RE, некоторые зачастую используются для обучения, а некоторые — для реализации задач приложения. Они обмениваются своими моделями через PMML или удаленную коммуникацию (Web-сервисы, JMI и т. д.). Java API предлагает доступ к моделям RE (как Xelopes `MiningModel`). Средствами Xelopes модели корректируются, анализируются, изменяются и перераспределяются за рабочий цикл. Следует отметить, что распределенные приложения Prudsys RE очень эффективны на практике. Такие комплексные приложения требуют строгой унификации и стандартизации, которая обеспечивается библиотекой Xelopes.

### 13.3.3. Приложение с открытым кодом — SpamAssassin

Проблема нежелательной электронной почты, зачастую сомнительного содержания, широко известна, и ситуация кажется безнадежной для получателей. Несмотря на соответствующие законы, их эффективное и законное осуществление вряд ли представляется возможным.

Люди, часто общающиеся посредством электронной почты, и, как результат, имеющие широкий диапазон контактных адресов, зачастую получают гораздо больше рекламных писем, чем реальных.

Получатели сталкиваются с проблемой разделения важной почты от спама, которую часто трудно распознать с первого взгляда.

Последние годы желание распознавать спам и автоматически отсеивать такую почту побудило к созданию различных программных проектов и коммерческих продуктов, снабженных в большей или меньшей степени "искусственным интеллектом" с установленным диапазоном эффективности. Одним из популярных почтовых фильтров является приложение с открытым кодом (Open Source) SpamAssassin, которое уже доказало свою эффективность во многих университетах и компаниях.

Ключевым компонентом в любом хорошем почтовом фильтре является его способность классифицировать почту на базе различных измеренных и синтезированных свойств (атрибутов), и таким образом корректно отличать спам от не спама с высокой вероятностью. Несомненно, качество почтового фильтра напрямую зависит от качества его классификационного алгоритма.

SpamAssassin использует статистическую фильтрацию, поддерживаемую внешними программами и online базами данных. SpamAssassin по умолчанию пытается усилить свои собственные правила с помощью Байесовского обучения (Bayesian learning) *(см. разд. 5.3.2)*, но Байесовское обучение наиболее эффективно при реальном взаимодействии с пользователем. Типично ожидается, что пользователь предоставит примеры почты-спама и полезной почты фильтру, который затем сможет изучить разницу между ними. Для этой цели SpamAssassin предоставляет инструмент командной строки sa-learn, через которую можно показать программе, что единичное письмо или все содержимое почтового ящика является спамом или полезной почтой.

Обычно пользователь будет перемещать нераспознанный спам в отдельную папку на какое-то время, а затем запускать sa-learn для папки с не спамом и спамом отдельно. В качестве альтернативы, если почтовый агент пользователя поддерживает это, то sa-learn может быть запущен для каждого письма в отдельности. Независимо от метода, используемого для выполнения обучения, SpamAssassin Байесовский тест позже назначит более высокий балл тем

письмам, которые похожи на ранее полученный спам (или, если быть более точными, тем письмам, которые отличаются от не спама таким же образом, как и ранее полученные спам-сообщения).

# Выводы

По результатам данной главы можно сделать следующие выводы.

- Методы Data Mining в реальном времени (или Real-Time Analytics) в отличие от статических методов обучаются динамически и основаны на обратной связи от прогноза, полученного с помощью предсказательной модели (постоянном переобучении).
- Адаптация моделей Data Mining использует подход, применяемый в математике, но в отличие от математики в бизнесе требуется не только создать точную модель, но и достичь бизнес-цели.
- Достоинства адаптивного подхода: требуется меньше статистических предположений для алгоритмов Data Mining; модели быстро приспосабливаются к изменяющейся окружающей среде; системы могут систематически собирать новые данные.
- Недостатки адаптивного подхода: требуется обширная IT-инфраструктура, особенно для осуществления обратной связи; требуются более сложные математические инструменты (нестационарные процессы).
- Рекомендательные системы — предшественники приложений Data Mining в реальном времени, представляют продукты или общее содержание, которые, вероятно, заинтересуют пользователя.
- Рекомендательные системы можно классифицировать по способу сбора данных: явный сбор данных и неявный сбор данных.
- Совместное фильтрование является методом информационного поиска (IR), который использует явно или неявно собранный/полученный набор пользовательских предпочтений как меру информационного качества/уместности.
- Усиление обучения рассматривается как набор состояний, в которых можно выполнять действия, и каждое действие переводит систему в новое состояние, получая некоторое значение — вознаграждение (цена перехода).
- Для усиления обучения используется модель агентов и среды, взаимодействующих друг с другом и влияющих друг на друга.
- В настоящее время наиболее популярными рекомендательными системами являются коммерческие рекомендательные системы Amazon.com, ProdSys RE и система с открытым кодом SpamAssassin.

ГЛАВА 14

# Извлечение знаний из Web — Web Mining

## 14.1. Web Mining

### 14.1.1. Проблемы анализа информации из Web

Всемирная паутина WWW в настоящее время является наиболее богатым источником информации и знаний. Она содержит огромное количество документов, данных, аудио- и видеофайлов. Однако пользователи Интернета сталкиваются с большими проблемами не только при анализе, но и при поиске нужной им информации. Выделяют следующие проблемы работы с информацией из Web.

- **Поиск значимой информации.** Пользователи в поиске информации могут самостоятельно перемешаться от сайта к сайту или пользоваться популярными в настоящее время поисковыми системами. Последние по введенным ключевым словам предоставляют списки ссылок на страницы, на которых представлена информация, соответствующая введенным ключевым словам. Однако использование поисковых систем имеет следующие проблемы [60]:
  - небольшой процент действительно нужной информации среди множества ссылок, которые предоставляют поисковые системы;
  - низкая повторяемость вызовов, связанная с невозможностью индексировать все Web-ресурсы. В результате возникают трудности поиска неиндексированной информации, которая может быть необходима для пользователя.
- **Создание новых знаний вне информации, доступной на Web.** Эта проблема может рассматриваться как часть проблемы поиска значимой информации. Она возникает уже после выполнения поиска информации и

связана с извлечением полезных знаний из того множества информации, которое было найдено поисковой системой по запросу пользователя.

- **Персонализация информации.** Данная проблема связана с типом и представлением информации в зависимости от смысла, вкладываемого пользователем (подобно тому, как люди отличают контекст и выдвигают предположения в процессе взаимодействия с Web).
- **Изучение потребителя или индивидуального пользователя.** Эта проблема связана с предоставлением пользователю именно той информации, которую он хочет получить. Для этого требуется настройка и персонализация поисковой системы для конкретного потребителя или пользователя.

## 14.1.2. Этапы Web Mining

Для решения перечисленных проблем используются различные технологии, напрямую или косвенно разрешающие их. К таким технологиям относятся: базы данных, информационный поиск, обработчики естественных языков и др. Технология Web Mining также направлена как на прямое, так и на косвенное решение перечисленных проблем.

*Web Mining — технология, использующая методы Data Mining для исследования и извлечения информации из Web-документов и сервисов [61].*

Выделяют следующие этапы применения Web Mining:

1. Поиск ресурсов — локализация неизвестных документов и сервисов в Web.
2. Извлечение информации — автоматическое извлечение определенной информации из найденных Web-ресурсов.
3. Обобщение — обнаружение общих шаблонов в отдельных и пересекающихся множествах сайтов.
4. Анализ — интерпретация найденных шаблонов.

Поиск ресурсов предполагает поиск различных Web-источников (преимущественно текстовых) по ключевым словам. Данный этап разделяют на два класса: поиск документов и поиск сервисов.

Большинство работ по поиску ресурсов сводится к автоматическому созданию поисковых индексов Web-документов. Для этих целей создавались роботы, индексирующие слова в документах и хранящие вычисленные индексы для дальнейшего их использования при обработке запросов пользователей. Наиболее популярными роботами считаются WebCrawler и AltaVista. Они способны сканировать миллионы документов и хранить индексы слов в этих документах. Существует много различных индексов, которые в настоящее время активно используются.

После того как ресурсы найдены, из них должна быть извлечена информация, подвергаемая анализу. Часто этот этап называют *препроцессинг*, т. к. он заключается в подготовке найденных ресурсов непосредственно к анализу. Такая подготовка заключается в преобразовании текстов, путем удаления стоп-слов, стеммингов, извлечением фраз и словосочетаний и т. п. Другими словами, результатом данного этапа должна быть информация, пригодная для анализа.

На этапе обобщения к обработанной информации применяются методы Data Mining. На этом этапе важную роль играет человек, учитывая также тот факт, что на последнем этапе он должен будет интерпретировать полученные результаты.

### 14.1.3. Web Mining и другие интернет-технологии

Web Mining, являясь инструментом для обработки и анализа Web-ресурсов, рассматривается в одном ряду с такими интернет-технологиями, как получение информации (Information Retrieval — IR) и извлечение информации (Information Extraction — IE). Однако, имея с ними много общего, Web Mining имеет также существенные отличия. Рассмотрим некоторые из них.

Технология IR заключается в получении документов из Web-среды, релевантных запросу пользователей (от англ. *Relevant — относящийся к делу*, что означает соответствие найденного ответа запросу, сделанному пользователем поисковой системы). При этом очень часто полученные документы включают в себя как релевантные, так и нерелевантные документы. Как уже упоминалось, для решения этой задачи строятся поисковые индексы. Для их построения используются различные методы, включающие в себя моделирование, классификацию и кластеризацию документов, фильтрацию и др. При этом для классификации и кластеризации документов могут быть использованы методы Data Mining (точнее Text Mining). С этой точки зрения можно говорить, что Web Mining является частью технологии IR.

Целью IE является извлечение необходимых фактов из Web-документов. Основное отличие этой технологии от IR заключается в том, что она работает с самим документом и ищет в нем релевантную информацию, в то время как IR работает с множеством документов, извлекая из него релевантные документы.

IE основное внимание уделяет структуре текстового документа, пытаясь извлечь из него ключевые понятия. Таким образом, IE и Web Mining может находиться в разных отношениях друг к другу. С одной стороны, для исследования структуры и извлечения основных понятий из текста могут быть использованы методы Text Mining. В этом случае Web Mining является частью IE. С другой стороны, структуризация документов методами IE позволяет сохранять его в реляционной базе данных, что, в свою очередь, позволяет

применить к нему методы Data Mining. Таким образом, IE может рассматриваться как технология препроцессинга на одном из этапов Web Mining.

Как следует из анализа, различные методы и технологии могут использоваться совместно, взаимно улучшая друг друга.

## 14.1.4. Категории Web Mining

В области Web Mining выделяют следующие направления анализа:

- извлечение Web-контента (Web Content Mining);
- извлечение Web-структур (Web Structure Mining);
- исследование использования Web-ресурсов (Web Usage Mining).

Извлечение Web-контента включает в себя методы извлечения полезной информации из Web-ресурсов, таких как содержание, данные, документы и др. Актуальность данного направления возрастает в связи с тем, что в настоящее время прослеживается тенденция предоставления компаниями доступа к своим ресурсам. Это относится не только к статической информации, представленной в виде HTML-страниц, но также к данным, хранящимся в БД компаний, и другим ресурсам (таким, например, как электронные магазины). Безусловно, остается часть данных, к которым доступ невозможен. К этой категории относятся и закрытая (конфиденциальная) информация, а также информация, которая не может анализироваться в виду своей динамичности (например, динамические страницы, формируемые по запросам пользователей).

Особенностью Web-ресурсов является разнородность представленной информации: текстовые файлы, изображение, звук, видео, метаданные, а также гиперссылки.

В связи с этим технология Web Mining тесно связана с другими направлениями Data Mining. Так, для анализа текстовой информации используются методы Text Mining, подробно описанного в *гл. 9*. Для анализа изображений, видео- и аудиоинформации используется Multimedia Mining.

Необходимо обратить внимание на отличия существующих точек зрения на технологию извлечения Web-контента со стороны информационного поиска (IR) и представления БД. С точки зрения IR данная задача решается для улучшения релевантности получаемых документов и используется для индексации документов. С точки зрения базы данных задача извлечения Web-контента позволяет структурировать документ, расширяя возможности запросов к нему.

В зависимости от решаемой задачи различаются и методы Web Mining. Кроме того, решение задачи извлечения Web-контента в процессе IR различается

в зависимости от вида анализируемых документов. Они могут быть структурированные и слабоструктурированные (почти структурированные).

При извлечении Web-структур строятся модели, отображающие взаимосвязи между Web-страницами. Модель основывается на топологии гиперссылок с или без описания этих ссылок. Такая модель может использовать категоризацию Web-страниц и быть полезна для генерации информации об отношении и подобности между Web-сайтами. Данная категория Web Mining может быть использована для распознавания авторских сайтов и обзорных сайтов по темам.

Исследование использования Web-ресурсов анализирует информацию, генерируемую в процессе пользовательских сессий (взаимодействия пользователя с Web-ресурсами) и поведений пользователей. В отличие от первых двух категорий Web Mining, которые работают с первичной информацией (Web-ресурсами), исследование использования Web работает со вторичной информацией, порождаемой как результат взаимодействия пользователей с Web-ресурсами. К таким источникам информации относятся протоколы доступа Web-серверов, протоколы прокси-серверов, протоколы браузеров, пользовательские профили, регистрационные данные, пользовательские запросы, куки, клики мышками, прокручивание и многое другое, что делает пользователь в процессе взаимодействия с Web-ресурсами.

В табл. 14.1 представлены описанные категории, данные, с которыми они работают, методы их применения и т. п. Более подробно эти категории будут рассмотрены в последующих разделах главы.

Необходимо заметить, что разница между описанными категориями не всегда достаточно четкая. Методы извлечения Web-контента могут использоваться для анализа как текста, так и ссылок и даже профилей. Профили пользователей по большей части используются для пользовательских приложений или персональных ассистентов. То же верно для методов извлечения Web-структур, которые используют информацию о ссылках в дополнение к структурам ссылок. Кроме того, можно сделать заключение о пересечениях ссылок, которые были запрошены в течение пользовательской сессии и получены из файлов регистрации, генерируемых сервером.

На практике все три категории Web Mining могут быть использованы как по отдельности, так и в комбинации друг с другом, Особенно это относится к первым двум категориям, т. к. часто документы содержат ссылки и для их извлечения должны использоваться методы извлечения Web-контента.

**Таблица 14.1.** Классификация задач Web Mining

| Характеристики задач | Задачи Web Mining | | | |
|---|---|---|---|---|
| | Извлечение Web-контента | | Извлечение Web-структур | Исследование использования Web-ресурсов |
| | В целях информационного поиска | В целях размещения в БД | | |
| **Тип данных** | Неструктурированные.<br>Слабоструктурированные | Слабоструктурированные.<br>Web-сайт как БД | Структуры ссылок | "Следы" взаимодействия |
| **Анализируемые данные** | Гипертекстовые и текстовые документы | Гипертекстовые документы | Структуры ссылок | Протоколы сервера.<br>Протоколы браузера |
| **Подходы к представлению данных** | Наборы слов, *n*-граммов.<br>Термины, фразы.<br>Понятия или онтологии.<br>Отношения | Отношения.<br>Маркированный граф. | Граф | Реляционные таблицы.<br>Графы |
| **Метод** | TFIDF и его варианты.<br>Машинное обучение.<br>Статические методы, в том числе и NLP | Частные алгоритмы.<br>NLP.<br>Модифицированные ассоциативные правила | Частные алгоритмы | Статистические.<br>Модифицированные ассоциативные правила.<br>Машинного обучения |
| **Прикладное применение задач** | Кластеризация.<br>Классификация.<br>Правила поиска извлечения.<br>Поиск шаблонов в тексте.<br>Моделирование пользователя | Поиск частных подструктур.<br>Обнаружение схем Web-сайтов | Кластеризация и классификация | Конструкция сайта, адаптация и управление.<br>Маркетинг.<br>Моделирование пользователей |

# 14.2. Методы извлечения Web-контента

## 14.2.1. Извлечение Web-контента в процессе информационного поиска

**Способы представления документов.** Методы извлечения Web-контента в процессе информационного поиска во многом зависят от типа анализируемых документов. Различают два основных типа: неструктурированные и почти структурированные. К неструктурированному типу относятся все текстовые документы, не имеющие определенной структуры. К почти структурированным относятся документы, имеющие структуру в целом, но позволяющую вхождение в структурный элемент неструктурированного текста. К таким документам относятся HTML, XML и др.

Большинство методов анализа неструктурированного текста использует представление текстового документа в виде множества или вектора слов. Данный подход также широко применяется в методах Text Mining. При этом в такие представления помещаются отдельные слова без учета их расположения, связи с другими словами, контекста и других лингвистических особенностей.

Каждому слову во множестве ставится в соответствие некоторое свойство. Данное свойство может иметь или логический тип, отражающий наличие или отсутствие слова в тексте, или числовое значение, отражающее частоту появления слова в тексте. Последующая обработка может быть связана с удалением пунктуации, нечастых слов, стоп-слов и др. Уменьшение числа свойств возможно за счет применения различных методов выбора свойств, основанных на расчете следующих метрик [62]:

- информационного прироста (information gain);
- полного количества информации (mutual information);
- перекрестной энтропии (cross entropy);
- вероятности успешного исхода (odds-ration).

Кроме большого размера модели, векторное представление документов имеет еще один существенный недостаток: оно не обрабатывает синонимы — документы считаются семантически далекими друг от друга, если в них нет одинаковых слов. Данный недостаток устраняется методом *скрытой семантической индексации* (Latent Semantic Indexing — LSI) [63]. Согласно этому методу, пространство термов сингулярным разложением (отбрасываются наименее значимые сингулярные значения) приводится к пространству ортогональных факторов (некоррелируемых "индексных термов"). Поскольку результирующие факторы играют роль "приведенных" термов, декомпозиция

"сближает" документы из одинаковых предметных областей. В результате документы одной тематики, но которые не используют одинаковые термины, размещаются в одном разделе, и их стемминг сокращает слова с общими морфологическими корнями. Например, слова "информирование", "информация", "информатор" и "информировано" приводятся к их общему корню "информ" — и только это слово используется как свойство документа, вместо четырех форм.

Необходимо обратить внимание, что эффективность вариантов препроцессинга, направленных на сокращение размера набора свойств, может быть различна для разных областей.

Кроме представления документа в виде вектора слов, возможны и другие представления [65]:

- использующие информацию о позиции слова в документе;
- использующие *n*-граммное представление (последовательности слов длины вплоть до *n*) (например, "морфологический корень" — 3-грамма),
- использующие целые фразы (например, "быстрая лиса исчезла из вида");
- использующие понятие документа категорий;
- использующие термины (например, "норма годового процента" или "Уолл-стрит");
- использующие гипернимы (hypernym — слово, являющееся более общим, абстрактным по отношению к данному) (лингвистический термин отношения "это есть" — "собака есть животное", поэтому "животное" — это hypernym "собаки");
- использующие адресные объекты (например, имена людей, даты, почтовые адреса, расположения, организации или URL).

Реляционное представление является по сути первым представлением, учитывающим порядок слов [66; табл. 14.3]. Данное представление более выразительно, чем позиционная логика. Например, во множественном представлении слову соответствует его частота. В представлении в виде отношений используется взаимосвязь между различными словами и их позициями, например "слово *X* левее слова *Y* в том же предложении".

Несмотря на то, что широко используются различные виды представлений естественных языков, в настоящий момент нет исследований, которые наглядно показали бы преимущества тех или иных видов представлений для разных типов задач текстовой классификации. Скотт и Матвин [65] сравнивают различные представления (множество слов, представление, основанное на фразах, гипернимы и др.), но они не нашли никаких их существенных достоинств или недостатков по отношению друг к другу.

В силу схожести задачи извлечения Web-контента с задачами Text Mining для ее решения используются методы, подробно описанные в *гл. 9*. В табл. 14.2 представлены методы, применяемые для решения задачи извлечения Web-контента. В ней для каждого метода приведены модели представления документов и прикладные задачи, в которых они могут быть использованы. Методы извлечения Web-контента находят применение в разных задачах:

- текстовой классификации и кластеризации;
- определения событий и трекинга;
- поиска извлекаемых шаблонов или правил;
- поиска шаблонов в текстовых документах.

Отдельно стоит остановиться на задаче определения событий и трекинге. Данная задача является частной задачей более широкого направления автоматизированной обработки новостных данных — Topic Detection and Tracking (TDT) [67]. В TDT выделяют следующие направления исследований:

- разбиение потока на сюжеты;
- идентификация новых событий;
- определение связей между новостными историями;
- отслеживание интересующей пользователя информации.

**Слабоструктурированные документы.** Извлечение Web-контента из слабоструктурированных документов использует более развитые средства представления текста. Это в первую очередь связано с тем, что в документах уже выделены некоторые структурные элементы. Практически все методы в этой области для представления документа используют HTML-структуры внутри документов. Некоторые методы используют также для представления гиперссылки между документами.

Как и в случае с неструктурированными документами, к полученным представлениям применяются общие методы Data Mining.

Область применения методов довольно широка:

- гипертекстовая классификация;
- классификации и кластеризации;
- изучение отношений между Web-документами;
- извлечение шаблонов или правила;
- поиск шаблонов и слабоструктурированных данных.

**Таблица 14.2.** Методы извлечения Web-контента из неструктурированных документов в целях информационного поиска

| Методы | Авторы | Представление документа | Применение |
|---|---|---|---|
| Эпизодические правила | Е. Ахонен (H. Ahonen) и др. [68] | Набор слов и их позиции | Поиск ключевых слов и ключевых фраз. Обнаружение грамматических правил |
| TFIDF | Д. Биллсус (D. Billsus) и М. Паззани (M. Pazzani) [69] | Набор слов | Текстовая классификация |
| | С. Думайс (S. Dumais) [70] | Набор слов | Текстовая классификация |
| Naive Bayes | Д. Биллсус (D. Billsus) и М. Паззани (M. Pazzani) [69] | Набор слов | Текстовая классификация |
| | С. Думайс (S. Dumais) [70] | Фразы | Текстовая классификация |
| | Е. Франк (E. Frank) [71] | Фразы и их позиции | Извлечение ключевых фраз из текстовых документов |
| Пропозициональные правила, основанные на системе индуктивного логического программирования | У. Кохен (W. Cohen) [66] | Реляционное | Текстовая классификация |
| Индуктивное логическое программирование | М. Юнкер (M. Junker) [72] | Реляционное | Текстовая классификация. Изучение правил извлечения |
| Деревья решений | У. Кохен (W. Cohen) [66] | Реляционное | Текстовая классификация |
| | У. Й. Нам (U. Y. Nahm) и Р. Муне (R. Moone) [73] | Набор слов | Предсказание (слова) взаимосвязи |
| | Й. Янг (Y. Yang) [74] | Набор слов и фразы | Обнаружение событий и трекинг |
| | Х. Каргупта (H. Kargupta) [75] | Набор слов с *n*-граммами | Текстовая классификация и иерархическая кластеризация |

**Таблица 14.2** *(продолжение)*

| Методы | Авторы | Представление документа | Применение |
|---|---|---|---|
| Ускоренные деревья решений | Ш. Вейс (S. Weiss) [76] | Набор слов | Текстовая категоризация |
| Ассоциативные правила | Р. Фелдман (R. Feldman) [77] | Термины | Поиск шаблонов в текстовых данных |
| Относительная энтропия | Р. Фелдман (R. Feldman) и И. Даган (I. Dagan) [78] | Категории понятия | Поиск шаблонов в текстовых данных |
| Максимизация энтропии | К. Нигам (K. Nigam) [79] | Набор слов | Текстовая классификация |
| Скрытая Марковская модель | Д. Фрейтаг (D. Freitag) и А. Маккалум (A. McCalum) [80] | Набор слов | Изучение модели извлечения |
| Статистические методы без учителя | Т. Хофман (T. Hofmann) [81] | Набор слов | Иерархическая кластеризация |
| Самоорганизующиеся карты | Т. Хонкела (T. Honkela) [64] | Набор слов с буквами | Текстовая кластеризация и кластеризация документов |
| Нейронные сети | Е. Винер (E. Wiener) [82] | Набор слов | Текстовая категоризация |
| Логистическая регрессия | Е. Винер (E. Wiener) [82] | Набор слов | Текстовая категоризация |
| Метод ближайшего соседа | Й. Янг (Y. Yang) [74] | Набор слов и фразы | Обнаружение событий и трекинг |
| Алгоритмы кластеризации | Й. Янг (Y. Yang) [74] | Набор слов и фразы | Обнаружение событий и трекинг |
| Неконтролируемая иерархическая кластеризация | Х. Каргупта (H. Kargupta) [75] | Набор слов с *n*-граммами | Текстовая классификация и иерархическая кластеризация |

**Таблица 14.2** *(окончание)*

| Методы | Авторы | Представление документа | Применение |
|---|---|---|---|
| Методы статистического анализа | Х. Каргупта (H. Kargupta) [75] | Набор слов с *n*-грамами | Текстовая классификация и иерархическая кластеризация |
| Базовая система правил | С. Скотт (S. Scott) и С. Мэтви (S. Matwi) [65] | Набор слов.<br>Фразы, гипернимы и синонимы | Текстовая классификация |
| Обучение на правилах | С. Содерланд (S. Soderland) [83] | Предложения и клаузы (элементарные предложения) | Изучение правил извлечения |
| Текстовая компрессия | И. Виттен (I. Witten) [84] | Название объекта | Классификации именованных сущностей |

**Таблица 14.3.** Методы извлечения Web-контента из слабоструктурированных документов в целях информационного поиска

| Методы | Авторы | Представление документа | Применение |
|---|---|---|---|
| Модифицированный Naive Bayes | М. Кравен (M. Craven) и др. [85] | Отношение и онтология | Гипертекстовая классификация.<br>Изучение отношения Web-страницы. |
| Индуктивное Логическое Программирование | | | Изучение правил извлечения |
| Алгоритмы классификации с и без учителя | Ф. Кримминс (F. Crimmins) и др. [86] | Фразы, URL и метаинформация | Иерархическая и графическая классификации.<br>Кластеризация |
| Правила изучения | Дж. Фюрнкранц (J. Furnkranz) [87] | Набор слов и информационные гиперссылки | Гипертекстовая классификация |
| Правила изучения | И. Мусли (I. Muslea) [88] | Множество слов, тегов, и позиций слов | Изучение правил извлечения |
| Обучение на правилах | С. Содерланд (S. Soderland) [83] | Предложения, фразы, и название объекта | Изучение правил извлечения |
| TFIDF | Т. Яахимс (T. Joachims) и др. [89] | Набор слов и информационные гиперссылки | Гипертекстовое предсказание |
| Изучение подкрепления | | | |
| Нейронные сети с усиленным обучением | Дж. Шавлик (J. Shavlik) и Т. Эллиси-Рэд (T. Eliassi-Rad) [90] | Локальный набор слов и отношения | Гипертекстовая классификация |
| Измененный алгоритм поиска ассоциативных правил | Л. Сайн (L. Singh) и др. [91] | Понятие и название объектов | Поиск шаблонов в слабоструктурированных текстах |

## 14.2.2. Извлечение Web-контента для формирования баз данных

Задача извлечения Web-контента для его размещения в базе данных относится к проблеме управления информацией и обработки запросов к ней [98]. Существуют три класса задач, относящихся к этой проблеме:

- моделирование и формирование запросов к Web;
- извлечение информации и интеграция;
- создание и реструктуризация Web-сайта.

Хотя первые две задачи относятся к категории извлечения Web-контента, не все методы, применяемые при их решении, относятся к этой категории. Это связано с отсутствием машинного обучения или использованием методов Data Mining в процессе их решения. Обычно методы извлечения Web-контента пытаются выявить структуру Web-документа или преобразовать его для сохранения в базе данных таким образом, чтобы улучшить информационное управление и сделать возможным запрос к нему.

С точки зрения размещения Web-контента в базе данных целью, большей частью, является построение модели данных и объединение их таким образом, чтобы поиск мог выполняться не только по ключевым словам, но и по запросам, более приближенным к естественному языку. Этого можно достичь построением схемы Web-документов, формированием хранилища, базы знаний или виртуальной базы данных. Исследования в этой области большей частью имеют дело со слабоструктурированными данными. Слабоструктурированные данные из представления базы данных часто ссылаются на данные, которые имеют некоторую структуру, но не жесткую схему [99; 100].

Из табл. 14.4 можно видеть, что методы извлечения Web-контента для целей базы данных используют представления, которые отличаются от представлений, используемых для целей информационного поиска. Данные методы в основном используют представления в виде *модели объектного обмена* (Object Exchange Model — OEM) [99].

Информация в OEM-структуре представляется в виде графа с именованными ребрами, а узлы соответствуют объектам. Для определенности будем называть их node-объектами или n-объектами. В простейшем варианте они могут быть атомарными или контейнерами. Атомарные n-объекты имеют только входящие связи и значение определенного типа, но не имеют зависимых объектов. У контейнеров нет значений, но есть зависимые объекты (исходящие связи). Каждый n-объект имеет уникальный идентификатор. На связи между ними, в общем случае, не накладывается никаких ограничений.

В большинстве своем рассмотренные методы применяются в задачах, связанных с выявлением, исследованием или формированием схем DataGuides [92].

DataGuide — сжатый вид схемы слабоструктурированных данных. Для практического применения и из-за сложности вычислений DataGuide часто аппроксимируется [99]. Некоторые приложения не решают задачу поиска глобальной схемы, они концентрируются на задачах поиска подсхем в слабоструктурированных данных.

**Таблица 14.4.** Методы извлечения Web-контента из слабоструктурированных документов в сохранения в базе данных

| Методы | Авторы | Представление документа | Применение |
|---|---|---|---|
| Частные алгоритмы | Р. Голдман (R. Goldman) и Дж. Видом (J. Widom) [92] | OEM | Поиск DataGuide в слабоструктурированных данных |
| | Ш. Грумбах (S. Grumbach) и Г. Мекка (G. Mecca) [93] | Строки и отношения | Поиск схемы в слабоструктурированных данных |
| | С. Нестеров (S. Nestorov) и др. [94] | OEM | Поиск типов иерархии в слабоструктурированных данных |
| Модифицированные алгоритмы поиска ассоциативных правил | Х. Тоивонен (H. Toivonen) [95] | OEM | Поиск используемых подструктур в слабоструктурированных данных |
| | Х. Лиу (H. Liu) и К. Ванг (K. Wang) [96] | OEM | Поиск шаблонов в слабоструктурированных данных |
| Атрибутно-ориентированное утверждение | О. Зайн (O. Zaiane) и Дж. Хан (J. Han) [97] | Отношения | Мультиуровневая база данных |

Другое применение методов данной категории Web Mining — это формирование *многослойной базы данных* (MLDB) [97], в которой каждый уровень создается обобщением низших уровней и использует специальный язык запросов для Web Mining, чтобы извлекать некоторые знания из MLDB.

Из-за различий представления Web-документов, используемых в извлечении Web-контента для их сохранения в БД, большинство из используемых методов отличаются от классических методов Data Mining, которые действуют на плоских данных и требуют дополнительной модификации. Исключение со-

ставляют методы ILP, которые могут работать с отношениями и графическим представлением данных. Для того чтобы действовать на реляционных или графических данных [93, 94, 97], используют вероятностные алгоритмы для обнаружения схем и для конструкции MLDB [96], применяют измененную версию алгоритмов построения ассоциативных правил [95] и модифицированную версию порядковой логики правил ассоциации [101].

Необходимо отметить, что существует также большая область, связанная с анализом мультимедийной информации, которая в большом объеме присутствует в Web-документах. Методы, позволяющие извлекать полезные знания из мультимедийной информации, объединяются в технологию Multimedia Mining и требуют отдельного рассмотрения.

# 14.3. Извлечение Web-структур

## 14.3.1. Представление Web-структур

Если в задаче извлечения Web-контента с целью его сохранения в базе данных интерес вызывает внутренняя структура Web-документа, то в задаче извлечения Web-структур, прежде всего, интерес вызывает структура гиперссылок в пределах Web-сети (междокументная структура).

Для решения этой задачи требуется представление документов и отношений между ними, учитывающее гиперссылки. В связи с этим такие модели имеют отличия от описанных ранее подходов в представлении отдельных документов.

Гиперссылки моделируются с разным уровнем детализации в зависимости от применения модели. В простейших моделях гиперссылки могут быть представлены как направленный граф:

$$G = (D, L),$$

где $D$ — это набор узлов, документов или страниц; $L$ — набор ссылок.

В зависимости от информации, которая учитывается при построении модели, они могут быть более или менее точными. Грубые модели могут не учитывать текст, сопровождающий гиперссылки. Наиболее точные модели учитывают не только текст документа (точнее модель документа), но и распределение термов в документе в зависимости от его принадлежности определенной предметной области.

В некоторых моделях исходные документы представляются как последовательность термов с исходящими от них гиперссылками. Такое представление может быть полезно, чтобы установить специфичные отношения между определенными термами и гиперссылками (а не между документами целиком).

В некоторых случаях целесообразно рассматривать распределение термов документов в зависимости от их тематики. Например, распределение термов документа из области "велосипедного спорта" полностью отличается от такого же документа, относящегося к теме "археология". В отличие от журналов по археологии или велосипедному спорту, Web-страницы не изолированы и могут ссылаться на документы по другим темам. Если необходимо учитывать связи не только между отдельными документами, но и между темами, то необходимо включать в модель также связи между темами.

Можно выделить три основные задачи, которые могут быть решены на основании анализа Web-структуры:

- оценка важности структуры Web (документа или узла), воздействие и влияние их друг на друга;
- поиск Web-документов с учетом гиперссылок, содержащихся в них;
- кластеризация структур для их возможного явного объединения.

## 14.3.2. Оценка важности Web-структур

Решение первой задачи в основном базируется на методах построения социальных сетей. Социальная сеть строится для измерения значимости того или иного индивидуума на основании его связей с другими индивидуумами. Такая социальная сеть представляется в виде графа, в котором узлы соответствуют индивидуумам, а дуги — их отношению между собой. При этом граф является взвешенным, т. е. каждой дуге ассоциируется вес, соответствующий силе влияния одного индивидуума другому. На основе анализа построенного таким образом графа вычисляется значимость каждого узла (индивидуума) на основании информации о дугах и их весах, входящих и исходящих от узла.

Как несложно увидеть, модель социальной сети очень близка модели, соответствующей структуре Web. В связи с этим и методы, применяемые к социальным сетям, могут использоваться для решения тех же задач на Web.

Начиная с 1996 г., методы анализа социальных сетей были применены к графу Web с целью идентифицировать самые значимые страницы, соответствующие пользовательскому запросу.

Л. Катц (L. Katz.) предложил для вычисления значимости Web-структур использовать пути, основанные на входящих ссылках. В соответствии с этой идеей количество путей длиной $r$ от узла $i$ к $j$ обозначается $P_{ij}^r$. Общее количество путей разной длины вычисляется по формуле:

$$Q_{ij} = \sum_{r=1}^{\infty} b^r P_{ij}^r,$$

величина $b^r < 0$ должна выбираться таким образом, чтобы обеспечить сходимость формулы для каждой пары. Значимость узла $j$ вычисляется как сумма количеств путей от всех узлов:

$$s_j = \sum_i Q_{ij}.$$

В матричной форме вычисление значимости каждого узла может быть записано в виде:

$$S = (I - bA)^{-1} - I,$$

где $I$ — единичная матрица; $A$ — матрица, содержащая веса связей между узлами.

Ч. Хаббелл (C. Hubbell) предложил подобную модель, основанную на обучении схемы распределения весов узлов сети. Априорная оценка значимости *j*-го узла обозначается $e_j$. Таким образом, Хаббелл описывает значимости узлов $\{s_j\}$ как набор таких значений, что процесс взаимодействия узлов сохраняет баланс, т. е. количество ссылок, входящих в узел $j$, взвешенное значением значимости узлов, из которых выходят ссылки, равны значимости узла $j$. Таким образом, значимости узлов могут быть вычислены решением системы линейных уравнений:

$$s_j = e_j + \sum_i A_{ij} s_i.$$

Если априорные оценки представить как вектор $\{e_j\}$, систему линейных уравнений можно записать в матричном виде:

$$S = (I - A^T)^{-1} e.$$

Для определения значимости влияния и воздействия Web-структур также широко используются метрики, применяемые для ранжирования найденных документов в поисковых системах. Так, широкое применение в данной категории задачи Web Mining нашла метрика, используемая поисковой системой Google — PageRank [102].

*PageRank* — статическая величина, предназначенная для оценки качества страниц на основании информации о количестве ссылок на нее. Данная метрика не зависит от запросов пользователей и вычисляется заранее для каждой страницы, что обеспечивает быстрое ее определение при обработке запроса пользователя.

За основу PageRank был выбран подход, используемый в библиометрике для оценки важности публикации автора по числу ее упоминаний в библиогра-

фических ссылках других авторов. Для применения метода к Web в него были внесены изменения: вес каждой ссылки учитывается индивидуально и нормируется по числу ссылок на ссылающейся странице.

Идея метода заключается в следующем: если перемещаться по Web-сети случайным образом, то при нахождении на странице $p$ пользователь может перескочить на другую страницу в сети, выбранную псевдослучайным образом, либо он может перейти по ссылке на текущей странице, не возвращаясь при этом и не посещая одну и ту же страницу дважды. Вероятность случайного скачка обозначается как $d$, тогда вероятность перехода по ссылке будет $1-d$. Таким образом, вероятность нахождения пользователя на странице $p$ можно вычислить по следующей формуле:

$$R_{j+1}(p) = d + (1-d)\sum_{i=1}^{k} R_j(p_i)/C(p_i),$$

где $R(p)$ — PageRank страницы; $C(p)$ — число исходящих ссылок на странице; $k$ — число ссылающихся на $p$ страниц; $d$ — коэффициент затухания (damping factor), обычно $0{,}1 < d < 0{,}15$.

Если масштабировать PageRank таким образом, что

$$\sum_{i=1}^{N} R(p_i) = 1,$$

где $N$ — число всех страниц, для которых производится расчет PageRank, то $R(p)$ можно рассматривать как распределение вероятности по всем страницам.

Для вычисления PageRank составляется матрица $M$ размером $N \times N$, где каждому элементу $m_{ij}$ матрицы присваивается значение $R_0(p) = 1/C(p)$ в том случае, если с $i$-й страницы имеется ссылка на $j$-ю, все оставшиеся элементы матрицы заполняются нулями. Таким образом, вычисление PageRank сводится к отысканию собственного вектора матрицы $M$, что достигается умножением матрицы $M$ на вектор $R_j$ на каждом шаге итерации. Введение коэффициента затухания гарантирует, что процесс сходится.

Другой популярной метрикой определения важности Web-страницы является *HITS* (Hyperlink-Induced Topic Search) [103]. Если PageRank вычисляется один раз глобально для всех страниц в индексе, то в рамках модели HITS предполагается, что важность страницы зависит от запроса, т. к. в разных тематических сообществах разные авторитеты. Поэтому HITS вычисляется локально для каждого запроса.

Этот алгоритм определяет важность Web-страниц не только по входящим ссылкам ("авторитетность страницы"), но и по исходящим ссылкам ("хабам"). Страницы по данному алгоритму разбиваются как бы на сообщества, которые предположительно являются тематическими сообществами. Внутри данных сообществ для каждой страницы $p$ вычисляется значение "авторитетности" $a_p$, т. е. "веса" каждой страницы как сумма "хаб"-весов ссылающихся страниц, и "хаб"-веса $h_p$ каждой страницы как сумма весов "авторитетности" цитируемых страниц:

$$a_p = \sum_{u \to p} h_u \text{ и } h_p = \sum_{p \to v} a_v .$$

Эти значения применяются при определении релевантности страницы запросу.

Из-за зависимости построения графа от запроса пользователя метод HITS медленнее, чем PageRank. Вариант этого метода был использован Дж. Дином (J. Dean) и М. Хенцигером (M. Henzinger), чтобы найти подобные Web-страницы, основываясь только на анализе ссылок. Они повысили скорость метода за счет выборки Web-графа из связанности серверов, который был построен на основании предварительного исследования существенных частей Web.

Расширение графа HITS иногда приводит к зашумлению темы. Например, Web-страницы по теме наград в кинематографе тесно связаны ссылками (и до некоторой степени относящейся к делу) страницами кинокомпаний. Хотя кинонаграды более узкая тема, чем кинофильмы, топовые кинокомпании появляются как победители после применения метода HITS. Это частично связано с тем, что в HITS (так же как и в методе PageRank) у всех ребер графа одинаковые значения. Зашумление может быть уменьшено за счет учета контента гиперссылки, который размещается около указателя (анкер-элемента HTML, связывающего Web-документы). Тогда можно сделать вывод, что ссылки, содержащие слова запроса пользователя (в нашем примере слово "награда"), более важны для этого запроса, чем другие ребра.

Системы автоматической компиляция ресурсов (Automatic Resource Compilation — ARC) и Clever включают подобную запросозависимую модификацию весов ребер. При этом результаты запроса значительно улучшаются [104].

К. Барат (K. Bharat) и М. Хенцигер (M. Henzinger) предложили другой способ объединения ссылок и текстового содержания, чтобы избежать зашумления графа [105]. Они для каждой страницы строят модель "векторного пространства". На каждом шаге расширения графа, в отличие от метода HITS, они не включают все узлы на расстоянии от предварительного результата запроса.

Вместо этого они сокращают расширение графа в узлах, чьи векторы термов являются выбросами относительно набора векторов, соответствующих документам, непосредственно восстановленным из поисковой машины [105]. В приведенном ранее примере можно было бы надеяться, что ответ на запрос "кинонаграда" от начального поиска по ключевому слову будет содержать больше страниц, связанных с награждениями, а не с компаниями; таким образом, распределение ключевых слов на этих страницах позволит данному алгоритму эффективно сократить узлы, которые являются выбросами и близки к терму "кинокомпания".

## 14.3.3. Поиск Web-документов с учетом гиперссылок

Другое применение задачи извлечения Web-структур призвано улучшить релевантность результатов поиска за счет учета гиперссылок в Web-документах.

Фрайз (Frisse) [106] рассматривал проблему поиска документа в автономных гипертекстах. Он предложил базовую эвристику, согласно которой гиперссылки могут увеличить релевантность найденных страниц и, следовательно, улучшить работу поисковой системы. В соответствии с ней релевантность страницы в гипертексте, соответствующая запросу, частично базируется на релевантности страниц, с которыми она связана. Данная эвристика была использована в алгоритме Мархиори (Marchiori's) HyperSearch [107], примененном для поиска Web-страниц. В нем значение релевантности для страницы $p$ вычисляется методом, который включает релевантность страниц, достижимых из $p$. При этом зависимость от релевантности достижимой страницы уменьшается за счет коэффициента затухания, уменьшенного экспоненциально с увеличением расстояния от страницы $p$.

Знание местоположения страницы $p$ в Web-графе, а точнее наличие информации "на кого она ссылается и какие страницы ссылаются на нее" увеличивает понимание ее контента. Это может помочь в лучшем сопоставлении запроса пользователя и ключевых слов документа. Для этого может использоваться текст, окружающий ссылку, как описатель указываемой страницы, используемый для определения ее релевантности.

Другим направлением работ, использующим гиперссылки для улучшения результатов поиска, является создание поискового формализма, который будет способен обрабатывать запросы, включающие предикаты текста и ссылок. Ароцена (Arocena), Менделзон (Mendelzon) и Михаила (Mihaila) [108] разработали структуру, поддерживающую Web-запросы, которые комбинируют стандартные ключевые слова с условиями структуры окружающей ссылки.

## 14.3.3. Кластеризация Web-структур

Кластеризация, основанная на ссылочной структуре Web-документов, фокусируется на декомпозиции явно представленной коллекции узлов на близкие поднаборы. Эта задача главным образом решается на небольших наборах документов, например, на документах одного сайта.

Для кластеризации применительно к Web-документам используются две функции подобия из библиометрики, вычисляемых для каждой пары документов $p$ и $q$:

- библиографическая связанность (bibliographic coupling) — количество документов, цитируемых обоими документами $p$ и $q$;
- взаимное цитирование (co-citation) — количество документов, которые цитируют оба документа $p$ и $q$.

Вэйс (Weiss) [109] обобщил две функции, применив их к ссылкам произвольной длины.

Несколько методов были предложены для решения этой задачи, чтобы выделить кластеры из набора узлов, аннотируемых такой информацией подобия. Смолл (Small) и Uhbaabc (Griffith) [110] используют поиск в ширину, чтобы вычислить связанные компоненты ненаправленного графа, в котором между двумя узлами строится дуга, если и только если для этой пары существует положительное значение взаимного цитирования. Питков (Pitkow) и Пиролли (Pirolli) [111] применили этот алгоритм для изучения структуры коллекции Web-страниц, основываясь на связях между ними.

Можно также использовать анализ основных компонентов [112, 113] и родственные методы сокращения измерений, такие как многомерное масштабирование, с целью сгруппировать коллекцию узлов. При таком подходе вычисления начинаются с матрицы $M$, содержащей информацию подобия между парами узлов и представления каждого узла в виде многомерного вектора $\{v_i\}$. При этом можно использовать несколько неглавных собственных векторов матрицы подобия $M$, чтобы описать подпространство с небольшим количеством измерений, в которое векторы $\{v_i\}$ могут быть спроецированы. К проекциям векторов могут быть применены геометрические или визуальные методы для выделения кластеров в маломерном пространстве. Теоремы линейной алгебры подтверждают, что проекция на первые $k$ собственных векторов обеспечивает минимальное искажение для всех $k$-мерных проекций данных. Смолл и Маккейн применили данный подход к журналам и данным о взаимном цитировании авторов. Питков и Пиролли применили метод понижения измерений к наборам Web-страниц, используя метрику взаимного цитирования.

Еще один подход, основанный на собственных векторах для кластеризации ссылочной структуры, был предложен Донасом (Donath), Хофманом (Hoffman) [114] и Фидлером (Fiedler) [115]. Спектральные методы разделения графа соотносят слабосвязанные сегменты ненаправленного $G$ графа с собственными значениями и собственными векторами его матрицы смежности $A$. Каждый собственный вектор матрицы $A$ имеет единственную координату для каждого узла графа $G$ и таким образом может рассматриваться как ассоциирование весов узлам графа. Каждый неглавный собственный вектор имеет как положительную, так и отрицательную координаты. Для кластеризации используется эвристика, которая заключается в том, что узлы, соответствующие большим положительным координатам данного собственного вектора, имеют тенденцию быть очень слабо связанными с узлами, соответствующими большим отрицательным координатам того же самого собственного вектора.

Другим направлением кластеризации является центроидное масштабирование. Метод предназначен для представления двух типов объектов в общем пространстве. Рассмотрим, например, ряд людей, которые ответили на вопросы исследования. Было бы удобно представить и людей и возможные ответы в общем пространстве таким образом, чтобы каждый человек был "ближе" к ответам, которые он или она выбрал; и каждый ответ был "ближе" к человеку, который выбрал его. Центроидное масштабирование, основанное на собственном векторе, позволяет достичь этого. В контексте информационного поиска скрытая семантическая индексация (Latent Semantic Indexing) применяла метод центроидного масштабирования к векторной модели представления документов. Это позволило представить термы и документы в общем маломерном пространстве, в котором естественно геометрически описанные кластеры часто отделяют множественный смысл термина запроса.

# 14.4. Исследование использования Web-ресурсов

## 14.4.1. Исследуемая информация

Процесс исследования использования Web-ресурсов обычно включает в себя только три фазы:

1. Препроцессинг.
2. Извлечение шаблонов.
3. Анализ шаблонов.

В отличие от предыдущих двух задач Web Mining, в этой задаче исходными данными являются не сами страницы и их содержимое, а информация, запи-

сываемая в результате взаимодействия пользователей с Web. Выделяют следующие типы данных, применяемые в задаче исследования использования Web-ресурсов:

- использование — данные, которые описывают использование страниц, такие как IP-адресса, ссылки на страницы, а также дату и время доступа к ним;
- пользовательские профили — данные, которые обеспечивают демографическую информацию (пол, возраст, социальное положение и др.) о пользователе, а также регистрационную информацию.

Данные об использовании Web собираются в различных источниках, которые можно разделить на следующие основные группы: серверы, клиенты, прокси.

На стороне сервера информация для анализа может извлекаться из логов, трафика сервера, куки-файлов, запросов пользователей и др.

Логи Web-серверов являются важным источником информации, т. к. они в явном виде содержат описание действий посетителей сайтов. Однако до сервера доходит не вся информация о действиях пользователей, так просмотр пользователем страниц, сохраненных у него в кэше, никак не отражается на сервере. Кроме того, информация, заключенная в HTTP Post-запрос, не будет сохраняться в логе сервера.

Для извлечения информации из входящего в сервер трафика может использоваться технология анализатора пакетов (packet sniffer). Данная технология позволяет извлекать информацию напрямую из TCP/IP-пакетов, поступающих на Web-сервер.

Куки-файлы автоматически генерируются для каждого пользовательского браузера, чтобы поддержать сессионное взаимодействие. Они хранят информацию, идентифицирующую пользовательскую сессию.

Пользовательские запросы содержат информацию о потребностях пользователей, искомых документах и др.

На стороне клиента потенциально может находиться достаточно много информации о поведении пользователей, начиная от просматриваемых страниц и заканчивая щелчками мыши. Однако основная трудность заключается в извлечении этой информации, т. к. пользователей сайтов бесконечно много и заранее невозможно предсказать, какие пользователи будут обращаться к сайтам.

В литературе [113] предлагаются подходы, основанные на использовании агентов (реализованных как Java-апплеты или Java-скрипты) и на изменении кода Web-браузеров. Подход, использующий агентов, предполагает его передачу на сторону клиента, где он будет записывать и отправлять на сервер информацию о действиях пользователя.

Такой подход имеет ряд недостатков:

- Java-апплеты и Java-скрипты имеют ограниченные возможности, выполняясь в рамках браузера (так называемой песочницы);
- не мониторятся действия пользователей до момента начала работы агента (время загрузки и время инициализации агента);
- отслеживаться может только взаимодействие одного пользователя (которому загружен агент) с одним сайтом (с которого загружен агент).

Подход, основанный на модификации существующих браузеров с целью встроить в них средства мониторинга действий пользователя, позволяет решить перечисленные проблемы. Однако он сталкивается с трудностями, связанными с тем, что не все браузеры являются Open Source (т. е. предоставляют исходный код), а следовательно, могут быть модифицированы. К браузерам с открытым кодом относятся такие популярные, как Mozaic или Mozilla. Ввиду этого появляется проблема, связанная с убеждением пользователей использовать именно такие модифицированные браузеры.

Прокси-серверы являются промежуточным уровнем между клиентом и сервером и используются для кэширования часто запрашиваемых пользователем страниц. Они могут хранить действительные запросы от нескольких клиентов к различным сайтам, которые не дошли до серверов. Такая информация может рассматриваться как поведение группы анонимных пользователей.

Ни один из видов источников не хранит в себе полной информации, а следовательно, не может рассматриваться как единственный поставщик данных для анализа. Для повышения качества анализа поведения пользователей в нем должна участвовать информация от всех видов источников данных.

Вся информация, поставляемая от перечисленных ранее источников, может быть описана в следующих терминах, определенных группой W3C Web Characterization Activity (WCA).

- *Пользователь* (user) — индивидуум, получающий доступ к файлу одного или нескольких Web-серверов через браузер. Сложность данного понятия связана с идентификацией уникального пользователя, т. к. он может получать доступ с разных клиентских машин, а также от его имени могут работать различные агенты.
- *Просмотр страницы* (page view) — включает в себя одновременное отображение нескольких файлов в браузере пользователя. Данное понятие ассоциирует одного пользователя и несколько файлов (объединенных в одной Web-странице, в том числе и фреймовой структурой), открываемых одним щелчком.
- *Потоки кликов* (click streams) — последовательность открытия Web-страниц (вызываемых кликами). Не всегда можно восстановить полную после-

довательность, основываясь только на информации со стороны сервера, т. к. запросы, обрабатываемые клиентом или прокси-сервером, "не известны".

- *Пользовательская сессия* — последовательность просматриваемых страниц (поток кликов) одним пользователем на разных Web-сайтах.
- *Сессии сервера* (server sessions) — набор просматриваемых страниц в рамках одной пользовательской сессии с одного Web-сервера. Для анализа обычно доступна только серверная сессия, ограниченная одним сервером.
- *Эпизоды* (episodes) — любой семантически значимый поднабор пользовательской или серверной сессии.

## 14.4.2. Этап препроцессинга

Для решения задачи исследования использования Web на этапе препроцессинга в массиве анализируемых данных должны быть выделены перечисленные сущности. Это значительно усложняет неполнота данных, получаемых с одного источника. Например, для идентификации пользователя недостаточно только IP-адреса, т. к. возможны следующие ситуации.

- *Один IP-адрес — несколько серверных сессий.* Интернет-провайдеры обычно имеют пул прокси-серверов, доступных пользователям. В одно и то же время пользователь может просматривать страницы одного сайта с различных прокси-серверов.
- *Несколько IP-адресов — одна серверная сессия.* Некоторые интернет-провайдеры или инструменты безопасности случайным образом каждый пользовательский запрос ассоциируют с разными IP-адресами. В результате одна серверная сессия ассоциируется с несколькими IP-адресами.
- *Несколько IP-адресов — один пользователь.* Пользователь может получать доступ к Web-ресурсам с разных машин, а следовательно, будет ассоциирован с разными IP-адресами. Это усложняет отслеживание повторяющихся запросов от одного пользователя.
- *Множество агентов — один пользователь.* Пользователь, работающий с более чем одним браузером или пользующийся агентами, даже с одной машины будет распознаваться как несколько пользователей.

При условии если удалось идентифицировать пользователя (с помощью кукифайлов, логина или анализа его IP/агента/пути), необходимо поток кликов от каждого пользователя разделить на сессии. Эта задача усложняется тем, что тяжело определить, когда пользователь покинул сайт, т. к. обычно при анализе запросы данного пользователя к другим серверам не доступны. Кроме того, часто сессия прерывается по тайм-ауту, если пользователь не возобновляет

**Таблица 14.5.** Пример серверного лога

| № | IP-адрес | Время | Метод | URL | Протокол | Статус | Ссылка | Браузер |
|---|---|---|---|---|---|---|---|---|
| 1 | 123.456.78.9 | 25/Apr/1998:03:04:41 -0500 | GET | A.html | HTTP/1.0 | 3290 | | Mozilla/3.04 (Win95, I) |
| 2 | 123.456.78.9 | 25/Apr/1998:03:05:34 -0500 | GET | B.html | HTTP/1.0 | 2050 | A.html | Mozilla/3.04 (Win95, I) |
| 3 | 123.456.78.9 | 25/Apr/1998:03:05:39 -0500 | GET | L.html | HTTP/1.0 | 4130 | | Mozilla/3.04 (Win95, I) |
| 4 | 123.456.78.9 | 25/Apr/1998:03:06:02 -0500 | GET | F.html | HTTP/1.0 | 5096 | B.html | Mozilla/3.04 (Win95, I) |
| 5 | 123.456.78.9 | 25/Apr/1998:03:06:58 -0500 | GET | A.html | HTTP/1.0 | 3290 | | Mozilla/3.01 (X11, I, IRIX6.2, IP22) |
| 6 | 123.456.78.9 | 25/Apr/1998:03:07:42 -0500 | GET | B.html | HTTP/1.0 | 2050 | A.html | Mozilla/3.01 (X11, I, IRIX6.2, IP22) |
| 7 | 123.456.78.9 | 25/Apr/1998:03:07:55 -0500 | GET | R.html | HTTP/1.0 | 8140 | L.html | Mozilla/3.04 (Win95, I) |
| 8 | 123.456.78.9 | 25/Apr/1998:03:09:50 -0500 | GET | C.html | HTTP/1.0 | 1820 | A.html | Mozilla/3.01 (X11, I, IRIX6.2, IP22) |
| 9 | 123.456.78.9 | 25/Apr/1998:03:10:02 -0500 | GET | O.html | HTTP/1.0 | 2270 | F.html | Mozilla/3.04 (Win95, I) |
| 10 | 123.456.78.9 | 25/Apr/1998:03:10:45 -0500 | GET | J.html | HTTP/1.0 | 9430 | C.html | Mozilla/3.01 (X11, I, IRIX6.2, IP22) |
| 11 | 123.456.78.9 | 25/Apr/1998:03:12:23 -0500 | GET | G.html | HTTP/1.0 | 7220 | B.html | Mozilla/3.04 (Win95, I) |
| 12 | 209.456.78.2 | 25/Apr/1998:05:05:22 -0500 | GET | A.html | HTTP/1.0 | 3290 | | Mozilla/3.04 (Win95, I) |
| 13 | 209.456.78.3 | 25/Apr/1998:05:06:03 -0500 | GET | D.html | HTTP/1.0 | 1680 | A.html | Mozilla/3.04 (Win95, I) |

взаимодействие с сервером через некоторый промежуток времени. Если ID сессии встраивается в каждый URI, то определение сессии устанавливается контентным сервером. В этом случае запрашиваемый контент хранится в поле запроса и сохраняется в логе на сервере. Однако в связи с тем, что для каждой сессии сервер может иметь переменные состояний, в которых хранит некоторую информацию о сессии, то не вся она доступна из URI. Также использование кэша и связанное с этим недохождение запросов до сервера усложняет выделение сессий. Эта проблема может быть решена только мониторингом запросов от клиентской стороны к кэшу. Ссылочные поля запросов могут быть использованы, чтобы определить, когда были просмотрены страницы из кэша.

В табл. 14.5 приведен пример серверного лога (первая колонка добавлена для удобства), на котором можно увидеть описанные ранее проблемы. IP-адрес 123.456.78.9 относится к трем серверным сессиям. Адреса 209.456.78.2 и 209.45.78.3 относятся к одной — четвертой сессии. Выделение 3-х сессий для одного адреса (с 1-й по 11-ю строку таблицы) возможно на основании совместно используемой информации о ссылке, по которой произошел переход с предыдущей страницы, и агенте. В результате запросы разбиваются на следующие сессии: A-B-F-O-G, L-R и A-B-C-J. Если сделать путь переходов непрерывным и добавить возвраты на предыдущие страницы, то в первую сессию добавятся еще две страницы A-B-F-O-F-B-G и одна страница добавится к третьей сессии A-B-A-C-J.

В данном примере без использования информации о куки-файлах, встроенной в запрос ID сессии, или информации со стороны клиента не удастся определить, что строки 12 и 13 принадлежат одной сессии, т. к. они различаются IP-адресами.

На этапе препроцессинга кроме выделения основных элементов, необходимых для дальнейшего анализа, часто выполняется фильтрация по контексту. Такая задача может возникнуть, например, если аналитика интересует использование Web-ресурсов по определенной тематике. Для такой фильтрации применяют методы анализа Web-контента *(см. разд. 14.2)*.

## 14.4.3. Этап извлечения шаблонов

Для извлечения шаблонов из информации об использовании Web-ресурсов применяются различные методы как классической статистики, так и относящиеся к области Data Mining.

Методы статистики часто используются для анализа посещения сайтов и трафиков. Так, анализ сессионных файлов позволяет выполнить различные виды дескриптивного статистического анализа (вычислить частоту, матема-

тическое ожидание, дисперсию и т. п.) для просмотров страниц, времени просмотра и длины навигационного пути. Многие инструменты анализа трафика позволяют получить такие характеристики, как наиболее часто посещаемые страницы, среднее время посещения страниц или средняя длина пути перемещения по страницам. Подобные отчеты могут содержать низкоуровневые ошибки, связанные с невозможностью определить неавторизированные точки входа или недействительные URI.

Такой вид получаемых знаний может быть весьма полезным для улучшения производительности систем, повышения безопасности систем, решения задач модификации сайтов и обеспечения поддержки для решения маркетинговых задач.

Методы генерации ассоциативных правил могут быть использованы для выявления наиболее часто совместно запрашиваемых страниц, объединенных одной серверной сессией. Эти страницы могут быть связаны не напрямую друг с другом (т. е. не иметь прямых ссылок друг на друга). Например, алгоритмом Apriori может быть обнаружена корреляция между пользователем, посетившим страницу с электронной продукцией, и пользователем, просматривающим страницу со спортивным оборудованием.

Наличие или отсутствие таких правил в области бизнеса и маркетинга может помочь Web-дизайнерам перестроить Web-сайт. Ассоциативные правила могут также служить в качестве эвристических правил, по которым выполняется упреждающая выборка документов, для уменьшения времени ожидания загрузки страниц с удаленного сайта.

Методы кластеризации в области исследования использования Web-ресурсов применяются как для кластеризации пользователей, так и для кластеризации страниц. Кластеризация пользователей позволяет группировать пользователей с похожим поведением просмотра страниц. Такие знания полезны для того, чтобы сделать выводы о демографии пользователей и выполнении маркетинговой сегментации рынка в электронной коммерции или обеспечении пользователей персональным Web-контентом.

Кластеризация страниц позволяет выявить группы страниц с близким по смыслу содержимым. Эта информация полезна для поисковых машин и персональных ассистентов. В обоих случаях пользователям могут предлагаться гиперссылки в соответствии с их запросом и историей запрашиваемой информации.

Методы классификации могут быть использованы для развития профилей пользователей, относящихся к определенному классу или категории. Это требует построения и выбора функции, которая бы наилучшим образом описывала свойства данного класса. Для этого могут быть использованы любые методы классификации: деревья решений, метод Байеса, SVM и др. Например,

методы классификации позволяют обнаружить следующее правило: 30 % пользователей, разместивших заказ в разделе "Продукты/Музыка", имеют возраст от 18 до 25 лет и проживают в крупных городах.

Методы обнаружения шаблонов в последовательностях применяются для выявления межсессионных шаблонов, в которых элементы следуют друг за другом в упорядоченном по времени множестве сессий и эпизодов. Данный подход может помочь в прогнозировании структуры будущих посещений, что в свою очередь помогает в правильном размещении рекламы, нацеленной на конкретную пользовательскую аудиторию. Также методы временного анализа могут быть использованы для анализа трендов, обнаружения точек изменения и анализа подобия.

Построение модели зависимостей также может широко использоваться применительно к Web. Целью такого анализа является разработка модели, включающей в себя наиболее значимые зависимости между разными переменными в области Web. Так, например, может быть построена модель зависимости между этапами, которые проходит посетитель, и фактом совершения покупки в интернет-магазине (т. е. модель, отличающая случайного посетителя от потенциального покупателя). Существует несколько вероятностных методов обучения модели, которые могут быть использованы для построения модели поведения пользователя при просмотре Web, включая скрытые модели Маркова (Hidden Markov Models) и Байесовские сети доверия (Bayesian Belief Networks).

Моделирование использования Web-ресурсов позволяет обеспечить не только теоретическую основу поведения пользователей, но и помочь в предсказании. Это способствует увеличению продаж продуктов, размещаемых на сайте, а также улучшению навигации.

### 14.4.4. Этап анализа шаблонов и их применение

Последним этапом в исследовании использования Web-ресурсов является анализ извлеченных шаблонов. Целью анализа является отфильтровать наиболее интересные шаблоны и отбросить ничего не значащие шаблоны. Методология анализа во многом зависит от области применения, в которой он выполняется. Более общей формой анализа шаблонов является механизм запроса знаний, такой как SQL. Другой метод заключается в загрузке данных в куб данных для применения к нему OLAP-операций. Методы визуализации, такие как раскрашивание или графическое изображение шаблонов, могут выделять характерные шаблоны или тренды в данных. Контент или информация о структуре может быть использована для фильтрации страниц, используе-

мых определенным образом, содержащих информацию определенного типа или страницы, имеющие определенную структуру гиперссылок.

В литературе [60] приводится классификация существующих систем анализа использования Web-ресурсов. Классификация выполняется по пяти характеристикам:

- источнику данных: сторона сервера, сторона клиента и прокси;
- типу данных: структуры, контент и информация об использовании;
- количеству пользователей: однопользовательские и многопользовательские;
- количеству сайтов: один или множество сайтов;
- области применения.

Выделяют следующие области применения систем анализа использования Web-ресурсов.

- *Персонализация* (Personalization) — обеспечивает для каждого пользователя индивидуальный подход и является одной из важнейших задач для многих Web-систем (например, систем электронной коммерции). Такая персонализация позволяет давать наиболее эффективные рекомендации пользователям в достижении их целей.
- *Улучшение систем* (System Improvement) — анализ использования Web-ресурсов позволяет рассматривать (выявлять закономерности и взаимосвязи) изменения трафика, обращения к страницам, поведение пользователей. Результаты анализа могут быть применены для разработки политики кэширования, балансировки нагрузки и распределения данных. Это позволяет повысить производительность систем. Кроме того, выявление в закономерностях поведения пользователей позволяет выявлять атаки на сайты и тем самым повышать их безопасность.
- *Модификация сайтов* (Site Modification) — анализ использования Web-ресурсов обеспечивает дизайнера сайта своего рода обратной связью от пользователей и информацией, необходимой для принятия решения об изменении структуры и его содержания.
- *Бизнес-интеллект* (Business Intelligence) — выполняет анализ информации об использовании пользователями данных с Web-сайтов, совместно с маркетинговой информацией из электронной коммерции. Выделяют четыре задачи, решаемых в этой области: привлечение новых пользователей, удержание пользователей, проведение перекрестных продаж и определение отказа пользователя от Web-сайта.

# Выводы

По результатам данной главы можно сделать следующие выводы.

- Web Mining включает в себя следующие этапы: поиск ресурсов, извлечение информации, обобщение и анализ.
- Различают следующие категории задач Web Mining: извлечение Web-контента, извлечение Web-структур и исследование использования Web-ресурсов.
- Извлечение Web-контента может проводиться в целях информационного поиска и с целью сохранения его в базе данных, также различают неструктурированные и слабоструктурированные Web-документы.
- Для извлечения Web-контента из неструктурированных документов используют модели их представления и методы, заимствованные из Text Mining.
- Извлечение Web-контента для формирования базы данных делится на три подзадачи: моделирование и формирование запросов к Web, извлечение информации и интеграция, создание и реструктуризация Web-сайтов.
- В задаче извлечения Web-структур для представления Web используют направленные и ненаправленные графы.
- Выделяют три основных класса задач, которые могут быть решены анализом Web-структур: оценка важности структуры, поиск на основе информации о гиперссылках и кластеризация.
- В решении задачи извлечения структуры Web используются подходы из области социальных сетей, библиометрики, ранжирования документов и т. п.
- В задаче исследования использования Web анализу подвергаются вторичные данные о взаимодействии пользователя с Web: протоколы работы, куки, авторизация и т. п.
- Существуют два основных подхода анализа использования Web-ресурсов: преобразование данных использования Web-сервера в реляционные таблицы до выполнения адаптированных методов Data Mining и использование информации из файла протокола непосредственно, применяя специальные методы предварительной обработки.

# ГЛАВА 15

# Средства анализа процессов — Process Mining

## 15.1. Автоматизация выполнения бизнес-процессов

### 15.1.1. Бизнес-процессы

Бизнес на сегодняшний день представляет собой достаточно сложный процесс, включающий в себя разного рода связанных друг с другом активностей, в том числе взаимодействующих с другими участниками бизнеса. Двести лет назад Адам Смит предложил разбивать индустриальное производство на простейшие и базовые операции. Он показал, что разделение труда способствует росту его производительности. Современные экономисты предлагают для облегчения понимания и управления бизнесом объединять разрозненные операции в единые процессы, называемые бизнес-процессами.

Впервые понятие бизнес-процесса было введено Майклом Хаммером и Джеймсом Чампи в работе "Реинжиниринг корпорации: Манифест революции в бизнесе" [123]. Они дали следующее определение:

*Бизнес-процесс — это совокупность различных видов деятельности, в рамках которой "на входе" используются один или более видов ресурсов, и в результате этой деятельности на "выходе" создается продукт, представляющий ценность для потребителя.*

Существует достаточно много определений данного термина, отличающихся тонкостями, но совпадающих в главном:

*Бизнес-процесс (БП) — последовательность операций, в ходе выполнения которых получается значимый для организации результат (продукты, услуги).*

Примерами таких бизнес-процессов являются: кредитные запросы, требования по страховке, электронная коммерция и др.

Автоматизация выполнения бизнес-процессов и привлечение информационных систем для их поддержки привели к тесной интеграции информационных технологий и бизнеса. В настоящее время широкую популярность получили следующие технологии:

- Business Process Management (BPM) — управление бизнес-процессами;
- Business Intelligence (BI) — бизнес-интеллект;
- Business Process Analysis (BPA) — анализ бизнес-процессов;
- Business Activity Monotoring (BAM) — мониторинг бизнес-деятельности.

Перечисленные технологии пересекаются и дополняют друг друга.

Представление БП в виде последовательности связанных друг с другом операций упростило задачу автоматизации бизнеса. Соответствующие информационные системы стали рассматривать бизнес-процессы как строго заданный поток работы. Системы такого класса получили название *системы Workflow*.

Согласно определению, данному некоммерческой коалицией WfMC (Workflow Management Coalition), объединяющей более 300 компаний *Workflow, — это автоматизация всего или части бизнес-процесса, в течение которого документы, информация или задачи передаются от одного участника к другому для их обработки в соответствии с набором процедурных правил*.

Сложность автоматизации бизнес-процесса связана с тем, что не все действия, выполняющиеся в его рамках, можно описать формально и, как следствие, автоматизировать. По степени формализуемости бизнес-процессы можно классифицировать на:

- структурированные;
- слабоструктурированные;
- неструктурированные.

*Структурированные процессы* предопределены заранее, и процесс их выполнения никогда не отклоняется от установленного порядка. Противоположностью данного типа процессов являются *неструктурированные процессы*, которые фокусируются на совместном использовании информации, и те, которые управляются данными. Их структура не может быть определена заранее, поэтому они не подходят для автоматического выполнения. С другой стороны, они дают больше свободы пользователям. *Слабоструктурированные процессы* находятся в промежутке между предыдущими двумя крайностями. Они имеют большую гибкость, чем структурированные процессы, т. к. их выполнение более неопределенно и им приходится иметь дело с исключе-

ниями и изменениями. Однако, в отличие от неструктурированных, они могут быть описаны с достаточной степенью формализации. Структура таких процессов моделируется заранее и скорее служит рекомендациями пользователям, но не строгим регламентом.

Можно сформулировать следующие требования, которые предъявляются к бизнес-процессу, чтобы его было возможно автоматизировать:

- *процесс можно выделить* из всей массы выполняемых на предприятии работ, заданий и действий;
- *процесс должен иметь структуру* — не быть вырожденным, состоящим из одной-единственной операции;
- *должны быть правила выполнения процесса*, которые можно сформулировать и формально описать, которые касаются последовательности выполнения операций, условий и предусмотренной реакции на внешние события;
- *процесс должен быть периодическим* — он должен повторяться на предприятии, иначе его не имеет смысл автоматизировать.

Если процесс удовлетворяет первым трем требованиям, то его можно формализовать. Последнее требование обеспечивает целесообразность автоматизации процесса.

## 15.1.2. Формализация бизнес-процессов

Как уже говорилось, поток работ (Workflow) — это формальное описание бизнес-процесса, которое используется для его автоматизации.

Организация WfMC выделяет следующие основные понятия, составляющие Workflow:

- *процесс* — состоит из шагов процесса или активностей, переходов, участников и данных процесса;
- *шаг процесса* — представляет собой часть работы, которая должна быть выполнена в рамках процесса;
- *переход* — является механизмом передачи управления от одного шага процесса к другому и представляет собой пару (шаг 1, шаг 2);
- *исполнитель* — участник процесса, который отвечает за запуск, выполнение и завершение некоторых шагов процесса;
- *данные* — переменные процесса, которые используются для инициализации процесса, а также хранения промежуточных результатов.

В зависимости от того, какие из перечисленных сущностей рассматривать как базовые — бизнес-процесс можно анализировать с разных точек зрения — перспектив:

- *перспектива управления потоком* (control-flow perspective) — соответствует набору шагов процесса;
- *перспектива данных* (data perspective) — соответствует набору переменных процесса, а также данных внешних информационных систем, которые используются при исполнении бизнес-процесса;
- *перспектива ресурсов* (resource perspective) — соответствует списку исполнителей, которые могут выполнить его шаги. При этом исполнителями могут быть как люди, так и информационные системы или специализированные устройства;
- *перспектива операций* (operational perspective) — соответствует списку элементарных действий, совершаемых исполнителями в рамках шага.

На практике в настоящее время широко используется только перспектива управления потоком. Рассмотрим ее более подробно.

Перспективу управления потоком можно представить как направленный граф. Узлы графа могут соответствовать двум сущностям бизнес-процесса: шагу процесса или маршрутному узлу. Узлы графа соединяются дугами, соответствующими переходам бизнес-процесса. По переходам перемещается точка управления (указатель на активный узел процесса), руководствуясь правилами в маршрутных узлах.

В узле, соответствующем шагу процесса, происходит выдача задания исполнителю (человеку или информационной системе) и ожидание ответа (сообщения о том, что работа выполнена). К узлу, соответствующему шагу процесса, может примыкать только один входящий и один исходящий переход.

Маршрутный узел соответствует разветвлению/слиянию точек управления. В таких узлах на основании содержащихся в маршрутных узлах правил выбирается следующий узел (узлы), в который будет передано управление. Поэтому с ними обязательно связано более одного входящего или исходящего перехода.

В выполняющемся бизнес-процессе одновременно может быть несколько точек управления. В соответствии с бизнес-логикой процесса точка управления в маршрутном узле может разделиться на несколько точек управления. Кроме того, точки управления могут ожидать друг друга в маршрутном узле и объединяться в нем в одну точку.

Описанный граф обычно представляется в виде сети Петри. Пример перспективы управления потоком в формализме сетей Петри изображен на рис. 15.1.

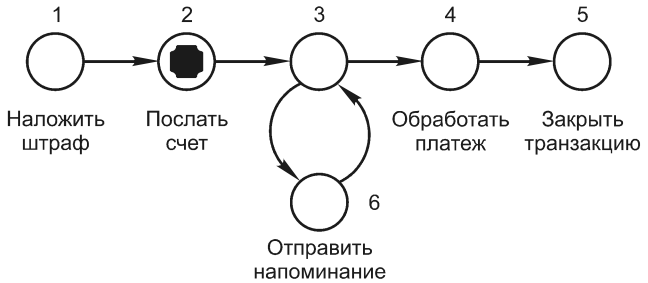



**Рис. 15.1.** Пример перспективы управления потоком бизнес-процесса:
1, 2, 4, 5, 6 — узлы, соответствующие шагам бизнес-процесса; 3 — маршрутный узел, в котором по определенным правилам принимается решение о переходе к узлу 4 или узлу 6. Узел 2 в текущий момент времени является активным, т. к. в нем расположена точка управления

## 15.1.3. Workflow-системы

Workflow-системы работают по одному общему принципу. Они получают на вход описание бизнес-процесса на формальном языке. Такое описание называется *схемой*. Как правило, схема представляется в графическом виде и является направленным графом (рис. 15.2).

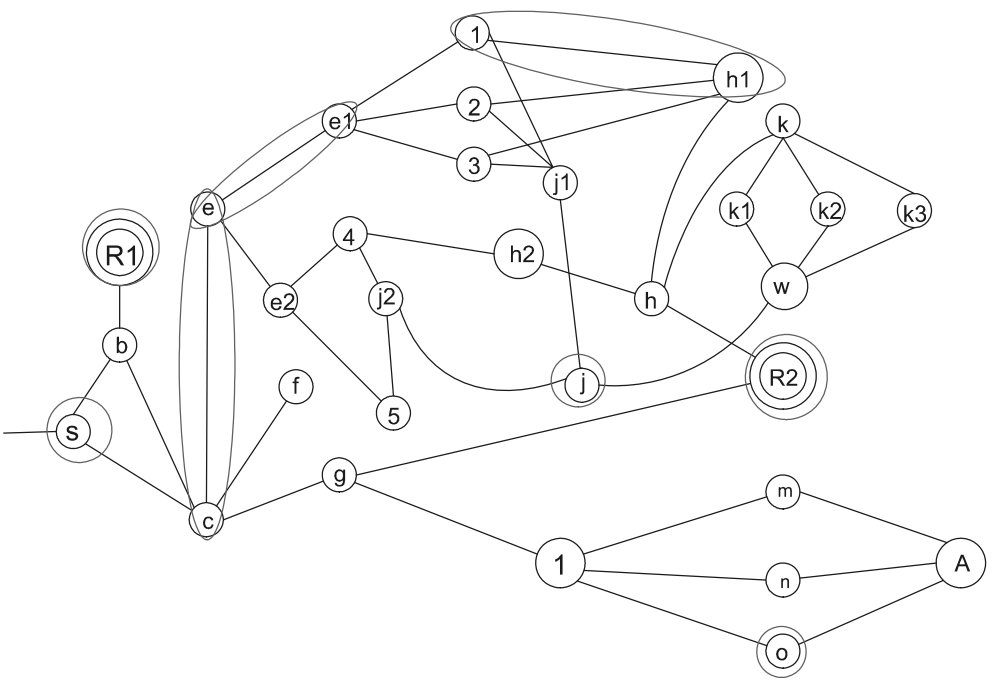



**Рис. 15.2.** Пример Workflow-схемы

По описанной схеме система переводит процесс из одного узла в другой. В зависимости от действий, выполняемых исполнителями системы, значения переменных процесса могут быть разными, следовательно, переходы в узлах маршрутизации будут отличаться. Таким образом, реальное выполнение процесса будет охватывать не все узлы схемы.

Узлы схемы, участвующие в выполнении реального процесса, называются *экземпляром процесса*. Системы класса Workflow обеспечивают выполнение множества экземпляров процесса, в том числе и одновременное. На рис. 15.3 представлен пример экземпляров процесса, отмеченных на схеме серыми и черными линиями.

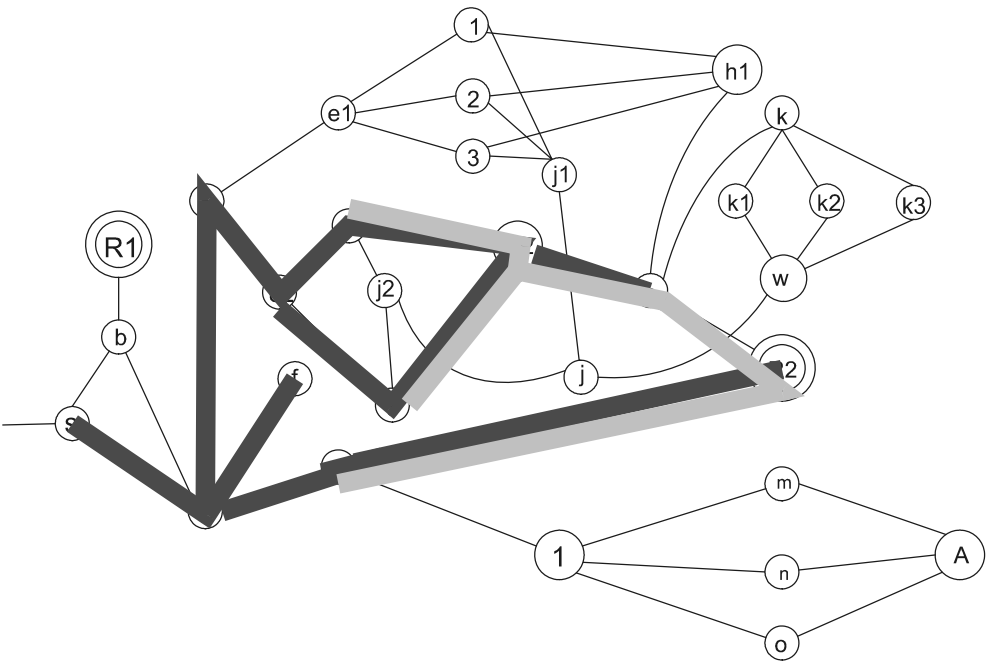



**Рис. 15.3.** Пример экземпляров процесса

## 15.1.4. Сервисно-ориентированная архитектура

В настоящее время все более популярной становится сервисно-ориентированная архитектура (service-oriented architecture — SOA). Основные принципы, лежащие в ее основе, во многом совпадают с принципами Workflow-систем. По этой причине последние все больше вытесняются системами, построенными на принципах SOA.

*Сервисно-ориентированная архитектура* — подход к разработке программного обеспечения, основанный на использовании сервисов (служб) со стандартизированными интерфейсами.

Компоненты систем могут быть распределены по разным узлам сети и предлагаются как независимые, слабо связанные, заменяемые сервис-приложениями.

Наиболее широко компоненты таких систем реализуются как Web-сервисы. *Web-сервис* (web service) — программная система, идентифицируемая строкой URI, чьи публичные интерфейсы и привязки определены и описаны языком XML. Описание этой программной системы может быть найдено другими программными системами, которые могут взаимодействовать с ней согласно этому описанию посредством сообщений, основанных на XML и передаваемых с помощью интернет-протоколов.

Концепция сервис-ориентированной архитектуры подразумевает, что отдельные Web-сервисы обладают определенной ограниченной функциональностью. А для решения более-менее сложных задач требуется использовать функциональность нескольких сервисов. Поэтому для описания взаимодействия сервисов и последовательности их выполнения используют *оркестровки* (Web Service Choreography).

Для описания систем и приложений, построенных по принципам сервисно-ориентированной архитектуры, на уровне модели бизнес-процесса ведущими IT-компаниями предлагались различные проекты стандартов:

- Wf-XML (от Workflow Management Coalition);
- WSFL (IBM Web Services Flow Language);
- XLANG (Microsoft's XLANG: Business modeling language for BizTalk);
- PIPs (RosettaNet's Partner Interface Process);
- и др.

К настоящему моменту наибольший вес имеют:

- BPEL4WS (Business Process Execution Language for Web Services), подготовленный IBM, Microsoft и BEA Systems;
- WSCI (Web Service Choreography Interface) корпорации Sun Microsystems.

Оба этих проекта были приняты в организации, стандартизирующей технологии архитектуры Web-сервисов. WSCI с 2002 г. развивается рабочей группой консорциума W3C (организована рабочая группа Web Services Choreography Working Group). Для развития BPEL4WS в 2003 г. в консорциуме OASIS был создан технический комитет — OASIS Web Services Business Process Execution Language TC (WS-BPEL TC).

Стандарты оркестровки опираются на стандартизованное описание Web-сервисов — WSDL. Web-сервисы, участвующие во взаимодействии, представлены своими интерфейсами.

### 15.1.5. Проектирование бизнес-процессов

Независимо от средств автоматизации бизнес-процессов (Workflow-системы или SOA) процесс постановки бизнес-процесса без привлечения автоматизированных средств достаточно трудоемок.

Жизненный цикл бизнес-процесса состоит из четырех этапов:

1. Проектирование процесса.
2. Конфигурирование процесса.
3. Выполнение процесса.
4. Диагностика процесса.

За проектирование процесса отвечает *бизнес-эксперт*. Он на основании своих знаний о предметной области и процессах, протекающих в ней, строит модель бизнес-процесса.

На этапе конфигурирования выполняется настройка построенной на предыдущем этапе модели для конкурентных условий эксплуатации. При этом учитываются ограничения и особенности системы, автоматизирующей данную модель.

Во время выполнения процесса его исполнители, взаимодействуя с информационной системой, решают автоматизированные задачи. При этом все экземпляры реально протекающих процессов сохраняются информационной системой.

Используя эту информацию, можно на этапе диагностики выполнить анализ реальных процессов в сравнении с построенной моделью. Такой анализ может позволить выявить отклонение модели от реальных процессов, пути улучшения или оптимизации модели и т. п.

На основании результатов анализа в модель могут быть внесены изменения. Таким образом, опять начинается этап проектирования и весь цикл повторяется.

## 15.2. Анализ процессов

### 15.2.1. Технология Process Mining

В программных средствах, автоматизирующих бизнес-процессы, вся информация о выполненных экземплярах процесса записывается в протоколы работы.

За все время работы программных средств протоколы работы накапливают большой объем информации о реальных процессах, выполняемых в компании. Безусловно, данная информация является ценной, а ее анализ позволяет извлечь новые знания о бизнес-процессах. Для этого к регистрационным журналам применяются адаптированные методы Data Mining.

Применение методов Data Mining для анализа информации о реальных процессах, выполняемых системами, автоматизирующими бизнес-процессы, получило в литературе название Process Mining. Часто в литературе также встречается и понятие Workflow Mining. Многие авторы сходятся во мнении, что эти два термина являются синонимами. Однако существуют и те, кто вкладывает в эти термины разный смысл. Они дают следующие определения:

- Workflow Mining — технология выявления часто встречающихся экземпляров процессов (шаблонов) из протоколов работы систем;
- Process Mining — технология построения формальных моделей экземпляров процессов по протоколам работы систем.

Как видно, данные определения действительно очень близки по смыслу. В обоих случаях требуется построение формального описания реального процесса, выполненного системой, по информации, представленной в ее протоколе работы. Это более близко к понятию Process Mining, поэтому в дальнейшем мы будем пользоваться данным термином.

Основная идея методов Process Mining изображена на рис. 15.4. Методы Process Mining применяются к протоколам работы информационных систем. В них отражается реальное выполнение бизнес-процессов через взаимодействие их исполнителей с информационными системами. Применение к ним методов Process Mining позволяет автоматически построить модели бизнес-процессов.

Построенные таким образом модели бизнес-процессов отражают реальность и доступны для восприятия и анализа человеку. На основании их анализа могут приниматься решения о внесении изменений в бизнес-процессы и/или о модернизации и настройке информационной системы.

Process Mining находит широкое применение во многих областях бизнеса и управления, т. к., в отличие от обычного моделирования, получаемые с помощью алгоритмов Process Mining модели строятся на основе записей о реально происходивших событиях, поэтому более адекватно отображают действительность и несут информацию о том, что происходило в реальности, а не о том, что было запланировано. Кроме того, в некоторых областях моделирование может быть затруднено или очень трудоемко, тогда как Process Mining позволяет автоматизировать процесс.

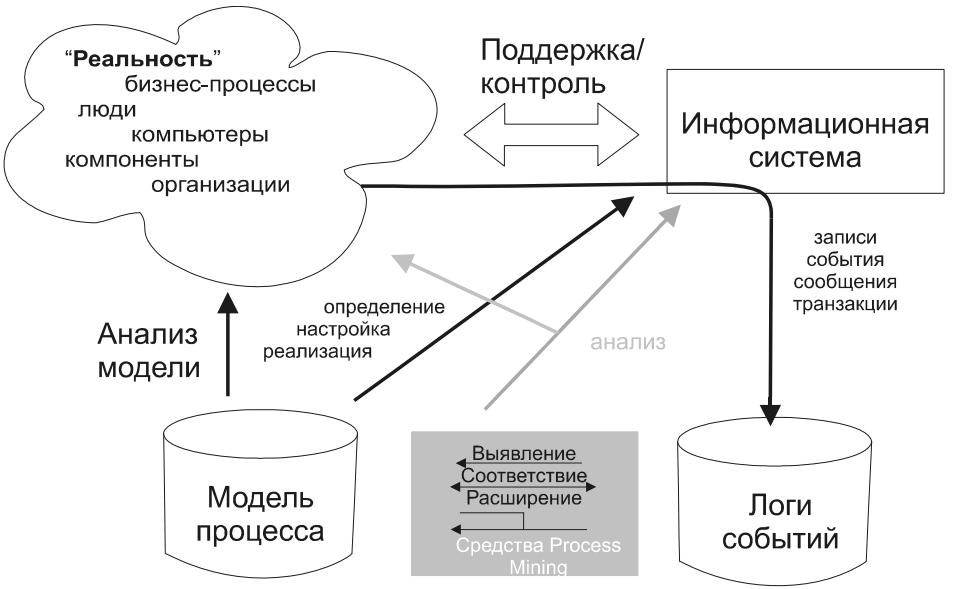



**Рис. 15.4.** Применение Process Mining

## 15.2.2. Анализ протоколов

Как видно из предыдущего раздела, источником информации для технологии Process Mining является протокол работы информационных систем. Однако для анализа может быть использован не каждый протокол. Нужно, чтобы в нем была информация, достаточная для применения методов Process Mining. К необходимым требованиям можно отнести следующее:

- все события, записанные в протоколе, должны быть идентифицированы с экземплярами процессов;
- все события должны быть упорядочены по времени их выполнения;
- разнотипные события должны различаться.

В табл. 15.1 приведен пример протокола работы информационной системы. В нем о каждом событии записана следующая информация: описание действия, шаг процесса, пользователь, инициировавший событие, время записи события. Кроме того, все действия сгруппированы по экземплярам процессов.

В данном протоколе все требования выполнены. Однако часто не требуется столь подробная информация. Для методов Process Mining, как правило, не имеет значения, кто выполнил те или иные действия. В табл. 15.2 представлен пример более простого протокола, к которому применимы методы Process Mining.

**Таблица 15.1.** Протокол работы системы Staffware

| Описание действия | Шаг процесса | Пользователь | yyyy/mm/dd/hh: mm |
|---|---|---|---|
| *Экземпляр 10* | | | |
| | Начало | bvdongen@staffw_e | 2002/06/19 12:58 |
| Регистр | Обработанный | bvdongen@staffw_e | 2002/06/19 12:58 |
| Рассмотрение анкеты | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 13:00 |
| Оценка | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 13:00 |
| Архив | Обработанный | bvdongen@staffw_e | 2002/06/19 13:00 |
| Архив | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 13:00 |
| | Закончено | | |
| *Экземпляр 9* | | | |
| | Начало | bvdongen@staffw_e | 2002/06/19 12:36 |
| Регистр | Обработанный | bvdongen@staffw_e | 2002/06/19 12:36 |
| Отправка анкеты | Обработанный | bvdongen@staffw_e | 2002/06/19 12:36 |
| Оценка | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 12:37 |
| Процесс выполнения | Обработанный | bvdongen@staffw_e | 2002/06/19 12:37 |
| Процесс выполнения | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 12:37 |
| Архив | Близко выпущенный | bvdongen@staffw_e | 2002/06/19 12:38 |
| | Закончено | | |

**Таблица 15.2.** Технологический процесс

| Идентификатор случая | Идентификатор задачи |
|---|---|
| Случай 1 | Задача A |
| Случай 2 | Задача A |
| Случай 3 | Задача A |
| Случай 3 | Задача B |
| Случай 1 | Задача B |

**Таблица 15.2** *(окончание)*

| Идентификатор случая | Идентификатор задачи |
|---|---|
| Случай 1 | Задача C |
| Случай 2 | Задача C |
| Случай 4 | Задача A |
| Случай 2 | Задача B |
| Случай 2 | Задача D |
| Случай 5 | Задача A |
| Случай 4 | Задача C |
| Случай 1 | Задача D |
| Случай 3 | Задача C |
| Случай 3 | Задача D |
| Случай 4 | Задача B |
| Случай 5 | Задача E |
| Случай 5 | Задача D |
| Случай 4 | Задача D |

В данном протоколе также выполнены все перечисленные ранее требования. В результате по нему может быть построена модель процесса, представленная на рис. 15.5.

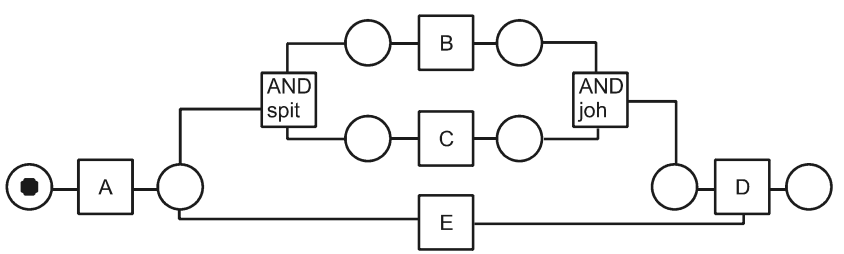



**Рис. 15.5.** Модель процесса, представленного в табл. 15.2

## 15.2.3. Стандарт записи протоколов MXML

Для закрепления требований и унификации протоколов, обрабатываемых алгоритмами Process Mining, был предложен стандарт записи протоколов MXML (Mining XML).

*MXML* — это расширяемый формат, основанный на языке разметки XML (eXtensible Markup Language). Он используется для представления и хранения информации в виде логов событий. Формат фокусируется на ключевой информации, необходимой для применения методов Process Mining, однако существует возможность расширения формата для записи дополнительной информации.

Структура MXML-формата изображена на рис. 15.6 в виде диаграммы классов.

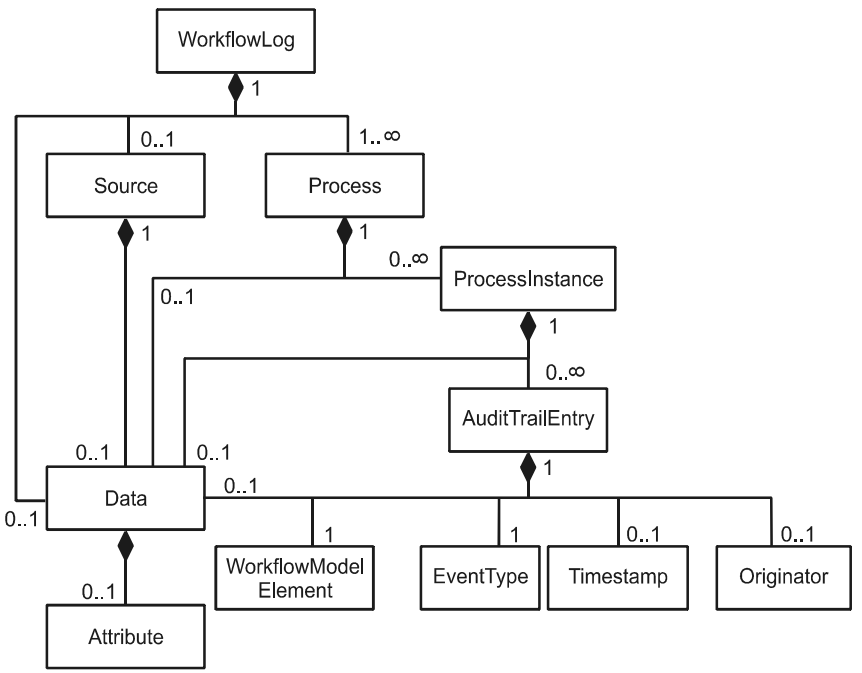



**Рис. 15.6.** UML-диаграмма MXML-формата

Корневым узлом каждого MXML-документа является `WorkflowLog`, представляющий лог-файл. Каждый `WorkflowLog` может содержать произвольное количество узлов `Process`. Каждый элемент типа `Process` является группой событий, которые произошли в течение выполнения какого-либо процесса. Однократные выполнения этого процесса представлены элементами типа `ProcessInstance`. Таким образом, каждый `ProcessInstance` представляет собой однократное протекание процесса. Каждый `ProcessInstance` содержит группу из произвольного количества элементов типа `AuditTrailEntry` (контрольные записи), каждая из которых соответствует уникальному событию в логе. Каждая контрольная запись должна содержать как минимум два элемента:

`WorkflowModelElement` (название задачи, которая была выполнена) и `EventType` (тип события, который описывает стадию выполнения задачи). Формат также поддерживает два необязательных, но, тем не менее, часто встречающихся поля. `Timestamp` содержит точную дату и время, когда событие произошло, а `Originator` идентифицирует ресурс, т. е. человека или информационную систему, которые являлись инициатором события. Расширяемое поле `Data` может содержать произвольное количество атрибутов, которые являются парами строк <название-значение>.

## 15.2.4. Задачи Process Mining

Главной целью Process Mining является автоматизированное выявление перспектив бизнес-процессов, и их представление в виде понятных человеку моделей. Это позволяет облегчить решение следующих задач, встающих в области автоматизации бизнес-процессов:

- построение модели процесса на основании имеющегося лога событий работающей информационной системы;
- проверка соответствия реального экземпляра процесса базовому;
- автоматическое восстановление систем после сбоев;
- улучшения и расширение процессов.

Построение модели бизнес-процесса позволяет упростить задачу формализации процессов и их автоматизации. При проектировании и создании Workflow-систем описание потока работ возлагается на эксперта в данной предметной области. Это достаточно сложная задача, и от качества ее решения во многом зависит успешность внедрения системы. Человек, описывающий процесс, должен не только очень хорошо представлять его себе, но и суметь его формализовать. Задача усложняется тем, что такие автоматизируемые процессы охватывают разные виды деятельности, в которых экспертами являются несколько человек.

Кроме сложности построения модели потока работ, возникает проблема, связанная с ее неактуальностью. Дело в том, что при описании модели, как правило, описывается то, "как должно работать", а не "как работает на самом деле".

При проверке процесса модель уже существует, требуется сравнить ее с логом событий и выявить различия. Проверка соответствия позволяет выявить отклонения, определить их местоположение в модели и оценить степень их серьезности.

Выявление отклонений реальных процессов может быть использовано для восстановления информационных систем после аварийных остановок. Анализ протоколов позволяет определить, что выполнялось системой до ее ава-

рийного завершения. Выявленные отклонения могут помочь в определении причин сбоя, а также восстановить результаты работы системы после ее восстановления.

Улучшение и расширение процессов также предполагает, что модель уже существует, требуется дополнить ее другими аспектами, или перспективами. Выявление шаблонных подпроцессов в реальных процессах может помочь в оптимизации как бизнес-процессов, так и модели. Например, работы, выполняемые, как правило, друг за другом, могут быть объединены внутри одного отдела или даже поручены одному человеку, что сократит время на коммуникацию и повысит эффективность управления. Также можно спроецировать на модель данные о производительности, что позволит выявить "узкие места" модели. Другим хорошим примером является алгоритм построения деревьев решений, который анализирует каждую точку выбора на заданной модели, просматривая лог и определяя, какая информация обычно известна на момент выбора. После этого используются классические методы Data Mining для определения того, какие именно элементы информации влияют на выбор. В результате строится дерево решений процесса.

## 15.2.5. Проблемы анализа протоколов

Анализ протоколов работы представляет собой достаточно сложный процесс. Во-первых, необходимо наличие полезной информации в протоколе, которая впоследствии извлекается.

Во-вторых, определенную сложность представляет разнообразие типов элементов схем потоков работ. Выделяют следующие типы элементов схемы:

- *последовательности* — представляют собой ситуации, когда задачи выполняются в заранее установленном порядке, одна за другой. Например, для модели на рис. 15.7 задачи "Открыть сайт" и "Просмотреть товары" образуют последовательность;
- *параллелизм* — означает, что выполнение двух и более задач независимо друг от друга и происходит параллельно (задача "Заполнить форму" может быть выполнена независимо от задач "Аутентификация" и "Создать учетную запись" для модели на рис. 15.7);
- *выбор* — это ситуация, когда выполняется только одна из задач, которые имеют такую возможность (задачи "Совершить покупку" и "Отменить покупку" на рис. 15.7);
- *циклы* — показывают, что некоторые части процессов повторяются много раз. Один из таких блоков образуют задачи "Просмотреть товары", "Добавить товар в корзину", "Вычислить сумму" на рис. 15.7;

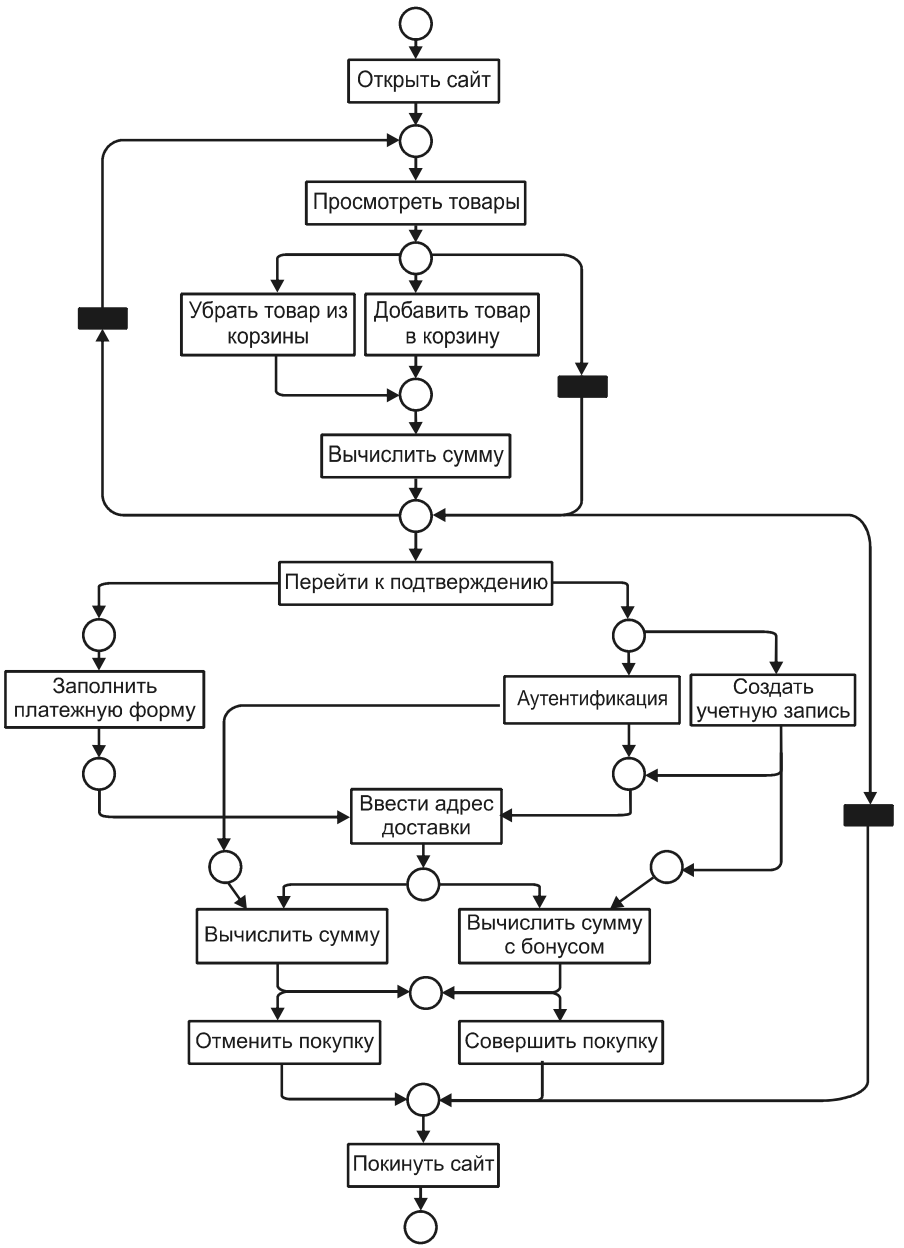



**Рис. 15.7.** Пример сети Петри, содержащей все типовые конструкции управления потоком, которые могут встретиться в бизнес-процессах

- *принудительный выбор* — это смесь синхронизации и выбора, под данную категорию подпадает, например, конструкция из "Вычислить сумму" и "Вычислить сумму с бонусом";
- *невидимые задачи* — это переходы, используемые для преодоления ограничений формализма сетей Петри, они не имеют соответствующих им событий в логе (черные прямоугольники на модели);
- *дублируемые задачи* — относятся к ситуациям, когда несколько задач имеют одинаковые метки (задача "Вычислить сумму" на модели).

В настоящее время ни один алгоритм Process Mining не обрабатывает все конструкции одинаково полноценно.

Еще одной проблемой, затрудняющей работу методов Process Mining, является шум, имеющийся в протоколах. Шум может появиться в двух ситуациях: события по каким-либо причинам были некорректно зарегистрированы (например, из-за временных сбоев в системе) или по причине возникновения исключительных ситуаций и записи о них в протокол. Присутствие шума может препятствовать корректному анализу протокола.

# 15.3. Методы Process Mining

## 15.3.1. Первые вероятностные методы Process Mining

Первые работы по Process Mining появились в 1995 г. Их авторами были Джонатан Кук (Jonathan E. Cook) и Александр Вольф (Alexander L. Wolf). Они были направлены на извлечение моделей процессов работы из протоколов работы в контексте программной инженерии. Данную технологию они назвали *исследование процессов* (process discovery).

Их основной целью было не построить полную и корректную модель процесса, а извлечь из протоколов работы наиболее часто встречаемые шаблоны. Извлекаемые шаблоны описывались как конечный автомат. В своих работах [124—128] они представили три алгоритма: RNet, KTail и Markov.

Из них только алгоритм Markov полностью разработан Куком и Вольфом, остальные являются адаптацией известных технологий к анализу протоколов работы. Алгоритм RNet использует нейронные сети. KTail реализует алгоритмический подход. Markov основывается на статическом и алгоритмическом подходах. Алгоритм Markov превосходит оба алгоритма. Наихудшие результаты показал алгоритм RNet.

Марковский алгоритм (Markov) использует теорию Марковских дискретных случайных процессов для нахождения наиболее вероятных последовательностей событий, после чего алгоритмически преобразует эти вероятности в со-

стояния и переходы между состояниями. В итоге он строит граф состояний в виде конечного детерминированного автомата.

Основная идея алгоритма Маркова заключается в использовании вероятностей последовательностей событий. В процессе выполнения алгоритма создаются таблицы вероятностей для последовательностей событий путем подсчета числа появлений одинаковых последовательностей в потоке событий. Далее те последовательности, вероятность и число появлений которых ниже установленного пользователем порога, отсекаются, а оставшиеся используются для создания конечного автомата.

Алгоритм включает следующую последовательность шагов.

**Шаг 1.** На первом шаге строятся таблицы вероятностей последовательностей событий путем прохода по потоку событий. При этом для каждой последовательности событий подсчитывается частота и число ее появлений в потоке событий.

Пример таблицы вероятностей для последовательности из двух событий приведен в табл. 15.3, для трех событий — в табл. 15.4. Последовательность соответствует комбинации строки и колонки, на пересечении в таблице указывается частота ее появления в протоколе. Так, для последовательности RC частота равна 0.5, а для последовательности RCE — 1.

**Таблица 15.3**

| | **R** | **C** | **E** |
|---|---|---|---|
| **R** | 0.00 | 0.50 | 0.50 |
| **C** | 0.54 | 0.00 | 0.46 |
| **E** | 0.42 | 0.58 | 0.00 |

**Таблица 15.4**

| | **R** | **C** | **E** |
|---|---|---|---|
| **RR** | 0.00 | 0.00 | 0.00 |
| **RC** | 0.00 | 0.00 | 1.00 |
| **RE** | 0.50 | 0.50 | 0.00 |
| **CR** | 0.00 | 0.00 | 1.00 |
| **CC** | 0.00 | 0.00 | 0.00 |
| **CE** | 0.33 | 0.67 | 0.00 |
| **ER** | 0.00 | 1.00 | 0.00 |
| **EC** | 1.00 | 0.00 | 0.00 |
| **EE** | 0.00 | 0.00 | 0.00 |

**Шаг 2.** На основе таблиц вероятностей создается направленный граф, именуемый *графом событий*. Он строится следующим образом: каждому типу событий сопоставляется своя вершина, далее для каждой последовательности событий, вероятность и число появлений которой превышают заданный

пользователем порог, создается уникально именуемое ребро от одного элемента последовательности до непосредственно следующего за ним элемента в этой последовательности.

Для последовательностей, приведенных в табл. 15.3, граф событий будет выглядеть так, как он изображен на рис. 15.8. Дуги помечены цифрами.

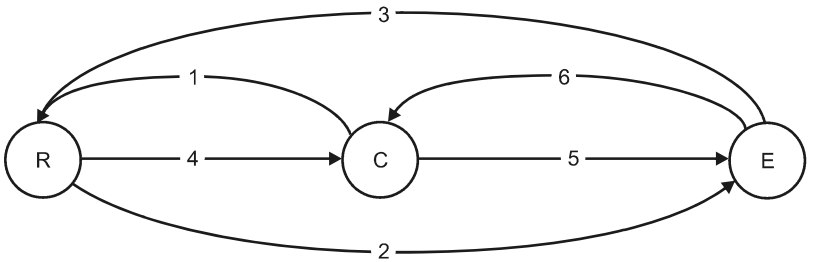



**Рис. 15.8.** Пример графа событий

**Шаг 3.** Предыдущий шаг может привести к появлению вершин с лишними ребрами, показывающими реально несуществующие пути. Для решения этой проблемы подобные вершины разбиваются на две или более вершин. Это делается путем поиска непересекающихся наборов входящих и выходящих ребер, которые имеют ненулевую вероятность, после чего вершина разбивается на набор вершин, количество которых соответствует количеству наборов.

В нашем примере после предыдущего шага на графе лишними несуществующими путями являются RCR, CRC, ERE, ECE. Для них в таблице вероятности значения равны 0, однако на графе они присутствуют. Чтобы устранить это несоответствие, вершины C и R разбиваются на две новых вершины. В результате, например, для новых вершин C к одной отходит ребро C-E, а к другой E-C, что в свою очередь убирает путь E-C-E с графа (рис. 15.9).

**Шаг 4.** Имеющийся граф событий $G$ преобразуется в граф $G'$ путем совершения следующих действий: каждому ребру в графе $G$ сопоставляется вершина в графе $G'$. Вершине присваивается уникальная метка, которая соответствовала ребру в графе $G$. Для каждой последовательности входящее ребро — вершина — исходящее ребро в графе $G$ создается ребро в графе $G'$ от вершины, соответствующей входящему ребру, до вершины, соответствующей исходящему ребру. Полученному ребру присваивается метка типа события, которому соответствовала вершина между входящим/исходящим ребрами в графе $G$ (рис. 15.10).

В нашем примере вершина 5 и прилегающие к ней ребра создаются из ребра с меткой "5" в графе событий, которое соединяет вершины C и E.

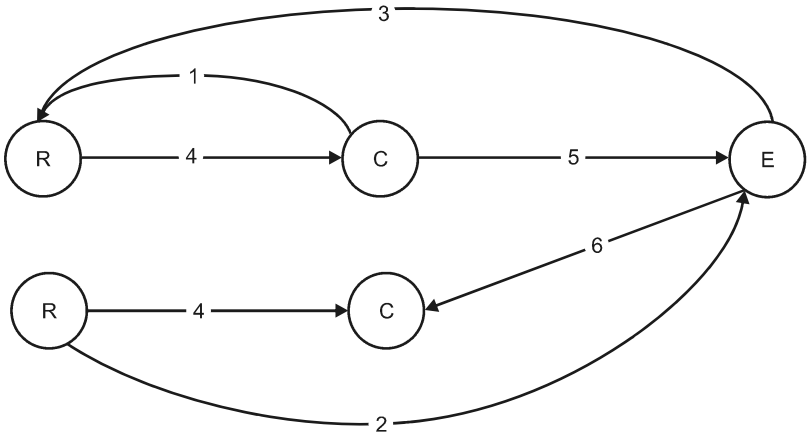



**Рис. 15.9.** Пример графа событий после преобразования

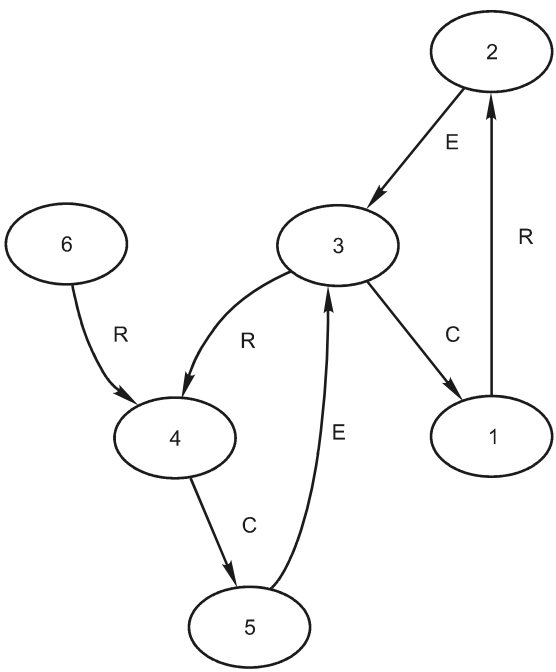



**Рис. 15.10.** Конечный детерминированный автомат, полученный в результате работы алгоритма Маркова

В работах [125, 128] авторы расширили алгоритм Маркова, чтобы извлекать параллельные элементы процессов. Было предложено идентифицировать элементы разделения и соединения потоков. Для этого проверяется таблица частот для непосредственного предшественника и следующего анализируе-

мого события. Кроме того, авторы описали четыре статических метрики, которые используются, чтобы различать элементы XOR/AND-split/join: энтропия, количество типов событий, причинная связь и периодичность.

Энтропия определяет, как соотносятся два типа событий. Это необходимо, чтобы установить прямого наследника и предшественника типа события.

Количество типов событий позволяет различить элементы AND/XOR-split/join. Обратите внимание, что AND-split — это элемент разделения, и его прямые последователи выполняются одновременно. В ситуации XOR-split время выполнения следующего элемента наступает после времени (или совпадает со временем) выполнения элемента XOR-split (при условии отсутствия шума в протоколе).

Причинная связь используется, чтобы отличить параллельные события и продолжительность циклов, состоящих из двух событий. Периодичность помогает идентифицировать точки синхронизации.

Благодаря вероятностной природе, подобные алгоритмы устойчивы к шумам.

Первым, кто применил технологию Process Mining к бизнес-системам, использующим в качестве Workflow-системы IBM MQ Series, был Агравал (Agrawal ) [129]. Предложенный им алгоритм предполагает, что протокол можно разбить на атомарные шаги. Извлеченная модель показывает зависимости между этими шагами, но не описывает семантики элементов разделитель/соединитель.

Данный подход требует цельную модель, имеющую начальную и конечную задачи. Это не является существенным ограничением, т. к. всегда можно предварительно добавить искусственную начальную и конечную задачи в начало и конец каждого экземпляра процесса. Данный алгоритм не обрабатывает дублируемые задачи и предполагает, что задача может появиться только один раз в экземпляре процесса. Поэтому для обработки циклов необходимо заново маркировать шаг процесса, преобразовав циклы в последовательности.

Как уже отмечалось, данный алгоритм нацелен на извлечение полной модели. Для этого он выполняет следующие шаги.

**Шаг 1.** Переименование повторяющихся меток в экземпляре процесса. Это позволяет корректно определить циклы.

**Шаг 2.** Построение графа зависимостей со всеми задачами, находящимися в протоколе. Здесь необходимо более детально рассмотреть, как устанавливается отношение зависимости. Алгоритм не только рассматривает появление задачи A рядом с B в одном и том же экземпляре процесса. Он рассматривает две задачи, которые могут также зависеть транзитивно во всем протоколе.

Так, если задача A следует за задачей C, а C за B, то B может зависеть от A, даже когда они никогда не появляются в том же самом экземпляре процесса.

**Шаг 3.** После построения графа отношения зависимости: удаление стрелки в обоих направлениях между двумя задачами; удаление строго связанных компонентов; применение транзитивного сокращения к подграфу (в главном графе), который представляет экземпляр процесса.

**Шаг 4.** Объединение перемаркированных задач в единый элемент.

Извлеченная модель имеет зависимости и булевы функции, ассоциированные с зависимостями. Эти булевы функции определяются, основываясь на других параметрах в протоколах. Для построения этих функций может использоваться классификационный алгоритм. В алгоритме шум обрабатывается сокращением дуг, которые появляются меньшее число раз, чем заданный порог.

Пинтер (Pinter) и Голани (Golani) в своих работах [130] развили подход Агравала. Главное отличие заключалось в том, что они рассматривают задачи, которые имеют начальное и конечное события. В этом случае обнаружение параллелизма становится более тривиальным.

Другое отличие заключается в том, что они рассматривают отдельные экземпляры процесса, чтобы установить отношения зависимости, в то время как Агравал и др. исследовали протокол как целое.

Хербст (Herbst) и Карагианнис (Karagiannis) предложили подход, который позволял обрабатывать дублирующиеся задачи [131, 132]. Они предложили три алгоритма: Merge-Seq, SplitSeq и SplitPar. Все три алгоритма могут извлекать модели с дублирующими задачами. Они извлекают модели процессов в два этапа.

На первом этапе строится *стохастический граф деятельности* (Stochastic Activity Graph, SAG), который описывает зависимости между задачами в протоколе потока работ. Этот граф имеет вероятности перехода, связанные с каждой зависимостью, но не имеет никакой информации об элементах AND/XOR-split/join. Вероятность перехода указывает возможность, что одна задача следует за другой. Поиск основан на метрике правдоподобия протокола — likelihood (LLH). Метрика LLH оценивает, насколько хорошо (правдоподобно) модель отражает поведение, запечатленное в протоколе событий.

На втором этапе выполняется преобразование SAG в язык описания, используемый в системе бизнес-моделирования ADONIS (Adonis Definition Language, ADL). ADL — язык с блочной структурой для описания модели Workflow. Преобразование SAG в ADL необходимо для создания хорошо формализованной Workflow-модели.

MergeSeq и SplitSeq [132] хорошо работают для извлечения последовательных моделей процесса. SplitPar [131] может также извлекать параллельные модели процесса.

Алгоритм MergeSeq строит SAG снизу вверх. Он начинает с графа, который имеет ветвь для каждого уникального случая процесса в протоколе и последовательно применяется к узлам в графе.

Алгоритм SplitSeq строит граф сверху вниз. Он начинает с графа, который моделирует поведение в протоколе, но не содержит никаких дублированных задач. Алгоритм применяет набор операций разделения к узлам в SAG.

Алгоритм SplitPar можно рассматривать как расширение SplitSeq. Он также является нисходящим. Однако из-за того, что он предназначается для извлечения параллельных процессов, операции деления выполняются на уровне экземпляра процесса вместо SAG.

## 15.3.2. Метод построения дизъюнктивной Workflow-схемы

### Основные понятия

В случаях, когда процессы имеют слишком сложную структуру, чтобы быть представленными одним графом, целесообразно использовать подход, основанный на кластеризации, т. е. разбиении модели на части. Существует несколько разных подходов к такому разбиению, одним из них является *дизъюнктивная Workflow-схема*, предложенная Греко и др. [133, 134]. Кластеризация в ней производится путем поиска таких переходов в модели, допущение об обязательном срабатывании (не срабатывании) которых позволит сделать вывод о недостижимости значительной части модели, что позволит убрать ее из рассмотрения (т. е. поделить модель на части).

Метод построен на формальной модели, описывающей процессы и протоколы. В данной модели набор различных задач, из которых состоит процесс $P$, и набор различных возможных последовательностей, в которых они должны выполняться, чаще всего моделируется посредством подходящего размеченного графа.

Граф управления потоком процесса $P$ — это набор $CF(P) = \langle A, E, a_0, F \rangle$, где $A$ — ограниченный набор задач, $E \subseteq (A - F) \times (A - \{a_0\})$ — отношение порядка среди задач, $a_0 \in A$ — начальная задача, $F \subseteq A$ — набор конечных задач.

Любой связанный подграф $I = \langle A_I, E_I \rangle$ графа управления потоком, такой что $a_0 \in A_I$ и $A_I \in F \neq \varnothing$, соответствует одному из возможных экземпляров (instance) процесса $P$.

Для того чтобы промоделировать ограничения на возможные экземпляры процесса, описание процесса часто обогащается локальными или глобальны-

ми ограничениями, например, такими как: задача должна (не должна) прямо (непрямо) следовать за выполнением набора других задач.

Определим понятие *локальные ограничения* с помощью трех функций $A \to \mathrm{N}$: $IN$, $OUT_{\min}$ и $OUT_{\max}$, каждая из них сопоставляет узлу натуральное число, следующим образом. Пусть входная степень вершины — это $InDegree(a) = |\{(b,a) | (b,a) \in E\}|$, а выходная степень вершины — $OutDegree(a) = |\{(a,b) | (a,b) \in E\}|$, тогда:

$$\forall a \in A - \{a_0\},\ 0 < IN(a) \le InDegree(a),$$

$$\forall a \in A - F,\ 0 < OUT_{\min}(a) \le OUT_{\max}(a) \le OutDegree(a),$$

$$IN(a_0) = 0 \text{ и } \forall a \in F,\ OUT_{\min}(a) = OUT_{\max}(a) = 0.$$

Семантика ограничений выглядит следующим образом: выполнение задачи $a$ может начаться не ранее, чем, как минимум, $IN(a)$ предшествующих задач были выполнены.

Типовые конструкции:

- если $IN(a) = InDegree(a)$, то $a$ называется *AND-соединением*, так так она может быть выполнена только после того, как все ее предшественники были выполнены;
- если $IN(a) = 1$, то $a$ называется *OR-соединением*, так так она может быть выполнена, как только любая из предшествующих ей задач завершит свое выполнение.

Сразу же после завершения задача $a$ должна активировать подмножество исходящих дуг, мощность (cardinality) которого находится в пределах $OUT_{\min}(a)$ и $OUT_{\max}(a)$.

Если $OUT_{\max}(a) = OutDegree(a)$, то $a$ является полным ветвлением, а если к тому же $OUT_{\min}(a) = OUT_{\max}(a)$, то $a$ является детерминированным ветвлением (*AND*-разделение), так как она активирует все последующие задачи. Наконец, если $OUT_{\max}(a) = 1$, то $a$ является исключающим ветвлением (*XOR*-разделением), так как она активирует только одну из исходящих дуг.

Глобальные ограничения существенно разнообразнее локальных, и их представление сильно зависит от конкретной области применения, к которой относится моделируемый процесс. Таким образом, они часто представляются с использованием сложных формализмов, основанных на подходящей логике и связанной с ней семантике.

Простым, но достаточно выразительным способом описания глобальных ограничений является представление глобального ограничения в виде пары

$\langle t,a\rangle$, где $t$ и $a$ — непересекающиеся наборы задач. Запись означает, что задачи из набора $a$ не могут быть (или должны) быть выполнены, до того как будут выполнены все задачи из набора $t$.

Пусть $P$ — процесс, *Workflow-схемой процесса* $P$, обозначаемой как $WS(P)$, является кортеж $\langle CF(P),C_L(P),C_G(P)\rangle$, где $CF(P)$ — граф управления потоком процесса $P$, а $C_L(P)$ и $C_G(P)$ — наборы локальных и глобальных ограничений соответственно. Если имеется подграф $I$ графа $CF(P)$, а также ограничение $c$ из множества $C_L(P)\cup C_G(P)$, то запись $I|=c$ означает, что $I$ удовлетворяет ограничению $c$. Более того, если $I|=c$ для всех $c\in C_L(P)\cup C_G(P)$, то граф $I$ называется экземпляром $WS(P)$, что обозначается как $I|=WS(P)$. Если из контекста понятно, о каком процессе идет речь, то Workflow-схема может быть обозначена как $WS=\langle CF,C_L,C_G\rangle$.

Проблема построения модели. Пусть $A_P$ — набор идентификаторов задач процесса $P$. Допустим, что фактическая Workflow-схема $WS(P)$ для процесса $P$ неизвестна, тогда проблема заключается в определении правильной схемы из множества Workflow-схем, для которых $A_P$ является набором узлов. Для того чтобы формализовать данную проблему, предварительно необходимо дать определения ряду других понятий и записей.

*Workflow-следом* $s$ для набора $A_P$ называется строка из множества $A_P^*$, представляющая последовательность задач. Пусть $s$ — след, тогда под обозначением $s[i]$ будем понимать $i$-задачу в последовательности $s$, а под обозначением $length(s)$ будем понимать длину $s$. Множество всех задач в $s$ обозначается как $tasks(s)=\bigcup_{1\le i\le length(s)}s[i]$. Тогда Workflow-лог процесса $P$ — $L_P$ — это набор Workflow-следов $\sum_p:L_p=[s\mid s\in A_p^*]$, являющийся единственным источником информации для выявления схемы $WS(P)$.

Чтобы придать проблеме выявления $WS(P)$ более конкретную форму, нужно определить, какой язык будет использоваться для описания глобальных ограничений в $C_G(P)$, таким образом, проблема зависит от синтаксиса. Вследствие этого, чтобы разработать общий подход, удобнее будет использовать альтернативный, не зависящий от синтаксиса, способ описания глобальных ограничений. Для решения данной проблемы заменим целевую схему $WS(P)$ набором альтернативных схем, которые не имеют глобальных ограничений, однако прямо моделируют различные варианты выполнения, предписывае-

мые глобальными ограничениями. Основная идея заключается в том, чтобы сначала получить из Workflow-лога начальную Workflow-схему, глобальные ограничения которой не учитываются, а затем пошагово детализировать ее в набор специфических схем, каждая из которых моделирует класс следов, имеющих те же характеристики, что и глобальные ограничения.

Пример: допустим имеется глобальное ограничение, означающее, что задача $c$ не может быть выполнена после завершения выполнения $a$ и $b$, подходящее уточнение начальной схемы будет состоять из двух схем: в первой можно достичь одновременно $a$ и $b$, но не $c$, а во второй $c$ является достижимой, а $a$ и $b$ нет. При таком подходе набор ограничений представляется в виде различных вариантов выполнения, часто встречающихся в логе. В результате Workflow-схема рассматривается как объединение нескольких схем более простых вариантов протекания потоков без глобальных ограничений.

Пусть $P$ — процесс. *Дизъюнктивная Workflow-схема процесса* $P$, обозначаемая $WS^{\vee}(P)$, — это множество Workflow-схем $\{WS^1,...,WS^m\}$ процесса $P$, причем $WS^j = \langle CF^j, C_L^j, \varnothing \rangle$, где $i \le j \le m$. Экземпляр любой из Workflow-схем $WS^j$ является также экземпляром $WS^{\vee}$, что обозначается как $I \models WS^{\vee}$.

При условии, что известен $L_P$, целью является выявление дизъюнктивной Workflow-схемы $WS^{\vee}$, наиболее "близкой" к некоторой "идеальной "схеме $WS(P)$, которая является полным отражением реального потока. Данное утверждение можно формализовать путем введения критериев полноты и правильности, которые будут служить ограничением для Workflow-схем, позволяющим принимать во внимание исключительно следы из лога. Очевидно, что предварительно нужно определить механизмы для принятия решения, может ли рассматриваемый след из множества $L_P$ быть выведен из реального протекания потока $WS^{\vee}$.

Пусть $s$ — Workflow-след из множества $L_P$, $WS^{\vee}$ — дизъюнктивная Workflow-схема и подграф $I = \langle A_i, E_i \rangle$ — экземпляр $WS^{\vee}$. Тогда будем говорить, что $s$ совместим с $WS^{\vee}$ через $I$, что обозначается как $I \models^I WS^{\vee}$, если $t$ является топологической сортировкой $I$, т. е. $t$ — упорядочивание задач в $A_i$, такое что для каждой $(a,b) \in E_I$, $i < j$, где $s[i] = a$, а $s[j] = b$. Кроме того, будем говорить, что $s$ совместим с $WS^{\vee}$, что обозначается как $s \models WS^{\vee}$, если существует $I$, такой что $I \models^I WS^{\vee}$.

Пусть $WS^{\vee}$ — дизъюнктивная Workflow-схема, а $L_P$ — лог процесса $P$, тогда:

- правильность:

$$soundness(WS^{\vee}, L_P) = \frac{|\{I \mid I|=WS^{\vee} \wedge \exists s \in L_P \; s.t. \; s|=^{I} WS^{\vee}\}|}{|\{I \mid I|=WS^{\vee}|},$$

  т. е. процент экземпляров процесса $P$, не имеющих соответствующих им следов в логе;

- полнота:

$$soundness(WS^{\vee}, L_P) = \frac{|\{s \mid s \in L_P \wedge s|=WS^{\vee}\}|}{|\{s \mid s \in L_P\}|},$$

  т. е. процент следов, имеющих какой-либо совместимый с ними след в логе.

Пусть имеются два вещественных числа $\alpha$ и $\sigma$, причем $0 < \alpha < 1$ и $0 < \sigma < 1$ (обычно $\alpha$ близко к 0, а $\sigma$ — к 1), тогда будем говорить, что:

- $WS^{\vee}$ является $\alpha$-*правильной* относительно лога $L_P$, если $soundness(WS^{\vee}, L_P) \le \alpha$, т. е. чем меньше, тем правильнее;

- $WS^{\vee}$ является $\sigma$-*полной* относительно лога $L_P$, если $completeness(WS^{\vee}, L_P) \ge \sigma$, т. е. чем больше, тем правильнее.

С учетом введенных понятий переформулируем задачу. Теперь она заключается в том, чтобы выявить дизъюнктивную схему $WS^{\vee}$ для некоторого процесса $P$, которая будет $\alpha$-правильной и $\sigma$-полной для заданных $\alpha$ и $\sigma$. Однако очевидно, что всегда существует тривиальная схема, удовлетворяющая данным условиям, она состоит из объединения протеканий процесса, каждое из которых моделирует соответствующий ему след из $L_P$. Однако подобная модель не будет представлять большой ценности, т. к. ее размер будет $|WS^{\vee}| = |L|$, где $|L| = |\{s \mid s \in L\}|$. Учитывая это, определим ограничение на набор подсхем в $WS^{\vee}$.

Пусть $L_P$ — Workflow-лог процесса $P$, $\alpha$ и $\sigma$ — вещественные числа, $m$ — натуральное число. Задача точного выявления процесса, обозначаемая как $EPD(P, \sigma, \alpha, m)$, состоит из поиска $\alpha$-правильной и $\sigma$-полной дизъюнктивной Workflow-схемы $WS^{\vee}$, такой что $|WS^{\vee}| \le m$.

Задача точного выявления процесса может быть решена за полиномиальное время только для тривиальных случаев $m=1$ или для больших $m$ (кроме случая, когда $P$ принадлежит классу $NP$).

**Теорема 1.** Имеет ли $EPD(P,\sigma,\alpha,m)$ решение, можно определить за полиномиальное время, зависящее от размера $L_P$, если $m \le 1$, или $m \ge |L|$, в противном случае задача $NP$-полная [134].

Переформулируем постановку задачи выявления процесса таким образом, чтобы она всегда имела решение.

Пусть $L_P$ — Workflow-лог процесса $P$, $\sigma$ — вещественное число, $m$ — натуральное число. Задача минимального выявления процесса, обозначаемая как $MPD(P,\sigma,m)$, состоит из поиска $\sigma$-полной дизъюнктивной Workflow-схемы $WS^{\vee}$, такой что $\left|WS^{\vee}\right| \le m$, и значение $soundness(WS^{\vee})$ минимально.

Задача теперь решаема, т. к. можно пожертвовать правильностью в пользу получения результата. Тем не менее она все еще трудноразрешима. Будем считать что числа, характеризующие правильность, соответствующим образом дискретизированы и являются положительными целыми числами, чтобы можно было представить $MPD$ как оптимизационную задачу класса $NP$.

**Теорема 2.** $EPD(P,\sigma,m)$ — оптимизационная задача класса $NP$, множество возможных решений которой не пустое [134].

Теперь можно перейти к проблеме $PD(P,\sigma,\alpha,m)$, которая заключается в поиске подходящей аппроксимации, т. е. $\sigma$-полной Workflow-схемы $WS^{\vee}$, где $\left|WS^{\vee}\right| \le m$, являющейся настолько правильной, насколько это возможно.

Чтобы выявить Workflow-схему процесса $P$ (проблема $PD(P,\sigma,\alpha,m)$), используется идея многократной пошаговой оптимизации схемы путем выявления глобальных ограничений, которые затем используются для того, чтобы провести различными вариантами выполнения, начиная с основной дизъюнктивной схемы $WS^{\vee}$, которая отвечает только за зависимости между задачами в $P$.

## Описание алгоритма

Алгоритм Process Discovery (листинг 15.1) получает $WS^{\vee}$, используя метод иерархической кластеризации, начинающийся с выявления графа управления потоком $CF_{\sigma}$, с помощью процедуры `minePrecedences`. Каждая Workflow-схема $WS_i^j$, которая в конечном счете добавляется в $WS^{\vee}$, имеет номер $i$ —

количество необходимых шагов оптимизации, а номер $j$ используется для различения схем на одном уровне оптимизации. $L(WS_i^j)$ — набор следов в кластере, определяемом $WS_i^j$. Следует также отметить, что начальная схема $WS_0^1$, содержащая все следы из набора $L_P$, вставляется в $WS^\vee$, а на шаге 3 модель оптимизируется выявлением локальных ограничений.

**Листинг 15.1. Алгоритм Process Discovery**

Вход: Проблема $PD(P, \sigma, m)$, натуральное число maxFeatures

Выход: Модель процесса

Алгоритм: выполнить последовательность шагов:

1. $CF_\sigma(WS_0^1) :=$ minePrecedences$(L_p)$
2. *let* $WS_0^1$
3. mineLocalConstraints$(WS_0^1)$
4. $WS^\vee := WS_0^1$ //Начать кластеризацию, имея только граф зависимостей
5. While $\left|WS^\vee\right| < m$ do

   5.1. $WS_i^j := leastSound(WS^\vee)$

   5.2. $WS^\vee := WS^\vee - \left\{WS_i^j\right\}$

   5.3. refineWorkflow$(i, j)$
6. return $WS^\vee$

Функция refineWorkflow выглядит следующим образом:

refineWorkflow$(i, j)$

1. F:=identifyRelevantFeatures$(L(WS_i^j, \sigma,$ maxFeatures, $CF_\sigma))$
2. $R(WS_i^j) := project(L(WS_i^j), F)$
3. $k := \left|F\right|$
4. If $k > 1$ then

   4.1. $j := \max\{j \mid WS_{i+1}^j \in WS^\vee\}$

   4.2. $\left\langle WS_{i+1}^{j+1}, ..., WS_{i+k}^{j+k}\right\rangle := k - means(R(WS_i^j))$

4.3. `For each` $WS_{i+1}^h$ `do`

4.3.1. $WS^\vee = WS^\vee \cup \{WS_{i+1}^h\}$

4.3.2. $CF_\sigma(WS_{i+1}^h) := \texttt{minePrecedences}(L(WS_{i+1}^h))$

4.3.3. $\texttt{mineLocalConstraints}(WS_{i+1}^h)$

5. `else // покинуть дерево`

5.1. $WS^\vee = WS^\vee \cup \left\{WS_i^j\right\}$

Алгоритм является "жадной" эвристикой, на каждом шаге он выбирает схему $WS_i^j \in WS^\vee$ для оптимизации функцией `refineWorkflow` исходя из того, какую из них наиболее выгодно оптимизировать.

Чтобы применить методы кластеризации (k-means алгоритм), процедура `refineWorkflow` транслирует лог $L(WS_i^j)$ в реляционную информацию с помощью процедур `identifyRelevantFeatures` и `project`. Если устанавливается наличие более чем одной релевантной особенности, то вычисляются кластеры $WS_{i+1}^{j+1}, ..., WS_{i+1}^{j+k}$ (где $j$ — максимальный индекс схем, уже добавленных в $WS^\vee$ на уровне $i+1$) с помощью применения k-means алгоритма к следам $L(WS_i^j)$, далее кластеры добавляются в дизъюнктивную Workflow-схему $WS^\vee$. В конце для каждой схемы, добавленной в $WS^\vee$, чтобы определить локальные ограничения, вызывается процедура `mineLocalConstraints`.

Алгоритм сходится не более чем за $m$ шагов и использует следующее свойство процедуры `refineWorkflow`: на каждом шаге оптимизации значение правильности уменьшается, т. е. алгоритм подходит ближе к оптимальному решению.

**Теорема 3.** Пусть $WS^\vee$ — дизъюнктивная Workflow-схема с $WS_i^j \in WS^\vee$, $WS_+^\vee$ — дизъюнктивная Workflow-схема, полученная оптимизацией $WS^\vee - \{WS_i^j\}$ процедурой `refineWorkflow`$(i,j)$, тогда $soundness(WS_+^\vee) \le \le soundness(WS^\vee)$ [134].

Основная идея алгоритма заключается в том, что число новых схем $k$, добавляемых на каждом шаге оптимизации, фиксируется. $k$ может находиться в интервале от минимум 2, что потребует нескольких шагов вычислений, до неограниченной величины, при которой результат будет возвращен всего через 1 шаг. Второй случай не всегда эффективнее, так так алгоритм класте-

ризации может работать медленнее с большим набором классов, теряя преимущество меньшего количества итераций. Весомым доводом в пользу небольшого значения $k$ : представление различных схем может быть оптимизировано путем сохранения древовидной структуры и хранения для каждого узла только его отличий от схемы родительского узла. Древовидное представление релевантно не только из-за уменьшения объема необходимой памяти, но и еще потому, что оно дает больше представления о свойствах моделируемых Workflow-следов и дает удобное описание глобальных ограничений.

Пусть $L_P$ — Workflow-лог задач $\sum p, A \subseteq \sum p$, $s$ — след в $L_P$. Началом (концом) $A$ в $s$, обозначаемым $b(A,s)$ ( $e(A,s)$ ), является индекс $i$ (если он существует), такой что $a = s[i]$ и $\forall a' \in A - \{a\}, a' = s[j]$, где $j > i$ ( $j < i$ ). Пусть заданы $B \subseteq \sum p$ и порог $\sigma$, будем говорить, что $A$ $\sigma$-предшествует $B$, что обозначается как $A \to_\sigma B$, если $\left|\{s \in L_P \mid e(A,s) < b(B,s)\}\right| / \left|L_P\right| \geq \sigma$.

Используя данную запись, можно описывать сложные взаимосвязи между задачами.

Пусть имеются две задачи $a$ и $b$, и порог $\sigma$, тогда будем говорить, что: $a$ и $b$ являются $\sigma$-параллельными задачами в $L_P$, что обозначается как $a|_\sigma b$, если есть задачи $a = a_1, a_2, ..., a_m = b$, где $m > 1$, такие что $\{a_i\} \to_\sigma \{a_{i+1}\}$ для $1 \leq i < m$ и $\{a_m\} \to_\sigma \{a_1\}$. $a$ $\sigma$-строго предшествует $b$ в $L_P$, что обозначается как $a \Rightarrow_\sigma b$, если $a$ и $b$ не являются $\sigma$-параллельными задачами, и если нет следов $s_1, ..., s_k$ в $L_P$, где $k \geq \sigma \times \left|Lp\right|$, такое что для каждого $s_i$ $b(\{a\}, s_i) < b(\{b\}, s_i)$ и $\forall j$ таких что $b(\{a\}, s_i) < j < b(\{b\}, s_i)$, $s_i[j] |_\sigma b$.

Параллельные задачи и строгие предшествования являются базовыми блоками, из которых строится процесс управления потоком. $\sigma$-управление потоком ($\sigma$-control flow) процесса $P$ — это граф $CF_\sigma(P) = \langle \sum p, E \rangle$, содержащий дугу $(a,b)$ в $E$ для каждой пары узлов $a$ и $b$, таких что или $a \Rightarrow_\sigma b$, или $\{a\} \to_\sigma \{b\}$, и не существует набора задач $\{h_1, ..., h_m\}$, где $a \Rightarrow_\sigma h_1$, $h_i \Rightarrow_\sigma h_{i+1}$ для $1 \leq i < m$ и $h_m \Rightarrow_\sigma b$.

Наконец, набор $\sigma$-локальных ограничений, обозначаемый как $C_{L\sigma}$, может быть получен, используя процесс управления потоком:

$$OUT_{\min}(a) = |\, succ(a) \,| - \max s \subseteq_{succ(a), \{a\} \to_\sigma S} \left|S\right|,$$

$$OUT_{\max}(a) = |\, succ(a) \,| - \min s \subseteq_{succ(a), \{a\} \to_\sigma S} \left|S\right|,$$

$$IN(a) = \min s \subseteq_{prec(a),\ S \to_\sigma \{a\}} \left|S\right|,$$

где $succ(a) = \{b \mid (a,b) \in E_\sigma\}$ и $prec(a) = \{b \mid (b,a) \in E_\sigma\}$, а $A \not\to_\sigma B$ означает, что $A \to_\sigma B$ не выполняется.

Ключевым моментом алгоритма кластеризации Workflow-следов является формализация процедур `identifyRelevantFeatures` и `project`.

Пусть $L$ — набор Workflow-следов, $CF_\sigma$ — полученный граф управления потоком точностью $\sigma$, тогда последовательность задач $[a_1,...,a_h]$ имеет $\sigma$-частоту в $L$, если $\left|\{s \in L \mid a_1 = s[i_1],...,a_h] = s[i_h] \wedge i_1 < ... < i_h\}\right| / |L| \geq \sigma$. Будем говорить, что $[a_1,...,a_h]$ $\sigma$-предшествует $a$ в $L$, если $[a_1,...,a_h]$ и $[a_1,...,a_h a]$ имеют $\sigma$-частоту в $L$.

Дискриминантным правилом (или особенностью) $\phi$ является выражение вида $[a_1,...,a_h] \not\to a$, такое что $[a_1,...,a_h]$ имеют $\sigma$-частоту в $L$, $a_h \to_\sigma a$ есть в $CF_\sigma$ и $[a_1,...,a_h]$ не $\sigma$-предшествует $a$ в $L$. $\phi$ является минимальным, если не существует такого $b$, где $[a_1,...,a_h] \not\to_\sigma b$, $b \not\to_\sigma a$, и не существует $j$, такого что $[a_1,...,a_h] \not\to_\sigma a$.

Выявление дискриминантных правил выполняется при помощи алгоритма (листинг 15.2). На каждом $k$-м шаге в $L_k$ сохраняются все последовательности $\sigma$-частоты, размер которых $k$. Конкретнее, на шагах 5—9 набор потенциальных последовательностей $M$, который сохраняется в $L_{k+1}$, получается путем объединения тех $L_k$, которые имеют отношения предшествования в $L_2$. Также шаг 7 предотвращает вычисление не минимальных неожиданных правил. Таким образом, только последовательности $\sigma$-частоты в $M$ включаются в $L_{k+1}$ (шаг 11), тогда как остальные определяют неожиданные правила (шаг 12). Процесс повторяется пока не найдены какие-либо другие часто встречающиеся следы. Правильность алгоритма определяется следующей теоремой.

**Теорема 4.** В алгоритме из листинга 15.2 до его завершения (шаг 16) множество $R$ должно содержать точно все последовательности задачи $\sigma$-частоты, множество $F$ должно содержать точно все минимальные дискриминантные правила.

**Листинг 15.2. Алгоритм IdentifyRelevantFeatures**

```
Вход: Набор логов L, порог σ, натуральное число maxFeatures, граф CF_σ
Выход: Набор минимальных "неожиданных" правил
```

Алгоритм: выполнить следующие шаги:

1. $L_2 := \{[ab] \mid a \xrightarrow[\sigma]{} b\}$
2. $k := 1$, $R := L_2$, $F := 0$
3. repeat
   - 3.1. $M := 0$; $k := k + 1$
   - 3.2. for all $\left[a_i ... a_j\right] \in L_k$ do
     - 3.2.1. for all $\left[a_j b\right] \in L_2$ do
       - 3.2.1.1. if $\left[a_{i+1} ... a_j\right] not \xrightarrow[\sigma]{}$ is not $F$ then
         - 3.2.1.1.1. $M := M \cup \left[a_i ... a_j b\right]$
   - 3.3. for all $p \in M$ do
     - 3.3.1. if $p$ это $\sigma$-частота в $L$ then $L_{k+1} := \{p\}$
     - 3.3.2. else $F := F \cup \left[a_i ... a_j\right] not \xrightarrow[\sigma]{} b$ // см. Теорему 2
4. until $L_{k+1} = 0$
5. return mostDiscriminant($F$)

Процедура mostDiscriminantFeatures($F$ набор "неожиданных" правил): набор "неожиданных" правил

1. $S' := L$, $S' := 0$
2. do
   - 2.1. let $\varphi = \arg\max_{\varphi' \in F} |\omega(\varphi', S')|$
   - 2.2. $F' := F' \cup \{\varphi\}$
   - 2.3. $S' = S' - \omega(\varphi', S')$
3. While($|S'| / |L_p| > \sigma$) and ($F'$ < maxFeatures)
4. Return $F'$

Следует отметить, что алгоритм IdentifyRelevantFeatures не прямо выдает $F$, а вызывает процедуру mostDiscriminantFeatures, целью которой является поиск подходящего подмножества, которое лучше дискриминирует следы в логе.

Это можно формализовать следующим образом. Пусть $\phi$-дискриминантное правило вида $[a_i,...,a_j] \not\rightarrow_\sigma b$, тогда отражение $\phi$ в $L$, что обозначается как $w(\phi, L)$, это множество логов, в которых встречается последовательность $[a_i,...,a_j]$. Более того, если задан набор правил $R$, то отражение $R$ в $L$ — это $\bigcap_{\phi \in R} w(\phi, L)$. Для фиксированного $k$ $R$ является самым дискриминантным $k$-набором особенностей, если $|R| = k$ и не существует $R'$ с $|w(R', L)| > |w(R, L)|$ и $|R'| = k$.

Следует отметить, что самый дискриминантный $k$-набор особенностей может быть вычислен за полиномиальное время путем рассмотрения всех возможных комбинаций особенностей $R$, содержащих $k$ элементов. Минимальное $k$, для которого наиболее дискриминантный $k$-набор особенностей $S$ покрывает все логи, т. е. $w(S, L) = L$, называется размерностью $L$, тогда как $S$ — наиболее дискриминантным набором особенностей.

**Теорема 5.** Пусть $L$ — набор следов, $n$ — размер $L$ (сумма размеров всех следов в $L$), и $F$ — набор особенностей. Тогда задача вычисления наиболее дискриминантного набора особенностей является $NP$-сложной [134].

Учитывая присущую проблеме сложность, перейдем к вычислению подходящей аппроксимации. Точнее, процедура `mostDiscriminantFeatures`, являющаяся имплементацией алгоритма поиска релевантных особенностей, вычисляет набор $F'$ дискриминантных правил с помощью эвристики, "жадно" выбирая особенность $\phi$, покрывающую максимальный набор следов, среди которых $S'$ не покрыты предыдущими выборками.

**Проецирование следов.** Набор релевантных особенностей $F$ может быть использован для представления каждого следа $s$ точкой в векторном пространстве $\mathrm{R}^{|F|}$, обозначаемой $\vec{s}$. Процедура `project` использует данный способ, переводя (`maps`) следы в пространство $\mathrm{R}^{|F|}$, в котором k-means алгоритм может работать.

## 15.3.3. $\alpha$-алгоритм

### Основная идея алгоритма

Ван дер Аалст и др. [135—140] разработали $\alpha$-алгоритм. Его главное отличие от других — то, что авторы доказывают, к какому классу моделей их подход будет гарантированно работать.

Они предполагают, что протокол должен быть без шума и должен быть дополнен отношением потоков. Так, если в исходной модели задача $A$ может

быть выполнена только до задачи $B$ (то есть $B$ следует за $A$), по крайней мере один экземпляр процесса в протоколе указывает на это поведение. Их подход доказывается в работе для класса структурированных сетей Workflow (SWF-сети) без коротких петель и неявных мест (см. более подробно [135]). $\alpha$-алгоритм работает, основываясь на бинарных отношениях в протоколе. Есть четыре отношения: следствие, причина, параллельность и несвязанность. Две задачи $A$ и $B$ имеют отношение следования, если они появляются друг за другом в протоколе. Это отношение — основное отношение, из которого происходят другие отношения. Две задачи $A$ и $B$ имеют причинное отношение, если $A$ следует за $B$, но $B$ не следует за $A$. Если $B$ также следует за $A$, то задачи имеют параллельное отношение. Когда $A$ и $B$ не вовлечены в перечисленные отношения, говорят, что они являются несвязанными.

Обратите внимание, что все отношения зависимости выведены на основании локальной информации в протоколе. Поэтому $\alpha$-алгоритм не может обрабатывать нелокальный "не свободный выбор". Кроме того, из-за того что $\alpha$-алгоритм работает, основываясь на наборах, он не может извлекать модели с дублируемыми задачами.

## Сети Петри

*Сеть Петри* — это набор $N=(P,T,F)$, где $P$ — конечное множество позиций, $T$ — конечное множество переходов, таких что $P\cap T=\varnothing$, $F\subseteq(P\times T)\cup(T\times P)$ — набор ориентированных дуг.

*Размеченная сеть Петри* — это пара $(N,s)$, где $N=(P,T,F)$ — сеть Петри, а $s$ — функция, задающая разметку сети $s\in P\to N$.

*Разметка сети* — это набор натуральных чисел (вектор), сопоставленных элементам множества $P$. Каждое число обозначает количество маркеров, сопоставленных данной позиции в текущей разметке.

Пусть $N=(P,T,F)$ — сеть Петри. Элементы множества $P\cup T$ называются *узлами*, или *вершинами сети*. Вершина $x$ является *входной* вершиной, для другой вершины $y$ тогда и только тогда, когда есть ориентированная дуга от $x$ к $y$ (т. е. $(x,y)\in F$, или $xFy$). Вершина $x$ является *исходящей* вершиной для другой вершины $y$ тогда и только тогда, когда $yFx$.

Для каждой вершины $x\in P\cup T$ существует множество входных вершин $\overset{N}{\bullet}x=\{y\mid yFx\}$ и множество исходящих вершин $x\overset{N}{\bullet}=\{y\mid xFy\}$ (индекс $N$ может быть опущен, если из контекста ясно, о какой сети Петри идет речь).

Графически сеть Петри изображается в виде двудольного ориентированного графа, в котором позиции изображаются в виде кружочков, а переходы — в виде прямоугольников. Разметка сети Петри изображается в виде точек внутри кружочков, соответствующих позициям. Количество точек внутри кружка соответствует числу маркеров, сопоставленных данной позиции в текущей разметке.

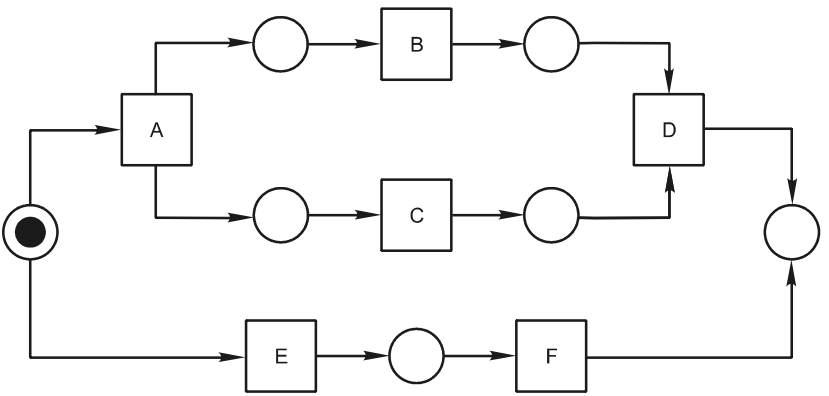



**Рис. 15.11.** Пример сети из позиций и переходов

На рис. 15.11 изображена сеть Петри, состоящая из 7 позиций и 6 переходов. Переход $A$ имеет одну входную и две выходные позиции, такой переход называется *AND-разделением*, а переход $D$, имеющий две входные и одну выходную позиции, называется *AND-соединением*.

Динамическое поведение размеченной сети Петри определяется правилом срабатывания.

Пусть $N=((P,T,F),s)$ — размеченная сеть Петри. Переход $t \in T$ считается возбужденным тогда и только тогда, когда $\bullet t \le s$, что обозначается как $(N,s)\left[t\right\rangle$. Переход $t$, являющийся возбужденным при текущей разметке $s$, может сработать, что приведет к изменению разметки: $s' = s - \bullet t + t \bullet$. Процесс срабатывания обозначается как $(N,s)\left[t\right\rangle(N, s - \bullet t + t\bullet)$.

При маркировке, изображенной на рис. 15.11, два перехода $A$ и $E$ возбуждены, однако только один из них может сработать. Если сработает $A$, то из входной позиции маркер будет удален, а на выходные позиции маркеры будут добавлены. В получившейся разметке переходы $B$ и $C$ будут возбуждены, причем они могут сработать параллельно (т. к. у них нет общих входных позиций).

Пусть $N=((P,T,F),s)$ — размеченная сеть Петри. Разметка $s$ называется *достижимой от начальной разметки* $s_0$ тогда и только тогда, когда сущест-

вует последовательность возбужденных переходов, срабатывание которых приведет к изменению разметки $s_0$ к $s$. Множество достижимых от $(N, s_0)$ разметок обозначается как $\left[N, s_0\right\rangle$.

Изображенная на рис. 15.11 размеченная сеть Петри имеет 6 достижимых разметок. Иногда удобно знать последовательность переходов, которые должны сработать для того, чтобы получилась некоторая искомая разметка. Далее используются следующие обозначения для таких последовательностей. Пусть $A$ — некоторый алфавит идентификаторов, последовательность длиной $n$ из натуральных чисел до числа $n \in N$ над алфавитом $A$ является функцией $\sigma : \{0,...,n-1\} \to A$. Последовательность длиной 0 называется пустой последовательностью и обозначается как $\varepsilon$. Для повышения читабельности непустая последовательность может записываться только из значений функции. Например, последовательность $\sigma = \{(0,a),(1,b),(2,b)\}$ при $a,b \notin A$ записывается как $aab$. Множество всех последовательностей произвольной длины из символов алфавита $A$ обозначается $A^*$.

Пусть $A$ — множество, $a_i \in A$ $(i \in N)$, $\sigma = a_0 a_1 ... a_{n-1} \in A^*$ — последовательность элементов $A$ длиной $n$, тогда:

1. $a \in \sigma$ тогда и только тогда, когда $a \in \{a_0 a_1 ... a_{n-1}\}$.
2. $first(\sigma) = a_0$, если $n \geq 1$.
3. $last(\sigma) = a_{n-1}$, если $n \geq 1$.

Пусть $(N, s_0)$, где $N = (P, T, F)$ — размеченная сеть Петри. Последовательность $\sigma \in T^*$ называется *последовательностью срабатываний сети* $(N, s_0)$ тогда и только тогда, когда для некоторого натурального числа $n \in N$ существуют разметки $s_1,...,s_n$ и переходы $t_1,...,t_n \in T$, такие что $\sigma = t_1,...,t_n$, и для каждого $i \in [0,n)$, $(N, s_i)\left[t_{i+1}\right\rangle (N, s_{i+1})$, где $s_{i+1} = s_i - {\bullet}t_{i+1} + t_{i+1}{\bullet}$ (при $n = 0$ предполагается, что $\sigma = \varepsilon$, и что $\varepsilon$ — последовательность срабатываний при $(N, s_0)$). Последовательность $\sigma$ называется *возбужденной при разметке* $s_0$, что обозначается как $(N, s_0)\left[\sigma\right\rangle$. Срабатывание последовательности $\sigma$ приведет к разметке $s_n$, что обозначается как $(N, s_0)\left[\sigma\right\rangle (N, s_n)$.

Сеть $N = (P, T, F)$ *слабо связана* тогда и только тогда, когда для любой пары узлов $x$ и $y$ из множества $P \cup T$ верно $x(F \cup F^{-1})^* y$, где $R^{-1}$ — инверсия, а $R^*$ — рефлексивное и транзитивное замыкание отношения $R$. Сеть

$N=(P,T,F)$ сильно связана тогда и только тогда, когда для любой пары узлов $x$ и $y$ из множества $P\cup T$ верно $xF^*y$.

Будем считать, что все сети слабо связаны и имеют, по крайней мере, две вершины. Сеть Петри, изображенная на рис. 15.11, связана, но не сильно.

Размеченная сеть Петри $(N,s_0)$, где $N=(P,T,F)$, *ограничена* тогда и только тогда, когда множество достижимых разметок $\lceil N,s\rangle$ конечно.

Размеченная сеть Петри $(N,s_0)$, где $N=(P,T,F)$, *безопасна* тогда и только тогда, когда для любого $s'\in\lceil N,s\rangle$ и $p\in P$, $s'(p)\le 1$. Безопасность подразумевает ограниченность.

Размеченная сеть Петри, изображенная на рис. 15.11, *безопасна* (и поэтому ограничена), т. к. ни одна из 6 достижимых разметок не имеет более одного маркера на одной позиции.

Пусть $(N,s)$, где $N=(P,T,F)$ — размеченная сеть Петри. Переход $t\in T$ называется *"живым"* в сети $(N,s)$, тогда и только тогда, когда есть достижимая разметка $s'\in\lceil N,s\rangle$, такая что $(N,s')\lceil t\rangle$. Сеть $(N,s)$ называется *"живой"*, если для каждой достижимой разметки $s'\in\lceil N,s\rangle$ и для каждого перехода $t\in T$ существует достижимая разметка $s''\in\lceil N,s'\rangle$, такая что $(N,s'')\lceil t\rangle$. То, что сеть "живая", предполагает, что все ее переходы "живые". Все переходы размеченной сети Петри, изображенной на рис. 15.11, "живые", однако сеть не является "живой", т. к. все переходы не могут сработать последовательно.

## Workflow-сети

Большинство Workflow-систем предоставляют стандартизированные структурные блоки, такие как *AND*-разделение, *AND*-соединение, *XOR*-разделение и *XOR*-соединение. Они используются для моделирования частичного, последовательного, условного и параллельного выполнений потоков задач. Очевидно, что сети Петри могут быть использованы для отслеживания процесса выполнения потоков задач. Задачи моделируются переходами, а зависимости — позициями и дугами. Каждой позиции соответствует условие, которое может быть использовано как предусловие или как постусловие для задач. *AND*-разделению соответствует переход с двумя или более исходящими позициями. *AND*-соединению соответствует переход с двумя или более входными позициями. *XOR*-соединению/разделению соответствуют позиции с двумя или более входными/выходными переходами. Учитывая тес-

ную связь между этими двумя понятиями, в дальнейшем будем считать понятия "задачи" и "переходы" взаимозаменяемыми.

Сеть Петри, которая моделирует очередность выполнения задач в потоке работ, называется *сетью потоков работ* (Workflow-сеть). Следует отметить, что Workflow-сеть определяет динамическое поведение каждого случая протекания потока задач в отдельности.

Пусть $N=(P,T,F)$ — сеть Петри, а $\bar{t}$ — переход, не являющийся частью $P \cup T$. $N$ — сеть потоков работ, или Workflow-сеть (WF-сеть), тогда и только тогда, когда выполняются следующие условия:

- $P$ содержит позицию $i$, такую что $\bullet i = \varnothing$ (начальная позиция);
- $P$ содержит позицию $o$, такую что $o\bullet = \varnothing$ (конечная позиция);
- сеть $\overline{N}=(P,T\cup\{\bar{t}\},F\cup\{(o,\bar{t}),(\bar{t},i)\})$ сильно связана.

Сеть Петри, изображенная на рис. 15.11, является WF-сетью. Даже если сеть удовлетворяет изложенным ранее синтаксическим требованиям и является WF-сетью, соответствующий ей процесс может содержать ошибки, такие как: тупики, задачи, которые никогда не выполняются, активные тупики, мусор, оставляемый процессом после завершения работы, и т. д. Учитывая это, определим критерий бездефектности (правильной завершаемости).

Пусть $N=(P,T,F)$ — WF-сеть с начальной позицией $i \in P$ и конечной позицией $o \in P$. $N$ является *бездефектной* тогда и только тогда, когда выполняются следующие условия:

- размеченная сеть $(N,[i])$ является безопасной;
- для любой разметки $s \in [N,[i]\rangle$ $o \in s$ подразумевает, что $s=[o]$, т. е. при достижении конечной вершины сети $N$ не должно быть маркеров в промежуточных позициях;
- для любой разметки $s \in [N,[i]\rangle$ $[o] \in [N,s\rangle$, т. е. конечная вершина достижима из любой разметки;
- $(N,[i])$ содержит только живые переходы.

Множество всех бездефектных WF-сетей обозначается как $W$.

WF-сеть, изображенная на рис. 15.11, бездефектна. Свойство бездефектности может быть проверено путем использования стандартных методов анализа, разработанных для сетей Петри. Фактически свойство бездефектности соответствует свойствам живости и безопасности для короткозамкнутой сети.

Пусть $N=(P,T,F)$ — сеть Петри с начальной разметкой $s$. Позиция $p \in P$ называется *скрытой* (имплицитной) в $(N,s)$ тогда и только тогда, когда

для всех достижимых разметок $s' \in [N,s\rangle$ и переходов $t \in p\bullet$ $s' \geq \bullet t \setminus \{p\} \Rightarrow s' \geq \bullet t$ .

Сеть Петри, изображенная на рис. 15.11, не содержит скрытых вершин, однако добавление позиции $p$, соединяющей переходы $A$ и $D$, приведет к появлению скрытой вершины. Распознавание скрытых вершин затруднено тем, что их добавление не оказывает никакого влияния на поведение сети.

*Структурированные Workflow-сети* (SWF-сети) являются подклассом WF-сетей, полученным путем наложения дополнительных ограничений. SWF-сети не должны содержать конструкций, изображенных на рис. 15.12.

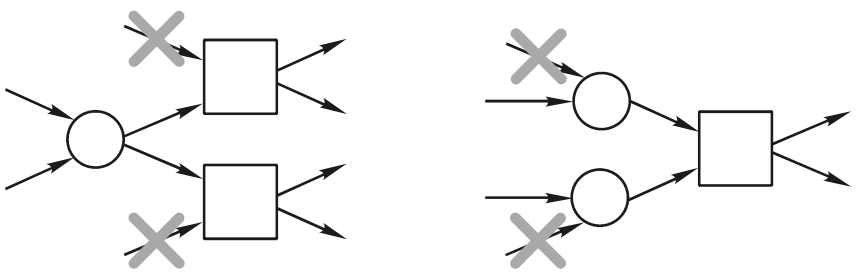


**Рис. 15.12.** Конструкции, запрещенные в SWF-сетях

Левый структурный блок иллюстрирует ограничение, что выбор и синхронизация не должны встречаться. Если два перехода имеют общую выходную позицию и таким образом "сражаются" за один и тот же маркер, у них никогда не должно быть синхронизации. Это означает, что "позиции-выборы" (позиции с несколькими выходными переходами) никогда не должны контролироваться синхронизациями.

Правый структурный блок, изображенный на рис. 15.12, иллюстрирует ограничение, что если есть синхронизация, все прямо предваряющие ей переходы должны сработать до момента синхронизации, т. е. не разрешается наличие синхронизаций, прямо перед *XOR*-соединением.

Структурированные сети потоков задач определяются следующим образом: WF-сеть $N = (P,T,F)$ является SFW-сетью тогда и только тогда, когда выполняются следующие условия:

- для всех $p \in P$ и $t \in T$, для которых $(p,t) \in F$: $|\mathrm{p}\bullet| > 1$, предполагается, что $|\bullet\,\mathrm{t}| = 1$;
- для всех 1 и $t \in T$, для которых $(p,t) \in F$: $|\bullet\,\mathrm{t}| > 1$, предполагается, что $|\bullet\,\mathrm{p}| = 1$;
- сеть $N$ не содержит скрытых вершин.

## Логи событий

Отправным пунктом для любого алгоритма Process Mining является лог событий (event log). Лог событий является множеством следов событий (event trace). Каждый след соответствует однократному протеканию процесса (case или process instance). Нужно отметить, что лог может содержать одинаковые следы события, что означает, что при различных протеканиях процесса события следовали одинаковому "пути" в процессе.

Пусть $T$ — набор задач, тогда $\sigma \in T^*$ — *след событий*, а функция $L: T^* \to N$ — *лог событий*. Для любого $\sigma \in dom(L)$ $L(\sigma)$ — количество появлений $\sigma$. Множество логов событий обозначается как $L$.

Здесь и далее используются следующие условные обозначения:

- $dom(f)$ — область определения функции $f$;
- $rng(f)$ — область значений функции $f$.

$\sigma \in L$ подразумевает, что $\sigma \in dom(L) \wedge L(\sigma) \geq 1$.

**Пример.** Допустим $L = [abcd, acbd, abcd]$ — лог для сети, изображенной на рис. 15.14, тогда $L(abcd) = 2$, $L(acbd) = 1$, $L(ab) = 0$.

На основании информации, содержащейся в логе, можно делать заключения о взаимных отношениях между событиями. В случае $\alpha$ -алгоритма любые две задачи в логе $L$ должны находиться в одном из четырех возможных отношений упорядочивания:

- $>_L$ — "следует за";
- $\to_L$ — "строго следует за";
- $|_L$ — "параллельно";
- $\#_L$ — "не связано".

Данные отношения выделяются из локальной информации в логе событий. Отношения упорядоченности определяются следующим образом: пусть $L$ — лог событий над набором задач $T$, т. е. $L: T^* \to N$, $a, b \in T$:

- $a >_L b$ тогда и только тогда, когда существует след $\sigma = t_1 t_2 t_3 ... t_n$, такой что $\sigma \in L$ и $t_i = a$, а $t_{i+1} = b$, где $i \in \{1, ..., n-1\}$;
- $a \to_L b$ тогда и только тогда, когда $a >_L b$ и $b \not>_L a$;
- $a \#_L b$ тогда и только тогда, когда $a \not>_L b$ и $b \not>_L a$.

Гарантией того, что лог событий содержит минимум информации, необходимый для успешного выявления потока задач, служит понятие полноты лога,

которое определяется следующим образом: пусть $N=(P,T,F)$ — бездефектная WF-сеть, т. е. $N \in W$. $L$ является логом событий для $N$ тогда и только тогда, когда $dom(L) \in T^*$ и каждый след $\sigma \in L$ является последовательностью срабатывания в $N$, начинающейся в начальной позиции $i$ и заканчивающейся в конечной позиции $o$, т. е. $(N,[i])\left[\sigma\right\rangle(N,[o])$. $L$ является *полным логом событий* для сети $N$ тогда и только тогда, когда для любого лога $L'$ для $N$: $>_{L'} \subseteq >_L$.

Для сети, изображенной на рис. 15.14, одним из возможных полных логов является следующий: $L=\{abcd, acbd, ef\}$. Из данного лога можно получить информацию о таких отношениях порядка:

- $a >_L b$, $a >_L c$, $b >_L c$, $b >_L d$, $c >_L b$, $c >_L d$ и $e >_L f$;
- $a \rightarrow_L b$, $a \rightarrow_L c$, $b \rightarrow_L d$, $c \rightarrow_L d$ и $e \rightarrow_L f$;
- $b \|_L c$ и $c \|_L b$;
- $x \#_L y$ и 1 для $x \in \{a,b,c,d\}$ и $y \in \{e,f\}$.

## Описание α-алгоритма

Данный алгоритм считается основным алгоритмом Process Mining (хотя и не самым эффективным), т. к. он является единственным, имеющим под собой теоретическую базу. Существует класс распознаваемых им моделей и доказательство того, что он может выявить модели этого класса из логов при условии наличия достаточной информации.

Пусть $L$ — лог событий для набора задач $T$, тогда $\alpha(L)$ определяется следующим образом:

$$T_L = \{t \in T \mid \exists_{\sigma \in L} t \in \sigma\},\ T_I = \{t \in T \mid \exists_{\sigma \in L} t = first(\sigma)\},$$

$$T_O = \{t \in T \mid \exists_{\sigma \in L} t = last(\sigma)\},$$

$$X_L = \{(A,B) \mid A \subseteq T_L \wedge B \subseteq T_L \wedge \forall_{a \in A}\ \forall_{b \in B}\ a \rightarrow_L b \wedge$$

$$\wedge \forall_{a_1,a_2 \in A}\ a_1 \#_L a_2 \wedge \forall_{b_1,b_2 \in B}\ b_1 \#_L b_2\},$$

$$Y_L = \{(A,B) \in X_L \mid \forall_{(A',B') \in X_L}\ \forall_{b \in B}\ A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B')\},$$

$$P_L = \{p(A,B) \mid (A,B) \in Y_L\} \cup \{i_L, o_L\},$$

$$F_L = \{(a, p(A,B)) \mid (A,B) \in Y_L \wedge a \in A\} \cup \{(p(A,B), b) \mid (A \mid B) \in Y_L \wedge b \in B\} \cup$$

$$\cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_O\},$$

$$\alpha(L) = \{P_L, T_L, F_L\}.$$

Схема алгоритма:

1. Просматриваются содержащиеся в логе следы событий. Создается множество переходов ($T_L$) WF-сети.

2. Создается множество выходных переходов $T_I$ для начальной позиции.

3. Создается множество входных переходов $T_O$ для конечной позиции.

4. Создается множество $X_L$, используемое для определения позиций в распознаваемой WF-сети. На данном шаге $\alpha$-алгоритм определяет, какие переходы связаны между собой отношением casual. В каждом кортеже $(A, B)$ из множества $X_L$ каждый переход в множестве $A$, который строго следует за всеми переходами в множестве $B$, и ни один переход множества $A$ ($B$) не следует за другими переходами данного множества в каких-либо последовательностях срабатываний.

   Эти ограничения на элементы множеств $A$ и $B$ позволяют корректно выявлять такие виды конструкций, как $AND$-разделения/соединения и $XOR$-разделения/соединения. Также следует отметить, что корректное выявление $XOR$-разделений/соединений требует совмещения вершин.

5. Создается множество $Y_L$, используемое для определения позиций в распознаваемой WF-сети. На данном шаге $\alpha$-алгоритм отбирает наибольшие вершины из множества $X_L$ для подмножества $Y_L$, т. е. определяется окончательный набор вершин для распознаваемой сети (исключая начальную $i_L$ и конечную $o_L$ вершины сети).

6. Создаются позиции сети.

7. К созданным на предыдущем шаге позициям подсоединяются их входные/выходные переходы.

8. Полученная сеть возвращается алгоритмом.

Пусть $N = (P, T, F)$ — бездефектная WF-сеть, т. е. $N \in W$. Пусть $\alpha$ — $\alpha$-Mining-алгоритм, устанавливающий соответствие между логами событий $N$ и бездефектными WF-сетями, т. е. $\alpha : L \to W$. Если для любого полного лога событий $L$ сети $N$ алгоритм возвращает сеть $N$ (без учета названий позиций), то можно говорить, что $\alpha$ может распознать $N$.

Следует отметить, что ни один Mining-алгоритм не может найти имена позиций. Поэтому имена позиций игнорируются, т. е. говорят, что $\alpha$ может распознать $N$ без учета названий позиций.

## Класс распознаваемых моделей

**Теорема.** Пусть $N=(P,T,F)$ — бездефектная WF-сеть, и $L$ — полный лог событий сети $N$. Если для всех $a,b\in T$, $a\bullet\cap\bullet b=\varnothing$ или $b\bullet\cap\bullet a=\varnothing$, то $\alpha(L)=N$ без учета названий позиций.

Из теоремы следует, что $\alpha$-алгоритм гарантированно всегда распознает класс структурированных WF-сетей без коротких циклов.

В случае циклов длиной в одну задачу это происходит вследствие того, что $B>B$ подразумевает, что $B\to B$ невозможно (рис. 15.16).

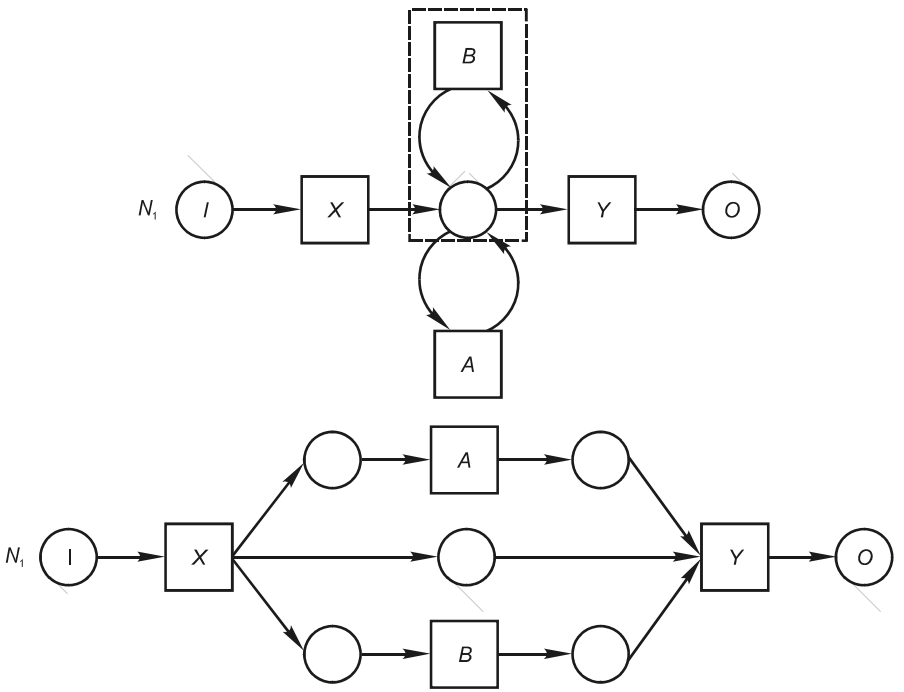



**Рис. 15.13.** Результат распознания $\alpha$-алгоритмом циклов длиной в одну задачу

А в случае с циклами длиной в две задачи $A>B$ и $B>A$ подразумевает, что $A\,|\,B$ и $B\,|\,A$, вместо $A\to B$ и $B\to A$ (рис. 15.17).

Кроме коротких циклов, $\alpha$-алгоритм также не может обрабатывать невидимые задачи, дублируемые задачи и принудительный выбор. С чем это связано см. подробнее в [135].

Также следует отметить, что $\alpha$-алгоритм весьма уязвим к шуму, т. е. к вкраплениям неправильно записанной или просто неверной информации в логе.

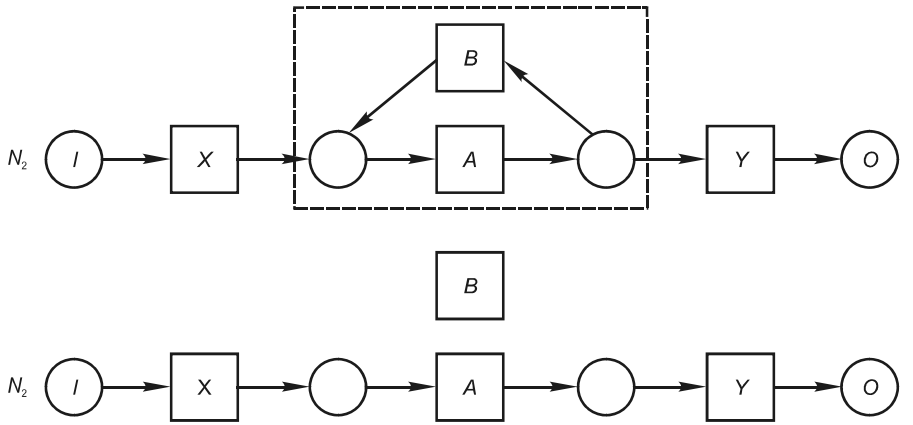



**Рис. 15.14.** Результат распознания $\alpha$ -алгоритмом циклов длиной в две задачи

Существуют модификации $\alpha$-алгоритма, которые помогают избавиться от некоторых его недостатков, в частности существует алгоритм Heuristics Miner, в котором учитываются не только бинарные отношения, связывающие события между собой, но и частота их появления. На практике это выглядит следующим образом: если $A > B$ встречается чаще, чем $B > A$, то вероятность того, что $A \to B$ выше, чем вероятность того, что $B \to A$.

В работах [137, 138] $\alpha$-алгоритм расширен для извлечения коротких циклов. В них авторы, добавляя больше ограничений к понятию законченности протоколов, переопределяют бинарные отношения.

$\alpha$-алгоритм был первоначально реализован в инструменте EMiT [140]. Впоследствии этот алгоритм стал программным расширением $\alpha$-алгоритма в библиотеке ProM. Наконец, важно заметить, что $\alpha$-алгоритм не принимает во внимание частоту отношения. Он только проверяет наличие отношения. Это также одна из причин, почему $\alpha$-алгоритм не является устойчивым к шуму.

Подход Weijters и др. [141] может быть рассмотрен как расширение $\alpha$-алгоритма. Он работает, основываясь на отношении следования. Однако, чтобы вывести остающиеся отношения (причину, параллельность и несвязанность), он подсчитывает частоту отношения следования в протоколе. По этой причине данный подход может обрабатывать шум. Главная идея применяемой эвристики — если задача $A$ чаще следует за задачей $B$, задача $B$ менее часто следует за $A$, то вероятность, что $A$ является причиной для $B$, выше. Поскольку алгоритм главным образом работает, основываясь на бинарных отношениях, нелокальные конструкции "не свободный выбор" не могут быть выявлены. Алгоритм был первоначально реализован в системе Little Thumb [142].

Ван Донджен (B. F. van Dongen) и др. [143, 144] ввели многошаговый подход к извлечению цепей процессов, ведомых событиями (Eventdriven Process Chains — EPCs). Первый шаг состоит в извлечении модели процесса для каждой трассы в экземпляре протокола. Чтобы сделать так, подход сначала удостоверяется, что ни одна задача не появляется более одного раза в трассе. Это означает, что каждому экземпляру задачи в трассе назначается уникальный идентификатор. После этого шага алгоритм выводит бинарные отношения, как $\alpha$-алгоритм. Эти бинарные отношения выводятся на уровне протокола. Основанный на этих отношениях подход строит модель для каждого трека протокола. Эти модели показывают определенный порядок между экземплярами задач в треке. Обратите внимание, что на уровне трека никакой выбор не возможен, потому что все задачи (экземпляры) в треке были действительно выполнены. Второй шаг выполняет агрегирование (или слияние) извлеченных моделей, в течение первого шага, для каждого трека. В основном, агрегируются идентификаторы, которые ссылаются на экземпляры одной и той же задачи. Алгоритм различает три типа точек разделения/соединения: *AND*, *OR* и *XOR*. Тип точки разделения/соединения устанавливается, основываясь на подсчете ребрами агрегированных задач. Если точка разделения появляется так же часто, как его прямые преемники, то тип точки разделения *AND*. Если появление его преемников составляет количество раз, которое эта точка была выполнена, то это *XOR*. Иначе тип *OR*.

Вен (Wen) и др. [145, 146] написали два расширения для $\alpha$-алгоритма *(см. разд. 2.2.7)*. Первое расширение — $\alpha$-алгоритм [145] — может извлекать структурированные сети потоков работ (SWF-сети) с короткими петлями. Расширение базируется на условии, что задачи в протоколе являются неатомными. Подход использует пересекающиеся времена выполнения задач, чтобы различить параллелизм и короткие петли. $\alpha$-алгоритм был реализован как алгоритм Tsinghuaalpha, в библиотеке ProM.

Второе расширение — $\alpha$++-algorithm [146] — извлекает сети Петри с местными или нелокальными конструкциями non-freechoice. Это расширение развивается в работах [137, 138]. Главная идея подхода состоит в рассмотрении окна с размером, больше чем 1, чтобы установить зависимости между задачами в протоколе.

Подходы сравнивались на основании их способности обрабатывать простейшие конструкции в моделях процесса, а также присутствии шума в протоколе. Наиболее проблемные конструкции, которые не могут быть извлечены всеми подходами: петли (особенно произвольные), невидимые задачи, "не освобождают выбор" (особенно нелокальные) и двойные задачи.

Петли и невидимые задачи не могут быть добыты, главным образом потому, что они не поддерживаются представлением, которое используется подходами.

"Не свободный выбор" не обнаруживается в протоколах событий, потому что большинство подходов, основываются на анализе локальной информации. Нелокальный "не свободный выбор" требует, чтобы методы рассматривали более отдаленные отношения между задачами.

Двойные задачи не могут быть извлечены, потому что много подходов принимают отношение "один к одному" между задачами в протоколах и их ярлыках.

Наконец, шумом нельзя должным образом заняться, потому что много методов не учитывают частоту зависимостей задачи.

## 15.3.4. Методы на основе генетических алгоритмов

### Проблема дублируемых задач

Большинство алгоритмов Process Mining для выявления перспективы управления потоком страдают серьезным недостатком. Данный недостаток заключается в том, что в теории, которая лежит в основе эвристических алгоритмов, практически всегда делается допущение, что одному уникальному событию должна соответствовать ровно одна вершина на модели (т. е. все вершины графов имеют уникальные метки). В результате, по мере прохода по логу, если алгоритмы встречают события, имеющие одинаковые имена, но разные события-предшественники и события-потомки, то в результирующей модели данные события объединяются в одну вершину, к которой подсоединяются все предшественники и все потомки (рис. 15.15).

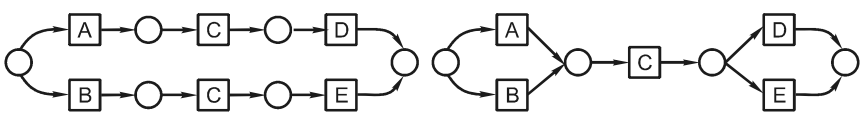



**Рис. 15.15.** Результат попытки выявления дублируемых задач эвристическими алгоритмами

На данный момент существует всего несколько алгоритмов, позволяющих выявлять дублируемые задачи, в частности генетический алгоритм [147].

### Описание алгоритма

Генетический алгоритм предъявляет следующие требования к выявляемой модели:

- модель должна иметь ровно одну начальную задачу;
- модель должна иметь ровно одну конечную задачу;
- начальная вершина должна иметь семантику *XOR*-соединение/разделение, в терминах сетей Петри это означает, что начальная задача должна иметь ровно одну входную и одну выходную позиции;

- конечная вершина должна иметь семантику *XOR*-соединение/разделение, в терминах сетей Петри это означает, что начальная задача должна иметь ровно одну входную и одну выходную позиции.

Как нетрудно заметить, данные требования можно легко удовлетворить для любой модели, добавив в начало и в конец искусственные начальную и конечную задачи.

Алгоритм работает следующим образом (рис. 15.16): вначале с помощью какого-нибудь эвристического алгоритма строится начальная популяция особей. Каждой особи сопоставляется мера соответствия желаемому результату, которая характеризует ее качество. В данном случае особь — это возможная модель процесса, а степень соответствия — функция, которая оценивает, насколько хорошо особь воспроизводит поведение в логе. Популяции эволюционируют путем выбора наилучших особей и создания новых особей, используя генетические операторы, такие как оператор пересечения (crossover — совмещение частей двух и более особей) и оператор мутации (случайная модификация особи).

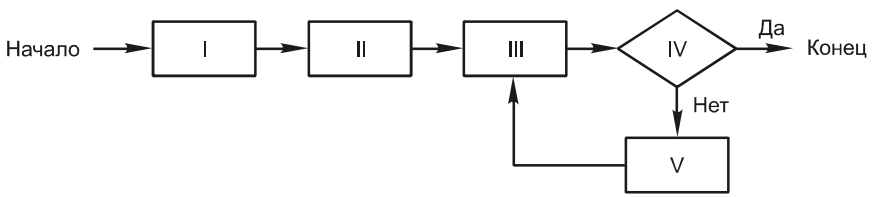



**Рис. 15.16.** Схема работы генетического алгоритма:
I — прочитать лог; II — построить начальную популяцию;
III — вычислить степень соответствия каждой особи;
IV — остановиться и вернуть наиболее соответствующих задаче особей;
V — создать следующую популяцию, используя элитизм и генетические операторы

Каждая особь в популяции представлена матрицей причинности (в ProM используется термин эвристическая сеть). Табл. 15.5 иллюстрирует матрицу причинности для модели процесса, изображенного на рис. 15.17.

**Таблица 15.5.** Модель в виде матрицы причинности

| Задача | Входные переходы | Выходные переходы |
|---|---|---|
| A | {{A}} | {{F,B,E},{E,C},{G}} |
| B | {{A}} | {{D}} |
| C | {{A}} | {{D}} |
| D | {{F,B,E},{E,C},{G}} | {} |
| E | {{A}} | {{D}} |

**Таблица 15.5** *(окончание)*

| Задача | Входные переходы | Выходные переходы |
|---|---|---|
| F | {{A}} | {{D}} |
| G | {{A}} | {{D}} |

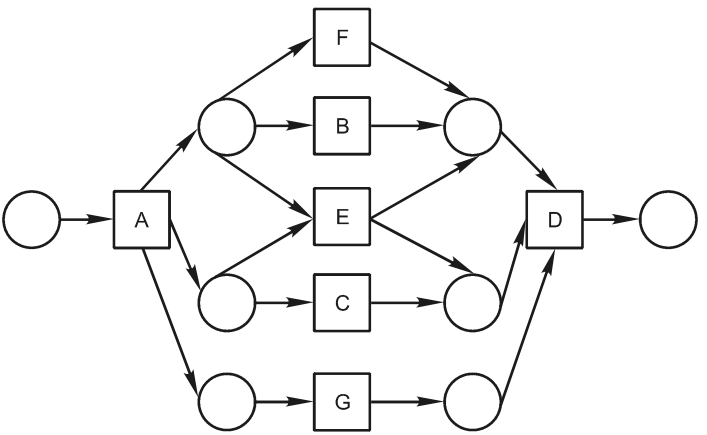



**Рис. 15.17.** Модель в виде сети Петри

Задачи из одного подмножества входного/выходного множеств находятся в *XOR*-отношении между собой, а задачи из разных входных/выходных подмножеств находятся в *AND*-отношении между собой.

В алгоритме существуют несколько вариантов получения начальной популяции (табл. 15.6) и использования генетических операторов.

**Таблица 15.6.** Возможные варианты получения начальной популяции

| | **Эвристики используются, чтобы установить зависимости между задачами** | **Зависимости между задачами устанавливаются случайным образом** |
|---|---|---|
| **Эвристики, основанные на причинной информации, используются, чтобы установить количество дубликатов на задачу** | Причинные эвристики (дубликаты + дуги) | Причинные эвристики (дубликаты) |
| **Эвристики, основанные на информации о порядке следования, используются, чтобы установить количество дубликатов на задачу** | Эвристики следования (дубликаты + дуги) | |

Мера соответствия является комбинацией различных метрик. В целом, для того чтобы получить модель, адекватно отображающую дублируемые задачи, используются следующие принципы:

- модель штрафуется за большое количество одновременно возбужденных переходов (как уже было отмечено ранее, неправильное распознавание дублируемых задач приводит к примыканию к одному переходу большого количества входных и выходных позиций);
- модель штрафуется за количество дублируемых задач с общими входными и выходными задачами (тоже чтобы не получалось больших узлов).

## Сравнение эффективности алгоритмов Process Mining

Так как за редким исключением алгоритмы Process Mining являются эвристическими, оценим их эффективность, исходя из теоретических соображений о том, какие типовые конструкции они могут выявлять, и основных принципов их работы. Сравнение описанных ранее алгоритмов приведено в табл. 15.7.

**Таблица 15.7.** Анализ эффективности алгоритмов Process Mining

| | **Вероятностные алгоритмы** | **Дизъюнктивная Workflow-схема** | **α-алгоритм** | **Генетический алгоритм для дублируемых задач** |
|---|---|---|---|---|
| Подход:<br>— многошаговый<br>— одношаговый | +<br>+ | +<br>+ | +<br>+ | +<br>+ |
| Последовательности | + | + | + | + |
| Выбор | + | + | + | + |
| Параллелизм | + | + | + | + |
| Циклы | + | | + | + |
| Принудительный выбор:<br>— локальный<br>— не локальный | + | + | + | +<br>+ |
| Невидимые задачи | + | + | | + |
| Дублируемые задачи | | | | + |
| Шум | + | + | | + |

Как видно из таблицы, только генетический алгоритм может распознавать все типовые конструкции и, в особенности, дублируемые задачи.

Как уже было отмечено, из всех алгоритмов Process Mining только генетический алгоритм для выявления дублируемых задач может распознавать все типовые конструкции. Остальные алгоритмы Process Mining на это не способны, что резко ограничивает область их применения, фактически они подходят только для очень узкого класса реальных задач.

К сожалению, генетический алгоритм тоже нельзя считать эффективным, т. к. у него есть другой недостаток, нехарактерный для остальных алгоритмов Process Mining, предназначенных для выявления перспективы управления потоком: его выполнение занимает очень большое время, что фактически сводит на нет все его преимущества.

Проблема больших временных затрат возникает вследствие того, что одной из основных метрик, используемой для вычисления меры соответствия и вычисляемой генетическим алгоритмом на каждом шаге, является процент правильно завершенных следов в логе. Это означает, что алгоритм как бы "проверяет" на каждом шаге свои результаты, "прокручивая" все имеющиеся в логе следы, через каждую особь (модель) в популяции. Хотя такой подход позволяет адекватно оценить меру соответствия модели логу, это также вызывает как минимум прямую зависимость между размером лога и временем выполнения алгоритма, что делает нецелесообразным его использование для серьезных задач из реальной жизни.

# 15.4. Библиотека алгоритмов Process Mining — ProM

## 15.4.1. Архитектура ProM

Данный раздел посвящен описанию использованных в работе сторонних программных средств. Подобными программными средствами являются ProM Framework (который был использован для построения моделей и анализа логов), а также ProM Import Framework (который послужил базой для реализации конвертера логов игрового сервера).

ProM Framework — это набор программных средств, направленных на решение основных задач Process Mining, т. е. получение и анализ информации о бизнес-процессах из лог-файлов. Данный набор средств является кроссплатформенным приложением, написанным на языке Java. Он состоит из базовой части, предоставляющей пользователю единую среду и графический интерфейс, и набора различных алгоритмов, реализованных в виде подклю-

чаемых модулей (plug-in). На данный момент ProM Framework содержит более чем 190 встроенных подключаемых модулей и поддерживает добавление дополнительных пользовательских модулей.

ProM Framework разработан в Техническом университете Эйндховен и является свободно распространяемым приложением с открытым кодом.

Основная концепция организации ProM Framework заключается в том, что каждый подключаемый модуль реализует только свою специфическую задачу, а среда отвечает за их взаимодействие между собой.

Для реализации этой концепции в ProM Framework используется единый MXML-формат внутренней организации лог-файлов, а также автоматическое подключение/отключение подключаемых модулей, которое опирается на информацию о входных/выходных данных каждого подключаемого модуля и то, какие объекты для обработки открыты в графическом интерфейсе в данный момент времени.

ProM поддерживает следующие типы подключаемых модулей:

- *подключаемые модули для импортирования* (import plug-ins) — позволяют использовать во время работы ранее созданные в других средствах модели, а также подключать их к уже открытым в среде логам;
- *подключаемые модули для выявления* (mining plug-ins) — реализуют алгоритмы извлечения информации из лог-файлов и представления ее в виде различных моделей;
- *подключаемые модули для экспортирования* (export plug-ins) — позволяют сохранять полученные на фазах извлечения и анализа информации модели, а также производить операции над логами;
- *подключаемые модули для анализа* (analysis plug-ins) — позволяют анализировать открытые в среде лог-файлы и модели;
- *подключаемые модули для преобразования* (conversion plug-ins) — позволяют преобразовывать созданные модели из одного формата в другой.

Кроме того, ProM Framework содержит набор лог-фильтров, предназначенных для очистки логов от ненужной или не актуальной для конкретной задачи информации.

Также в связке с ProM Framework распространяется отдельное приложение ProM Import Framework, предназначенное для преобразования лог-файлов различных форматов к используемому в ProM Framework единому формату MXML.

ProM Framework предоставляет пользователю полноценный многооконный графический интерфейс, позволяющий одновременно:

- открывать, просматривать и редактировать лог-файлы;
- импортировать, просматривать, сохранять модели;
- управлять выполнением алгоритмов;
- и т. д.

Наличие возможности добавления дополнительных подключаемых модулей, полноценный графический интерфейс и удобный MXML-формат делают ProM Framework удобной платформой для реализации и тестирования алгоритмов Process Mining. Также следует отметить наличие подробной справочной документации о создании пользовательских подключаемых модулей к ProM, что существенно упрощает их разработку.

Существенным минусом ProM является весьма ограниченная документация по использованию встроенных подключаемых модулей, что несколько затрудняет их применение. Также стоит отметить отсутствие полноценного интерфейса командной строки, что делает применение ProM в автоматизированном режиме мало возможным, однако данный недостаток не актуален для этой работы.

На сегодняшний день ProM уже содержит более 200 подключаемых модулей, реализующих множество алгоритмов, предназначенных для решения задач Process Mining. Имеющейся функциональности более чем достаточно для всестороннего анализа бизнес-процессов, что делает данное приложение подходящей платформой для проведения обсуждаемого исследования.

## 15.4.2. ProM Import Framework

Как уже было отмечено ранее, ProM Framework работает только с лог-файлами в MXML-формате, что, в свою очередь, создает проблему приведения существующих лог-файлов от разнообразных информационных систем к данному формату. Для решения этой проблемы в связке с ProM Framework распространяется набор программных средств ProM Import Framework.

ProM Import предоставляет разработчику подключаемых модулей графический интерфейс и выходной поток для автоматизированной записи лог-файлов на диск с возможностью применения различных свойств анонимности, позволяющих при желании скрыть или изменить часть информации в выходном потоке, независимо от процесса преобразования. Это дает возможность сконцентрироваться исключительно на задаче преобразования логов в MXML-формат, что делает данный комплекс средств привлекательной платформой для реализации этой задачи. По аналогии с ProM, ProM Import также является кроссплатформенным приложением, написанным на языке Java.

ProM Import имеет гибкую, расширяемую архитектуру. Для каждого формата лог-файлов создается отдельный подключаемый модуль для импорта (import

filter plug-in). Каждый подключаемый модуль представляет собой отдельный класс, наследующий от абстрактного суперкласса `ImportFilter` набор методов, которые он может вызывать в своем конструкторе, чтобы оповещать систему о своих конфигурационных свойствах и внешних зависимостях. Непосредственно для процедуры преобразования подключаемому модулю передается объект, являющийся реализацией интерфейса `FilterEnvironment`, соединяющий подключаемый модуль и основные возможности Framework во время процедуры преобразования.

ProM Import работает по принципу конвейера. Все его элементы реализуют интерфейс `LogFilter`, который позволяет гибко управлять процессом записи в выходные файлы. Данный интерфейс предоставляет методы начала и окончания записи лог-файлов, процессов, протеканий процессов и контрольных записей. `LogFilter` также осуществляет форматирование в MXML, но непосредственно запись выполняют потомки данного класса.

Класс `FilterManager` группирует набор подключаемых модулей, предоставляет доступ к ним и обеспечивает их конфигурацию в Framework для абстрактного доступа и изменения. `ImportController`, включающий `FilterManager`, обеспечивает видимость конфигурационной информации для всего приложения, а также удовлетворяет запросы фильтров во внешних ресурсах. Класс `ImportFilterFrame` реализует основной графический интерфейс приложения.

ProM Import предоставляет однооконный графический интерфейс. Список основных элементов управления выглядит следующим образом:

- список имеющихся подключаемых модулей, с помощью которого пользователь может выбрать нужный ему модуль;
- текстовое поле с названием выбранного подключаемого модуля;
- текстовое поле с описанием выбранного подключаемого модуля;
- текстовое поле с ФИО и контактными данными автора выбранного подключаемого модуля;
- список свойств данного подключаемого модуля;
- кнопка **Start**, с помощью нее пользователь может запустить процесс преобразования;
- кнопка **Abort**, с помощью нее пользователь может остановить запущенный процесс преобразования;
- кнопка **Reset**, с помощью нее пользователь может перезапустить процесс преобразования;
- кнопка **Help**, с помощью нее пользователь может открыть HTML-страницу с описанием подключаемого модуля;

- текстовое поле, содержащее путь к выходному каталогу, куда будут записываться файлы, получаемые в результате преобразования;
- кнопка **Change** открывает диалог выбора выходного каталога;
- кнопка **Anonymiser** открывает диалог свойств, влияющих на степень анонимности выходного потока MXML-файлов;
- меню **Main**, **Filter**, **Tools** и **Help** дублируют функциональность перечисленных ранее элементов управления.

В целом графический интерфейс ProM Import достаточно удобен и не содержит ничего лишнего.

# Выводы

По результатам данной главы можно сделать следующие выводы.

- Бизнес-процесс (БП) — последовательность операций, в ходе выполнения которых получается значимый для организации результат (продукты, услуги).
- Workflow — это автоматизация всего или части бизнес-процесса, в течение которого документы, информация или задачи передаются от одного участника к другому для их обработки в соответствии с набором процедурных правил.
- Workflow включает в себя следующие понятия: процесс, шаг процесса, переход, исполнитель, данные.
- Общий принцип работы Workflow-системы заключается в следующем: система получает на вход описание бизнес-процесса на формальном языке (схему) и согласно ему шаг за шагом выполняет операции, включенные в бизнес-процесс.
- Сервисно-ориентированная архитектура — подход к разработке программного обеспечения, основанный на использовании сервисов (служб) со стандартизированными интерфейсами.
- Workflow Mining — технология выявления часто встречающихся экземпляров процессов (шаблонов) из протоколов работы систем.
- Process Mining — технология построения формальных моделей экземпляров процессов по протоколам работы систем.
- MXML-формат — это расширяемый формат, основанный на языке разметки XML (eXtensible Markup Language). Он используется для представления и хранения информации в виде логов событий.
- Существуют следующие проблемы анализа процессов: наличие необходимой информации в протоколе, разнообразие типов элементов схем (после-

довательности, параллелизм, выбор, циклы, принудительный выбор, невидимые задачи, дублируемые задачи), наличие шума в протоколах.

- Вероятностные методы извлечения моделей из протоколов основаны на вычислении частоты появления последовательностей элементов.
- Дизъюнктивная Workflow-схема использует подход, основанный на кластеризации, т. е. разбиении модели на части, что удобно при анализе больших схем.
- $\alpha$-алгоритм работает, основываясь на бинарных отношениях (следствие, причина, параллельность и несвязанность) в протоколе, и является единственным алгоритмом Process Mining, для которого определен и доказан класс моделей, к которым он может быть применен.
- Генетические алгоритмы используют идею эволюционирования популяций и создания новых особей, применяя генетические операторы (такие как оператор пересечения и оператор мутации) для улучшения моделей, извлекаемых из протоколов.
- Алгоритмы Process Mining неодинаково успешно решают проблемы: вероятностные алгоритмы не работают с дублируемыми задачами, дизъюктивная Workflow-схема не обрабатывает циклы и дублируемые задачи, $\alpha$-алгоритм не работает с невидимыми и дублируемыми задачами, а генетические алгоритмы решают все проблемы, но являются довольно трудоемкими.
- Большинство алгоритмов Process Mining реализованы в свободно распространяемой библиотеке ProM, которая работает с протоколами, реализующими стандарт MXML.

# ПРИЛОЖЕНИЯ

# ПРИЛОЖЕНИЕ 1

# Нейронечеткие системы

## П1.1. Способы интеграции нечетких и нейронных систем

Нейронечеткие или гибридные системы, включающие в себя нечеткую логику, нейронные сети, генетические алгоритмы и экспертные системы, являются эффективным средством при решении большого круга задач реального мира.

Каждый интеллектуальный метод обладает своими индивидуальными особенностями (например, возможностью к обучению, способностью объяснения решений), которые делают его пригодным только для решения конкретных специфических задач.

Например, нейронные сети успешно применяются в распознавании моделей, но они неэффективны в объяснении способов достижения своих решений.

Системы нечеткой логики, которые связаны с неточной информацией, успешно применяются при объяснении своих решений, но не могут автоматически пополнять систему правил, которые необходимы для принятия этих решений.

Эти ограничения послужили толчком для создания интеллектуальных гибридных систем, где два или более методов объединяются для того, чтобы преодолеть ограничения каждого метода в отдельности.

Гибридные системы играют важную роль при решении задач в различных прикладных областях. Во многих сложных областях существуют проблемы, связанные с отдельными компонентами, каждый из которых может требовать своих методов обработки.

Пусть в сложной прикладной области имеется две отдельные подзадачи, например задача обработки сигнала и задача вывода решения, тогда нейронная

сеть и экспертная система будут использованы соответственно для решения этих отдельных задач.

Интеллектуальные гибридные системы успешно применяются во многих областях, таких как управление, техническое проектирование, торговля, оценка кредита, медицинская диагностика и когнитивное моделирование. Кроме того, диапазон приложения данных систем непрерывно растет.

В то время, как нечеткая логика обеспечивает механизм логического вывода из когнитивной неопределенности, вычислительные нейронные сети обладают такими заметными преимуществами, как обучение, адаптация, отказоустойчивость, параллелизм и обобщение.

Для того чтобы система могла обрабатывать когнитивные неопределенности так, как это делают люди, нужно применить концепцию нечеткой логики в нейронных сетях. Такие гибридные системы называются *нечеткими нейронными* или *нечетко-нейронными сетями*.

Нейронные сети используются для настройки функций принадлежности в нечетких системах, которые применяются в качестве систем принятия решений.

Нечеткая логика может описывать научные знания напрямую, используя правила лингвистических меток, однако много времени обычно занимает процесс проектирования и настройки функций принадлежности, которые определяют эти метки.

Обучающие методы нейронных сетей автоматизируют этот процесс, существенно сокращая время разработки и затраты на получение данных функций.

Теоретически нейронные сети и системы нечеткой логики равноценны, поскольку они взаимно трансформируемы, тем не менее на практике каждая из них имеет свои преимущества и недостатки.

В нейронных сетях знания автоматически приобретаются за счет применения алгоритма вывода с обратным ходом, но процесс обучения выполняется относительно медленно, а анализ обученной сети сложен ("черный ящик").

Невозможно извлечь структурированные знания (правила) из обученной нейронной сети, а также собрать особую информацию о проблеме для того, чтобы упростить процедуру обучения.

Нечеткие системы находят большое применение, поскольку их поведение может быть описано с помощью правил нечеткой логики, таким образом, ими можно управлять, регулируя эти правила. Следует отметить, что приобретение знаний — процесс достаточно сложный, при этом область изменения каждого входного параметра необходимо разбивать на несколько интервалов; применение систем нечеткой логики ограничено областями, в которых допустимы знания эксперта и набор входных параметров достаточно мал.

Для решения проблемы приобретения знаний нейронные сети дополняются свойством автоматического получения правил нечеткой логики из числовых данных.

Вычислительный процесс представляет собой использование следующих нечетких нейронных сетей. Процесс начинается с разработки "нечеткого нейрона", который основан на распознавании биологических нейронных морфологий согласно механизму обучения. При этом можно выделить следующие три этапа вычислительного процесса нечеткой нейронной сети:

- разработка нечетких нейронных моделей на основе биологических нейронов;
- модели синоптических соединений, которые вносят неопределенность в нейронные сети;
- разработка алгоритмов обучения (метод регулирования синоптических весовых коэффициентов).

На рис. П1.1 и П1.2 представлены две возможные модели нечетких нейронных систем.

Полученное лингвистическое утверждение интерфейсный блок нечеткой логики преобразует во входной вектор многоуровневой нейронной сети. Нейронная сеть может быть обучена вырабатывать необходимые выходные команды или решения.

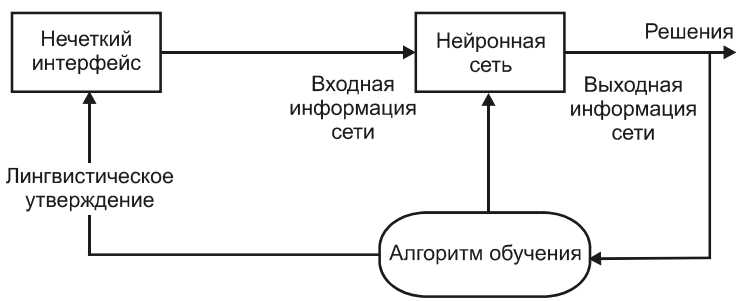



**Рис. П1.1.** Первая модель нечеткой нейронной системы

Многоуровневая нейронная сеть запускает интерфейсный механизм нечеткой логики.

Основные обрабатываемые элементы нейронной сети называют искусственными нейронами, или просто нейронами. Сигнал с нейронных входов $x_j$ считается однонаправленным, направление обозначено стрелкой, то же касается нейронного выходного сигнала.

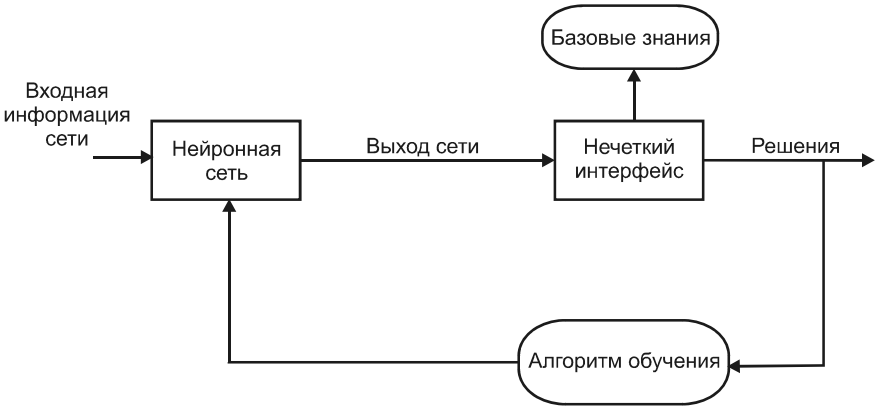



**Рис. П1.2.** Вторая модель нечеткой нейронной системы

Простая нейронная сеть представлена на рис. П1.3. Все сигналы и веса задаются вещественными числами.

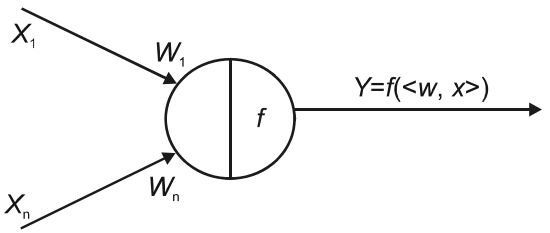



**Рис. П1.3.** Простая нейронная сеть

Входные нейроны не изменяют входной сигнал, поэтому выходные и входные параметры совпадают.

При взаимодействии с весовым коэффициентом $w_i$ для сигнала $x_i$ получаем результат $p_i = w_i x_i$, $i = 1, ..., n$. Элементы входной информации $p_i$ складываются и в результате дают входное значение для нейрона:

$$\text{net} = p_1 + ... + p_n = w_1x_1 + ... + w_nx_n.$$

Нейрон применяет свою передаточную функцию $f$, которая может быть сигмоидальной функцией вида:

$$f(t) = \frac{1}{1+e^{-t}}$$

для вычисления выходного значения:

$$y = f(\text{net}) = f(w_1x_1 + ... + w_nx_n).$$

Эту простую нейронную сеть, которая производит умножение, сложение и вычисляет сигмоидальную функцию $f$, назовем *стандартной нейронной сетью*.

*Гибридная нейронная сеть* — это нейронная сеть с нечеткими сигналами и весами, и нечеткими передаточными функциями (рис. П1.4). Однако: (1) можно объединить $x_i$ и $w_i$, используя другие непрерывные операции; (2) сложить компоненты $p_i$ с помощью других непрерывных функций; (3) передаточная функция может иметь вид любой другой непрерывной функции.

Обрабатывающий элемент гибридной нейронной сети называется *нечетким нейроном*.

Следует отметить, что все входные, выходные параметры и веса гибридной нейронной сети представляют собой вещественные числа из интервала [0, 1].

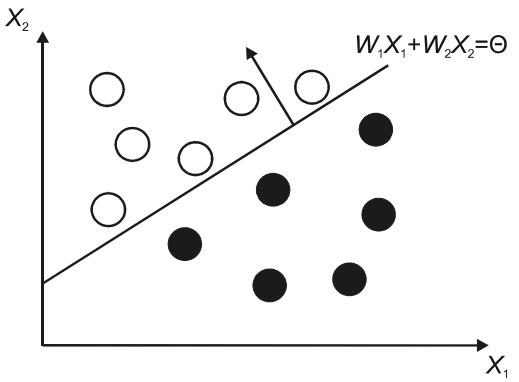



**Рис. П1.4.** Передаточная функция гибридной нейронной сети

# П1.2. Нечеткие нейроны

**Определение 1 — нечеткий нейрон И.** Сигналы $x_i$ и $w_i$ объединяются оператором максимума и дают:

$$p_i = \max\{w_i, x_i\},\ i = 1, 2.$$

Элементы входной информации $p_i$ объединяются с помощью оператора минимума и в результате дают выходную информацию нейрона:

$$y = \min\{p_1, p_2\} = \min\{w_1 \vee x_1, w_2 \vee x_2\}.$$

**Определение 2 — нечеткий нейрон ИЛИ.** Сигнал $x_i$ и вес $w_i$ объединяются оператором минимума:

$$p_i = \min\{w_i, x_i\},\ i = 1, 2.$$

Элементы входной информации $p_i$ объединяются с помощью оператора максимума и в результате дают выходную информацию нейрона:

$$y = \max\{w_1 \wedge x_1,\ w_2 \wedge x_2\}.$$

**Определение 3 — нечеткий нейрон ИЛИ (максимум произведения).** Сигнал $x_i$ и вес $w_i$ объединяются оператором умножения:

$$p_i = w_i\, x_i,\ i = 1, 2.$$

Элементы входной информации $p_i$ объединяются с помощью оператора максимума и в результате дают выходную информацию нейрона (рис. П1.5):

$$y = \max\{w_1\, x_1,\ w_2\, x_2\}.$$

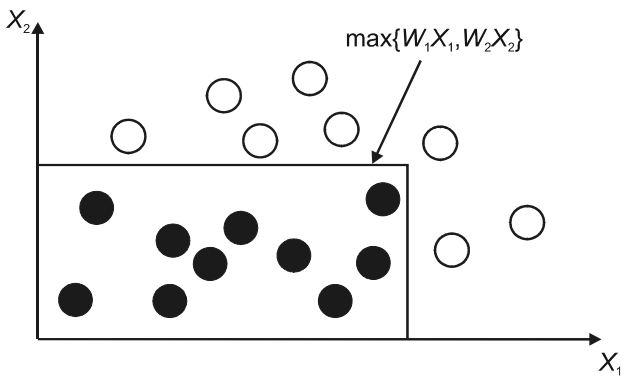



**Рис. П1.5.** Передаточная функция нечеткого нейрона ИЛИ

Нечеткие нейроны И и ИЛИ осуществляют стандартные логические операции над значениями множества. Роль соединений заключается в том, чтобы различить конкретные уровни воздействия, которое может быть оказано отдельными входными параметрами на результат их объединения.

Известно, что стандартные сети являются универсальными аппроксиматорами, т. е. они могут аппроксимировать любую непрерывную функцию на компактном множестве с любой точностью. Задача с таким результатом является неконструктивной и не дает информации о том, как построить данную сеть.

Гибридные нейронные сети применяются для реализации правил нечеткой логики IF-THEN конструктивным путем.

Хотя гибридные нейронные сети не способны использовать напрямую стандартный алгоритм вывода с обратным ходом, они могут быть обучены методами наискорейшего спуска распознавать параметры функций принадлежности, представляющих собой лингвистические термины в правилах.

# П1.3. Обучение методами спуска

Процедура обучения исправлению ошибок представляет собой всего лишь концепцию. Данная процедура состоит в следующем: в течение обучения входная информация поступает в сеть, где по возможным путям проходит преобразование, выдавая множество выходных значений.

Далее полученные экспериментально выходные значения сравниваются с теоретическими значениями и вычисляется несоответствие. Если экспериментальные и теоретические значения совпадают, то параметры сети не изменяются. Однако если эти значения расходятся, необходимо произвести изменения соединений в соответствии с полученным несоответствием.

Пусть функция $f$: $R \to R$ дифференцируемая, всегда возрастает в направлении своей производной и убывает в противоположном направлении.

В методе спуска для минимизации функции следующий шаг $w^{n+1}$ должен удовлетворять условию:

$$f(w^{n+1}) < f(w^n).$$

То есть значение функции $f$ в точке $w^{n+1}$ должно быть меньше значения функции на предыдущем шаге $w^n$.

В процедуре обучения исправлению ошибок на каждой итерации метода спуска рассчитывается направление спуска (противоположное направление производной) от точки $w^n$, следовательно, при достаточно малых $\eta > 0$ должно выполняться неравенство:

$$f(w^n - \eta f'(w^n)) < f(w^n),$$

где $w^{n+1}$ есть вектор

$$w^{n+1} = w^n - \eta f'(w^n).$$

Пусть функция $f$: $R^n \to R$ вещественная.

В методе спуска последующая итерация $w^{n+1}$ должна удовлетворять условию:

$$f(w^{n+1}) < f(w^n).$$

То есть значение функции $f$ в точке $w^{n+1}$ меньше, чем ее значение в предыдущем приближении $w^n$.

На каждой итерации метода спуска рассчитывается направление спуска в точке $w^n$ (направление, противоположное направлению производной), это означает, что при достаточно малых $\eta > 0$ должно выполняться неравенство:

$$f(w^n - \eta f'(w^n)) < f(w^n),$$

где $w^{n+1}$ есть вектор

$$w^{n+1} = w^n - \eta f'(w^n).$$

**УПРАЖНЕНИЕ 1.** Минимизировать функцию ошибки, заданную формулой:

$$E(w_1, w_2) = \frac{1}{2}\left[(w_2 - w_1)^2 + (1 - w_1)^2\right].$$

Найти аналитически вектор градиента:

$$E'(w) = \begin{bmatrix} \partial_1 E(w) \\ \partial_2 E(w) \end{bmatrix}.$$

Найти аналитически вектор весовых коэффициентов, который минимизирует функцию ошибок так, что $E'(w) = 0$.

Применить метод наискорейшего спуска для минимизации функции $E$.

РЕШЕНИЕ 1. Вектор градиента функции $E$:

$$E'(w) = \begin{bmatrix} (w_1 - w_2) + (w_1 - 1) \\ (w_2 - w_1) \end{bmatrix} = \begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix}$$

и единственное решение уравнения:

$$\begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Метод наискорейшего спуска для минимизации функции E:

$$\begin{bmatrix} w_1(t+1) \\ w_2(t+1) \end{bmatrix} = \eta \begin{bmatrix} 2w_1(t) - w_2(t) - 1 \\ w_2(t) - w_1(t) \end{bmatrix}.$$

где $\eta$ — константа обучения, а $t$ — указатель номера итерации.

То есть:

$$w_1(t+1) = w_1(t) - \eta(2\, w_1(t) - w_2(t) - 1),$$
$$w_2(t+1) = w_2(t) - \eta(2\, w_2(t) - w_1(t)).$$

# П1.4. Нечеткие схемы рассуждений

Пусть экспертная система на основе продукционных правил имеет вид:

$\Re_1$: if $x$ is $A_1$ and $y$ is $B_1$ then $z$ is $C_1$

$\Re_2$: if $x$ is $A_2$ and $y$ is $B_2$ then $z$ is $C_2$

...

$\Re_n$: if $x$ is $A_n$ and $y$ is $B_n$ then $z$ is $C_n$

fact: $x = x_0$ and $y = y_0$

consequence: $z$ is $C$

где $A_i$ и $B_i$ — нечеткие множества, $i = 1, ..., n$.

Процедура получения нечеткой выходной информации такой базы знаний включает следующие три этапа:

- найти границы применения каждого правила;
- найти выходные параметры каждого правила;
- объединить отдельные выходные параметры правил для получения полной выходной информации системы.

Суджено (Sugeno) и Такаги (Takagi) используют следующие правила:

$\Re_1$: if $x$ is $A_1$ and $y$ is $B_1$ then $z_1 = a_1x + b_1y$;

$\Re_2$: if $x$ is $A_2$ and $y$ is $B_2$ then $z_2 = a_2x + b_2y$.

Границы применения правил вычисляются по формулам:

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),\ \alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

затем выходная информация каждого отдельного правила выводится из отношения (см. рис. П1.6):

$$z_1 = a_1 x_0 + b_1 y_0,\ z_2 = a_2 x_0 + b_2 y_0,$$

и действие четкого управления выражается как

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \beta_1 z_1 + \beta_2 z_2,$$

где $\beta_1$ и $\beta_2$ — нормированные значения $\alpha_1$ и $\alpha_2$ по отношению к сумме $(\alpha_1 + \alpha_2)$, т. е.:

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2},\ \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}.$$

ПРИМЕР 1. Проиллюстрируем метод рассуждений Суджено на следующем примере:

if $x$ is SMALL and $y$ is BIG then $o = x - y$;

if $x$ is BIG and $y$ is SMALL then $o = x + y$;

if $x$ is BIG and $y$ is BIG then $o = x - 2y$,

где функции принадлежности SMALL и BIG определяются так:

$$\mathrm{SMALL}(\upsilon)=\begin{cases}1 & \text{if } \upsilon \le 1\\ 1-\dfrac{\upsilon-1}{4} & \text{if } 1\le \upsilon \le 5\\ 0 & \text{othewise}\end{cases};$$

$$\mathrm{BIG}(u)=\begin{cases}1 & \text{if } u \ge 5\\ 1-\dfrac{(5-u)}{4} & \text{if } 1\le u \le 5\\ 0 & \text{othewise}\end{cases}.$$

Предположим, что имеются следующие входные параметры: $x_0 = 3$ и $y_0 = 3$. Каковы будут при этом выходные параметры системы?

Граница применения первого правила:

$$\alpha_1 = \min\{\mathrm{SMALL}(3), \mathrm{BIG}(3)\} = \min\{0{,}5,\ 0{,}5\} = 0{,}5,$$

характерный выходной параметр первого правила:

$$o_1 = x_0 - y_0 = 3 - 3 = 0.$$

Граница применения второго правила:

$$\alpha_1 = \min\{\mathrm{BIG}(3), \mathrm{SMALL}(3)\} = \min\{0{,}5,\ 0{,}5\} = 0{,}5,$$

характерный выходной параметр второго правила:

$$o_2 = x_0 + y_0 = 3 + 3 = 6.$$

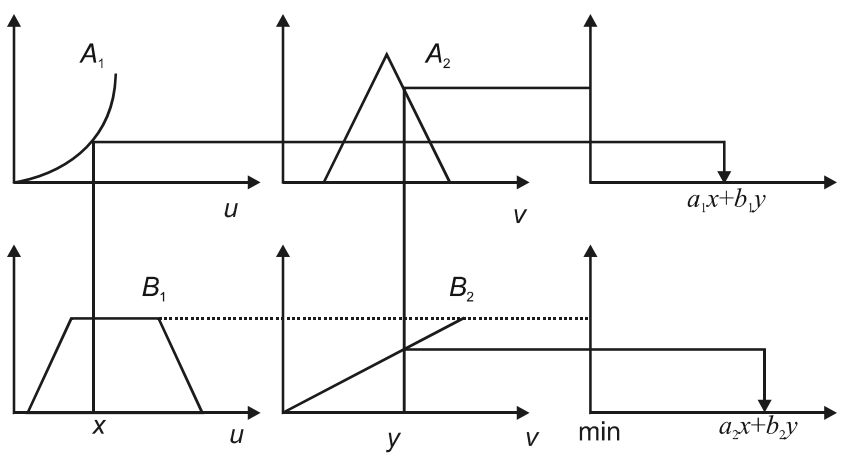



**Рис. П1.6.** Выходная информация правила

Граница применения третьего правила:

$$\alpha_1 = \min\{\text{BIG}(3), \text{BIG}(3)\} = \min\{0{,}5,\ 0{,}5\} = 0{,}5,$$

характерный выходной параметр третьего правила:

$$o_3 = x_0 + 2\,y_0 = 3 + 6 = 9.$$

выходной параметр системы, o, вычисляется из уравнения:

$$o = \frac{0 \times 0{,}5 + 6 \times 0{,}5 + 9 \times 0{,}5}{1{,}5} = 5{,}0\ .$$

В качестве примера покажем способ построения гибридной нейронной сети (названной *Jang-адаптивной сетью*), которая по функциональности является эквивалентом интерфейсного механизма Суджено.

Гибридная нейронная сеть, по вычислительным алгоритмам идентичная механизму Суджено, отражена на рис. П1.7.

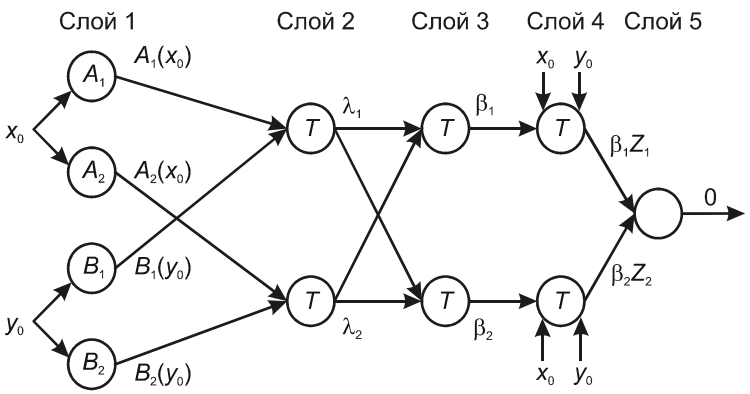



**Рис. П1.7.** Гибридная нейронная сеть

Для простоты будем учитывать только два правила и два лингвистических значения для каждого входного параметра.

- **Уровень 1.** Выходной параметр узла представляет собой степень соответствия данного входного параметра лингвистической метке, связанной с этим узлом. Обычно выбираются колоколообразные функции принадлежности:

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u - a_{i1}}{b_{i1}}\right)^2\right],$$

$$B_i(\upsilon) = \exp\left[-\frac{1}{2}\left(\frac{\upsilon - a_{i2}}{b_{i2}}\right)^2\right],$$

определяющие лингвистические термины, где $\{a_{i1}, a_{i2}, b_{i1}, b_{i2}\}$ — *множество параметров*.

По мере изменения значений этих параметров соответственно меняются колоколообразные функции, принимая таким образом различные формы функций принадлежности для лингвистических меток $A_i$ и $B_i$.

На самом деле, любые такие непрерывные функции принадлежности, как трапециевидные и треугольные, также являются квантифицируемыми вариантами узловых функций данного уровня. Параметры этого уровня относятся к *исходным параметрам*.

- **Уровень 2.** Для каждого узла вычисляется мощность действия соответствующего правила.

  Выходная информация, помещаемая на вершину нейрона, составляет:

$$\alpha_1 = A_1(x_0) \times B_1(y_0) = A_1(x_0) \wedge B_1(y_0),$$

  а выходная информация основания нейрона:

$$\alpha_2 = A_2(x_0) \times B_2(y_0) = A_2(x_0) \wedge B_2(y_0).$$

  Оба узла данного уровня имеют метку $T$, потому что можно выбрать другие $t$-нормы для моделирования логического оператора *и*. Узлы этого уровня называются *узлами правил*.

- **Уровень 3.** Каждый узел данного уровня имеет метку $N$, указывая на нормированность границ применения правил.

  Выходная информация вершины нейрона нормализует (при этом осуществляются операции сложения и нормализации нормированности границ) границу применения первого правила:

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}.$$

  Выходная информация основания нейрона нормирует. При этом осуществляется операция сложения и нормализации нормированности границ:

$$\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}.$$

- **Уровень 4.** Выходная информация вершины нейрона является результатом нормированной границы применения и отдельного выходного параметра первого правила:

$$\beta_1 z_1 = \beta_1(a_1 x_0 + b_1 y_0).$$

Выходная информация вершины нейрона является результатом нормированной границы применения и отдельного выходного параметра второго правила:

$$\beta_2 z_2 = \beta_2(a_2 x_0 + b_2 y_0).$$

- **Уровень 5.** Для отдельного узла данного уровня рассчитывается полная выходная информация как сумма всех входных сигналов, т. е.:

$$o = \beta_1 z_1 + \beta_2 z_2.$$

Если задана нечеткая обучающая последовательность:

$$\{(x^k, y^k), k = 1, ..., K\},$$

то параметры гибридной нейронной системы (которая определяет форму функций принадлежности исходных условий) могут быть изучены с помощью метода спуска.

> ***Внимание!***
> Данная архитектура и процедура обучения называются ANFIS (нечеткая интерфейсная система на основе адаптивной сети Jang).

Функция ошибок для модели $k$ может быть задана следующим образом:

$$E_k = \frac{1}{2} \times \left( y^k - o^k \right)^2,$$

где $y^k$ — желаемый результат, а $o^k$ — экспериментальное значение, полученное при расчете гибридной нейронной сети.

Имеем *упрощенную нечеткую схему рассуждений*; если характерная выходная информация правил задана crisp-числами, то можно использовать их весовую сумму (где веса представляют собой firing-мощность соответствующих правил) для определения полной выходной информации системы:

$$\Re_1: \text{if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_1 \text{ then } o = z_1$$

$$...$$

$$\Re_m: \text{if } x_1 \text{ is } A_m \text{ and } x_2 \text{ is } B_m \text{ then } o = z_m$$

$$\text{fact: } x_1 = u_1 \text{ and } x_2 = u_2$$

$$\text{consequence: } o = z_0$$

где $A_{ij}$ — нечеткие множества.

Получаем значение $z_0$ из начального содержания базы данных $\{u_1, u_2\}$ и из базы правил нечеткой логики, используя упрощенную нечеткую схему рассуждений как среднее из выходной информации отдельно взятых правил:

$$o = z_0 = \frac{z_1\alpha_1 + \ldots + z_m\alpha_m}{\alpha_1 + \ldots + \alpha_m},$$

где граница применения *i*-го правила определяется следующим образом:

$$\alpha_i = A_i(u_1) \wedge B_i(u_2).$$

# П1.5. Настройка нечетких параметров управления с помощью нейронных сетей

Нечеткие рассуждения используются во многих областях. Для реализации нечеткого контроллера необходимо определить функции принадлежности, представляющие лингвистические термины лингвистических правил вывода (рис. П1.8).

Рассмотрим лингвистический термин "примерно один". Очевидно, что соответствующее нечеткое множество должно быть унимодальной функцией с максимумом в точке 1. Для нахождения максимума ни форма, которая может быть треугольной или гауссовской, ни диапазон значений, которые определяют функцию принадлежности, не позволяют определить понятие "примерно один".

Как правило, главный эксперт имеет некоторые соображения о диапазоне значений функций принадлежности, но он уже может рассуждать о немного измененном диапазоне.

> ***Внимание!***
> Эффективность нечетких моделей, представляющих собой нелинейные отношения входа/выхода, зависит от нечеткого разделения входного пространства.

В связи с этим, настройка функций принадлежности становится важным вопросом для нечеткого контроллера. Далее задача настройки может быть представлена как задача оптимизации нейронных сетей, а *генетические алгоритмы* предоставляют возможные пути решения этой задачи.

Прямой подход заключается в определении точной формы функций принадлежности нескольких переменных, которые в свою очередь могут быть изучены с помощью нейронной сети.

Согласно этой идеи функции принадлежности принимают вид функций симметричных треугольников, зависящих от двух параметров, один из которых определяет максимум данной функции, второй задает ширину основания функции.

Оба подхода требуют множества экспериментальных данных в виде правильных кортежей входа/выхода и подробного описания правил, включающего предварительное определение соответствующих функций принадлежности.

Опишем простой метод обучения функций принадлежности антецедента (предыдущий член отношения) и консеквента (последующий член отношения) нечетких правил IF-THEN.

Предполагается, что неизвестное нелинейное отображение, выполняемое нечеткой системой, может быть представлено в следующем виде:

$$y^k = f\left(x^k\right) = f\left(x_1^k, \ldots, x_n^k\right),$$

для $k = 1, ..., K$, т. е. имеется следующая обучающая последовательность:

$$\{(x^1, y^1), ..., (x^K, y^K)\}.$$

Для моделирования неизвестного отображения $f$ применим упрощенное нечеткое правило IF-THEN следующего вида:

$$\Re_i: \text{ if } x_1 \text{ is } A_{i1} \text{ and } ... \text{ and } x_n \text{ is } A_{in} \text{ then o} = z_i$$

$i = 1, ..., m$, где $A_{ij}$ — нечеткие числа треугольной формы, а $z_i$ — вещественные числа.

В данном случае слово "упрощенное" означает, что выходная информация правил выхода представляется crisp-числами и поэтому становится возможным использование весовой суммы (где веса есть мощности действия соответствующих правил) для получения общей выходной информации системы.

Положим, o есть выход нечеткой системы, соответствует входу $x$. Соответственно, firing-уровень $i$-го правила, обозначенный через $\alpha_i$, определяется оператором произведения следующим образом:

$$\alpha_i = \prod_{j=1}^{n} Aij\left(x_j\right),$$

а выход системы вычисляется как

$$\text{o} = \frac{\sum_{i=1}^{m} \alpha_i z_i}{\sum_{i=1}^{m} \alpha_i}.$$

Чаще всего ошибка для $k$-го обучающего образа задается формулой

$$E = \frac{1}{2}\left(\text{o} - y\right)^2,$$

где o — рассчитанный выход нечеткой системы $\Re$, соответствующий входному образу $x$, а $y$ — желаемый результат.

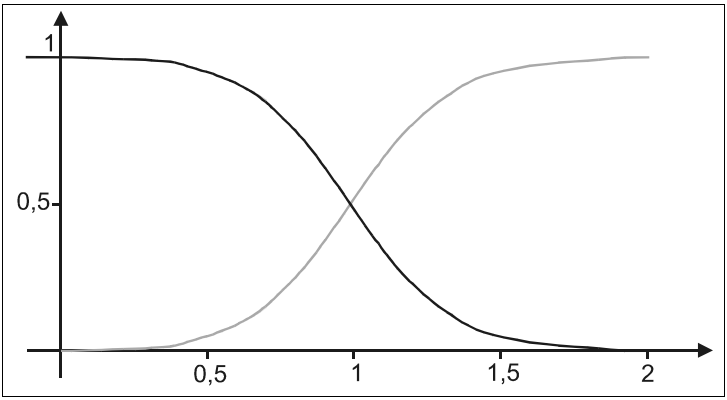



**Рис. П1.8.** Симметричная функция принадлежности

Метод наискорейшего спуска применяется для обучения $z_i$ в консеквентной части нечеткого правила $\Re_l$, т. е.:

$$z_i(t+1) = z_i(t) - \eta \frac{\partial \mathrm{E}}{\partial z_i} = z_i(t) - \eta(\mathrm{o} - y) \frac{\alpha_1}{\alpha_1 + \ldots + \alpha_m},$$

для $i = 1, ..., m$, где $m$ — обучающая константа, а $t$ указывает количество регулировок $z_i$. Проиллюстрируем описанный ранее процесс настройки на простом примере.

Пусть заданы два нечетких правила с одним входным и одним выходным параметрами:

$$\Re_1\text{: if } x \text{ is } A_1 \text{ then } \mathrm{o} = z_1;$$

$$\Re_2\text{: if } x \text{ is } A_2 \text{ then } \mathrm{o} = z_2,$$

где нечеткие области $A_1$ SMALL и $A_2$ BIG имеют сигмовидные функции принадлежности вида:

$$A_1(x) = \frac{1}{1 + \exp(-b(x-a))},$$

$$A_2(x) = \frac{1}{1 + \exp(b(x-a))}.$$

Здесь $a$ и $b$ — параметры $A_1$ и $A_2$.

В этом случае уравнение $A_1(x) + A_2(x) = 1$ справедливо для всех $x$ областей $A_1$ и $A_2$.

Общий выход системы вычисляется по формуле:

$$\text{o} = \frac{A_1(x) z_1 + A_2(x) z_2}{A_1(x) + A_2(x)}.$$

Весовые коэффициенты определяются следующим образом:

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E}{\partial z_1} = z_1(t) - \eta(\text{o} - y)\, A_1(x),$$

$$z_2(t+1) = z_2(t) - \eta \frac{\partial E}{\partial z_2} = z_2(t) - \eta(\text{o} - y)\, A_2(x^k),$$

$$a(t+1) = a(t) - \eta \frac{\partial E(a,b)}{\partial a},$$

$$b(t+1) = b(t) - \eta \frac{\partial E(a,b)}{\partial b},$$

где

$$\frac{\partial E(a,b)}{\partial a} = (\text{o} - y)\frac{\partial \text{o}^k}{\partial a} = (\text{o} - y)\frac{\partial}{\partial a}\left[z_1 A_1(x) + z_2 A_2(x)\right] =$$

$$= (\text{o} - y)\frac{\partial}{\partial a}\left[z_1 A_1(x) + z_2 (1 - A_1(x))\right] = (\text{o} - y)(z_1 - z_2)\frac{\partial A_1(x)}{\partial a} =$$

$$= (\text{o} - y)(z_1 - z_2)\, b\, A_1(x)\, A_2(x);$$

$$\frac{\partial E(a,b)}{\partial b} = (\text{o} - y)(z_1 - z_2)\frac{\partial A_1(x)}{\partial b} = -(\text{o} - y)(z_1 - z_2)(x - a)\, A_1(x)\, A_2(x).$$

Это означает, что чем больше нечетких терминов (следовательно, правил) используется в базе правил, тем ближе будет выходной параметр к требуемым значениям аппроксимируемой функции.

**УПРАЖНЕНИЕ 2.** Предположим, что неизвестное отображение, производимое нечеткой системой, может быть представлено в виде:

$$y = f(x_1, x_2),$$

и заданы следующие две обучающие пары вход/выход:

$$\{(1{,}1;\ 1),\ (2{,}2;\ 2)\}$$

(т. е. если входной вектор (1, 1), тогда желаемый результат равен 1, а если входной вектор (2, 2), то желаемый результат будет равняться 2).

Для моделирования неизвестного отображения $f$ применим четыре нечетких правила IF-THEN.

$$\text{if } x_1 \text{ is SMALL and } x_2 \text{ is SMALL then o} = ax_1 - bx_2;$$
$$\text{if } x_1 \text{ is SMALL and } x_2 \text{ is BIG then o} = ax_1 + bx_2;$$
$$\text{if } x_1 \text{ is BIG and } x_2 \text{ is SMALL then o} = bx_1 + ax_2;$$
$$\text{if } x_1 \text{ is BIG and } x_2 \text{ is BIG then o} = bx_1 - ax_2,$$

где функции принадлежности нечетких чисел SMALL и BIG задаются так:

$$\text{SMALL}(\upsilon) = \begin{cases} 1 - \dfrac{\upsilon}{2}, & \text{if } 0 \le \upsilon \le 2 \\ 0 & \text{otherwise;} \end{cases},$$

$$\text{BIG}(\upsilon) = \begin{cases} 1 - \dfrac{2-\upsilon}{2}, & \text{if } 0 \le \upsilon \le 2 \\ 0 & \text{otherwise,} \end{cases},$$

где $a$ и $b$ — неизвестные параметры.

Общий выход системы рассчитывается с помощью механизма рассуждений Суджено.

Построить функции ошибок $E_1(a, b)$, $E_2(a, b)$ для первой и второй обучающей пары.

РЕШЕНИЕ 2. Пусть (1, 1) является входом нечеткой системы. Границы применения правил рассчитываются по формулам:

$$\alpha_1 = \text{SMALL}(1) \wedge \text{SMALL}(1) = 0{,}5;$$
$$\alpha_2 = \text{SMALL}(1) \wedge \text{BIG}(1) = 0{,}5;$$
$$\alpha_3 = \text{BIG}(1) \wedge \text{SMALL}(1) = 0{,}5;$$
$$\alpha_4 = \text{BIG}(1) \wedge \text{BIG}(1) = 0{,}5;$$

а выход системы:

$$o_1 = \frac{a+b}{2}.$$

Определим меру ошибки первой обучающей модели как:

$$E_1(a,b) = \frac{1}{2}\left(\frac{a+b}{2} - 1\right)^2,$$

а в случае второй обучающей модели имеем:

$$\alpha_1 = \text{SMALL}(2) \wedge \text{SMALL}(2) = 0;$$
$$\alpha_2 = \text{SMALL}(2) \wedge \text{BIG}(2) = 0;$$
$$\alpha_3 = \text{BIG}(2) \wedge \text{SMALL}(2) = 0;$$
$$\alpha_4 = \text{BIG}(2) \wedge \text{BIG}(2) = 1.$$

Выход системы рассчитывается по формуле $o_2 = 2b - 2a$.

Мера ошибки для второй обучающей модели определяется как

$$E_2(a,b) = \frac{1}{2}(2b - 2a - 2)^2.$$

**УПРАЖНЕНИЕ 3.** Предположим, что неизвестное отображение, производимое нечеткой системой, может быть представлено в виде:

$$y^k = f(x^k) = f(x_1^k, \ldots, x_n^k),$$

для $k = 1, \ldots, K$, т. е. имеется следующая обучающая последовательность:

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}.$$

Для моделирования неизвестного отображения $f$ применим три упрощенных нечетких правила IF-THEN следующего вида:

$$\text{if } x \text{ is SMALL then o} = z_1;$$
$$\text{if } x \text{ is MEDIUM then o} = z_2;$$
$$\text{if } x \text{ is BIG then o} = z_3,$$

где лингвистические термины $A_1 = \text{SMALL}$, $A_2 = \text{MEDIUM}$, а $A_3 = \text{BIG}$ имеют треугольную форму функций (рис. П1.9):

$$A_1(\upsilon) = \begin{cases} 1, & \text{if } \upsilon \le c_1 \\ \dfrac{c_2 - x}{c_2 - c_1}, & \text{if } c_1 \le \upsilon \le c_2 \\ 0 & \text{othewise,} \end{cases}$$

$$A_2(\upsilon) = \begin{cases} \dfrac{x - c_1}{c_2 - c_1}, & \text{if } c_1 \le \upsilon \le c_2 \\ \dfrac{c_3 - x}{c_3 - c_2}, & \text{if } c_2 \le \upsilon \le c_3 \\ 0 & \text{othewise,} \end{cases}$$

$$A_3(u) = \begin{cases} 1, & \text{if } u \geq c_3 \\ \dfrac{x - c_2}{c_3 - c_2}, & \text{if } c_2 \leq u \leq c_3 \\ 0 & \text{othewise.} \end{cases}.$$

Примените метод наискорейшего спуска для настройки исходных параметров $\{c_1, c_2, c_3\}$ и консеквентные параметры $\{y_1, y_2, y_3\}$.

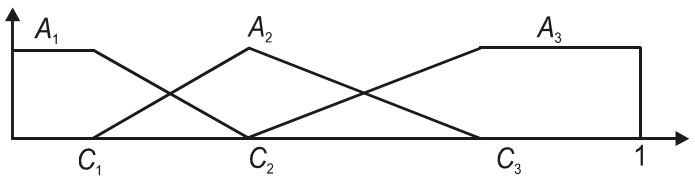



**Рис. П1.9.** Лингвистические термины $A_1$, $A_2$ и $A_3$

РЕШЕНИЕ 3. Пусть $x$ есть вход нечеткой системы. Границы применения правил вычисляются как:

$$\alpha_1 = A_1(x),\ \alpha_2 = A_2(x),\ \alpha_3 = A_3(x),$$

а выход системы рассчитывается по формуле:

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3} = \frac{A_1(x) z_1 + A_2(x) z_2 + A_3(x) z_3}{A_1(x) + A_2(x) + A_3(x)} =$$

$$= A_1(x) z_1 + A_2(x) z_2 + A_3(x) z_3,$$

где необходимо использовать тождество:

$$A_1(x) + A_2(x) + A_3(x) = 1$$

для всех $x \in [0, 1]$.

Определим меру ошибки для $k$-й обучающей модели как обычно:

$$E_k = E_k(c_1, c_2, c_3, z_1, z_2, z_3) = \frac{1}{2}\left(o^k(c_1, c_2, c_3, z_1, z_2, z_3) - y^k\right)^2,$$

где $o^k$ — рассчитанный выход нечеткой системы, соответствующий входной модели $x^k$, а $y^k$ — желаемый выход, $k = 1, ..., K$.

Метод наискорейшего спуска используется для обучения $z_i$ в консеквентной части $i$-го нечеткого правила, т. е.:

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E_k}{\partial z_1} = z_1(t) - \eta\left(o^k - y^k\right) A_1\left(x^k\right),$$

где $x^k$ — вход системы, $\eta > 0$ — обучающая константа, а $t$ — количество регулировок $z_i$.

Таким же образом можно настроить центры для $A_1$, $A_2$ и $A_3$.

$$c_1(t+1) = c_1(t) - \eta \frac{\partial E_k}{\partial c_1},$$

$$c_2(t+1) = c_2(t) - \eta \frac{\partial E_k}{\partial c_2},$$

$$c_3(t+1) = c_3(t) - \eta \frac{\partial E_k}{\partial c_3},$$

где $\eta > 0$ — обучающая константа, $t$ — количество регулировок параметров.

Частную производную функции ошибок $E_k$ по $c_1$ можно записать так:

$$\frac{\partial E_k}{\partial c_1} = \left(o^k - y^k\right)\frac{\partial o^k}{\partial c_1} = \left(o^k - y^k\right)\frac{(x - c_1)}{(c_2 - c_1)^2}(z_1 - z_2),$$

при $c_1 \le xk \le c_2$, и 0 — в противном случае.

Можно заметить, что регулировку центра невозможно произвести независимо от других центров, поскольку неравенство:

$$0 \le c_1(t+1) < c_2(t+1) < c_3(t+1) \le 1$$

должно выполняться для всех $t$.

# П1.6. Нейронечеткие классификаторы

Обычный подход к классификации моделей включает в себя кластеризацию обучающих образцов и сопоставление кластеров данным категориям. Сложность и ограничения предыдущего механизма в большей степени касаются отсутствия эффективного пути определения граничных областей между кластерами.

Эта проблема становится наиболее трудной в случае увеличения количества свойств, положенных в основу классификации.

Напротив, нечеткая классификация предполагает, что граница между двумя соседними классами является непрерывной с перекрывающей областью, в которой любой объект частично присутствует в каждом из классов. Данная точка зрения не только соответствует многим реальным приложениям, в которых категории имеют нечеткие границы, но и обеспечивает простое представление возможного деления множества пространства свойств.

Вкратце используются нечеткие правила IF-THEN для описания классификатора. Предположим, что $K$ структуры $x_p = (x_{p1}, ..., x_{pn})$, $p = 1, ..., K$ заданы из двух классов, где $x_p$ — $n$-мерный нечеткий вектор. Типичная нечеткая классификация правил для $n = 2$:

if $x_{p1}$ is SMALL and $x_{p2}$ is VERY LARGE then $x_p = (x_{p1}, x_{p2})$ belongs to Class $C_1$;

if $x_{p1}$ is LARGE and $x_{p2}$ is VERY SMALL then $x_p = (x_{p1}, x_{p2})$ belongs to Class $C_2$,

где $x_{p1}$ и $x_{p2}$ — свойства модели (или объекта) $p$, SMALL и VERY LARGE — лингвистические термины, характеризующиеся соответствующими функциями принадлежности.

Граница применения правила

$$\Re_i: \text{ if } x_{p1} \text{ is } A_i \text{ and } x_{p2} \text{ is } B_i \text{ then } x_p = (x_{p1}, x_{p2}) \text{ belongs to Class } C_i.$$

Что касается данного объекта $x_{pi}$, он интерпретируется как *степень принадлежности* $x_p$ к $C_i$.

Данная граница применения, обозначенная через $\alpha_i$, обычно определяется как:

$$\alpha_i = A_i(x_{p1}) \wedge A_2(x_{p2}),$$

где $\wedge$ — треугольная норма, моделирующая логическую связку "и".

По существу, нечеткое правило дает смысловое выражение качественных сторон человеческого сознания.

Основываясь на результатах сопоставления антецедентов правил и входных сигналов, ряд нечетких правил запускается параллельно с различными значениями мощностей действия.

Отдельно выполненные действия собираются вместе с помощью комбинационной логики. Более того, необходимо, чтобы система имела способность к обучению при обновлении и к тонкой настройке самой себя на основе вновь поступающей информации.

Задача нечеткой классификации заключается в создании соответствующего нечеткого деления пространства свойств. В данном случае слово "соответствующего" означает, что набор неправильно классифицированных моделей очень мал или отсутствует.

База правил должна быть улучшена за счет удаления неиспользуемых правил.

Проблема двуклассной классификации представлена на рис. П1.10. Предположим, что нечеткое деление для каждого входного свойства состоит из трех лингвистических терминов:

{SMALL, MEDIUM, BIG},

которые описываются треугольными функциями принадлежности.

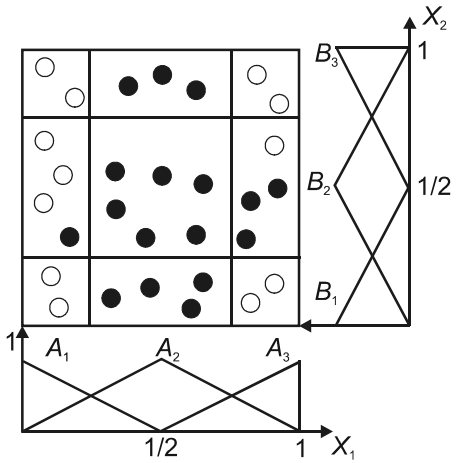



**Рис. П1.10.** Исходное нечеткое деление с 9 нечеткими подпространствами и 2 неправильно классифицированными моделями. Закрашенные и пустые кружки отображают данные модели класса 1 и класса 2 соответственно

Оба исходных нечетких деления из рис. П1.10 удовлетворяют полноте значением 0,5 для каждого входного параметра, а модель $x_p$ классифицирована в классе $j$, если существует, по крайней мере, одно правило для класса $j$ в базе правил, мощность действия которых (определяемая минимумом $t$-нормы) по отношению к $x_p$ больше или равна 0,5.

Таким образом, правило создается по найденной для данной входной модели $x_p$ комбинации нечетких множеств, каждое из которых обладает высокой степенью принадлежности для соответствующего входного свойства. Если эта комбинация не совпадает с антецедентами уже существующего правила, тогда создается новое правило.

Однако создание правила может произойти и в случае неверно выполненного нечеткого деления или в случае недостаточного количества лингвистических терминов для входных свойств. Тогда некоторые модели могут быть классифицированы ошибочно.

Следующие 9 правил могут быть созданы из исходных нечетких делений, показанных на рис. П1.10:

$\Re_1$: if $x_1$ is SMALL and $x_2$ is BIG then $x_p = (x_1, x_2)$ belongs to Class $C_1$;

$\Re_2$: if $x_1$ is SMALL and $x_2$ is MEDIUM then $x_p = (x_1, x_2)$ belongs to Class $C_1$;

$\Re_3$: if $x_1$ is SMALL and $x_2$ is SMALL then $x_p = (x_1, x_2)$ belongs to Class $C_1$;

$\Re_4$: if $x_1$ is BIG and $x_2$ is SMALL then $x_p = (x_1, x_2)$ belongs to Class $C_1$;

$\Re_5$: if $x_1$ is BIG and $x_2$ is BIG then $x_p = (x_1, x_2)$ belongs to Class $C_1$;

$\Re_6$: if $x_1$ is MEDIUM and $x_2$ is SMALL then $x_p = (x_1, x_2)$ belongs to Class $C_2$;

$\Re_7$: if $x_1$ is MEDIUM and $x_2$ is MEDIUM then $x_p = (x_1, x_2)$ belongs to Class $C_2$;

$\Re_8$: if $x_1$ is MEDIUM and $x_2$ is BIG then $x_p = (x_1, x_2)$ belongs to Class $C_2$;

$\Re_9$: if $x_1$ is BIG and $x_2$ is MEDIUM then $x_p = (x_1, x_2)$ belongs to Class $C_2$,

где используются лингвистические термины SMALL для $A_1$ и $B_1$, MEDIUM для $A_2$ и $B_2$ и BIG для $A_3$ и $B_3$.

Однако такой же уровень ошибок может быть достигнут рассуждением: если $x_1$ — MEDIUM, тогда модель $(x_1, x_2)$ принадлежит классу 2 независимо от значения $x_2$, т. е. следующие 7 правил дают тот же результат классификации:

$\Re_1$: if $x_1$ is SMALL and $x_2$ is BIG then $x_p$ belongs to Class $C_1$;

$\Re_2$: if $x_1$ is SMALL and $x_2$ is MEDIUM then $x_p$ belongs to Class $C_1$;

$\Re_3$: if $x_1$ is SMALL and $x_2$ is SMALL then $x_p$ belongs to Class $C_1$;

$\Re_4$: if $x_1$ is BIG and $x_2$ is SMALL then $x_p$ belongs to Class $C_1$;

$\Re_5$: if $x_1$ is BIG and $x_2$ is BIG then $x_p$ belongs to Class $C_1$;

$\Re_6$: if $x_1$ is MEDIUM then $x_p$ belongs to Class $C_2$;

$\Re_7$: if $x_1$ is BIG and $x_2$ is MEDIUM then $x_p$ belongs to Class $C_2$.

Рис. П1.11 демонстрирует пример нечетких делений (3 лингвистических термина для первого входного свойства и 5 — для второго), которые корректно классифицируют модели.

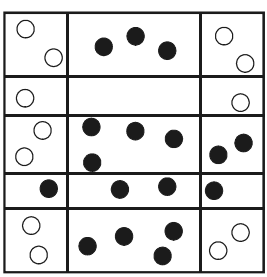


**Рис. П1.11.** Пример нечетких делений

Сан и Джанг в работе "C.-T. Sun and J.-S. Jang. A neuro-fuzzy classifier and its applications, in: Proc. IEEE Int. Conference on Neural Networks, San Francisco, 1993 94-98" предлагают нечеткий классификатор на основе адаптивной сети для решения проблем нечеткой классификации.

Рис. П1.12 представляет архитектуру данного классификатора с двумя входными параметрами $x_1$ и $x_2$. Обучающие данные подразделяются на два класса: $C_1$ и $C_2$, каждый вход представляется двумя лингвистическими терминами; таким образом получаем четыре правила.

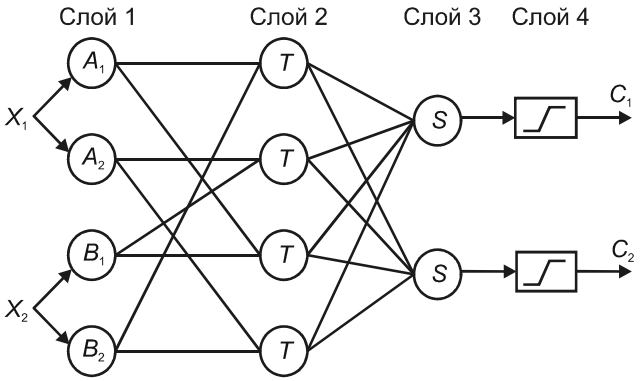



**Рис. П1.12.** Архитектура нечеткого классификатора

- **Уровень 1.** Выходной параметр узла представляет собой степень соответствия данного входного параметра лингвистической метке, связанной с этим узлом.

  Обычно выбираются колоколообразные функции принадлежности:

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u - a_{i1}}{b_{i1}}\right)^2\right],$$

$$B_i(v) = \exp\left[-\frac{1}{2}\left(\frac{v - a_{i2}}{b_{i2}}\right)^2\right],$$

  определяющие лингвистические термины, где $\{a_{i1}, a_{i2}, b_{i1}, b_{i2},\}$ — множество параметров.

  По мере изменения значений этих параметров соответственно меняются колоколообразные функции, принимая, таким образом, различные формы функций принадлежности для лингвистических меток $A_i$ и $B_i$.

- **Уровень 2.** Каждый узел создает сигнал, соответствующий конъюнктивной комбинации отдельных степеней соответствия. Выходной сигнал — это мощность действия нечеткого правила относительно объекта, который должен быть классифицирован.

> ***Внимание!***
> В большинстве систем классификаций моделей и поиска данных оператор конъюнкции играет важную роль, а его трактовка зависит от контекста.

Так как не существует единого оператора, который может быть применим для всех приложений, то для рассмотрения этого динамического свойства построения классификатора можно использовать параметризованные *t*-нормы.

Все узлы этого уровня обозначаются меткой *T*, т. к. есть возможность выбора любой *t*-нормы для моделирования логического оператора *и*. Узлы этого уровня называются *узлами правила*.

Свойства могут быть объединены путем компенсирования. Например, можно использовать обобщенный *p*-mean, предложенный Дукоф и Педриз (Dyckhoff и Pedrycz):

$$\left(\frac{x^p + y^p}{2}\right)^{1/p}, p \geq 1.$$

Возьмем линейную комбинацию мощностей действия правил уровня 3 и применим сигмоидальную функцию уровня 4 для расчета степени принадлежности определенному классу.

Если задано обучающее множество:

$$\{(x^k, y^k), k = 1, ..., K\},$$

где $x^k$ относится к *k*-й входной модели и

$$y^k = \begin{cases} (1, 0)^T \text{ if } x^k \text{ belongs to Class 1} \\ (0, 1)^T \text{ if } x^k \text{ belongs to Class 2} \end{cases},$$

тогда параметры гибридной нейронной сети (которые определяют форму функций принадлежности исходных условий) могут быть изучены с помощью методов спуска.

Функция ошибки для модели может быть определена как:

$$E_k = \frac{1}{2}\left[\left(o_1^k - y_1^k\right)^2 + \left(o_2^k - y_2^k\right)^2\right],$$

где $y^k$ — желаемый выход, а $o^k$ — подсчитанный результат гибридной нейронной сети.

# ПРИЛОЖЕНИЕ 2

# Особенности и эффективность генетических алгоритмов

## П2.1. Методы оптимизации комбинаторных задач различной степени сложности

В общем случае оптимизация или поиск наилучшего значения (набора параметров) некоторой заданной целевой функции является достаточно сложной задачей. Сложность оптимизации обусловливается, прежде всего, видом целевой функции, которая может иметь как глобальный, так и локальный оптимумы.

В настоящее время не существует метода оптимизации, который позволил бы решить любую задачу (был универсальным) и при этом однозначно определен как лучший среди других методов по точности решения.

По степени приближения к точному решению, а также по характеру пространства поиска задачи могут быть разделены на следующие категории.

- *Комбинаторные задачи* — характеризуются конечным и дискретным пространством поиска. Сущность любой комбинаторной задачи можно сформулировать следующим образом: найти на множестве $X$ элемент $x$, удовлетворяющий совокупности условий $K(x)$, в предположении, что пространство поиска $X$ содержит некоторое конечное число различных точек.
- *Общие задачи без ограничений* — имеют нелинейное и неограниченное пространство поиска. Методы оптимизации для таких задач обычно полагаются на правильность аналитической формулировки целевой функции. Оптимизация функции без ограничений заключается в максимизации или минимизации некоторой функции $U(x_1, ..., x_p)$.
- *Общие задачи с ограничениями* — могут быть сформулированы как задачи минимизации функции $U(x_1, ..., x_p)$ при следующих ограничениях:

$g_i(x_1, ..., x_p) \geq 0$ для $1 \leq i \leq m$, $h_j(x_1, ..., x_p) = 0$ для $1 \leq j \leq n$. Обычно задачи с ограничениями могут быть сведены к задачам без ограничений с помощью метода штрафов.

Если пространство поиска содержит конечное число точек, то наиболее точное решение может быть уверенно получено методом полного перебора. Этот метод имеет один очевидный недостаток — сложность вычислений, а следовательно, время, затрачиваемое на нахождение оптимального решения, существенно зависит от размерности пространства поиска. Метод перебора может быть достаточно эффективным только в небольшом пространстве поиска. А если предположить, что за одну секунду может быть выполнен миллиард операций ($10^9$) и при этом процесс случайного поиска начался 15 млрд лет назад, то к настоящему времени можно было бы протестировать около $10^{27}$ точек из пространства поиска. Одна точка такого пространства будет представлять собой бинарную строку длиной $L$ ($10^{27} \approx 2^{90}$).

Градиентные методы, являющиеся основой линейного и нелинейного, динамического программирования, а также численного анализа, более универсальны, но менее точны. При этом усложнение ландшафта пространства поиска приводит к снижению эффективности таких методов. Методы градиента не гарантируют получение единственного оптимального решения, за исключением случая, когда пространство отображения является выпуклым и не допускает появления второстепенных вершин, плато и т. д.

С другой стороны, *эвристические* методы, к которым относятся генетические алгоритмы (ГА), являются наиболее универсальными, поэтому не гарантируют нахождения глобального оптимума, являющегося единственным решением задачи.

Характеристикой задачи и, соответственно, основой для классификации методов оптимизации является также *сложность* задачи. По степени сложности однозначно выделяются следующие задачи.

- *Линейные задачи* — сложность которых определяется как $O(n)$, где $n$ — размерность входных данных задачи.
- *Полиномиальные задачи* ($P$) — для них известен алгоритм, сложность которого составляет полином заданной, постоянной и не зависящей от размерности входной величины $n$ степени.
- *Экспоненциальные задачи* — сложность которых не менее порядка $f^n$, где $f$ — константа или полином от $n$.

Однако существует большое число задач, которые не попадают ни в один из перечисленных классов. Сложность решения таких задач не может быть определена априорно. К ним относятся: оптимизация пути коммивояжера, оптимальная загрузка емкости, оптимизация маршрутов, инвестиций и т. д.

В общем случае задача оптимизации в настоящее время не может быть отнесена к какому-либо классу.

ГА являются *стохастическим эвристическим* методом, в котором вероятность выбора состояния $S(t+1)$ зависит от состояния $S(t)$ и косвенно от предыдущих состояний. Стохастические методы позволяют решать широкий класс таких задач, поскольку не требуют жесткой формализации. Следует отметить, что стохастические методы оптимизации используются для решения *NP*-сложных комбинаторных задач, т. е. таких задач, к которым сводима любая задача из класса *NP*. При этом *NP*-сложные задачи не обязательно относятся к классу *NP*.

Каждый из стохастических и эвристических методов имеет свои достоинства и недостатки, обусловленные формулировкой и размерностью решаемой задачи. При этом математически доказано, что для комбинаторных задач оптимизации средняя эффективность всех алгоритмов для всех возможных задач одинакова. На рис. П2.1 приведена классификация эвристических и стохастических алгоритмов, результаты оценки эффективности которых приведены на рис. П2.2 и П2.3.

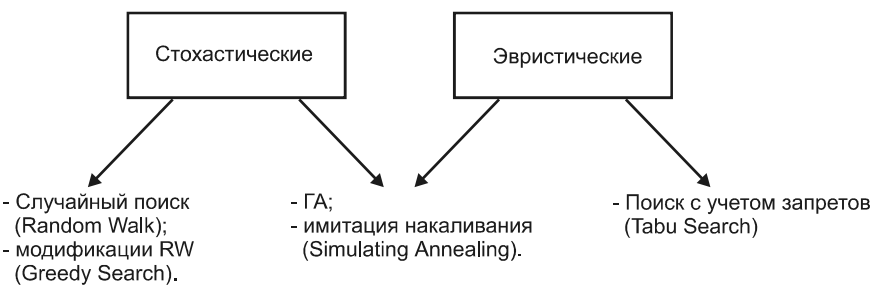



**Рис. П2.1.** Классификация эвристических и стохастических алгоритмов

Результаты решения задачи оптимизации (при одном запуске) на примере минимизации числа передающих станций при максимальной зоне радиоохвата с помощью методов случайного поиска (Random Walk), "жадного" поиска (модификация случайного поиска, Greedy Search), имитации накаливания (Simulating Annealing), а также поиска с учетом запретов (Tabu Search) и ГА представлены на графике зависимости качества решения от количества вычислений функции или числа шагов выполнения алгоритма (рис. П2.2 и П2.3).

Полученные результаты, усредненные по 10 запускам, доказывают справедливость утверждения о сравнимости эффективности всех перечисленных алгоритмов поиска глобального оптимума. Вместе с тем результаты, полученные при одном запуске, говорят о наибольшей эффективности двух методов поиска — ГА и поиска с учетом запретов.

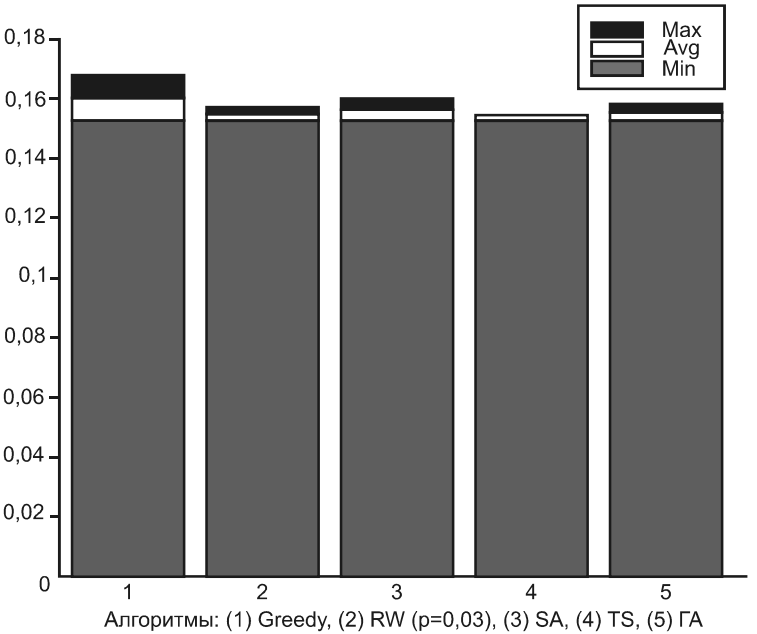



**Рис. П2.2.** График зависимостей качества решений задачи от числа шагов выполнения алгоритмов

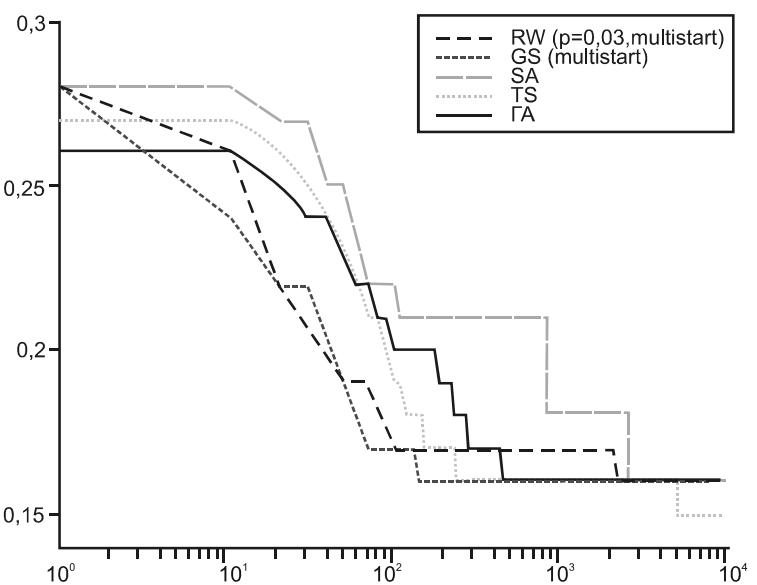



**Рис. П2.3.** Динамика получения результата

В соответствии с классификацией, приведенной на сайте **http://www.uwasa.fi /cs/publications/2NWGA/node40.html**, ГА являются стохастическим методом подкласса Markov Chain Monte Carlo (MCMC), в котором вероятность выбора состояния $S(t + 1)$ зависит от состояния $S(t)$ и косвенно от предыдущих состояний. К этому же подклассу стохастических методов могут быть отнесены: случайный поиск, "жадный" поиск и имитация накаливания.

Стохастические методы позволяют решать широкий класс таких задач, поскольку не требуют жесткой формализации.

По классификации, приведенной в [8] и на сайте **http://www.ee.cornell.edu /~bhaskar/msthesis/**, все перечисленные ранее методы, а также поиск с учетом запретов могут быть отнесены к методам *поиска соседей* (Neighborhood Search). В соответствии с этой классификацией методы объединяет общий принцип перехода из текущего состояния в последующее, заключающийся в выборе следующего состояния из определенного набора.

Несмотря на некоторые различия в классификациях, ГА и поиск с учетом запретов имеют общую основу. Их объединяет использование эвристики для перехода из текущего состояния в последующее. Особенностью ГА является работа с пространством поиска с помощью комбинирования решений, а поиска с учетом запретов — использование памяти состояний.

Эффективностью обоих методов обусловлено появление метода HGT (гибридной стратегии), использующей поиск с учетом запретов для повышения эффективности генетических операторов рекомбинации и мутации [4]. Вместе с тем, если по эффективности отмеченные методы сравнимы, то по надежности поиск с учетом запретов уступает ГА, поскольку для данного метода качество решения существенно зависит от начального состояния (или решения). Поэтому начальное решение с высоким значением оценочной функции (в случае решения задачи минимизации) может быстро привести к желаемому решению, а начальное решение с низким значением оценочной функции — существенно снизить скорость поиска.

Анализ результатов использования ГА позволяет выделить следующие условия, при выполнении которых задача решается эффективно:

- большое пространство поиска, ландшафт которого является негладким (содержит несколько экстремумов);
- сложность формализации оценки качества решения функцией степени пригодности;
- многокритериальность поиска;
- поиск приемлемого решения по заданным критериям в отличие от поиска единственного оптимального.

# П2.2. Сущность и классификация эволюционных алгоритмов

## П2.2.1. Базовый генетический алгоритм

Эволюционные алгоритмы, моделирующие процессы естественной эволюции, были предложены уже в 60-х годах прошлого века. Их особенностью является то, что они опираются на естественную эволюцию в природе, используя основные ее механизмы (отбор или селекцию, скрещивание и мутацию). Известны утверждения: "алгоритм является хорошим оптимизационным методом, потому что его принцип используется в природе", и наоборот: "алгоритм не может быть хорошим оптимизационным методом, потому что вы не находите его в природе".

Моделирование процесса естественной эволюции для эффективной оптимизации является первостепенной задачей теоретических и практических исследований в области эволюционных алгоритмов.

В 70-х годах прошлого века независимо друг от друга появились два различных направления в области эволюционных алгоритмов: генетический алгоритм Холланда и эволюционные стратегии (ЭС) Реченберга и Швефела. Эволюционные стратегии используют операторы селекции и мутации, а если использовать биологические термины, то эволюционная стратегия моделирует естественную эволюцию с помощью непарной репродукции (рис. П2.4).

**Эволюционные стратегии ($\mu + \lambda$).**
Шаг 1. Создание первоначальной популяции размера $\lambda$.
Шаг 2. Вычисление пригодности $F(x_i)$ $i = 1, \ldots, \lambda$.
Шаг 3. Селекция (отбор) $\mu < \lambda$ лучших индивидов.
Шаг 4. Создание $\lambda / \mu$ потомков каждого из $\mu$ индивидов с небольшими вариациями.
Шаг 5. Возврат к шагу 2.

**Рис. П2.4.** Разновидность эволюционных алгоритмов — эволюционные стратегии

Алгоритмы поиска, которые моделируют парную репродукцию, называются *генетическими алгоритмами*. Парная репродукция характеризуется рекомбинацией двух родительских строк для создания потомков. Эта рекомбинация называется *скрещиванием*.

Предпочтение разных генетических операторов в ЭС и ГА определило отношение к используемому размеру популяции. Так, Холланд подчеркивал важность рекомбинации в больших популяциях, в то время как Реченберг и Швефел, главным образом, рассматривали мутацию в очень маленьких популяциях.

При работе с ГА решения задачи должны быть представлены в виде строки с бинарной, целочисленной или вещественной кодировкой. Способ кодирования предполагает работу со строками фиксированной или переменной длины, возможна также и контекстно-зависимая кодировка. Основным отличием генетических программ (ГП) от ГА является работа с деревьями решений. При этом в ГП отсутствует необходимость в генетическом представлении задачи. Такая схема представления вносит гибкость в описание структур данных, однако решения могут стать очень объемными без улучшения производительности. Это справедливо и для эволюционных программ (ЭП).

На рис. П2.5 приведен базовый или стандартный ГА (СГА), предложенный Холландом, который явился основой для различных модификаций.

**СГА.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции индивидов $P(0) = x_1^0, \ldots, x^0_N$, $t = 0$.
Шаг 2. Вычисление средней пригодности $f_{cp}(t) = \sum_i^N f(x_i)/N$. Вычисление нормализованного значения степени пригодности $f(x_i)/f_{cp}(t)$ для каждого индивида.
Шаг 3. Назначение каждому индивиду $x_i$ вероятности $p(x_i,t)$ пропорционально нормализованной пригодности. Выбор $N$ векторов из $P(t)$, используя полученное распределение. Это дает набор отобранных родителей.
Шаг 4. Формирование случайным образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания, а также других генетических операторов, таких как мутация, для формирования новой популяции $P(t + 1)$.
Шаг 5. $t = t + 1$, возврат к шагу 2.

**Рис. П2.5.** Стандартный генетический алгоритм

## П2.2.2. Последовательные модификации базового генетического алгоритма

Как показывает анализ, модификации ГА отличаются, прежде всего, способом селекции индивидов. В основных модификациях ГА несколько способов селекции используется для достижения различных целей — упрощения формирования промежуточной популяции, распараллеливания работы алгоритма, возможности анализа и предсказания поведения ГА. Было произведено сравнение четырех различных схем селекции (для СГА и SSGA, рассматриваемых далее), показавшее, что эффективность всех методов примерно одинакова. Таким образом, в настоящее время абсолютно лучший метод селекции не определен.

Модификация стандартного варианта ГА (Steady State GA) [Whitley и Kauth, 1988] затронула способ формирования *промежуточной популяции* (Mating Pool), являющейся результатом отбора (селекции) для формирования наследников с помощью генетических операторов. SSGA не формируют промежу-

точную популяцию как стандартный ГА, а осуществляют последовательно выбор пары наилучших индивидов, применяя к ним генетические операторы с целью формирования наследников, которые заменяют худшие индивиды популяции. Данная модификация ГА представлена на рис. П2.6.

**SSGA.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции $P(0) = x_1^0, \ldots, x_N^0$, $t = 0$.
Шаг 2. Вычисление относительной (нормализованной) степени пригодности $f_н(x_i) = f(x_i)/\sum_i^N f(x_i)/N$.
Шаг 3. Выбор пары из лучших индивидов. Выбор худшего индивида. Применение скрещивания и мутации к выбранной паре лучших индивидов. Результат замещает худший индивид.
Шаг 4. $t = t + 1$, возврат к шагу 2.

**Рис. П2.6.** Steady State GA

При проектировании ГА могут быть выгодно использованы знания, полученные селекционерами в области искусственной селекции. Генетические алгоритмы селекционеров (ГАС) моделируют именно искусственную селекцию. ГАС представлен на рис. П2.7, где под виртуальным селекционером понимается некоторый механизм селекции, который и является основным отличием ГАС от стандартного ГА.

**ГАС.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции $P(0)$ размером $N$, $t = 0$.
Шаг 2. Виртуальный селекционер отбирает $T$ % популяции для создания потомков. Это дает набор отобранных родителей.
Шаг 3. Формирование случайным образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания и мутации, формируя новую популяцию $P(t + 1)$.
Шаг 5. $t = t + 1$, возврат к шагу 2.
Шаг 6. Возврат к шагу 3.

**Рис. П2.7.** Генетический алгоритм селекционеров

Селекция основывается преимущественно на статистических методах, которые позволяют выполнить теоретический анализ и прогнозировать эффективность механизмов селекции, мутации и рекомбинации с помощью введенных уравнений селекции, реакции на селекцию и понятия наследственности.

Еще одна модификация ГА затрагивает решение многокритериальных задач. Многокритериальный ГА (МГА) также является модификацией стандартного ГА и отличается способом селекции, поскольку при отборе пар родителей в этом случае используется не один, а несколько критериев. При этом предла-

гается большое число вариантов схем селекции и соответственно вариантов МГА. На рис. П2.8 приведен вариант МГА, предложенный Schaffer в 1984 г., — векторный ГА (VEGA). Сравнительные оценки показывают, что по эффективности VEGA имеет средние показатели, однако не оценивалась вычислительная сложность для различных вариантов МГА, по которой VEGA может существенно улучшить свои показатели.

**Многокритериальный ГА.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции $P(0) = x^0_1, \ldots, x^0_N$, $t = 0$.
Шаг 2. Последовательное выполнение шагов 2.1–2.3.
Шаг 2.1. Вычисление значения степени пригодности каждого индивида по критерию $i = 1, \ldots, k$.
Шаг 2.2. Для $j$ от 1 до $N/k$ осуществление селекции индивида из популяции в промежуточную популяцию.
Шаг 2.3. Возврат к шагу 2.1, если $I < k$.
Шаг 3. Формирование случайным образом из данного набора $N/2$ пар. Применение к каждой паре скрещивания, а также других генетических операторов, таких как мутация, формируя новую популяцию $P(t + 1)$.
Шаг 4. $t = t + 1$, возврат к шагу 2.

**Рис. П2.8.** Многокритериальный генетический алгоритм

## П2.2.3. Параллельные модификации базового генетического алгоритма

Стандартный ГА представляет собой строго синхронизованный последовательный алгоритм, который в условиях большого пространства поиска или сложного ландшафта пространства поиска может быть неэффективен по критерию времени. Эту проблему позволяет решить другой вид ГА — параллельный генетический алгоритм (ПГА). Следует отметить, что любая последовательная модификация стандартного ГА может быть преобразована в параллельную.

По степени распараллеливания можно выделить следующие типы параллельных ГА:

- ПГА на базе популяции;
- ПГА на базе подпопуляций;
- ПГА на базе индивидов.

*ПГА на базе популяции* сохраняет стандартную структуру ГА, работающего с целой популяцией, распараллеливание реализуется на этапе скрещивания и мутации (см. шаг 4, рис. П2.5). По степени распараллеливания процессов можно выделить следующие модели [8]:

- синхронная модель "ведущий-ведомый", где главный процесс хранит целую популяцию в собственной памяти, выполняет селекцию, скрещивание и мутацию, но оставляет вычисление степени пригодности новых индивидов *k* подчиненным процессам;
- полусинхронная модель "ведущий-ведомый", где новый индивид обрабатывается по мере освобождения одного из процессов;
- асинхронная параллельная модель, где индивиды популяции хранятся в общей памяти, к которой можно обращаться *k* параллельным процессам. Каждый процесс выполняет оценку степени пригодности, а также генетические операции.

Каждый процесс работает независимо от других. Единственное отличие между этой моделью и стандартным ГА заключается в механизме селекции [4]. Очевидным в этом случае является вариант использования $N/2$ параллельных процессоров при популяции в $N$ индивидов. Тогда каждый процессор дважды случайным образом выбирает два индивида из общей памяти и оставляет лучшего. Два выбранных индивида затем подвергаются скрещиванию, мутации и оценке степени пригодности. Возникающие в результате наследники размещаются в общей памяти.

**Распределенный ПГА.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции индивидов и разделение на подпопуляции $SP_1, \ldots, SP_N$.
Шаг 2. Формирование структуры подпопуляций.
Шаг 3. Для $SPi$ , $i = 1, \ldots, N$ — выполнение параллельно шагов 3.1–3.3.
Шаг 3.1. Применение в течение $m$ поколений селекции и генетических операторов.
Шаг 3.2. Перемещение $k$ хромосом в соседние подпопуляции.
Шаг 3.3. Получение хромосом из соседних подпопуляций.
Шаг 4. Возврат к шагу 3.

**Рис. П2.9.** Распределенный параллельный генетический алгоритм

Особенность *ПГА на базе подпопуляций* заключается в использовании независимых конкурирующих подпопуляций, которые обмениваются индивидами с заданной частотой (распределенный ПГА, рис. П2.9). При этом каждый процессорный блок выполняет последовательный ГА с собственной подпопуляцией, при условии максимизации одной общей для всех функции степени пригодности. В этом случае для обмена индивидами должна быть определена структура связей подпопуляций. С точки зрения оценки и сравнения эффективности может быть рассмотрен вариант распределенной модели, в которой обмен индивидами не осуществляется. Результаты, представленные в [5], свидетельствуют о большей эффективности распределенного ПГА по сравнению с этим частным случаем, а также со стандартным ГА.

Существенным недостатком модели может стать снижение степени разнообразия при интенсивном обмене индивидами. С другой стороны, недостаточно частое перемещение может привести к преждевременной сходимости подпопуляций. При построении такой модели важно определить следующее:

- связи между процессорами для обмена индивидами;
- частоту обмена индивидами (оптимальной является частота обмена через 20 поколений [5]);
- степень перемещения или число обмениваемых индивидов (оптимальным является 20 % подпопуляции [5]);
- способ селекции индивида для обмена;
- критерий, по которому полученный индивид сможет заменить члена подпопуляции.

С точки зрения времени и даже числа поколений, затрачиваемых на решение задачи, ПГА эффективнее стандартного ГА, но при этом некоторые задачи могут быть слишком простыми для ПГА. Параллельный поиск имеет смысл в том случае, если пространство поиска большое и сложное [3]. Увеличение числа процессоров в данной модели улучшает скорость сходимости, но не качество решения.

*ПГА на базе индивидов* имеют одну строку индивида, постоянно находящуюся в каждом процессорном элементе (ячейке). Индивиды выбирают пары и рекомбинируют с другими индивидами в их непосредственном ближайшем окружении (по вертикали и горизонтали). Выбранный индивид затем совмещается с индивидом, постоянно находящимся в ячейке. В результате формируется один наследник, который может или не может заменить индивида в ячейке в зависимости от выбранной схемы замещения. Таким образом, модель является полностью распределенной и не нуждается в централизованном управлении (рис. П2.10).

**ПГА на базе индивидов.**
Шаг 0. Определение генетического представления задачи.
Шаг 1. Создание первоначальной популяции индивидов и формирование структуры популяции.
Шаг 2. Локальное повышение каждым индивидом своей производительности (hill-climbing).
Шаг 3. Выполнение каждым индивидом селекции с целью поиска пары.
Шаг 4. Применение к паре скрещивания, а также других генетических операторов, таких как мутация.
Шаг 5. Локальное повышение наследником своей производительности (hill-climbing).
Замещение наследником родителя в соответствии с заданным критерием качества.
Шаг 6. Возврат к шагу 3.

**Рис. П2.10.** Параллельный генетический алгоритм на базе индивидов

При работе с моделью на базе индивидов необходимо задать:

- структуру связей ячеек;
- схему селекции;
- схему замещения.

Исследования этой модели показали, что для сложных задач она способна обеспечить лучшие решения, чем стандартный ГА.

## П2.3. Классификация генетических алгоритмов

В ходе исследований в области генетических алгоритмов и эволюционных алгоритмов в целом появилось большое количество направлений, и их число непрерывно растет.

Классификация ЭА и основные модификации стандартного ГА, приведенного на рис. П2.5, отражены на рис. П2.11.

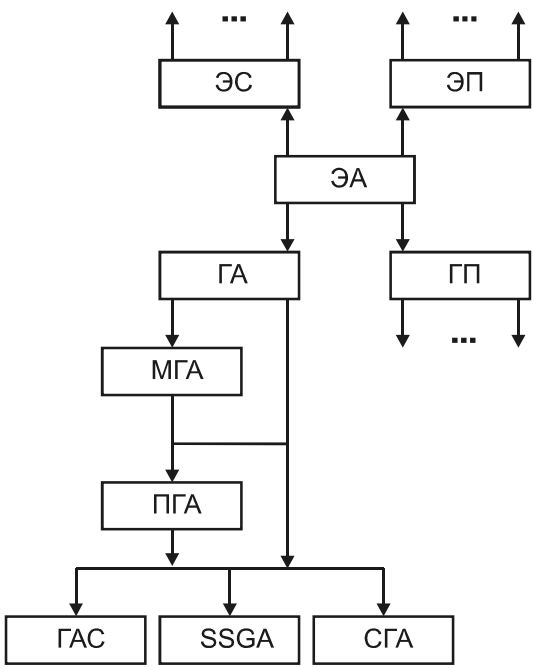



**Рис. П2.11.** Классификация эволюционных алгоритмов

# П2.4. Особенности генетических алгоритмов, предпосылки для адаптации

Исследования в области ГА и ЭА в целом позволяют выделить важную особенность данных методов — эффективность ЭА существенно зависит от таких взаимосвязанных параметров, как вероятность применения генетических операторов, их тип и размер популяции.

ГА являются вероятностным методом направленного поиска и поддерживают сведения об индивидуальных точках в пространстве поиска (ландшафте), известном как популяция. Как отмечалось ранее, поиск может быть рассмотрен как итерационная программа, применяемая с целью создания лучших индивидов с помощью таких операторов, как селекция, скрещивание (рекомбинация) и мутация.

Стратегии мутации и скрещивания различны. Мутация основывается на случае. Результат даже одного шага мутации всегда непредсказуем. Результат скрещивания менее случаен, поскольку при этом скрещиваются только строки, находящиеся в одной популяции. При этом поиск с помощью скрещивания приводит к сходимости популяции и способен локализовать оптимум без применения мутации только при достаточно большом размере популяции.

Эти операторы обычно являются статическими, т. е. их параметры и вероятность использования фиксированы в начале и остаются постоянными до завершения работы алгоритма. Однако имеются доказательства того, что недостаточно один раз установить набор операторов и постоянно использовать его далее, не существует такого набора операторов, который бы являлся оптимальным для всех задач. Существуют и доказательства того, что оптимальный набор операторов для данной задачи будет зависеть от степени сходимости популяции и будет меняться во времени. Основываясь на теоретических и практических подходах, некоторые авторы предложили различные методы адаптивного управления операторами.

ГА, как метод направленного поиска, на каждом шаге генерируют новые точки в пространстве поиска для дальнейшего развития популяции. Каждой точке в пространстве поиска соответствует уникальное значение степени пригодности. Следовательно, можно говорить о пространстве поиска как ландшафте функции степени пригодности. При этом популяция дает оценку эффективности работы алгоритма, которую можно определить как *неоднородную функцию распределения вероятности* (НФРВ).

В отличие от однородного распределения вероятности, характеризующего случайный поиск, НФРВ отражает возможные взаимодействия между членами популяции (или степень вовлечения точек пространства поиска в даль-

нейший процесс поиска посредством рекомбинации двух или более членов популяции).

Генетический поиск может рассматриваться как программа, состоящая из пошагового выполнения двух процессов: формирования промежуточной популяции (модификации популяции) с помощью селекции и генерации нового набора точек с помощью генетических операторов рекомбинации и мутации.

Таким образом, эффективность алгоритма зависит от двух факторов: от поддержания эффективной популяции и от соответствия НФРВ ландшафту степени пригодности и эффективной популяции. Первый из этих факторов зависит от размера популяции и алгоритма селекции. Второй будет зависеть от действия операторов и связанных с ними параметров в данной популяции.

Большое внимание было уделено тому, чтобы найти подходящие варианты операторов и их параметров, которые бы работали в широком диапазоне прикладных задач. В первой работе [DeJong, 1975] определялся тестовый набор из пяти функций для исследования различных характеристик (непрерывные/прерывные, выпуклые/вогнутые, унимодальные/мультимодальные, квадратичные/неквадратичные, с низкой размерностью/с большой размерностью, детерминированные/стохастические), предлагался набор параметров, который бы успешно работал для решения целого ряда прикладных задач. Был определен следующий набор параметров: размер популяции от 50 до 100; вероятность мутации — 0,6; вероятность скрещивания — 0,001. Также была подчеркнута эффективность элитизма и двухточечного скрещивания.

Однако последующие исследования, применяющие "мета-ГА" для определения подходящих значений [Grefenstette, 1986] или использующие полное тестирование [Schaffer, 1989], привели к различным выводам. Grefenstette использовал "мета-ГА" для исследования параметров с помощью описанных ранее пяти функций. Его набор параметров был следующим: размер популяции — 30; вероятность мутации — 0,01; вероятность скрещивания — 0,95.

Schaffer исследовал 6 размеров популяции, 10 уровней скрещивания, 7 уровней мутации и два типа скрещивания, т. е. 840 различных установок. Он отметил, что двухточечное скрещивание не хуже, а иногда лучше одноточечного скрещивания. В небольшой популяции производительность сильно зависит от уровня мутации и меньше — от уровня скрещивания. Когда размер популяции увеличивается, чувствительность к мутации снижается. Им был установлен следующий набор параметров: размер популяции от 20 до 30; вероятность мутации — от 0,005 до 0,01; вероятность скрещивания — от 0,75 до 0,95.

Тем временем теоретический анализ оптимальных размеров популяции [Goldberg, 1985] формализовал утверждение, что размер популяции зависит от размера пространства поиска.

Интенсивность использования мутации и скрещивания может и должна быть привязана к размеру популяции. Эмпирически исследователи [Schaffer, 1989] нашли почти оптимальное сочетание параметров:

$$\ln N + 0{,}93 \ln m + 0{,}45 \ln n = 0{,}56,$$

где $N$ — размер популяции, $m$ — уровень мутации, $n$ — длина стратегии.

Это выражение может быть аппроксимировано:

$$N \times m \times n = 1{,}7.$$

Однако и данное выражение не является окончательным, поскольку рассматривается вне конкретной задачи.

Мутация становится более эффективной, чем скрещивание, когда размер популяции небольшой, и наоборот [Spears и Anand, 1991]. Другие факторы, такие как схема представления, селекция, функция степени годности, также отражаются на эффективности использования мутации и скрещивании. При этом теоретически невозможно определить, какие именно операторы использовать для решения конкретной задачи.

Последующие несколько лет исследований привели к появлению ряда новых операторов, среди которых можно выделить *однородное скрещивание*, которое выполняется на $L/2$ точках скрещивания строки длиной $L$ [Syswerda, 1989] и является более эффективным [Spears William M., Adapting Crossover in Evolutionary Algorithms].

Результаты исследований позволили сформировать два важных вывода:

- значения НФРВ являются результатом выполнения генетических операторов (в частности, скрещивания) и зависят от их типа [Eshelman, 1989]. Практические результаты были подтверждены анализом различных механизмов рекомбинации [DeJong и Spears, 1990, 1992; Spears и DeJong, 1991];
- вероятность того, что новая точка, сгенерированная с помощью генетических операторов, будет более эффективна, чем предыдущая родительская, также зависит от типа генетического оператора. Значение вероятности меняется также от поколения к поколению [Schaffer и Eshelman, 1991]. Кроме того, вероятность отражает соответствие между НФРВ и ландшафтом степени пригодности популяции.

Эти исследования вместе с исследованиями эволюционных стратегий, в которых уже использовались адаптивные операторы (различные версии этих алгоритмов с адаптивной оценкой мутации оказались достаточно эффективными для решения задач оптимизации функций [Back, 1991; Schwefel, 1981], последние исследования подтвердили эту точку зрения на эффективность му-

тации [Baeck & Schwefel, 1993]), повысили интерес к возможностям алгоритмов, которые способны адаптировать операторы или параметры. Цель введения механизма адаптации состоит также и в том, чтобы согласовать значения НФРВ, формируемые алгоритмом, и ландшафт степени пригодности.

# П2.5. Классификация адаптивных ГА

Адаптивные ГА можно классифицировать по трем направлениям (рис. П2.12).

- основа управления адаптацией (внешнее управление или самоадаптация);
- область адаптации (применяется ли адаптация ко всей популяции, только к отдельным членам популяции и т. д.);
- основа адаптации (операторы, параметры и т. д.).

## П2.5.1. Основа адаптации

Большинство адаптивных алгоритмов основывается на СГА и SSGA, при этом чаще всего исследования проводятся на генетических операторах скрещивания и мутации. Направленный поиск в ГА в основном сосредоточен на исследовании новых областей пространства поиска и эксплуатации предварительно изученных эффективных областей (или гиперпланов) пространства поиска.

В алгоритме исследования высокая вероятность назначается неисследованным областям, в то время как в эксплуатационном алгоритме НФРВ представляет собой накопленную информацию об областях с большей степенью пригодности и гиперпланах пространства поиска. При этом НФРВ алгоритма исследования способна измениться быстрее, чем в эксплуатационном алгоритме.

Однако диапазон, в котором наследники отличаются от их родителей, управляется не только типом оператора, но и вероятностью их использования. Таким образом, для данной популяции и данного оператора можно настраивать форму НФРВ между предельными значениями исследования и эксплуатации, изменяя вероятность использования операторов.

Этот вывод привел в дальнейшем к сосредоточению на адаптации операторов рекомбинации и мутации, поскольку, изменяя степень разрушения, вызванного операторами, можно адаптировать популяцию, которая заменяется на каждом шаге при использовании простых и эффективных механизмов селекции.

Можно выявить ряд направлений в исследованиях того, как именно следует осуществлять адаптацию в ГА:

- самый простой класс — ГА, которые используют фиксированный набор операторов и адаптируют вероятности использования операторов. Хорошо известны работы, устанавливающие зависимости:
  - вероятности мутации от времени [Fogarty, 1989];
  - вероятностей использования нескольких простых операторов (однородное скрещивание, мутация и т. д.) от их эффективности на последних поколениях [Davis, 1989];

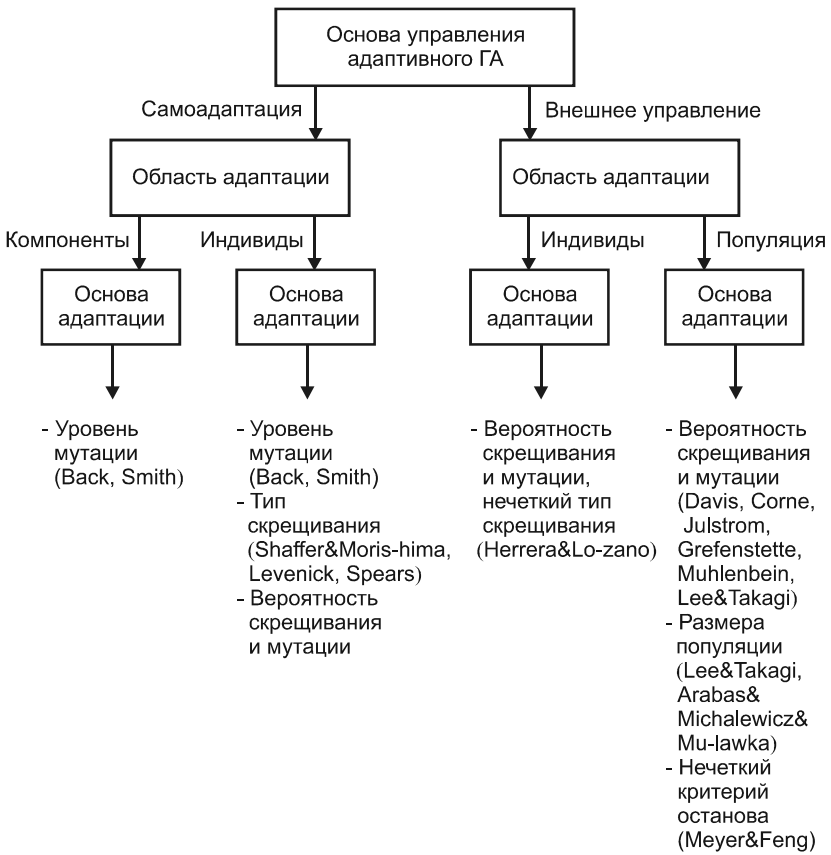



**Рис. П2.12.** Классификация генетических алгоритмов

- дальнейшее развитие этого подхода заключается в алгоритмах, которые поддерживают несколько популяций или подпопуляций, использующих различные наборы операторов и параметров. Общий подход заключается в "мета-ГА" (или многоуровневом ГА, в котором одновременно может

существовать несколько популяций, расположенных по некоторой иерархии) и определении параметров для каждой подпопуляции [Grefenstette, 1986];

- адаптации в ГА могут подвергаться не только вероятности применения операторов, типы операторов, но и такие параметры, как размер популяции и критерий останова. За счет внешнего управления при этом возможно обеспечение большей гибкости и учет большего количества зависимостей, в том числе и неявных.

## П2.5.2. Область адаптации

Адаптация в ГА может осуществляться на уровнях:

- популяции, когда параметры ГА являются общими для всей популяции, и следовательно, изменения в НФРВ происходят со стороны всей текущей популяции;
- индивидов, когда значения НФРВ формируются со стороны каждого члена популяции отдельно, но одним способом;
- компонентов, где на НФРВ можно влиять со стороны каждого компонента каждого члена популяции различным способом.

### Адаптация на уровне популяции

Адаптация на уровне популяции предполагает использование фиксированного набора генетических операторов, таких, что их параметры могут изменяться во времени. Наиболее важным параметром при этом является вероятность применения генетического оператора. К этому уровню адаптации относятся уже упомянутые "мета-ГА", а также конкурирующие подпопуляции ГАС, например [Schlierkamp-Voosen и Muhlenbein, 1994].

В [Fogarty, 1989] внешне определенная форма используется, чтобы уменьшить уровень мутации с течением времени. Однако этот подход не является универсальным и не применяется к другим генетическим операторам. Хорошо известен и популярен подход [Davis, 1989; Corne, 1994; Julstrom, 1995], который заключается в том, чтобы хранить статистику по эффективности наследников, сгенерированных различными операторами, по отношению к их родителям. Эффективные операторы при этом "вознаграждаются" — увеличивается вероятность их использования. Этот подход требует дополнительного пространства памяти.

В [Lee и Takagi, 1993] для управления различными параметрами, в том числе и размером популяции, используются нечеткие правила, основанные на относительной эффективности самых лучших, худших и средних значений текущей популяции.

## Адаптация на уровне индивидов

Альтернативный подход к адаптации базируется на рассмотрении отдельных членов популяции. Глобальная адаптация может изменить вероятность мутации для целой популяции, в то время как алгоритм с адаптацией на уровне индивидов использует параметры ГА, такие как тип или вероятность применения генетических операторов, как переменные величины по отношению к индивидам, а не ко всей популяции.

К этому уровню адаптации относится механизм скрещивания, в котором добавочные биты используются для кодирования точек скрещивания [Schaffer и Morishima, 1987]. Согласно этому механизму, в $L$-битовые индивиды добавляются $L$ дополнительных бит. Эти добавочные биты используются для определения точек скрещивания ("1" обозначает, что в этом месте осуществляется скрещивание). Добавочные биты представляют собой маску для определения точек скрещивания двух родителей, которая эволюционирует вместе с решениями. Здесь результаты непосредственно зависят от числа точек скрещивания и длины строки индивидов. В [Levenick, 1995] исследуется аналогичный механизм, но с дополнительным кодированием бит для изменения вероятности скрещивания.

Альтернативный метод управления скрещиванием (однобитовая адаптация) используется в [Spears, 1995]. Здесь один бит служит для управления выбором однородного или двухточечного скрещивания.

Более популярной является идея самоадаптивного уровня мутации, заимствованная из эволюционных стратегий. Она заключается в добавлении бит для кодирования уровня мутации [Back, 1992].

В [Srinivas и Patnaik, 1994] предлагается вариант адаптации вероятностей операций мутации или скрещивания. При этом вероятности применения генетических операторов к индивиду зависят от максимальной относительной степени пригодности и средней степени пригодности популяции через коэффициенты:

$$P_c = K_1 (f_{\max} - f') / (f_{\max} - f_{\text{ср}}), \text{ если } f' \le f_{\text{ср}},$$
$$P_c = K_3, \text{ если } f' > f_{\text{ср}},$$
$$P_m = K_2 (f_{\max} - f) / (f_{\max} - f_{\text{ср}}), \text{ если } f \le f_{\text{ср}},$$
$$P_m = K_4, \text{ если } f' > f_{\text{ср}},$$

где $K_1$, $K_2$, $K_3$, $K_4 \le 1$, а $f'$ — большая степень пригодности из двух родителей.

В [Herrea&Lozano, 1998] предлагается вариант адаптации вероятности скрещивания и мутации на основе нечетких баз знаний, при этом нечеткие базы знаний представляют собой популяцию верхнего уровня (мета-ГА).

### Адаптация на уровне компонентов

Принципиальное преимущество этого направления — обеспечение лучшей и более точной настройки НФРВ, связанной с каждым индивидом.

Этот подход был испытан в самоадаптивном механизме мутации [Back, 1992] в сравнении с уровнем адаптации на уровне индивидов. В некоторых обстоятельствах результаты были лучше, но на других ландшафтах дополнительные расходы на обучение, связанные со всеми параметрами мутации, замедлили поиск. Такой механизм мог бы быть очень эффективен при правильно определенных компонентах, т. е. верно выбранном уровне детализации.

При адаптации на уровне компонентов за основу может быть взят подход [Schaffer и Morishima] в части добавления битов дополнительного пространства к представлению индивидов для того, чтобы определить, могут ли два смежных гена быть разбиты скрещиванием. Отличие заключается в том, что при формировании нового индивида блоки генов могут быть выбраны из всей популяции (а не только из двух родителей). Этот процесс эволюции может быть рассмотрен как адаптация стратегии рекомбинации на уровне компонентов, т. к. индивиды здесь рассматриваются только с позиции общего генофонда, из которого собираются новые индивиды. Это используется в алгоритме [Smith и Fogarty, 1996a], который предполагает самоадаптацию уровня мутации, применяя некоторый уровень мутации для каждого блока или компонента.

## П2.5.3. Основа управления адаптацией

С точки зрения основы управления адаптацией ГА можно разделить на ГА с внешним механизмом адаптации и ГА с самоадаптацией, при которой в качестве механизма адаптации используется селекция.

В самоадаптивных алгоритмах основой для определения дальнейшей траектории развития является относительная степень пригодности индивида и значения НФРВ. В адаптивных алгоритмах с внешним управлением эта основа приобретает форму статистики эффективности алгоритма. При этом механизм, используемый для генерирования новых стратегий, основанных на этих признаках, внешне обеспечивается в форме обучающего алгоритма или набора фиксированных правил. Такой механизм позволяет учитывать совокупность зависимостей как параметров генетических алгоритмов, так и внутреннего механизма генных зависимостей, ландшафта степени пригодности и обеспечивает более гибкую адаптацию к текущему процессу эволюции, соответствующему конкретному запуску ГА.

В качестве иллюстрации внешнего механизма можно использовать уже упомянутый пример адаптации вероятностей генетических операторов и размера популяции с помощью нечетких правил [Lee и Takagi, 1993].

Известен и другой вариант генетического алгоритма с адаптацией размера популяции с помощью нечеткой логики (GAVaPS) [Arabas, Michalewicz, Mulawka, 1994]. Этот алгоритм использует понятие возраста индивида, который является эквивалентным числу поколений. При этом возраст индивидов заменяет понятие селекции. Исключение индивида происходит, когда возраст превышает значение срока службы. При вычислении срока службы может также учитываться текущее состояние ГА. Это состояние описывается средним, максимальным и минимальным значениями степени пригодности в текущей популяции. При таком подходе более высокие значения срока службы назначаются индивидам, имеющим значения степени пригодности выше среднего. Однако выбор оптимальной стратегии вычисления срока службы является открытой проблемой и требует дальнейших исследований.

Адаптация с помощью нечетких правил может быть распространена также:

- непосредственно на механизм скрещивания (нечеткое скрещивание) [Herrera & Lozano, 1995b];
- на нечеткий критерий останова [Meyer & Feng, 1994].

В последнем случае адаптивный механизм предполагает оценку качества текущего решения по отношению к оптимальному решению на основании имеющейся статистики и принятие решения на базе нечеткого вывода о завершении или продолжении работы ГА.

Таким образом, адаптация известна и широко используется исследователями ГА как на уровне популяции, так и на уровне индивидов и компонентов.

Адаптация на уровне индивидов является разумным сочетанием возможности соэволюции популяции параметров управления, с одной стороны, и средней степенью детализации популяции, с другой, что отражается на быстродействии алгоритма.

Большинство адаптивных ГА с внешним управлением на основе нечеткой логики используют адаптацию уровня популяции. Адаптивные механизмы, основанные на нечеткой логике на уровне индивидов, могут быть интересны для корректировки параметров управления генетическими операторами [Herrera, 1998].

## П2.6. Двунаправленная интеграция ГА и нечетких алгоритмов продукционного типа

Двунаправленная интеграция ГА и нечетких алгоритмов продукционного типа может быть создана таким образом, что принесет пользу обоим методам. Хорошо известно, что использование нечеткой логики позволяет принимать решения в условиях неопределенности. Нечеткая логика обеспечивает базис для представления различных форм знаний в неопределенных средах, позво-

ляет моделировать взаимодействие переменных информационной системы. ГА дают возможность обучения, глобального и локального поиска.

Использование нечеткой логики для улучшения работы ГА позволяет адаптировать различные параметры ГА к текущим условиям пространства поиска (рис. П2.13).

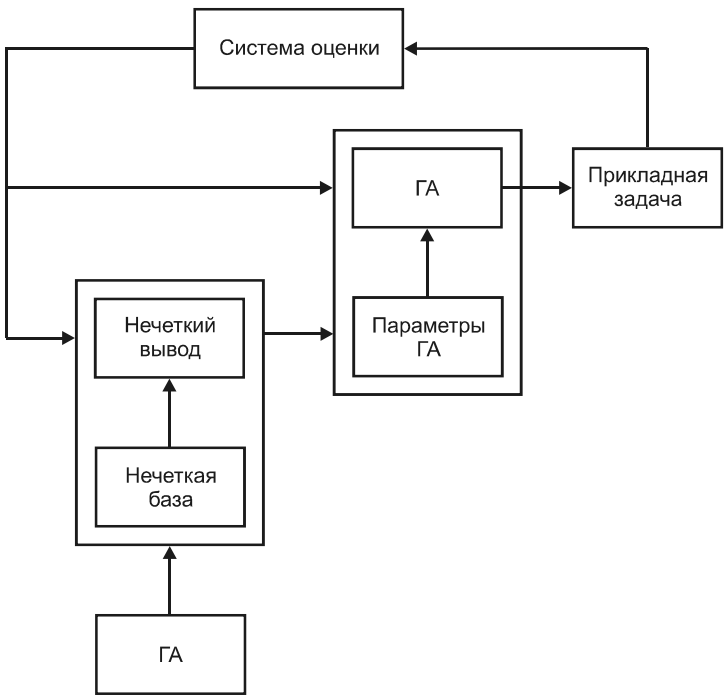



**Рис. П2.13.** Схема организации ГА с функцией адаптации (обратной связи)

Использование ГА для оптимизации задач, моделируемых с помощью нечеткой логики, предполагает настройку нечетких баз знаний, включая нечеткие лингвистические термы и правила (рис. П2.14).

Первая попытка спроектировать нечеткую систему с помощью ГА была сделана в работе [Karr], который использовал ГА для настройки последовательности нечетких правил при решении задачи стабилизации инверсного маятника. Естественно, что такое решение не в полной мере использовало возможности ГА по настройке самих функций принадлежностей лингвистических термов. Следующая система, созданная [Lee and Takagi], дала большую степень свободы ГА для проектирования нечетких баз правил. Вопросы формирования нечетких баз знаний и являются основой двунаправленной интеграции ГА и НЛ.

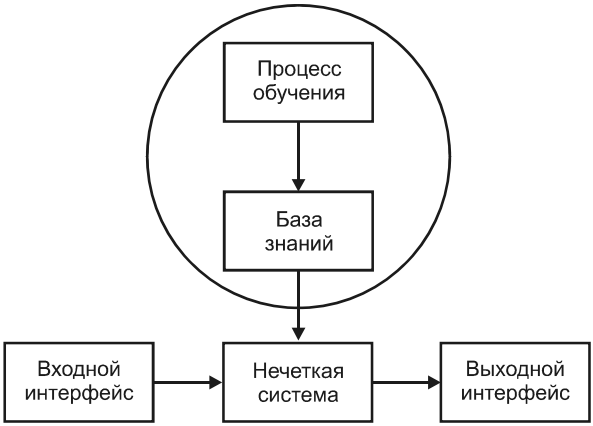



**Рис. П2.14.** Использование ГА для оптимизации задач

Для разработки любой системы, основанной на НЛ, в том числе и адаптивного ГА, необходимо определить следующее.

1. Входы и выходы системы — входными переменными (входами) могут быть различные показатели эффективности популяции и различные отношения этих показателей: меры разброса значений в популяции, максимальная, средняя и минимальная степени пригодности и т. д. В работе [Herrera, Lozano, Verdegay, 1993] в качестве входных переменных используются два параметра:

   - генотипический разброс, который определяется как

   $$ED = (d_{\text{cp}} - d_{\min}) / (d_{\max} - d_{\min}),$$

   где $d_{\text{cp}}$, $d_{\min}$, $d_{\max}$ — среднее, минимальное и максимальные значения отклонений эффективности индивидов C$i$ от лучшего значения в популяции $P$ размером $N$:

   $$d_{\text{cp}} = (1/N) \sum_{i=1}^{N} d\,(C_{\text{best}}, Ci),$$

   $$d_{\max} = \max\{d(C_{\text{best}}, Ci)|, Ci \in P\},$$

   $$d_{\min} = \min\{d(C_{\text{best}}, Ci)|, Ci \in P\}.$$

   Диапазон значений $ED$ — [0, 1]. Если значение $ED$ небольшое, то большинство индивидов в популяции располагается вокруг лучшего значения, таким образом достигается сходимость популяции.

   - фенотипический разброс, который измеряет отношение между самой лучшей и средней степенью пригодности — $PDM1 = f_{\text{best}} / f_{\text{cp}}$.

Текущие параметры управления могут также рассматриваться как входы. Например, нечеткая схема управления может включать следующие отношения для управления размером популяции:

- ЕСЛИ (средняя пригодность) / (самая лучшая пригодность) БОЛЬШАЯ, ТО размер популяции должен увеличиться;
- ЕСЛИ (самая плохая пригодность) / (средняя пригодность) МАЛЕНЬКАЯ, ТО размер популяции должен уменьшиться;
- ЕСЛИ мутация МАЛЕНЬКАЯ И популяция МАЛЕНЬКАЯ, ТО размер популяции должен увеличиться.

В [Xu, Vukovich, Ichikawa, Ishii, 1993, 1994] входными параметрами являются текущее число поколений и размер популяции.

Выходы указывают значения параметров управления или изменений в этих параметрах, так в [Xu, Vukovich, Ichikawa, Ishii, 1993, 1994] выходными параметрами являются вероятности скрещивания и мутации. При этом обычно количество нечетких баз знаний соответствует числу выходных переменных. Например, в [Herrera, Lozano, Verdegay, 1993] две базы знаний используются для управления степенями изменений $\delta p_e$ и $\delta\eta_{\min}$, вероятностью скрещивания $p_e$ и интенсивностью селекции $\eta_{\min}$.

2. Лингвистические термы — каждый вход и выход должен иметь связанный набор лингвистических термов. Значение этих меток определено через функции принадлежности нечетких множеств. В [Herrera, Lozano, Verdegay, 1993] лингвистические термы и функции принадлежности заданы так, как показано на рис. П2.15.

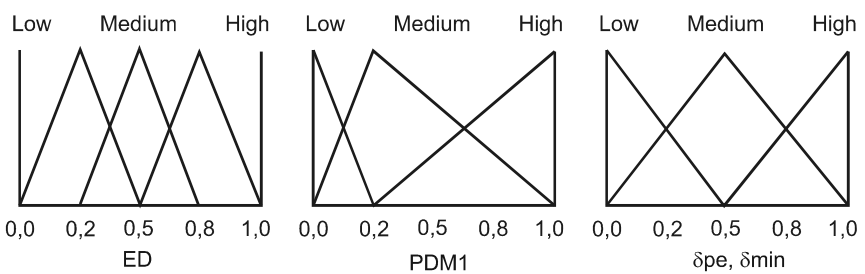



**Рис. П2.15.** Лингвистические термы

Имеется несколько различных методов задания формы функций принадлежности. В простейшем случае смежные функции принадлежности накладываются друг на друга таким образом, что центр предыдущей функции принадлежности совпадает с крайней отметкой следующих. Исполь-

зуя такую кодировку, только $n$–1 центров функции принадлежности должны быть определены (рис. П2.16). Может быть использовано и более гибкое задание функций принадлежностей — через все определяющие точки функций принадлежностей [Cordon, Herrera, Lozano, 1992].

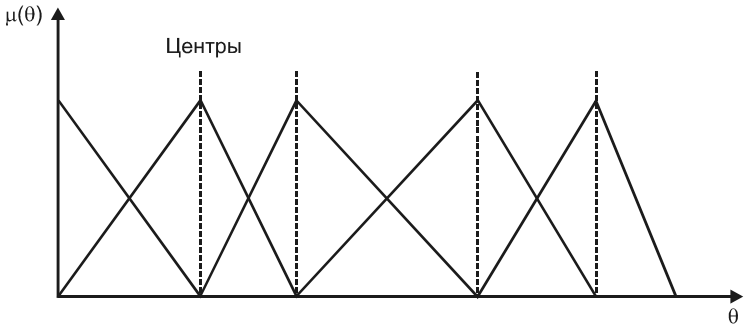



**Рис. П2.16.** Центры функции принадлежности

3. Нечеткие правила — после выбора входов и выходов и определения лингвистических термов и функций принадлежности должны быть определены нечеткие правила, описывающие отношения между ними, например, так, как это показано в табл. П2.1 [Herrera, Lozano, Verdegay, 1993] или табл. П2.2 [Xu, Vukovich, Ichikawa, Ishii, 1993, 1994].

**Таблица П2.1**

| $\mathbf{P_e}$ | **PDM1** | | |
|---|---|---|---|
| ED | Low | Medium | High |
| Low | Small | Small | Medium |
| Medium | Big | Big | Medium |
| High | Big | Big | Medim |
| $\boldsymbol{\eta_{min}}$ | **PDM1** | | |
| ED | Small | Medium | Large |
| Low | Small | Medium | Big |
| Medium | Small | Big | Big |
| High | Small | Small | Big |

Задать нечеткую базу знаний, включающую задание базы правил, а также лингвистических термов и их функций принадлежности, можно следующими

способами: с использованием опыта и знаний экспертов или с помощью автоматической методики обучения для тех случаев, где знание или экспертиза недоступны.

**Таблица П2.2**

| **Вероятность скрещивания** | **Размер популяции** | | |
|---|---|---|---|
| Поколение | Small | Medium | Big |
| Short | Medium | Small | Small |
| Medium | Large | Large | Small |
| Long | Very Large | Very Large | Large |
| **Вероятность мутации** | **Размер популяции** | | |
| Поколение | Small | Medium | Big |
| Short | Large | Medium | Small |
| Medium | Medium | Small | Very Small |
| Long | Small | Very Small | Very Small |

Используя автоматическую методику, отношения и функции принадлежности могут быть автоматически определены из сложного взаимодействия между параметрами управления ГА и эффективностью ГА. В этом случае ГА применяется для настройки нечеткой базы знаний. Настройка может быть произведена в следующих режимах:

- в режиме offline [Lee & Takagi], когда каждая строка индивида представляет собой закодированное задание функций принадлежностей и набора правил (базы правил);
- в режиме online, когда настройка базы знаний осуществляется в процессе решения задачи [Herrea & Lozano, 1998].

Этим определяется существенное преимущество адаптации ГА на уровне индивидов в отличие от адаптации на уровне популяции — адаптация на уровне индивидов позволяет производить online-настройку нечетких баз знаний, которые, в свою очередь, используются для адаптации параметров ГА. Это обусловлено возможностью на каждом шаге работы ГА получить оценку эффективности набора из $N/2$ нечетких баз знаний, где $N$ — размер популяции.

Выполнив анализ имеющейся информации, можно сформулировать те узловые направления развития адаптивных ГА на основе нечеткой логики, которые еще недостаточно изучены и требуют рассмотрения:

- определение нечеткой базы данных, описывающей принципиальные особенности ГА для управления процессом сходимости популяции;
- определение нечеткой базы данных, позволяющей достигнуть равновесия исследования и эксплуатации;
- определение нечетких баз знаний, учитывающих действие каждого генетического оператора в соответствии с поведением остальных;
- повышение эффективности нечетких критериев останова;
- определение нечетких генетических операторов: селекции, скрещивания, мутации.

Комплексно решить эти проблемы до последнего времени не удавалось. Однако, в конечном счете, все перечисленные направления сводятся к уровню нечеткого управления генетическими операторами.

Использование в ГА нечеткого адаптивного оператора скрещивания позволит предотвращать преждевременную сходимость популяции, поддерживая необходимый и зависящий от текущих условий уровень разнообразия популяции, учитывать влияние других генетических операторов.

# ПРИЛОЖЕНИЕ 3

# Описание прилагаемого компакт-диска

Компакт-диск содержит документацию по основным стандартам Data Mining, перечисленным далее.

- CWM — спецификация стандарта организации хранилищ данных CWM. Подробную информацию по стандарту можно найти по адресу: **http://www.omg.org/cwm/**.
- PMML — спецификация стандарта записи моделей Data Mining в виде XML-файла — PMML версия 2.0. Подробную информацию по стандарту можно найти по адресу: **http://www.dmg.org/pmml-v3-0.html**.
- CRISP — спецификация методологии разработки систем Data Mining. Подробную информацию по стандарту можно найти по адресу: **http://www.crisp-dm.org/index.htm**.

Драйверы и библиотеки, необходимые для работы GUI-интерфейса библиотеки Xelopes, прилагаются согласно следующему списку:

- JFreeChart — свободно распространяемая графическая Java-библиотека JFreeChart версии 0.9.8, использующая для отображения графических объектов 2D API. Подробную информацию по библиотеке можно найти по адресу: **http://www.jfree.org/jfreechart/index.html**;
- JGraph-2.1-java1.4 — библиотека JGraph версии 2.1 для Java версии 1.4 предназначена для отображения и редактирования графики. Подробную информацию можно найти по адресу: **http://sourceforge.net/projects/jgraph/**;
- DirectX — драйверы DirectX версии 8.0. Подробную информацию по библиотеке можно найти по адресу: **http://www.microsoft.com/directx/default.asp**;
- java3d-1_3_1-beta-windows-i586-directx-sdk.exe — Java-адаптер 3D-графики к драйверам DirectX для операционных систем семейства Win32. Под-

робную информацию по библиотеке можно найти по адресу: **http://java.sun.com/jdk/**.

На прилагаемом диске находится лабораторный практикум, который может быть использован как для самостоятельного изучения алгоритмов Data Mining, так и в рамках учебного процесса. Практикум включает пять лабораторных работ, использующих библиотеку Xelopes и GUI-интерфейс к ней:

- № 1 — знакомство с GUI-интерфейсом библиотеки алгоритмов Data Mining;
- № 2 — выполнение анализа данных методами Data Mining;
- № 3 — создание программ анализа данных с использованием алгоритмов Data Mining;
- № 4 — реализация алгоритмов построения моделей unsupervised;
- №5 — реализация алгоритмов построения моделей supervised.

Библиотека Xelopes и документация также находятся на прилагаемом диске. Xelopes — свободно распространяемая библиотека алгоритмов Data Mining от компании Prudsys. Подробную информацию можно найти по адресу: **http://www.zsoft.ru/rus/index.php?kat=xelopes_intro**. Далее указан состав прилагаемой библиотеки:

- XELOPES_Java.zip — библиотека Xelopes;
- Xelopes.exe — графический интерфейс к библиотеке Xelopes, его установка происходит автоматически.

Полное описание структуры диска приведено в табл. П3.1.

**Таблица П3.1**

| Папка | Содержание |
|---|---|
| \Drivers | Драйверы и библиотеки, необходимые для работы GUI-интерфейса библиотеки Xelopes |
| \Java 1.6.X | Дистрибутив Java версии 1.6 |
| \Labs | Лабораторный практикум по алгоритмам Data Mining |
| \Standard | Стандарты Data Mining |
| \Xelopes | Свободно распространяемая библиотека алгоритмов Data Mining от компании Prudsys |
| \Eclipse | Свободно распространяемая среда разработки Eclipse |

Для инсталляции библиотеки Xelopes и GUI-интерфейса к ней необходимо выполнить следующую последовательность действий:

1. Запустить файл Xelopes.exe из папки Xelopes.
2. Задать папку назначения (по умолчанию диск C:\, для стабильной работы лучше этот параметр не менять). По окончании автоматически запустится установка JDK 1.6, в случае, если на вашей машине он не установлен, необходимо продолжить установку, если JDK установлен, то следует отказаться от переустановки.

Программа установки дальше сама проведет инсталляцию.

После установки на диске C:\ появятся следующие папки: eclipse3.3 и Workspace, а также файл jdk-6u7-windows-i586-p.exe который следует удалить.

Для запуска и настройки GUI-интерфейса необходимо выполнить следующие шаги:

1. Открыть каталог eclipse3.3 и запустить файл eclipse.exe.
2. Выбрать рабочую область (по умолчанию c:\workspace) и нажать **OK**.
3. Если все сделано правильно, в Package Explorer-е появится проект XelopesGuiLast.
4. В Eclipse на панели задач выполнить **Run | Run History | Xelopes**, после чего запустится приложение.

# Список литературы


1. Балашов Е. П., Куприянов М. С., Барсегян А. А. Лингвистические модели в биотехнических системах. — В кн.: Модели выбора альтернатив в нечеткой среде. — Рига: РПИ, 1980, с. 109—111.
2. Барсегян А. А. Нежесткое ситуационное управление микроклиматом в теплицах. — В кн.: Разработка основных подсистем автоматизированной системы управления микроклиматом в зимних теплицах с использованием ЭВМ. Отчет по НИР 536. Гос. рег. № У92618, гл. 4, Л., 1982, с. 30—35.
3. Барсегян А. А. Реализация процессов классификации и вывода в лингвистических процессорах. — В кн.: Управление при наличии расплывчатых категорий. — Пермь: ППИ, 1982, с. 64—66.
4. Барсегян А. А. Устройство обработки лингвистических таблиц решений. — В кн.: Совершенствование устройств и методов приема и передачи информации. — Ростов-Ярославский: ЯПИ, 1982, с. 69.
5. Барсегян А. А., Бялый В. С., Виноградов В. Б., Куприянов М. С. Подход к организации терминальных процессоров для человеко-машинного управления. — В кн.: Диалог "Человек-ЭВМ". — Л.: ЛИАП, 1982, с. 83—86.
6. Барсегян А. А., Семенов В. Н. Многоуровневая система управления на основе БИС многофункциональной памяти. — В кн.: Синтез и проектирование многоуровневых систем управления. Часть 2. — Барнаул: АГУ, 1982, с. 133—135.
7. Кузьмин В. Б. Построение нечетких групповых отношений. — М: Наука, 1988.
8. Куприянов М. С., Неддермайер У., Барсегян А. А. Использование таблиц решений для проектирования быстродействующих микропроцессорных систем. — В кн.: Микропроцессорные системы. — Л.: ЛДНТП, 1981, с. 78—86.

9. Куприянов М. С., Ярыгин О. Н. Построение отношения и меры сходства нечетких объектов. — Техническая кибернетика, № 3, 1988.
10. Носов В. А. Комбинаторика и теория графов. Учебное пособие. — Московский государственный институт электроники и математики (Технический университет), Москва, 1999.
11. Теоретические основы системы STARC Ver. 3.3, DATA-CENTER. — Екатеринбург, 1992.
12. Alex Berson, Stephen J. Smith. Data Warehousing, Data Mining & OLAP. McGraw-Hill, 1997.
13. Colin Shearer. The CRISP-DM Model: The New Blueprint for Data Mining. JOURNAL OF DATA WAREHOUSING Volume 5 Number 4 Fall 2000.
14. Common Warehouse Metamodel (CWM) Specification. OMG. Version 1.0, 2 February 2001.
15. CRISP-DM 1.0. Step-by-step data mining guide. SPSS, 2000.
16. Wolpert D. H., Macready W. G.. No Free Lunch Theorems for Optimization //IEEE Transactions on Evolutionary Computation, 1997. — vol. 1, № 1.
17. Erik Thomsen. OLAP Solutions. Jhon Wiley & Sons, Inc. 2002.
18. Ian W., Elbe F. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Department of computer science University of Waikato.
19. Jim Melton, Andrew Eisenberg. SQL Multimedia and Application Packages (SQL/MM).
20. Mark F. Hornick JSRs: Java Specification Requests Detail JSR 73 Data Mining API **http://web1.jcp.org/en/jsr/detail?id=73&showPrint**.
21. Michael J. A. Berry, Gordon Linoff. Data Mining Techniques. Jhon Wiley & Sons, Inc. 1997.
22. Pei-Hsin Wu, Wen-Chih Peng and Ming-Syan Chen. Mining Sequential Alarm Patterns in a Telecommunication Database. Department of Electrical Engineering, National Taiwan University Taipei, Taiwan, ROC.
23. Ralph Kimball. The Data Warehouse Toolkit. Second Edition. Jhon Wiley & Sons, Inc. 2002.
24. Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning. Springer. 2001.
25. Inmon W. H. Building the Data Warehouse. Third Edition. Jhon Wiley & Sons, Inc. 2002.
26. XELOPES Library Documentation. Version 1.1. prudsys AG. Germany. Chemnitz May 26, 2003.

27. Artificial Intelligence — A Guide to Intelligent Systems, Michael Negnivitsky, Addison-Wesley, Pearson Education Limited, 2002.
28. A Modified Fuzzy C-Means Algorithm for Bias Field Estimation and Segmentation of MRI Data, Mohamed N. Ahmed, Member, IEEE, Sameh M. Yamany, Member, IEEE, Nevin Mohamed, Aly A. Farag, Senior Member, IEEE, Thomas Moriarty, IEEE Transactions on medical imaging, Vol. 21, No. 3, March 2002.
29. Fuzzy C-Means Clustering using spatial information with application to remote sensing, P. Dulyakarn, Y. Rangsanseri, Department of Telecommunications Engeneering, Faculty of Engeneering, King Mongkut's Institute of Technology Ladkranbang, Bangkok 10520, Thailand, 22nd Asian Conference on Remote Sensing, 5—9 Singapore, 2001.
30. Learning Fuzzy Classification Rules from Labeled Data, Johannes A. Roubos, Magne Setnes, Janos Abonyic, 2001.
31. Neuro-Fuzzy Classification Initialized by Fuzzy Clustring, D. Nauck, F. Klawonn, Department of Computer Science, Technical University of Braunschweig, Bueltenweg 74-75, D-38106 Braunschweig, Germany.
32. Clustering Methods. Applications of Multivariate Statistical Analysis. Jiangsheng Yu, Institute of Computational Linguistics, Peking University, Beijing, 100871.
33. Fuzzy Clustering. Hard-c-Means, Fuzzy-c-Means, Gustafson-Kessel. Olaf Wolkenhauer, Control Systems Centre.
34. Fuzzy c-Means Clustering with Regularization by K-L Information, H. Ichihashi, K. Miyagishi, K. Honda, Industrial Engineering, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuencho, Sakai, Osaka 599-8531 Japan, 2002.
35. Gaussian Mixture PDF Approximation and Fuzzy c-Means Clustering with Entropy Regularization, H. Ichihashi, K. Honda, N. Tani, College of Engineering Osaka Prefecture University, 2000.
36. Comparation of Fuzzy c-means Algorithm and New Fuzzy Clustreing and Fuzzy Merging Algorithm, L. Zhang, Computer Science Departament, University of Nevada, Reno, NV 89557, 2001.
37. Fuzzy Cluster Analysis with Cluster Repulsion, H. Timm, C. Borgelt, C. Doring, R. Kruse, Dept. of Knowledge Processing and Language Engineering, Otto-von-Guericke-University of Magdeburg, Universitatsplatz 2, D-39106 Magdeburg, Germany, iws.cs.uni-magdeburg.de, 2001.
38. Neural Fuzzy Systems, Robert Fuller, Donner Visiting professor, Abo Akademi University, Abo, 1995.

39. Daniel A. Keim. Information Visualization and Visual Data Mining. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, vol. 7, № 1, January-March 2002.

40. Sachinopoulou A. Multidimensional Visualization. Technical research centre of Finland, ESPOO, VTT, 2001.

41. Fayyad U. and Piatetsky-Shapiro G., "From Data Mining to Knowledge Discovery: An Overview", Advances in knowledge Discovery and Data Mining, Fayyad U., Piatetsky-Shapiro G.

42. Haralampos Karanikas, Babis Theodoulidis. Knowledge Discovery in Text and Text Mining Software. Centre for Research in Information Management. Department of Computation, UMIST, P.O. Box 88, Manchester, M60 1QD, UK.

43. Некрестьянов И. Г. Тематико-ориентированные методы информационного поиска. Диссертация. — СПб.: 2000.

44. Salton G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, 1989.

45. Hahn U., Mani I. The challenges of automatic summarization. IEEE Computer, 33(11):29—35, 2000.

46. Salton G. et al. "Automatic Text Structuring and Summarization", Information Processing & Management, Vol. 33, No. 2, 1997, pp. 193—207.

47. Kruengkrai C., Jaruskulchai C. Generic text summarization using local and global properties of sentences //IEEE/WIC International ConferenWeb Intelligence (WI'03). Halifax. Canada, 2003.

48. Grishman R. Information Extraction: Techniques and Challenges.Computer Science DepartmentNew York University New York, NY 10003, U.S.A.

49. Шабанов В. И., Андреев А. М. Метод классификации текстовых документов, основанный на полнотекстовом поиске // Труды первого российского семинара по оценке методов информационного поиска. Под ред. И. С. Некрестьянова. — СПб.: НИИ Химии СПбГУ, 2003.

50. Franklin Stan, Graesser Art. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents: Thes. Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

51. Agent Technology Glossary. Revision 0.2. OMG Document.

52. Agent Technology in OMA. OMG Document, 1999.

53. FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2001.

54. Philippe G. Ciarlet. The Finite Element Method for Elliptic Problems. North-Holland, Amsterdam, 1978.

55. D. Goldberg; D. Nichols, B. M. Oki, D. Terry (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM 35 (12): 61—70, 1992.

56. R.S. Sutton, A.G. Barto. Reinforcement Learning. An Introduction. MIT Press, Cambridge, London, 1998.

57. R. Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. Machine Learning Journal, 2000.

58. R. Munos. Error Bounds for Approximate Value Iteration, Technical Report CMAP 527, Ecole Polytechnique, 2004.

59. W. Bangerth, R. Rannacher. Adaptive Finite Element Methods for Differential Equations. Lectures in Mathematics. ETH Zürich. 2003.

60. S. Chakrabarti, Data mining for hypertext: A tutorial survey. ACM SIGKDD Explorations, 1(2):1—11, 2000.

61. O. Etzioni. The world wide web: Quagmire or gold mine. Communications of the ACM, 39(11):65—69, 1996.

62. Feature selection for unbalanced class distribution and Naive Bayes (1999) by Marko Grobelnik.

63. Indexing by Latent Semantic Analysis Scott Deerwester. Graduate Library School. University of Chicago.

64. T. Honkela, S. Kaski, K. Lagus и T. Kohonen. Websom — self-organizing maps of document collections. In Proc. Of Workshop on Self-Organizing Maps 1997, pp. 310—315, 1997.

65. S. Scott и S. Matwin. Feature engineering for text classification. In Proceedings of the 16th International Conference on Mashine Learning ICML-99, 1999.

66. W. W. Cohen. Learning to classify English text with ILP methods. In Advances in Inductive Logic Programming (Ed. L. De. Raedt), IOS Press, 1995.

67. Connel M., Feng A., Kumaran G., Raghavan H., Shah C., Allan J., Umass at TDT2004. Proc. DARPA Topic Detection and Tracking Workshop, Gaithersburg, December 2004.

68. H. Ahonen, O. Heinonen, M. Klemettinen, and A. Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In Advances in Digital Libraries (ADL'98), Santa Barbara Calfornia, USA April 1998, 1998.

69. D. Billsus and M. Pazzani. A hybrid user model for news story classification. In Proceedings of the Seventh International Conferece on User Modeling (UM'99) Banff, Canada, 1999.

70. S. Dumais, J. Piatt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In Proceedings of the 1998 ACM 7th nterational conference on Information and knowledge managemt, pages 148—155, Washington United States, 1998.

71. E. Frank, G. W. Paynter, I. H. Witten, C Gutwin, and C. G. NeviManning. Domain-specific keyphrase extraction. In Proceedings of 16th Inernational Joint Conference on Artificial Intelligece (IJCAI-99), pages 668—673, 1999.

72. M. Junker, M. Sintek, and M. Rinck. Learning for text categorization and information extraction with ip. In Proceedings of the Workshop on Learning Language in Logic, Bled, Slovenia, 1999.

73. U. Y. Nahm and R. J. Mooney. Ua mutually beneficial integration of data mining and information extraction. In Proceedings of the Seventeenth National Conference Artificial Intelligence (AAAI-00), 2000.

74. Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. IEEE Intelligent Systems, 14(4):32—43, 1999.

75. H. Kargupta, I. Hamzaoglu, and B. Stafford. Distributed data mining using an agent based architecture. In Proceedings of Knowledge Discovery and Data Mining, pages 211—214. AAAI Press, 1999

76. S. M. Weiss, C Apte, F. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. IEEE Intelligent Systems 14(4):63—69, 1999.

77. R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liph-stat, M. Rajman, Y. Schler, and Zamir. Text mining at the term level. In Principles of Data Mining and Knowledge Discovery, Second European Symposium, PKDD'98, volume 1510 of Lecture Notes Computer Science, pages 56—64. Springer, 1998.

78. R. Feldman and I. Dagan. Knowledge discovery in textual databases (kdt). In Proceedings of the First International Conferece on Knowledge Discovery and Data Mining (KDD-95), pages 112—117 Montreal, Canada, 1995.

79. K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In Proceedings the International Joint Conference on Artifical Inteligence IJICAI-99 Workshop on Machine Learning for Information Filtering, pages 61—67, 1999.

80. D. Freitag and A. McCallum. Information extraction with hmms and shrinkage. In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction, 1999.

81. T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99, pages 682—687, 1999.

82. E. Wiener, J. Pedersen, and S. Weigend. A neural network approach to topic spotting. In Proceedings of the 4th Symposium on Document Analysis and Information Retrieval (SDAIR 95), pages 317—332, 1995.

83. S. Soderland. Learning information extraction rules for semistructured and free text. Machine Learning, 34(l-3):233—272, 1996.

84. I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. In Data Compression Conference 1999, pages 198—207, 1999.

85. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI98), pages 509—516, 1998.

86. F. Crimmins, A. Smeaton, T. Dkaki, and J. Mothe. Tetrafusion: Information discovery on the internet. IEEE Intelligent Systems, 14(4):55—62, 1999.

87. J. Furnkranz. Exploiting structural information for text classification on the WWW. In Advances in Intelligent Data Analysis, Third International Symposium, IDA-99, pages 487—498, 1999.

88. I. Muslea, S. Minton, and C. Knoblock. Wrapper induction for semistructured, web-based information sources. In Proceedings of the Conference on Automatic Learning and Discovery CONALD-98, 1998.

89. T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-97, pages 770—777, 1997.

90. J. Shavlik, T. Eliassi-Rad. Intelligent agents for web-based tasks: An advice-taking approach. In Working Notes of the AAAI/ICML-98 Workshop on Learning for Text Categorization, Madison, WI, pages 588—589, 1999.

91. L. Singh, B. Chen, R. Haight, P. Scheuermann, and K. Aoki. A robust system architecture for mining semistructured data. In Proceeding of The Second Int. Conference on Knowledge Discovery and Data Mining 1998, pages 329—333, 1998.

92. R. Goldman and J. Widom. Approximate DataGuides. In Proceedings of the Workshop on Query Processing for Semistructured Data and non-Standard Data For mat, 1999.

93. S. Grumbach and G. Mecca. In search of the lost schema. In Database Theory — ICDT'99, 7th International Conference, pages 314—331, 1999.

94. S. Nestorov, S. Abiteboul, and R. Motwani. Inferring structure in semistructured data. SIGMOD Record 26(4), 1997.

95. H. Toivonen. On knowledge discovery on graph-structured data. In Workshop on Knowledge Discovery from Advanced Databases (KDAD'99), pages 26—31, 1999.

96. H. Liu, K. Wang. Discovering association of structure from semistructured objects. To appear in IEEE Transaction Knowledge and Data Engineering, 1999.

97. O. Zaiane and J. Han. Webml: Querying the worldwide web for resources and knowledge. In Proc. ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98), pages 9—12, 1998.

98. D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. SIGMOD Record, 2(3):59—74, 1998.

99. S. Abiteboul. Querying semistructured data. In F. N. Afrati and P. Kolaitis, editors, Database Theory — ICDT'97, 6 International Conference, Delphi Greece, January 8—10 1997, Proceedings, volume 1186 of Lecture Notes in Computer Science, pages 1—18. Springer, 1997.

100. P. Buneman. Semistructured data. In Proceeding of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12—14,1997, Tucson, Arizona, pages 117—121. ACM Press, 1997.

101. D. E. Appelt and D. Israel. Introduction to information extraction technology. In Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99, Tutorial, 1999.

102. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In Seventh International World Wide Web Conference, Brisbane, Australia, 1998.

103. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In Proc. Of ACM-SIAM Symposium on Discrete Algorithms, 1998, pages 668—677, 1998.

104. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In 7th World-wide web conference(WWW7), 1998.

105. K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 104—111, Aug. 1998.

106. M. E. Frisse, "Searching for information in a hypertext medical handbook", Communications of the ACM, 31(7), pp. 880—886.

107. M. Marchiori, "The quest for correct information on the Web: Hyper search engines", Proc. 6th International World Wide Web Conference, 1997.

108. G. O. Arocena, A. O. Mendelzon, G.A. Mihaila, "Applications of a Web query language", Proc. 6th International World Wide Web Conference, 1997.

109. R. Weiss, B. Velez, M. Sheldon, C. Nemprempre, P. Szilagyi, D. K. Giord, "HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering", Proceedings of the Seventh ACM Conference on Hypertext, 1996.

110. H. Small, B. C. Grith, "The structure of the scientific literatures I. Identifying and graphing specialties", Science Studies 4(1974), pp. 17—40.

111. J. Pitkow, P. Pirolli, "Life, death, and lawfulness on the electronic frontier", Proceedings of ACM SIGCHI Conference on Human Factors in Computing, 1997.

112. H. Hotelling, "Analysis of a complex statistical variable into principal components", J. Educational Psychology, 24(1933), pp. 417—441.

113. I. T. Jollie. Principal Component Analysis. Springer-Verlag, 1986.

114. W. E. Donath, A. J. Homan, "Lower bounds for the partitioning of graphs", IBM Jour-nal of Research and Development, 17(1973).

115. M. Fielder, "Algebraic connectivity of graphs", Czech. Math. J., 23(1973), pp. 298—305.

116. J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. GKDD Exporati 1(2), 2000.

117. J. Borges and M. Levene. Data mining of user navigation patterns. In Proceedings the WEBKDD'99 Workshop on Web Usage Analysis and User Profiling, August 15 1999, San Diego, USA, pages 31—36, 1999.

118. J. Borges and M. Levene. Mining association rules in hypertext databases. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), August 27—31 1998, New York City, New York, USA, 1998.

119. A. Btichner, M. Baumgarten, S. Anand, M. Mulvenna, and J. Hughes. Navigation pattern discovery from internet data. In Proceedings of the WEBKDD 99 Workshop Usage Analysis and User Profiling, August 15 1999, San Diego, CA, USA, 1999.

120. J. Borges and M. Levene. Data mining of user navgation patterns. In Proceedings of the WEBKDD'99 Workshop on Web Usage Analysis and User Profiling, August 15 1999, San Diego, USA, pages 31—36, 1999.

121. P. Langley. User modeling in adaptive interfaces. In Proceedings of the Seventh International Conference on User Modeling, pages 357—370, 1999.

122. M. Spiiopoulou. Data mining for the web. In Principles of Data Mining and Knowledge Discovery, Second European Symposium, PKDD'99, pages 588—589, 1999.

123. Hammer M., Champy J. Reengineering the corporation: a manifesto for business revolution. — New York, NY: HarperBusiness, 1993. — 223.

124. J. E. Cook. Process Discovery and Validation Through Event-Data Analysis. PhD thesis, 1996.

125. J. E. Cook, Z. Du, C. Liu, and A. L. Wolf. Discovering Models of Behavior for Concurrent Workows. Computers in Industry, 53(3):297—319, 2004.

126. J. E. Cook and A. L. Wolf. Automating Process Discovery Through Event-Data Analysis. In ICSE'95: Proceedings of the 17th international conference on Software engineering, pages 73—82, New York, NY, USA, 1995. ACM Press.

127. J. E. Cook and A. L. Wolf. Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, 7(3):215—249, 1998.

128. J. E. Cook and A. L. Wolf. Event-Based Detection of Concurrency. In Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6), pages 35—45, New York, NY, USA, 1998. ACM Press.

129. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workow Logs. In I. Ramos G. Alonso H.-J. Schek, F. Saltor, editor, Advances in Database Technology — EDBT'98: Sixth International Conference on Extending Database Technology, volume 1377 of Lecture Notes in Computer Science, pages 469—483, 1998.

130. M. Golani and S. S. Pinter. Generating a Process Model from a Process Audit Log. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, International Conference on Business Process Management (BPM 2003), volume 2678 of Lecture Notes in Computer Science, pages 136—151, 2003.

131. J. Herbst. A Machine Learning Approach to Workow Management. In R.L. de Mntaras and E. Plaza, editors, Proceedings 11th European Conference on Machine Learning.

132. J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. International Journal of Intelligent Systems in Accounting, Finance and Management, 9:67—92, 2000.

133. G. Greco, A. Guzzo, and L. Pontieri. Mining Hierarchies of Models: From Abstract Views to Concrete Speci_cations. In W.M.P. van der Aalst, B. Bena-

tallah, F. Casati, and F. Curbera, editors, Business Process Management, volume 3649, pages 32—47, 2005.

134. G. Greco, A. Guzzo, L. Pontieri, and D. Sacca. Mining Expressive Process Models by Clustering Workow Traces. In H. Dai, R. Srikant, and C. Zhang, editors, PAKDD, volume 3056 of Lecture Notes in Computer Science, pages 52—62. Springer, 2004.
135. W.M.P. van der Aalst and B.F. van Dongen. Discovering Workow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002), volume 2480 of Lecture Notes in Computer Science, pages 45—63. Springer-Verlag, Berlin, 2002.
136. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1128—1142, 2004.
137. A. K. Alves de Medeiros, B. F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm. In L. Baresi, S. Dustdar, H. Gall, and M. Matera, editors, Ubiquitous Mobile Information and Collaboration Systems (UMICS 2004), volume 3272 of Lecture Notes in Computer Science, pages 154—168. Springer-Verlag, Berlin, 2004.
138. A. K. Alves de Medeiros, B. F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process Mining: Extending the α-algorithm to Mine Short Loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.
139. B. F. van Dongen, A. K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM Framework: A New Era in Process Mining Tool Support. In G. C. and P. Darondeau, editors, ICATPN, volume 3536 of Lecture Notes in Computer Science, pages 444—454. Springer, 2005.
140. B. F. van Dongen and W.M.P. van der Aalst. EMiT: A Process Mining Tool. In Jordi Cortadella andWolfgang Reisig, editors, ICATPN, volume 3099 of Lecture Notes in Computer Science, pages 454—463. Springer, 2004.
141. W.M.P. van der Aalst and A.J.M.M. Weijters. Chapter 10: Process Mining. In M. Dumas, W.M.P. van der Aalst, and A.H. ter Hofstede, editors, Process-Aware Information Systems: Bridging People and Software Through Process Technology. John Wiley & Sons, Inc., 2005.
142. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workfkow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering, 10(2):151—162, 2003.

143. B. F. van Dongen and W.M.P. van der Aalst. Multi-phase Process Mining: Building Instance Graphs. In Paolo Atzeni, Wesley W. Chu, Hongjun Lu, Shuigeng Zhou, and Tok Wang Ling, editors, ER, volume 3288 of Lecture Notes in Computer Science, pages 362—376. Springer, 2004.

144. B. F. van Dongen and W.M.P. van der Aalst. Multi-phase Process mining: Aggregating Instance Graphs into EPCs and Petri Nets. In Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workow and Business Process Management (PNCWB), 2005.

145. L. Wen, J. Wang, W.M.P. van der Aalst, Z. Wang, and J. Sun. A Novel Approach for Process Mining Based on Event Types. BETA Working Paper Series, WP 118, Eindhoven University of Technology, Eindhoven, 2004.

146. L. Wen, J. Wang, and J. Sun. Detecting Implicit Dependencies Between Tasks from Event Logs. In Xiaofang Zhou, Jianzhong Li, Heng Tao Shen, Masaru Kitsuregawa, and Yanchun Zhang, editors, APWeb, volume 3841 of Lecture Notes in Computer Science, pages 591—603. Springer, 2006.

147. A. K. Alves de Medeiros. Genetic Process Mining. — Eindhoven : Technische Universiteit Eindhoven, 2006. — Proefschrift.

148. J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In Wayne Read, Jay W. Larson, and A. J. Roberts, editors, Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006, volume 48 of ANZIAM J., pages C725—C740, 2006.

149. J. Garcke, M. Griebel, and M. Thess. Data Mining with Sparse Grids. In Computing, 67:225—253, 2001.

150. F. Girosi, M. Jones, T. Poggio. Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines. AI Memo No. 1430. Artificial Intelligence Laboratory, MIT, 1993.

151. S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. Dokl. Akad. Nauk SSSR 4, 240—243, 1963.

# Предметный указатель


## C

Cluster 160
Conference on Data Systems Languages 19

## D

Data Base Task Group 19
Data Mart 34
Data Mining 18, 68
◊ churn detection 78
◊ descriptive model 82
◊ fraud detection 77, 78, 81
◊ java 60, 243, 267
◊ machine learning 70
◊ predictive model 81
◊ Real-Time 325
◊ SQL Server 2000 266
◊ supervised learning 70
◊ unsupervised learning 70
◊ web usage mining 77
◊ базовые методы 84
◊ биоинформатика 79
◊ генетические алгоритмы (ГА) 87
◊ задачи 69
◊ машинное обучение 70
◊ метод конечного элемента 326
◊ нейронные сети 89
◊ нечеткая логика 84
◊ обучение:
▫ без учителя 70
▫ с учителем 70
◊ описательные модели 82
◊ очистка данных 44
◊ практическое применение 77
◊ предсказательные модели 81
◊ секвенциальный анализ 78, 143
◊ стандарт CRISP 251
◊ стандарт CWM 242
◊ стандарт JDMAPI 267
◊ стандарт PMML 258
◊ стандарт SQL/MM 264
◊ стандарты 242
◊ статический 328
◊ усиление обучения 336
Dendrograms 167

## E

ETL-процесс 39
Executive Information Systems 23

## H

Hard-c-means 172

## I

Information Management System 19
Integrated Database Management System 19

## M

Multi-dimensional conceptual view 50
◊ dimensions 51
◊ drill up 53
◊ measures 51
◊ rotate 52
◊ slice 52
MXML 394

## N

Neighborhood Search 471

## O

OLAP 54
◊ HOLAP 66
◊ MOLAP 60
◊ ROLAP 63
◊ архитектура системы 59
◊ схема "звезда" 63
◊ схема "снежинка" 63
OLAP 18
OLE DB 242, 273
OLTP 18, 23

## P

PageRank 367
Process Mining 382, 389
◊ α-алгоритм 415, 423
◊ MXML 393
◊ ProM 432
◊ Workflow 384
◊ Workflow-сеть 420
◊ Workflow-система 383
◊ алгоритм IdentifyRelevantFeatures 414
◊ алгоритм Process Discovery 409
◊ алгоритм Маркова 399
◊ генетический алгоритм 428
◊ дизъюнктивная Workflow-схема 404
◊ исследование процессов 398
◊ лог событий 422
◊ сервисно-ориентированная архитектура (SOA) 388
◊ сеть Петри 416
◊ структурированная Workflow-сеть 421
Prudsys RE 345

## S

SpamAssassin 348
SQL 18, 62, 154, 273
Support Vector Machines (SVM) 128

## T

Text Mining 211
◊ N-грамма 214
◊ анализ понятий 216
◊ аннотирование 230
◊ извлечение ключевых понятий 215
◊ категоризация 223
◊ лексический анализ 216
◊ навигация по тексту 215
◊ стемминг 213
◊ стоп-слово 213

## V

Visual Mining 192
◊ "лица Чернова" 204
◊ 2D/3D-образы 197
◊ геометрические преобразования 197
◊ гипердоли 200
◊ глифы 205
◊ деревья 208
◊ диаграмма разброса данных 197
◊ динамическое проецирование 198
◊ древовидные карты 208
◊ заполнение пространства 205
◊ иерархические образы 197
◊ иерархические оси 207
◊ интерактивная фильтрация 198
◊ интерактивное искажение 199
◊ интерактивное комбинирование 199
◊ канонические деревья 209
◊ линейчатые фигуры 203
◊ масштабирование образов 198
◊ мозаика 206

◊ наложение измерений 207
◊ отображение иконок 203
◊ параллельные координаты 201
◊ поверхностные и объемные графики 201
◊ рекурсивные шаблоны 206
◊ текстуры и растры 203
◊ точки и матрицы 199
◊ цветные иконки 204

## W

Web Mining 350
◊ HITS 368
◊ PageRank 367
◊ важность Web-структур 366
◊ извлечение Web-контента 358, 363
◊ представление документов 356
Workflow Mining 390

## А

Агент 303
Агенты:
◊ -аналитики 309
◊ мобильность 311
◊ мобильные (МА) 304
◊ общительность 311
◊ свойства 303
Агломеративные алгоритмы 168
Агрегированные данные 37
Алгоритм:
◊ 1R- 106
◊ C4.5 117
◊ ID3 114
◊ Naive Bayes 108
◊ покрытия 119
◊ Apriori 152
◊ AprioriTid 157
◊ k-means 172
Анализатор пакетов 373
Ассоциативные правила:
◊ достоверность (confidence) 150
◊ поддержка (support) 150
◊ улучшение (improvement) 151

## Б

Базовый генетический алгоритм 472
Библиотека Xelopes 271
Бизнес-процесс 382

## В

Визуальный анализ 192
Витрина данных (ВД) 34

## Д

Дендрограмма 167
Деревья решений 104
Дивизимные алгоритмы 170

## И

Информационные потоки 39
Информационные системы руководства 23

## К

Классификационные правила 104
Кластер 159
Кластеризация по Гюстафсону-Кесселю 179
Концепция Захмана 38

## М

Математическая функция 105
Меры близости 164
◊ Евклидово расстояние 164
◊ пиковое расстояние 165
◊ расстояние Махаланобиса 164

Меры близости *(прод.)*:
◊ расстояние по Хеммингу 164
◊ расстояние Чебышева 164
Метаданные 38
Метод поиска соседей 471
Механизм управления транзакциями 22
Многоагентные системы 304
◊ стандарт FIPA 305
◊ стандарт MASIF 304
Многомерная модель данных 50
◊ вращение 52
◊ гиперкуб 51
◊ консолидация 53
◊ мера 51
◊ оси измерений 51
◊ срез 52

## Н

Нейронная сеть:
◊ метод рассуждений Суджено 449
◊ нечеткий нейрон 445
Неоднородная функция распределения вероятности (НФРВ) 479

## О

Оперативные источники данных (ОИД) 29
Очистка данных 41
◊ профайлинг данных 44

## П

Правила Кодда 19, 54
Профайлинг данных 44
Процесс переноса данных 39

## Р

Рекомендательные машины 330
Реляционная БД 19

## С

Системы мобильных агентов (СМА) 304, 307
◊ JADE 307
Совместное фильтрование 331
СППР (Система поддержки принятия решений) 15, 24
Стохастический граф деятельности (SAG) 403
СУБД (Система управления базами данных) 18

## Т

Теория нормализации БД 21
Тест FASMI 58
Транзакция 22

## У

Уравнение Белмана 337

## Х

Хранилище данных (ХД) 29
◊ концепция 47
Хэш-дерево 156

## Э

Эволюционные алгоритмы 472
Эвристические методы 468