А. А. Барсегян
М. С. Куприянов
В. В. Степаненко
И. И. Холод

Методы и модели анализа данных: OLAP и Data Mining

- *Хранилища данных*
- *OLAP – оперативный анализ*
- *Data Mining – интеллектуальный анализ*
- *Методы решения задач классификации, кластеризации и поиска ассоциативных правил*

УЧЕБНОЕ ПОСОБИЕ

bhv®

А. А. Барсегян
М. С. Куприянов
В. В. Степаненко
И. И. Холод

Методы и модели анализа данных: OLAP и Data Mining

Рекомендовано УМО вузов по университетскому политехническому образованию в качестве учебного пособия по специальности 071900 «Информационные системы и технологии» направления 654700 «Информационные системы»

Санкт-Петербург
«БХВ-Петербург»
2004

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73
Б26

Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И.

Б26 Методы и модели анализа данных: OLAP и Data Mining. — СПб.: БХВ-Петербург, 2004. — 336 с.: ил.

ISBN 5-94157-522-X

В книге освещены основные направления в области анализа данных: организация хранилища данных, оперативный (OLAP) и интеллектуальный (Data Mining) анализ данных. Приведено описание методов и алгоритмов решения основных задач анализа: классификации, кластеризации и др. Описание идеи каждого метода дополняется конкретным примером его применения. Представлены стандарты и библиотека алгоритмов Data Mining. Прилагается компакт-диск, содержащий описание стандартов Data Mining, библиотеку алгоритмов Xelopes, лабораторный практикум и необходимое для практической работы программное обеспечение.

Для студентов и специалистов в области анализа данных

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Алексей Данченко*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.04.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 27,09.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

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

ISBN 5-94157-522-X

Содержание

Предисловие авторов **9**

Data Mining и перегрузка информацией **11**

Глава 1. Системы поддержки принятия решений **13**

1.1. Задачи систем поддержки принятия решений 13
1.2. Базы данных — основа СППР 16
1.3. Неэффективность использования OLTP-систем для анализа данных 21
Выводы 26

Глава 2. Хранилище данных **27**

2.1. Концепция хранилища данных 27
2.2. Организация ХД 34
2.3. Очистка данных 39
2.4. Хранилища данных и анализ 45
Выводы 45

Глава 3. OLAP-системы **49**

3.1. Многомерная модель данных 49
3.2. Определение OLAP-систем 53
3.3. Концептуальное многомерное представление 54
3.3.1. Двенадцать правил Кодда 54
3.3.2. Дополнительные правила Кодда 55
3.3.3. Тест FASMI 57
3.4. Архитектура OLAP-систем 58
3.4.1. MOLAP 59
3.4.2. ROLAP 62
3.4.3. HOLAP 65
Выводы 66

Глава 4. Интеллектуальный анализ данных **67**

4.1. Добыча данных — Data Mining 67
4.2. Задачи Data Mining 68

4.2.1. Классификация задач Data Mining 68
4.2.2. Задача классификации и регрессии 70
4.2.3. Задача поиска ассоциативных правил 72
4.2.4. Задача кластеризации 74
4.3. Практическое применение Data Mining 76
4.3.1. Интернет-технологии 76
4.3.2. Торговля 76
4.3.3. Телекоммуникации 77
4.3.4. Промышленное производство 77
4.3.5. Медицина 78
4.3.6. Банковское дело 79
4.3.7. Страховой бизнес 80
4.3.8. Другие области применения 80
4.4. Модели Data Mining 80
4.4.1. Предсказательные (predictive) модели 80
4.4.2. Описательные (descriptive) модели 81
4.5. Методы Data Mining 83
4.5.1. Базовые методы 83
4.5.2. Нечеткая логика 83
4.5.3. Генетические алгоритмы 86
4.5.4. Нейронные сети 88
4.6. Процесс обнаружения знаний 89
4.6.1. Основные этапы анализа 89
4.6.2. Подготовка исходных данных 91
Выводы 93

Глава 5. Классификация и регрессия 95

5.1. Постановка задачи 95
5.2. Представление результатов 96
5.2.1. Правила классификации 96
5.2.2. Деревья решений 97
5.2.3. Математические функции 99
5.3. Методы построения правил классификации 99
5.3.1. Алгоритм построения 1-правил 99
5.3.2. Метод Naive Bayes 101
5.4. Методы построения деревьев решений 104
5.4.1. Методика "разделяй и властвуй" 104
5.4.2. Алгоритм покрытия 112
5.5. Методы построения математических функций 117
5.5.1. Общий вид 117
5.5.2. Линейные методы. Метод наименьших квадратов 119
5.5.2. Нелинейные методы 120
5.5.3. Support Vector Machines (SVM) 121
5.6. Карта Кохонена 124
Выводы 128

Глава 6. Поиск ассоциативных правил 129

6.1. Постановка задачи 129
6.1.1. Формальная постановка задачи 129

6.1.2. Сиквенциальный анализ 132
6.1.3. Разновидности задачи поиска ассоциативных правил 135
6.2. Представление результатов 137
6.3. Алгоритмы 141
6.3.1. Алгоритм Apriori 141
6.3.2. Разновидности алгоритма Apriori 146
Выводы 147

Глава 7. Кластеризация 149

7.1. Постановка задачи кластеризации 149
7.1.1. Формальная постановка задачи 152
7.1.2. Меры близости, основанные на расстояниях, используемые в алгоритмах кластеризации 154
7.2. Представление результатов 156
7.3. Базовые алгоритмы кластеризации 158
7.3.1. Классификация алгоритмов 158
7.3.2. Иерархические алгоритмы 159
Агломеративные алгоритмы 159
Дивизимные алгоритмы 161
7.3.3. Неиерархические алгоритмы 162
Алгоритм *k*-means (Hard-c-means) 162
Алгоритм Fuzzy C-Means 166
Кластеризация по Гюстафсону-Кесселю 168
7.4. Кластеризация данных при помощи нечетких отношений 174
7.4.1. Анализ свойств нечетких бинарных отношений применительно к анализу данных 174
Отношения и свойства отношений 174
Сравнение данных 179
Отношение α-толерантности 181
7.4.2. Отношение α-квазиэквивалентности 182
Построение шкалы отношения α-квазиэквивалентности как алгоритм анализа данных 190
Об использовании шкалы α-квазиэквивалентности для анализа данных 191
Примеры анализа данных при помощи шкалы α-квазиэквивалентности 192
Выводы 204

Глава 8. Стандарты Data Mining 209

8.1. Кратко о стандартах 209
8.2. Стандарт CWM 209
8.2.1. Назначение стандарта CWM 209
8.2.2. Структура и состав CWM 211
8.2.3. Пакет Data Mining 214
8.3. Стандарт CRISP 218
8.3.1. Появление стандарта CRISP 218
8.3.2. Структура стандарта CRISP 218
8.3.3. Фазы и задачи стандарта CRISP 220

8.4. Стандарт PMML ... 225
8.5. Другие стандарты Data Mining ... 233
8.5.1. Стандарт SQL/MM ... 233
8.5.2. Стандарт OLE DB для Data Mining ... 235
8.5.3. Стандарт JDMAPI ... 237
Выводы ... 237

Глава 9. Библиотека Xelopes ... 241

9.1. Архитектура библиотеки ... 241
9.2. Диаграмма Model ... 244
9.2.1. Классы модели для Xelopes ... 244
9.2.2. Методы пакета Model ... 246
9.2.3. Преобразование моделей ... 247
9.3. Диаграмма Settings ... 248
9.3.1. Классы пакета Settings ... 248
9.3.2. Методы пакета Settings ... 250
9.4. Диаграмма Attribute ... 250
9.4.1. Классы пакета Attribute ... 250
9.4.2. Иерархические атрибуты ... 251
9.5. Диаграмма Algorithms ... 252
9.5.1. Общая концепция ... 252
9.5.2. Класс *MiningAlgorithm* ... 253
9.5.3. Расширение класса *MiningAlgorithm* ... 254
9.5.4. Дополнительные классы ... 256
9.5.5. Слушатели ... 256
9.6. Диаграмма DataAccess ... 256
9.6.1. Общая концепция ... 257
9.6.2. Класс *MiningInputStream* ... 258
9.6.3. Классы Mining-векторов ... 258
9.6.4. Классы, расширяющие класс *MiningInputStream* ... 258
9.7. Диаграмма Transformation ... 259
9.8. Примеры использования библиотеки Xelopes ... 261
9.8.1. Общая концепция ... 261
9.8.2. Решение задачи поиска ассоциативных правил ... 264
9.8.3. Решение задачи кластеризации ... 266
9.8.4. Решение задачи классификации ... 268
Выводы ... 271

Приложение 1. Нейронечеткие системы ... 273

П1.1. Способы интеграции нечетких и нейронных систем ... 273
П1.2. Нечеткие нейроны ... 277
П1.3. Обучение методами спуска ... 279
П1.4. Нечеткие схемы рассуждений ... 280
П1.5. Настройка нечетких параметров управления с помощью нейронных сетей ... 286
П1.6. Нейронечеткие классификаторы ... 293

Приложение 2. Особенности и эффективность генетических алгоритмов.....299

П2.1. Методы оптимизации комбинаторных задач различной степени сложности.... 299
П2.2. Сущность и классификация эволюционных алгоритмов 304
П2.2.1. Базовый генетический алгоритм.....304
П2.2.2. Последовательные модификации базового генетического алгоритма.....305
П2.2.3. Параллельные модификации базового генетического алгоритма.....307
П2.3. Классификация генетических алгоритмов 310
П2.4. Особенности генетических алгоритмов, предпосылки для адаптации 311
П2.5. Классификация адаптивных ГА 314
П2.5.1. Основа адаптации 314
П2.5.2. Область адаптации 316
Адаптация на уровне популяции 316
Адаптация на уровне индивидов..... 317
Адаптация на уровне компонентов..... 318
П2.5.3. Основа управления адаптацией 318
П2.6. Двунаправленная интеграция ГА и нечетких алгоритмов продукционного типа..... 319

Приложение 3. Описание прилагаемого компакт-диска.....327

Список литературы.....331

Предметный указатель 335

Предисловие авторов

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

Действительно, управление предприятием, банком, различные сферы бизнеса, в том числе электронного, немыслимы 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. В частности, *разд. 7.4* подготовлен С. И. Елизаровым.

Книга ориентирована на студентов и специалистов, интересующихся современными методами анализа данных. Наличие в приложениях 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 применения информации систем, но и для разработки конкретных систем.

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

Благодарности:

Григорию Пятецкому-Шапиро — основателю направления Data Mining за поддержку и полезные замечания.

Доктору М. Тессу — одному из руководителей немецкой компании Prudsys за исключительно содержательные консультации по структуре книги и по содержанию ее отдельных частей.

Data Mining и перегрузка информацией

В 2002 году, согласно оценке профессоров из университета Berkeley, объем информации в мире увеличился на пять миллиардов 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 для 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.

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 — торгов-

ле, телекоммуникациях, 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 Гугл (Google) или другие поисковые системы (search engines) на просторах Интернета.

Виды информации, с которыми работают исследователи, включают 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), которая имеет приложения в таких разных областях, как биоинформатика, цифровые библиотеки и защита против 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. Ряд глав рассматривают 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, Январь 2004

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. Таким образом, можно выделить три основные задачи, решаемые в СППР:

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

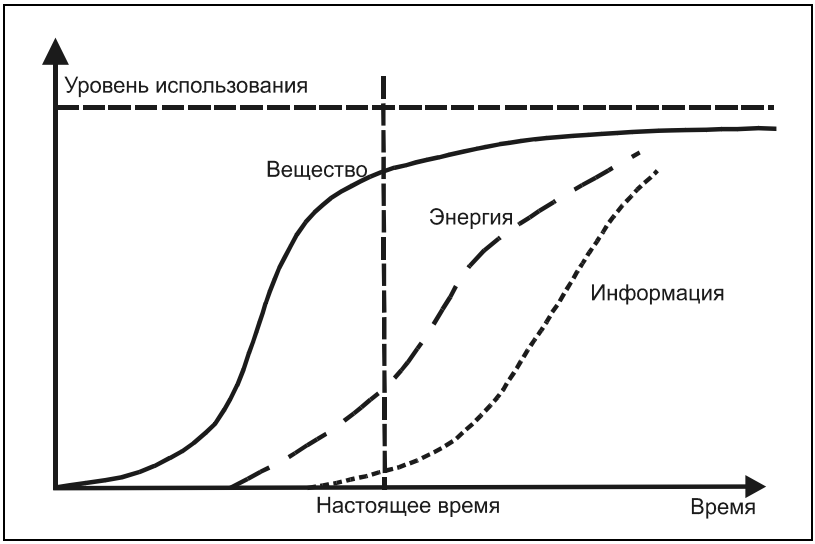

Рис. 1.1. Уровень использования человеком различных объектов материального мира

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, либо 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 обеспечивают выполнение функции поддержки принятия решений. Очевидно, что, с одной стороны, качество принятых решений зависит от квалификации аналитика. С другой — 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 (с определенной вероятностью) прогнозируют развитие некоторых процессов.

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

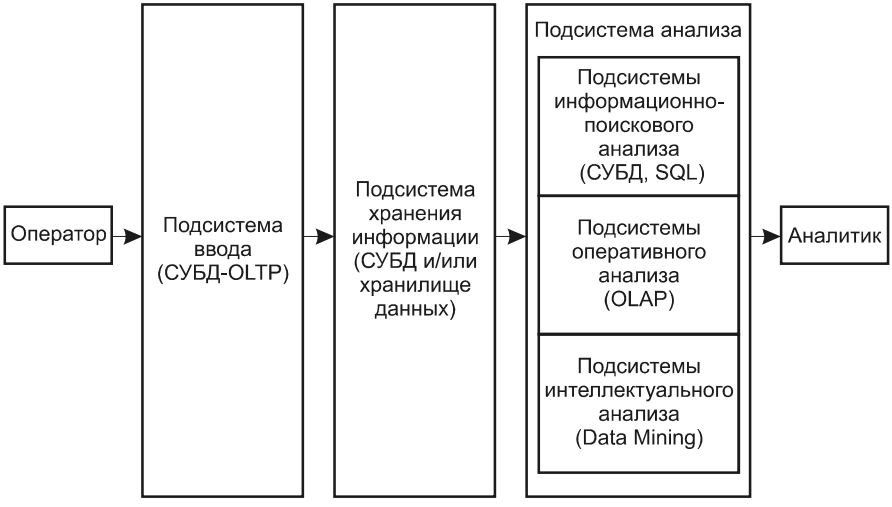

Рис. 1.2. Обобщенная архитектура системы поддержки принятия решений

Рассмотрим отдельные подсистемы более подробно.

Подсистема ввода данных. В таких подсистемах, называемых OLTP (On-line transaction processing), реализуется операционная (транзакционная) обработка данных. Для их реализации 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 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.

Системы, предоставляющие средства работы с БД, называются 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 элемента 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 NULL. Его нельзя путать с пустой строкой или со значением 0.

4. **БД должна включать в себя метаданные** — БД хранит два вида таблиц: пользовательские таблицы и системные таблицы. 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 структурных запросов 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 от 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 поддерживать 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 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 изображения, 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, 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, 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 в себе 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 таком 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 произвольные 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 в себе представления (view), 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, работа с OLTP-системами, как правило, выполняется в режиме реального

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 для систем OLTP, на рис. 1.4 — для систем анализа, отражающие загрузку процессора в течение дня.

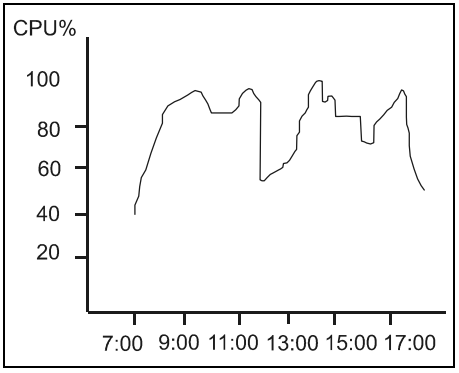

Рис. 1.3. Загрузка процессора для систем OLTP

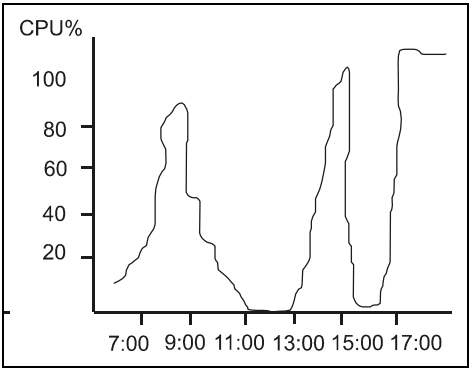

Рис. 1.4. Загрузка процессора для систем анализа

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, другими словами то, что необходимо аналитикам для анализа данных.

Противоречивость требований к OLTP-системам и системам, ориентированным на глубокий анализ информации, усложняет задачу интеграции их 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 разделяется на информационно-поисковый, оперативно-аналитический и интеллектуальный классы.
- 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 СППР.
- В настоящее время для объединения в рамках одной системы OLTP подсистем и подсистем анализа используется концепция хранилищ данных. Общая идея заключается в разделении БД для OLTP-систем 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 Девлин и Мэрфи. В 1992 г. Уильман Г. Инмон подробно описал данную концепцию в своей монографии "Построение хранилищ данных".

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 — является фундаментальным отличием 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 необходимости 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 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 больший 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 приводятся к единому формату. После такой обработки данные занимают гораздо меньший объем.

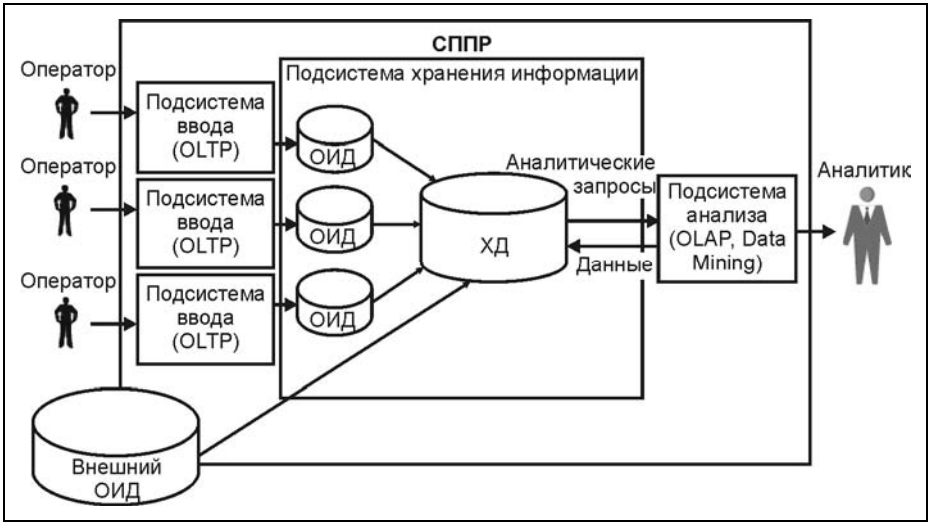

Рис. 2.1. Структура СППР с физическим ХД

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.

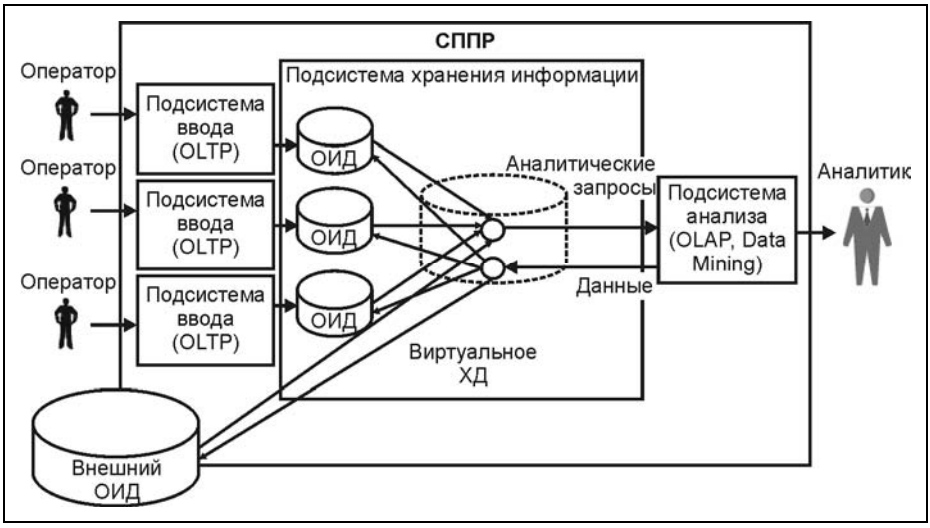

Рис. 2.2. Структура СППР с виртуальным ХД

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 аналитических запросов, либо к неверным результатам.

Выполнение сложных аналитических запросов над ОИД занимает большой объем ресурсов компьютеров, на которых они работают. 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 потребуется память объемом в 4,87 раза бóльшая, 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.

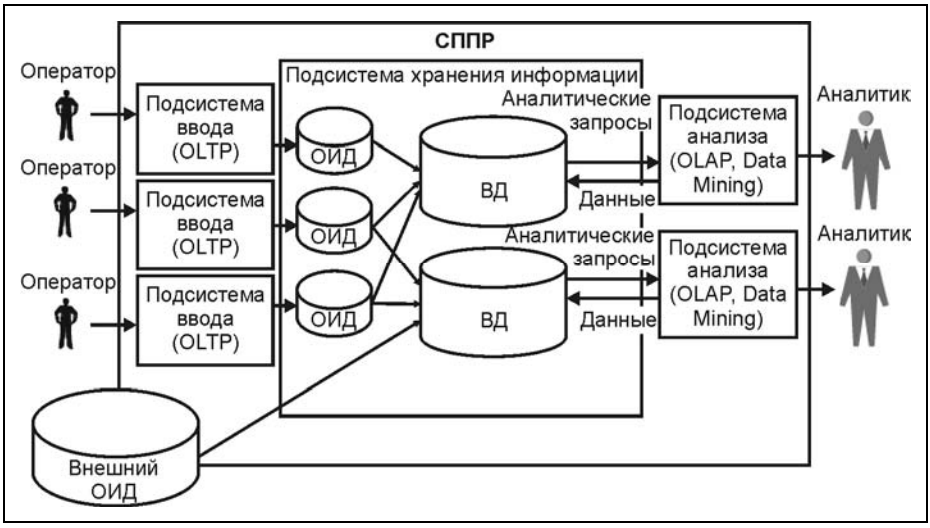

Рис. 2.3. Структура СППР с самостоятельными ВД

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.

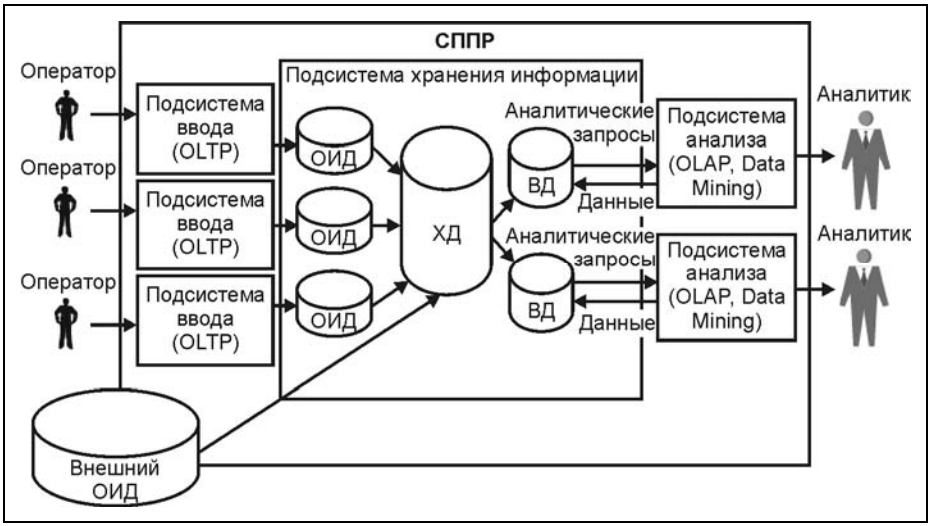

Рис. 2.4. Структура СППР с ХД и ВД

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.

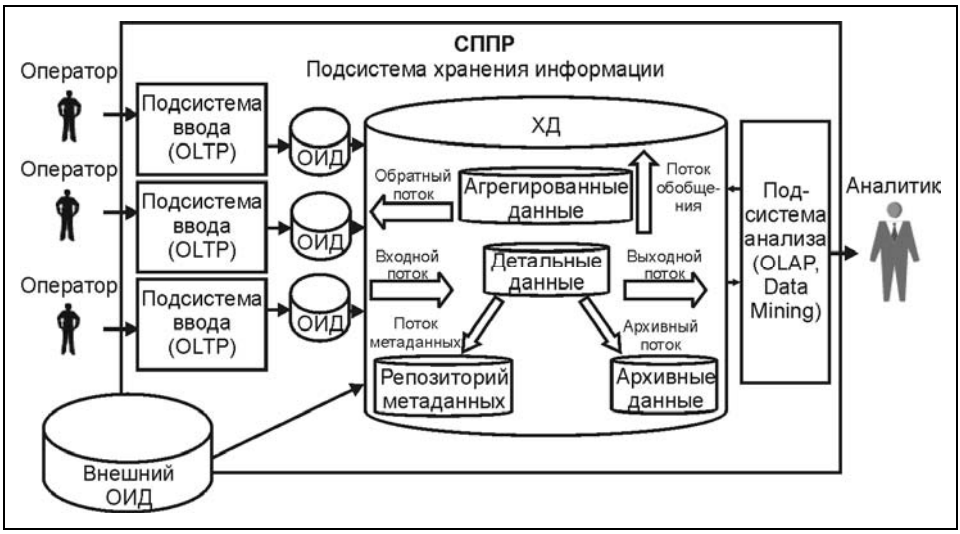

Рис. 2.5. Архитектура ХД

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.

Проведенные исследования показали, что 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 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 потоком 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.

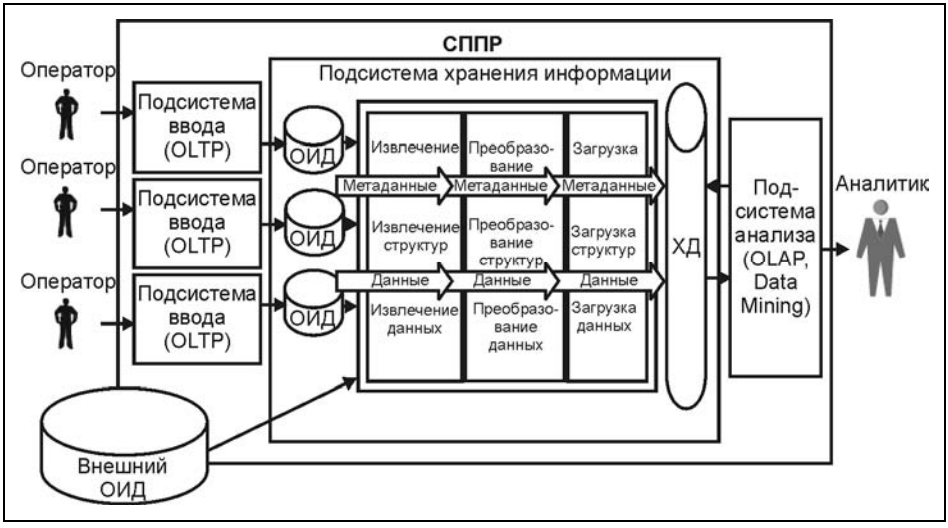

Рис. 2.6. ETL-процесс

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 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 — 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.

1. Орфографические ошибки (опечатки) — ошибки, возникающие при вводе информации. Они могут привести к неправильному пониманию, а также к искажению реальных данных. Например, при продаже товара вместо количества 1000 было введено 10 000 или вместо названия товара "Водка" было введено название "Вода".
2. Отсутствие данных — 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).
3. Фиктивные значения — 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.
4. 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.
5. Закодированные значения — сокращенная запись или кодировка реальных данных, используемая для уменьшения занимаемого места.
6. 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: для человека возраст не соответствует 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 нижеприведенные 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 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 данном 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 такие данные может породить еще более неверные результаты, и, возможно, лучшим решением будет отказ 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 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 % вероятность правила

"итого = количество × единицу" демонстрирует несоответствие и потребность в более детальном исследовании для 1 % 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 необходимости 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 за тем — 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 — данная 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 — данная 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 — данная 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. 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 их выгрузке, очистке, согласовании, транспортировке и загрузке.

Необходимо понимать, что концепция ХД:

- это не концепция анализа данных, скорее, это концепция подготовки данных для анализа;
- не предопределяет архитектуру целевой аналитической системы. Она говорит о том, какие процессы должны выполняться в системе, но не о том, 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 их организации и представления в целевой БД и режимах их использования. ХД — это концепция построения аналитической системы, но не концепция 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 анализ 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 Уильмана Г. 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 могут быть реализованы 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 *таблица БД*, уровень *одиночной БД*, уровень *множества БД*.

- Очистка данных включает следующие этапы: выявление проблем в данных, определение правил очистки, тестирование правил очистки, непосредственно 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 этих гипотез в законченные решения они 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, традиционные средства анализа, оперирующие данными, которые представлены в виде таблиц реляционной БД, не могут в полной мере удовлетворять таким требованиям. В 1993 г. Э. Ф. 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, например, 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.

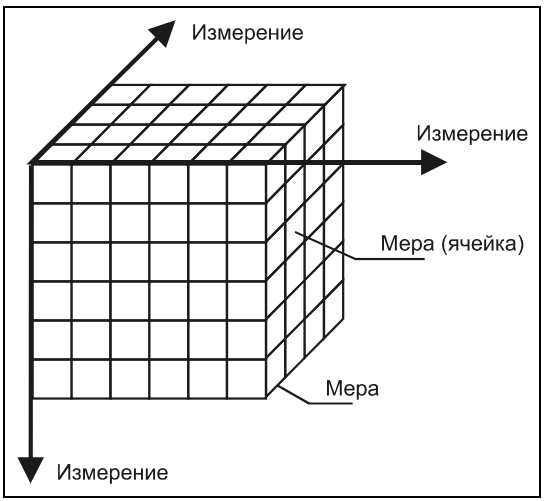

Рис. 3.1. Представление данных в виде гиперкуба

Таким образом, многомерную модель данных можно представить как гиперкуб (рис. 3.1) (конечно, название не очень удачно, 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) — формируется подмножество многомерного массива данных, соответствующее 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.

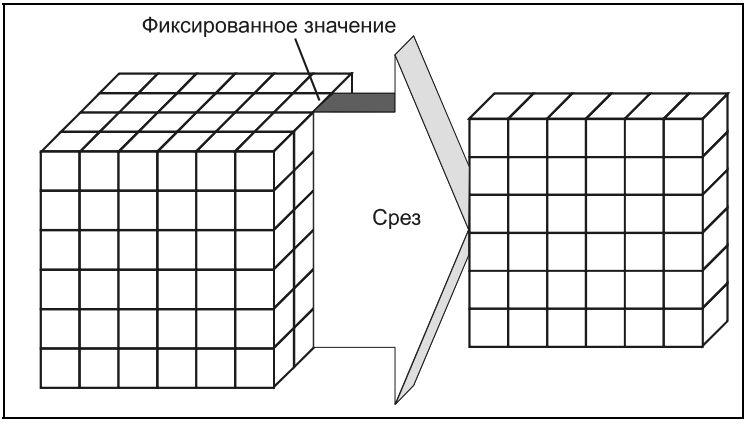

Рис. 3.2. Операция среза

- 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 для первого случая может служить такой отчет, для которого элементы измерения "Время" располагаются поперек экрана (являются заголовками столбцов таблицы), а элементы измерения "Продукция" — вдоль экрана (являются

заголовками 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 расположенным по горизонтали измерением "Время" и измерением "Продукция", расположенным по вертикали, в отчет, у которого по горизонтали представлено измерение "Время", а по вертикали — измерение "География" (синоним: Pivot).

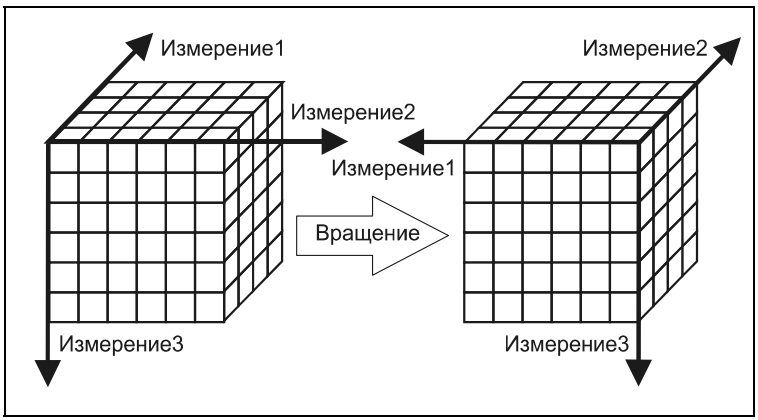

Рис. 3.3. Операция вращения

- 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.

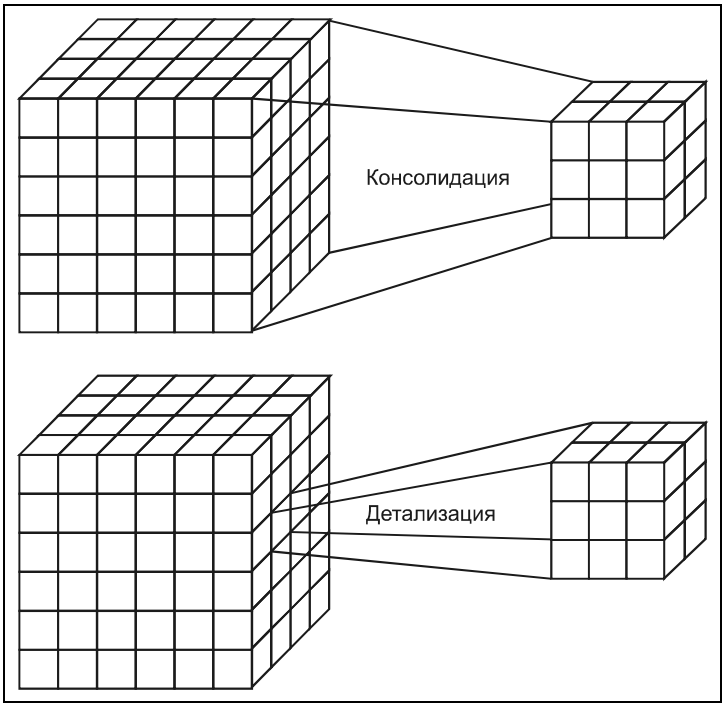

Рис. 3.4. Операции консолидации и детализации

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

Ниже 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 предоставлена любому измерению.
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 работать нескольким пользователям совместно с одной аналитической моделью или 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. отчеты должны представляться в любой возможной ориентации. 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 может использоваться до 19 измерений. Отсюда вытекает настоятельная рекомендация, чтобы аналитический инструмент мог одновременно предоставить хотя бы 15, а предпочтительно — 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 этих требований, послуживших де-факто определением OLAP, достаточно часто вызывает различные нарекания, например, правила 1, 2, 3, 6 являются требованиями, а правила 10, 11 — неформализованными пожела-

ниями. Таким образом, перечисленные 12 требований Кодда не позволяют точно определить OLAP. В 1995 г. Кодд к приведенному перечню добавил следующие шесть правил:

13. **Пакетное извлечение против интерпретации** — OLAP-система должна в равной степени эффективно обеспечивать доступ как к собственным, так и к внешним данным.
14. **Поддержка всех моделей OLAP-анализа** — OLAP-система должна поддерживать все четыре модели анализа данных, определенные Коддом: категориальную, толковательную, умозрительную и стереотипную.
15. **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.
16. **Сохранение результатов OLAP: хранение их отдельно от исходных данных** — OLAP-система, работающая в режиме чтения-записи, после модификации исходных данных должна результаты сохранять отдельно. Иными словами, обеспечивается безопасность исходных данных.
17. **Исключение отсутствующих значений** — OLAP-система, представляя данные пользователю, должна отбрасывать все отсутствующие значения. Другими словами, отсутствующие значения должны отличаться от нулевых значений.
18. **Обработка отсутствующих значений** — OLAP-система должна игнорировать все отсутствующие значения без учета их источника. Эта особенность связана с 17-м правилом.

Кроме того, 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);
- ❒ стандартная 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

Определенные ранее особенности распространены. Более известен тест FASMI (Fast of Shared Multidimensional Information), созданный в 1995 г. Найджелом Пендсом (Nigel Pendse) и Ричардом Критом (Richard Creeth) 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. представление анализируемых фактов как функций от большого числа их характеризующих параметров. Таким образом, они 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 с. При этом самые простые запросы обрабатываются в течение 1 с, и очень немногие более 20 с. Недавнее исследование в Нидерландах показало, что конечные пользователи воспринимают процесс неудачным, если результаты не получены по истечении 30 с. Они способны 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 ко-

личеством 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 множественный доступ для записи необходим, 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 1000 раз большим 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 — для реализации многомерной модели используют многомерные БД;
- ROLAP — для реализации многомерной модели используют реляционные БД;
- HOLAP — 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 многомерные 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	1500	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	2000	50
Женская гимназия № 1	07.03.99	Дмитрий А.	Маркеры	2250	150
Женская гимназия № 1	07.03.99	Алексей Ш.	Карандаши	230	10
Женская гимназия № 1	07.03.99	Алексей Ш.	Ручки	1000	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:

- 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 всего 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 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).

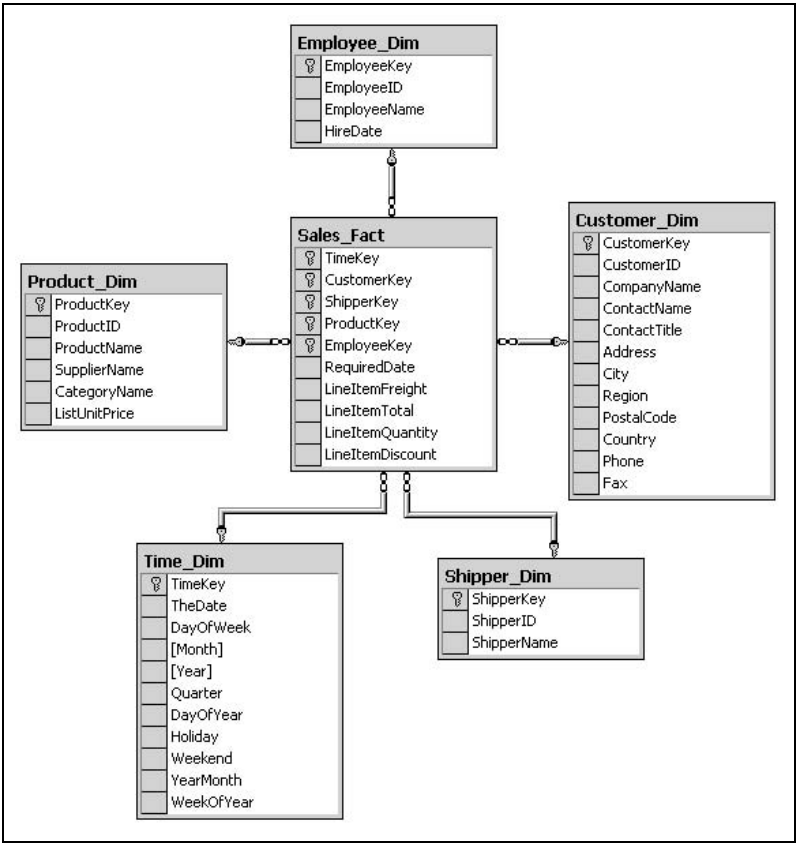

Рис. 3.5. Пример схемы "звезда"

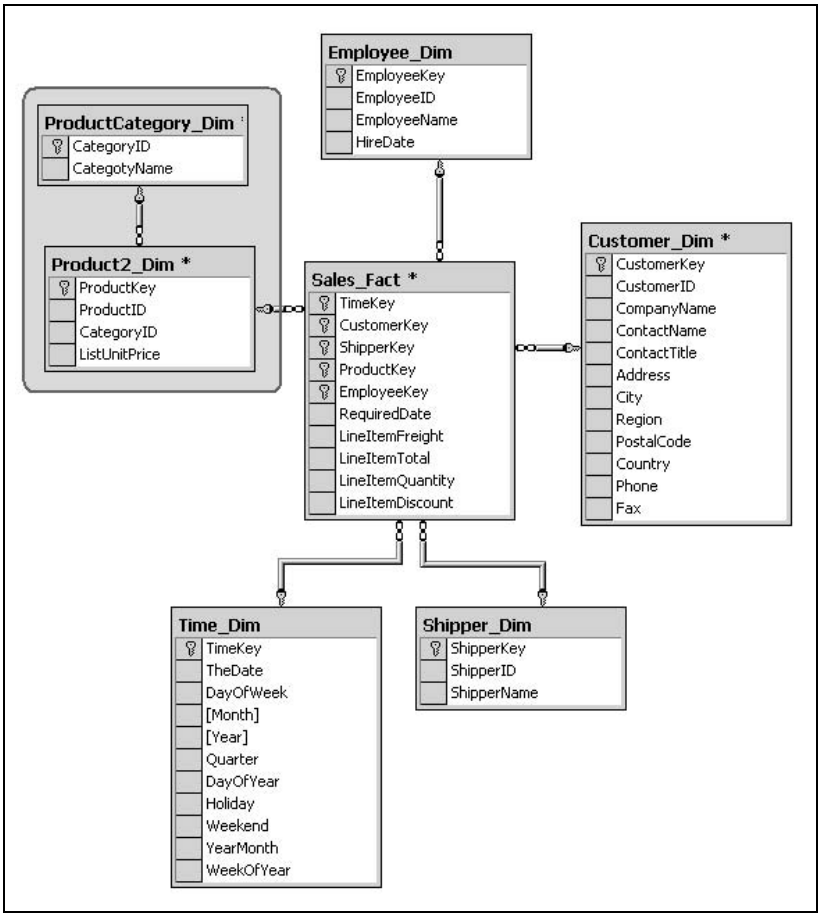

Рис. 3.6. Пример схемы "снежинка"

Основными составляющими таких схем 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 являются телефонный звонок или снятие денег со счета с помощью банкомата);
- факты, связанные с "моментальными снимками" (Snapshot facts). Основаны на состоянии объекта (например, банковского счета) в определенные моменты времени, например на конец дня или месяца. Типичными примерами таких фактов являются объем продаж за день или дневная выручка;

- факты, связанные с элементами документа (Line-item facts). Основаны на том или ином документе (например, счете за товар или услуги) и содержат подробную информацию об элементах этого документа (например, количестве, цене, проценте скидки);
- факты, связанные с событиями или состоянием объекта (Event or state facts). 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, в дальнейшем используемые для построения иерархий в измерениях куба, содержатся в таблицах измерений.

Таблицы измерений содержат неизменяемые либо 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 луч-

шей 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 довольно 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. Эти операции и многомерную модель 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 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 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 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. Ниже 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, заранее известно.
- 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 неудовлетворительно, 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. Отсюда и название — unsupervised learning. Достоинством таких задач является возможность их

решения без каких-либо предварительных знаний об анализируемых данных. К ним 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 — человеке): его месте работы, размере заработной платы, возрасте, составе 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 (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 определенных слов;
- значения цвета пикселов матрицы.

Зависимыми переменными в этих же примерах являлись:

- 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 построенная на ее основе функция классификации или регрессии будет точнее;
- 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 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.

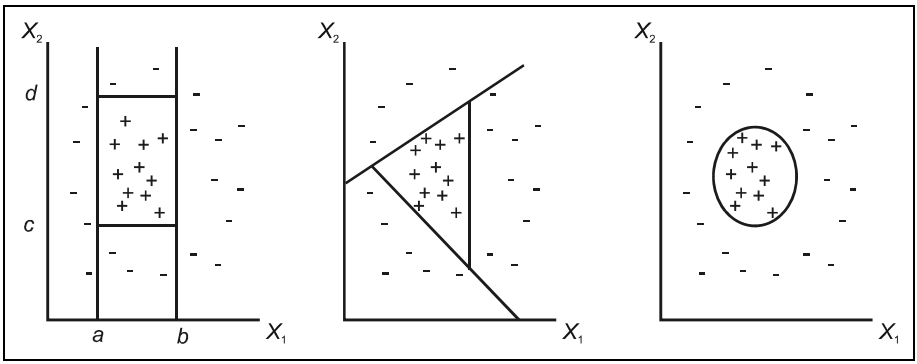

Рис 4.1. Классификация в двумерном пространстве

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 такая модель будет некорректно работать в дальнейшем 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 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, 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:

$$\{e_5, e_2, e_7, e_{13}, e_6, e_1, ...\},$$

где e_i — сбой с кодом i, 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. 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.

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 (характеристики) 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. С этой точки зрения 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 задачей.

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, просто для получения групп схожих объектов. После определения кластеров применяются другие методы Data Mining, для того чтобы попытаться установить, а 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, четко количественно очерченные объекты, а с другой — объекты, имеющие вероятностное или нечеткое описание.

Во-вторых, решение значительно зависит также и от представления кластеров и предполагаемых отношений объектов данных и кластеров. Так, необходимо учитывать такие свойства, как возможность/невозможность принадлежности объектов нескольким кластерам. Необходимо определение самого понятия принадлежности кластеру: однозначная (принадлежит/не принадле-

жит), 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...". Персонализация клиентов, 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 также 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. Это Web Content Mining и Web Usage Mining. В первом случае речь идет об автоматическом поиске и извлечении качественной информации из перегруженных "информационным шумом" источников Интернет, а также о всевозможных средствах автоматической классификации и аннотировании документов. Данное направление также называют Text Mining. Web Usage Mining направлен на обнаружение закономерностей в поведении пользователей конкретного Web-узла (группы узлов), в частности на то, какие страницы и в какой временной последовательности запрашиваются пользователями и какими группами пользователей.

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 сик-

венциальный анализ. Зная связи между покупками и временные закономерности, 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 кластеризацию и классификацию.

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 (fraud detection), 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, экономические кризисы и т. д). В общем, все то, что нарушает обычный ход вещей, и положено в основе Data Mining и статистики — *принцип прецедента*. Опыт работы компаний, предлагающих решения Data Mining для промышленного производства, также свидетельствует об успешности такой интеграции. Примером использования 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 изуче-

ния 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 использования 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, и 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 "прошлых" клиентов. С помощью инструментов Data Mining (деревья решений, кластерный анализ, нейронные сети и др.) банк может получить профили добросовестных и неблагонадежных заемщиков. Кроме того, возможно 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, так же 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. Также с помощью технологий Data Mining решается такая часто встречающаяся в страховании задача, как определение случаев мошенничества (fraud detection).

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 (predictive) модели

Предсказательные модели строятся на основании набора данных с известными результатами. Они используются для предсказания результатов на осно-

вании других наборов данных. При этом, естественно, требуется, чтобы модель работала максимально точно, была статистически значима и оправданна и т. д.

К ним 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;
- ❒ 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 (descriptive) модели

Описательные модели уделяют внимание сути зависимостей в наборе данных, взаимному влиянию различных факторов, т. е. на построении 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 ним относятся следующие виды моделей:

- ❒ регрессионные — 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;
- ❒ кластеризации — 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 больше похожи объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация;
- ❒ 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, что 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 — 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, Data Summarization — это нахождение 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;
- ассоциации — 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, теория баз данных, вполне закономерно, что большинство алгоритмов и методов Data Mining были разработаны на основе различных технологий и концепций. 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, 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. Основная их идея сводится к корреляционному, регрессионному и другим видам статистического анализа. Основным 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:

- неизвестность;
- неполнота (недостаточность, неадекватность);
- 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:

- неточность (неточность измерений значений определенной величины, выполняемых физическими приборами);
- 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:

- неопределенность значений слов (многозначность, расплывчатость, неясность, нечеткость). Она возникает в случае, если отображаемые одним и тем же словом объекты задачи управления различны;
- 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. Подобные модели получили название лингвистических.

Рассмотрим основные достоинства нечеткой логики, наиболее ярко проявляющиеся на примере общей задачи нечеткого управления. Если говорить кратко, нечеткая логика позволяет удачно представить мышление человека. Очевидно, что в повседневной деятельности человек никогда не пользуется формальным моделированием на основе математических выражений; он 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, причем используя 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:

- правила принятия решений являются условными высказываниями типа "если..., то..." и реализуются с помощью механизма логического вывода;
- 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;
- 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, сравнивая, 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 настрой-

ку 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 ниже 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.), упрощает обеспечение интегрированности, снижает требования приложений к быстродействию ЭВМ и 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 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 могут 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 (1994, p. 2) определяет 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 и (2) для хранения информации используются величины интенсивности межнейронных соединений, которые называются синаптическими весами.

Однако, как отмечает Риплей (1996), 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 созданием 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 существующих проблем.

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

- ❒ понимание и формулировка задачи анализа;
- ❒ подготовка данных для автоматизированного анализа (препроцессинг);
- ❒ применение методов Data Mining и построение моделей;
- ❒ проверка построенных моделей;
- ❒ интерпретация моделей человеком.

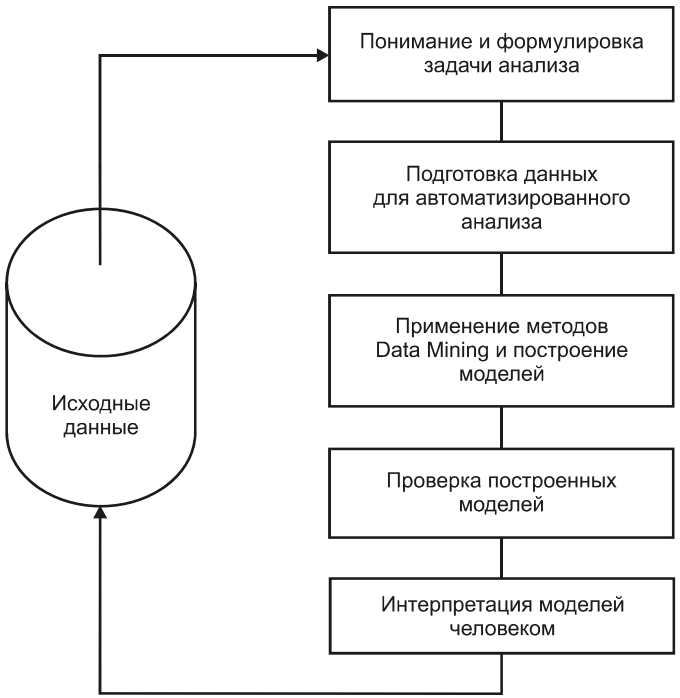

Рис. 4.2. Этапы интеллектуального анализа данных

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 ниже 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 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.

Рассмотренным этапом и завершается цикл Data Mining в строгом смысле этого слова. 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, для применения того или иного метода Data Mining к данным их необходимо подготовить к этому. Например, стоит 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 (не менее чем квадратичная, а нередко и экспоненциальная). В то же время зависимость времени от количества исследуемых объектов линейна или близка к линейной. Поэтому в качестве предобработки 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 также бывает необходимо провести предварительную очистку данных по строкам таблицы (записям). Любая реальная база данных обычно содержит ошибки, очень неточно 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 о искомой

зависимости невозможно. Эта предварительная обработка или препроцессинг данных и составляет второй этап.

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, доступны для интерпретации человеком.
- Методами Data Mining решаются три основные задачи: классификация и регрессия, поиск ассоциативных правил и кластеризация. По назначению они делятся на описательные и предсказательные задачи. 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, облегчить анализ.
- Методы Data Mining находятся на стыке различных направлений информационных технологий: статистики, нейронных сетей, нечетких множеств, генетических алгоритмов и др.
- Интеллектуальный анализ включает в себя следующие этапы: понимание и формулировка задачи анализа, подготовка данных для автоматизирован-

ного 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.
- Методы Data Mining могут эффективно использоваться в различных областях человеческой деятельности: бизнеса, медицины, науки, телекоммуникаций и др.

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

Таблица 5.1 *(окончание)*

Наблюдение	Температура	Влажность	Ветер	Игра
Солнце	Норма	Норма	Есть	Да
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_1 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

Несмотря на то что был назван способ определения значения зависимой переменной функцией классификации или регрессии, он необязательно может быть выражен математической функцией. Существуют следующие основные виды представления таких способов: классификационные правила, деревья решений и 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. Другое преимущество — относительная их независимость. В набор правил легко добавить новое 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, имеются правила:

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

В них объекты, удовлетворяющие условиям из второго правила, удовлетворяют и условиям первого правила. Однако вывод делается разный. Другими словами, в соответствии с этими правилами при одинаковых обстоятельствах 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.

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.

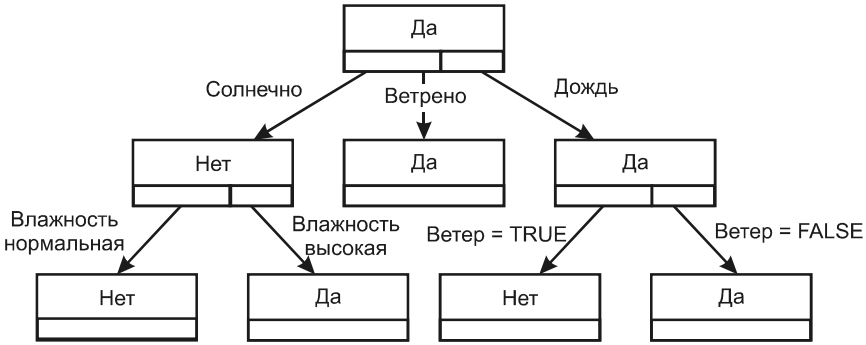

Рис. 5.1. Пример дерева решений

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 большему 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 большей 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. В этом случае анализируемые объекты рассматриваются как точки в $(n + 1)$-мерном пространстве. Тогда переменные объекта $i_j = \{x_1, x_2, ..., x_h, ..., x_m, y\}$ рассматривают как координаты, а функция имеет следующий вид:

$$y_i = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ...+ \omega_m x_m,$$

где ω_0, ω_1, ..., ω_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. В системе 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 объекты 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 алгоритм 1R 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 алгоритм 1R, 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 все же 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 условной 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:

$$\text{если } x_1 = c_h^1 \text{ и } x_2 = c_h^2 \text{ и ... } x_m = c_h^m \text{ тогда } y = c_r.$$

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

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

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:

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

то есть 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`:

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

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,
```

надо вычислить следующие условные вероятности:

$$P(\text{игра=да} \mid E) = P(\text{наблюдение = солнечно} \mid \text{игра = да}) \times$$
$$\times P(\text{температура = холодно} \mid \text{игра = да}) \times$$
$$\times P(\text{влажность = высокая} \mid \text{игра = да}) \times$$
$$\times P(\text{ветер = есть} \mid \text{игра = да})*P(\text{игра = да})/P(E);$$

$$P(\text{игра = нет} \mid E) = P(\text{наблюдение = солнечно} \mid \text{игра = нет}) \times$$
$$\times P(\text{температура = холодно} \mid \text{игра = нет}) \times$$
$$\times P(\text{влажность = высокая} \mid \text{игра = нет}) \times$$
$$\times P(\text{ветер = есть} \mid \text{игра = нет}) \times P(\text{игра = нет})/P(E).$$

Подставляя соответствующие вероятности, получим следующие значения:

$$P(\text{игра = да} \mid E) = 2/9 \cdot 3/9 \cdot 3/9 \cdot 3/9 \cdot 9/14 \,/\, P(E) = 0{,}0053 \,/\, P(E);$$
$$P(\text{игра = нет} \mid E) = 3/5 \cdot 1/5 \cdot 4/5 \cdot 3/5 \cdot 5/14 \,/\, P(E) = 0{,}0206 \,/\, P(E).$$

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 определяется 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 со средним μ и среднеквадратичным отклонением σ:

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

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 — подход к построению 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^n; T разбивается на подмножества T_1, T_2, ..., T_n, где каждое подмножество T_i содержит все объекты, имеющие значение c_h^i 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. количество объектов из других классов ("примесей") в каждом из этих множеств было минимальным. Разные алгоритмы реализуют различные 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. 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", 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 (L. Hyafill) и Р. Ривестом (R. Rivest) 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. 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 $|X|$ (количеством 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 x_h даст подмножества T_1, T_2, ..., T_m, при x_h равном соответственно c_{h1}, c_{h2}, ..., c_{hm}. 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).

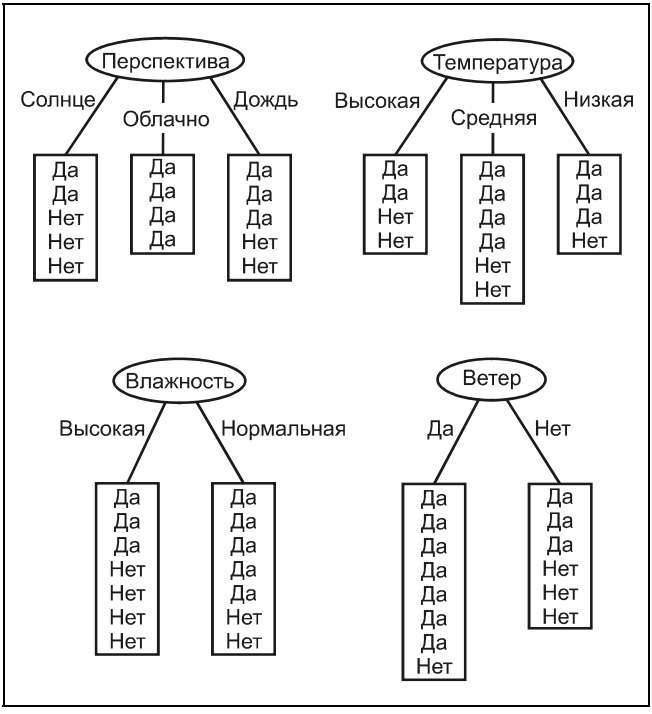

Рис. 5.2. Четыре варианта первоначального разбиения дерева для разных переменных

Пусть 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:

$$P = \frac{\text{freq}(c_r, I)}{|I|}.$$

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, это выражение дает количественную оценку в битах. Для данного примера:

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

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

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

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:

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

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 производится 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:

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

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

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

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

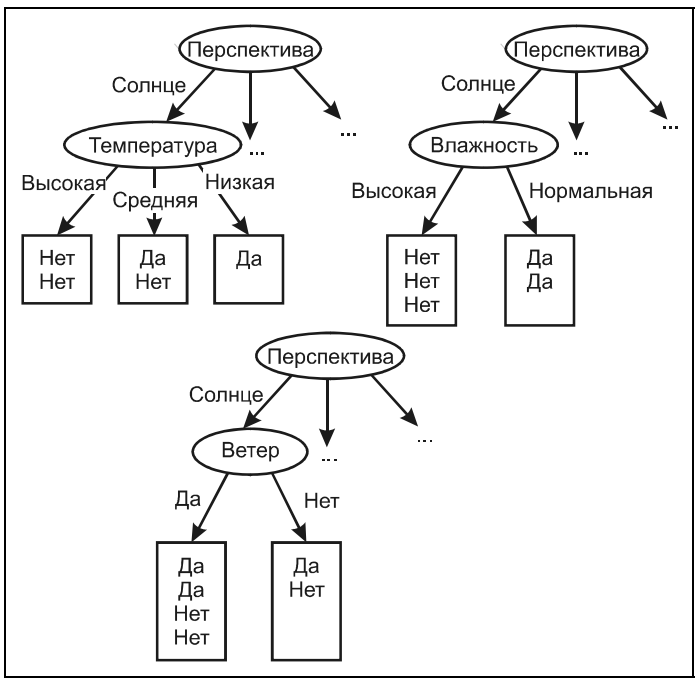

Рис. 5.3. Разбиение дерева на второй итерации для подмножества $T_{\text{солнечно}}$

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 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 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 Info_x будет иметь минимальное значение и, соответственно, критерий Gain(X) достигнет 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

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

Значит критерий 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 Info(T), имеем:

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

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

Рассмотрим следующее выражение:

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

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:

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

$$\mathrm{Info}_{xh}(T) = \sum_{i=1}^{m} |T_i|/(|T| - U)\, \mathrm{Info}(T_i).$$

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

Тогда критерий можно переписать:

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

Подобным образом изменяется и критерий gain ratio. Если проверка имеет n выходных значений, то критерий gain ratio считается как 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 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:

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

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.

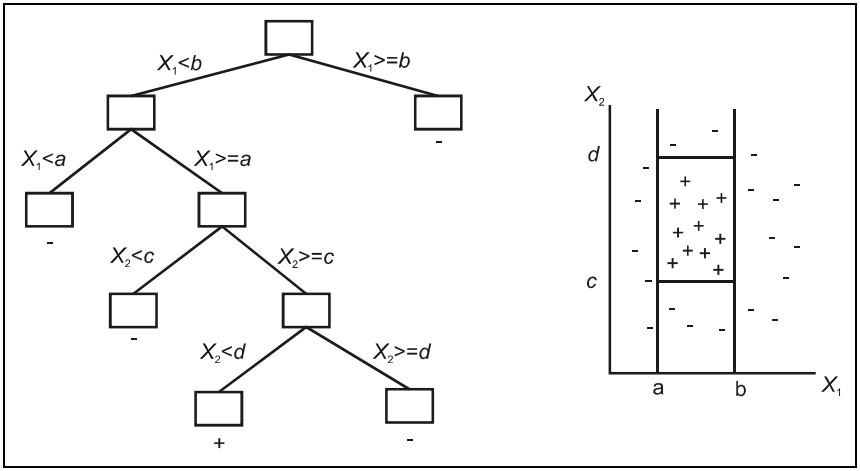

Рис. 5.4. Геометрическая интерпретация идеи алгоритма покрытия

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
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

Таблица 5.3 *(окончание)*

Возраст	Предписание	Астигматизм	Степень износа	Рекомендации
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, необходимо 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
Пожилой	Дальнозоркость	Да	Нормальный	Нет
Старческий	Близорукость	Да	Пониженный	Нет
Старческий	Близорукость	Да	Нормальный	Жесткие

Таблица 5.4 *(окончание)*

Возраст	Предписание	Астигматизм	Степень износа	Рекомендации
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.

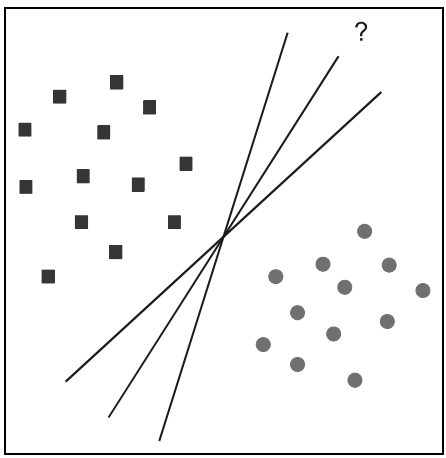

Рис. 5.5. Варианты линейного разделения обучающей выборки

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 можно формально описать как задачу выбора функции с минимальной степенью ошибки:

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

где F — множество всех возможных функций; $c(y, f(x_i))$ — 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 — 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, вводя положительный сдвиг (возможно, с зависимостью 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; далее первый класс 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:

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

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:

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

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

Ситуация усложняется в случае классификации с числом 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

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

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

Соответственно, минимизируем:

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

5.5.2. Линейные методы. Метод наименьших квадратов

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

$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n = \omega_0 + \sum \omega_j x_j,$$

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

Задача заключается в отыскании таких коэффициентов ω, чтобы удовлетворить условие (5.1). Например, при решении задачи регрессии, используя квадратичную функцию потерь (5.2), и множестве линейных функций F:

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

где $f_i : X \to \Re$.

Необходимо найти решение следующей задачи:

$$\min_{f \in F} R(f) = \min_{\omega \in \Re^n} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} \omega_i f_i(x_i) \right)^2.$$

Вычисляя производную $R(f)$ по ω и вводя обозначение $Y_{ij} := f_i(x_j)$, 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 ω. Рассмотренный пример иллюстрирует поиск оптимальной функции f методом наименьших квадратов.

5.5.2. 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).

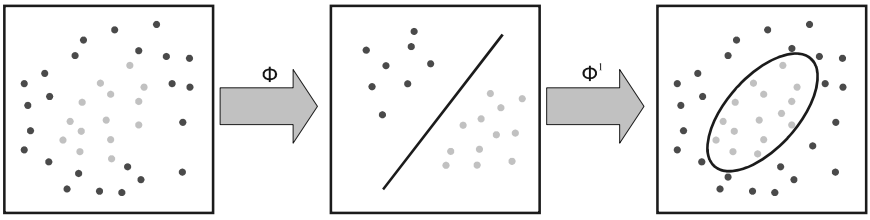

Рис. 5.6. Графическая интерпретация прямого и обратного преобразований из линейного пространства в нелинейное

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 такая зависимость не является линейной. В связи с этим построение нелинейных моделей с таким подходом будет неэффективным. Альтернативой ему может служить метод Support Vector Machines (SVM), не выполняющий отдельные преобразования всех объектов, а учитывающий это в расчетах.

5.5.3. 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 функцией $f(x)$), равноудаленная от точек, принадлежащих разным классам. Для двумерного пространства эту идею можно представить в виде, изображенном на рис. 5.7.

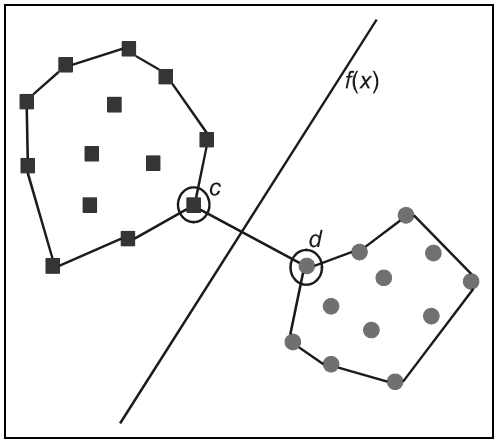

Рис. 5.7. Графическая интерпретация идеи метода SVM

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:

$$\begin{cases} < \omega, x > + b - y_i \le \varepsilon, \\ y_i - < \omega, x > - b \le \varepsilon \end{cases}$$

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

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

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

где $<\omega, x>$ — скалярное произведение векторов ω и x; b — константа, заменяющая коэффициент ω_0.

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

$$\|\omega\| = \sqrt{\langle \omega, \omega \rangle}.$$

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

$$\begin{cases} < \omega, x > + b - y_i \le \varepsilon \\ y_i - < \omega, x > - b \le \varepsilon \end{cases}.$$

Решением данной задачи является функция вида:

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

где α_i и α_i^* — положительные константы, удовлетворяющие следующим условиям:

$$\begin{cases} \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}.$$

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

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:

$$f(x) = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^* \right) k(x_i, x) + b .$$

Отличие от линейного варианта SVM здесь в том, что ω теперь находится не непосредственно, а с использованием преобразования Ф. 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.

5.6. Карта Кохонена

Иногда возникает задача анализа данных, которые с трудом можно представить в математической числовой форме. Это случай, когда нужно извлечь данные, принципы отбора которых заданы нечетко: выделить надежных партнеров, определить перспективный товар и т. п. Рассмотрим типичную для задач подобного рода ситуацию — предсказание банкротств. Предположим, что имеется информация о деятельности нескольких десятков банков (их открытая финансовая отчетность) за некоторый период времени. По окончании этого периода известно, какие из этих банков обанкротились, у каких отозвали лицензию, а какие продолжают стабильно работать (на момент окончания периода). И теперь необходимо решить вопрос о том, в каком из банков стоит размещать средства. Естественно, маловероятно желание разместить средства в банке, который может скоро обанкротиться. Значит, надо каким-либо образом решить задачу анализа рисков вложений в различные коммерческие структуры.

На первый взгляд, решить эту проблему несложно — ведь имеются данные о работе банков и результат их деятельности. Однако данная задача не так проста, поскольку имеющиеся сведения описывают прошедший период, а интерес представляет то, что будет в дальнейшем. Таким образом, на основании имеющихся у нас априорных данных необходимо получить прогноз на дальнейший период. Для решения этой задачи можно использовать различные методы.

Так, например, наиболее очевидным является применение методов математической статистики. Но при этом возникает проблема с количеством данных, ибо статистические методы хорошо работают при большом объеме априорных данных, а их в конкретном случае может оказаться недостаточно. При этом статистические методы не могут гарантировать успешный результат.

Другой путь решения этой задачи — применение нейронных сетей, которые можно обучить на имеющемся наборе данных. В этом случае в качестве исходной информации используются данные финансовых отчетов различных банков, а в качестве целевого поля — итог их деятельности.

Однако при использовании описанных выше методов результат навязывается, без попытки найти закономерности в исходных данных. В принципе все обанкротившиеся банки похожи друг на друга хотя бы тем, что они обанкротились. Значит, в их деятельности должно быть нечто общее, что привело их к этому итогу. Следовательно, можно попытаться выявить эти закономерности с тем, чтобы использовать их в дальнейшем. Сразу же возникает вопрос о путях хождения данных закономерностей. Если использовать методы статистики, надо определить, какие критерии "похожести" использовать, что может потребовать каких-либо дополнительных знаний о характере задачи.

Однако существует метод, позволяющий автоматизировать все действия по поиску закономерностей — метод анализа с использованием самоорганизующихся *карт Кохонена*. Рассмотрим, как решаются такие задачи и как карты Кохонена находят закономерности в исходных данных. Для общности рассмотрения будем использовать термин объект (например, объектом может быть банк, как в рассмотренном ранее примере, но описываемая методика без изменений подходит для решения и других задач, например: анализа кредитоспособности клиента, поиска оптимальной стратегии поведения на рынке и т. д.).

Каждый объект характеризуется набором различных параметров, которые описывают его состояние. В частности, для данного примера параметрами будут сведения из финансовых отчетов. Эти параметры часто имеют числовую форму или могут быть приведены к ней.

Таким образом, необходимо на основании анализа параметров объектов выделить схожие объекты и представить результат в форме, удобной для восприятия.

Все эти задачи решаются самоорганизующимися картами Кохонена. Рассмотрим подробнее, как они работают. Для простоты будем считать, что объекты имеют 3 признака (на самом деле их может быть любое количество).

Теперь представим, что данные третьего параметра являются координатами объектов в трехмерном пространстве (в том самом пространстве, которое окружает нас в повседневной жизни). Тогда каждый объект можно представить в виде точки в этом пространстве (во избежание проблем с различным масштабом по осям пронормируем все эти признаки в интервал [0, 1] любым подходящим способом), в результате чего все точки попадут в куб единичного размера. Отобразим эти точки (рис. 5.8).

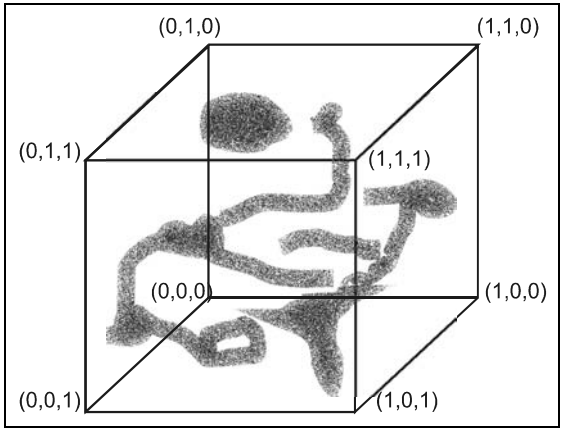

Рис. 5.8. Расположение объектов в пространстве

Из рисунка видно, как расположены объекты в пространстве, причем легко заметить участки, где объекты группируются, т. е. у них схожи параметры, а значит, и сами эти объекты, скорее всего, принадлежат одной группе. Но так легко можно поступить только в случае, когда признаков немного (попробуйте, например, изобразить четырехмерное пространство). Значит, необходимо найти способ, которым можно преобразовать данную систему в простую для восприятия, желательно двумерную систему (трехмерную картинку невозможно корректно отобразить на плоскости бумажного листа) так, чтобы соседние в искомом пространстве объекты оказались рядом и на полученной картинке. Для этого используем самоорганизующуюся карту Кохонена. В первом приближении ее можно представить в виде сети, изготовленной из резины (рис. 5.9).

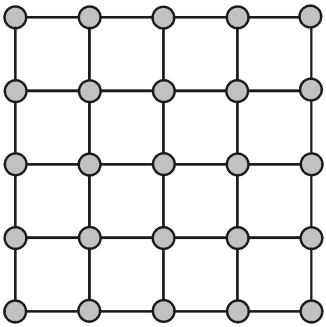

Рис. 5.9. Карта Кохонена

Эту сеть, предварительно "скомканную", бросаем в пространство признаков, где уже имеются объекты, и далее поступаем следующим образом: берем один объект (точку в этом пространстве) и находим ближайший к нему узел сети. Далее этот узел подтягивается к объекту (т. к. сетка "резиновая", то вместе с этим узлом так же, но с меньшей силой подтягиваются и соседние узлы). Затем выбирается другой объект (точка), и процедура повторяется. В результате получим карту, расположение узлов которой совпадает с расположением основных скоплений объектов в исходном пространстве. Кроме того, полученная карта обладает следующим замечательным свойством — узлы ее расположились таким образом, что объектам, похожим между собой, соответствуют соседние узлы карты (рис. 5.10). Теперь определяем, в какие узлы карты попали объекты. Это также определяется ближайшим узлом — объект попадает в тот узел, который находится ближе к нему. В результате описанных операций объекты со схожими параметрами попадут в один узел или в соседние узлы. Таким образом, можно считать, что решена задача поиска похожих объектов и их группировки.

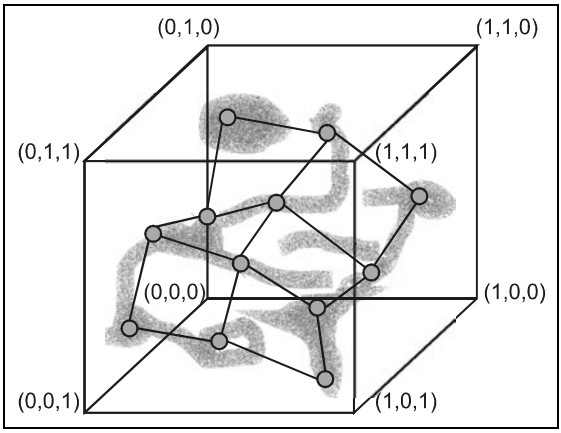

Рис. 5.10. Вид пространства после наложения карты

Но на этом возможности карт Кохонена не заканчиваются. Они позволяют также представить полученную информацию в простой и наглядной форме путем нанесения раскраски. Для этого раскрашиваем полученную карту (точнее, ее узлы) цветами, соответствующими интересующим нас признакам объектов. Возвращаясь к примеру с классификацией банков, можно раскрасить одним цветом те узлы, куда попал хоть один из банков, у которых была отозвана лицензия. Тогда после нанесения раскраски получим зону, которую можно назвать зоной риска, и попадание интересующего нас банка в эту зону говорит о его ненадежности.

Но и это еще не все. Можно также получить информацию о зависимостях между параметрами. Нанеся на карту раскраску, соответствующую различным статьям отчетов, можно получить так называемый атлас, хранящий в себе информацию о состоянии рынка. При анализе, сравнивая расположение цветов на раскрасках, порожденных различными параметрами, можно получить полную информацию о финансовом портрете банков-неудачников, процветающих банков и т. д.

При всем этом описанная технология является универсальным методом анализа. С ее помощью можно анализировать различные стратегии деятельности, производить анализ результатов маркетинговых исследований, проверять кредитоспособность клиентов и т. д.

Таким образом, имея перед собой карту и владея информацией о части исследуемых объектов, можно достаточно достоверно судить о малознакомых объектах. Нужно узнать, что на самом деле представляет собой новый партнер? Отобразим его на карте и посмотрим на соседей. В результате можно извлекать информацию из базы данных, основываясь на нечетких характеристиках.

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 алгоритма 1R 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.

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:

$$T = \{i_j \mid i_j \in I\}.$$

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:

$$D_{i_j} = \{T_r \mid i_j \in T_r; j = 1..n; r = 1..m\} \subseteq D .$$

В данном примере множеством транзакций, содержащих объект `вода`, является следующее множество:

$$D_{вода} = \{\{\text{чипсы, вода, пиво}\},$$
$$\{\text{кокосы, вода, орехи}\},$$
$$\{\text{орехи, кокосы, чипсы, кокосы, вода}\}\}.$$

Некоторый произвольный набор объектов (itemset) обозначим следующим образом:

$$F = \{i_j \mid i_j \in I; j = 1..n\} .$$

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:

$$D_F = \{T_r \mid F \subseteq T_r; r = 1..m\} \subseteq D .$$

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 Supp(F):

$$\text{Supp}(F) = \frac{|D_F|}{|D|}.$$

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 $\text{Supp}_{\min}$. Набор называется частым (large itemset), если значение его поддержки больше минимального значения поддержки, заданного пользователем:

$$\text{Supp}(F) > \text{Supp}_{\min}.$$

Таким образом, при поиске ассоциативных правил требуется найти множество всех частых наборов:

$$L = \{F \mid \text{Supp}(F) > \text{Supp}_{\min}\}.$$

В данном примере частыми наборами при $\text{Supp}_{\min} = 0{,}5$ являются следующие:

{чипсы} $\text{Supp}_{\min} = 0{,}5$;
{чипсы, вода} $\text{Supp}_{\min} = 0{,}5$;
{кокосы} $\text{Supp}_{\min} = 0{,}75$;
{кокосы, вода} $\text{Supp}_{\min} = 0{,}5$;
{кокосы, вода, орехи} $\text{Supp}_{\min} = 0{,}5$;
{кокосы, орехи} $\text{Supp}_{\min} = 0{,}75$;
{вода} $\text{Supp}_{\min} = 0{,}75$;
{вода, орехи} $\text{Supp}_{\min} = 0{,}5$;
{орехи} $\text{Supp}_{\min} = 0{,}75$.

6.1.2. Сиквенциальный 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:

$$S = \{..., i_p, ..., i_q, ...\}, \text{ где } q < p.$$

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:

$$S = \{..., i_p, ..., i_q, ...\}, \text{ где } q < p, \text{ а } i_q = i_p.$$

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 сиквенциального анализа является поиск всех частых последовательностей:

$$L = \{S \mid \text{Supp}(S) > \text{Supp}_{\min}\}.$$

Основным отличием задачи сиквенциального 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).

Сиквенциальный 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_{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).

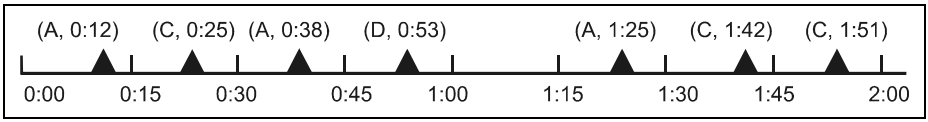

Рис. 6.1. Последовательность сбоев на телекоммуникационной станции

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.

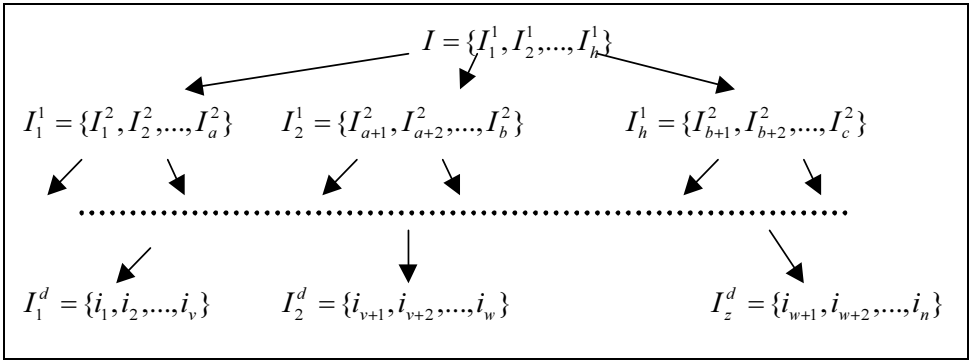

Рис. 6.2. Иерархическое представление объектов множества *I*

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 $F = \{I^g \mid I^g \in I^{g+1}\}$, можно рассматривать

также смешанные наборы объектов и групп $F = \{i, I^g \mid i \in I^g, \in I^{g+1}\}$. Это позволяет расширить анализ и получить дополнительные знания.

При иерархическом построении объектов можно варьировать характер поиска изменяя анализируемый уровень. Очевидно, что чем больше объектов в 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 $X => Y$ 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

$$\mathrm{Supp}_{\text{ если (вода, кокосы) то (орехи)}} = \mathrm{Supp}_{\text{ \{вода, кокосы, орехи\}}} = 1/2.$$

Достоверность (confidence) — показывает вероятность того, что из наличия в транзакции набора X следует наличие в ней набора Y. Достоверностью правила $X => Y$ 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 $X => Y$. 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 $\text{Supp}_F > \text{Supp}_{\min}$:

$$L_k = \{(F_1, \text{Supp}_1), (F_2, \text{Supp}_2), \ ..., (F_q, \text{Supp}_q)\},$$

  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 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 $Supp_{min}$.

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 ($p.item_{k-1} < q.item_{k-1}$). При этом первые все $k-2$ элемента обоих наборов одинаковы ($p.item_1 = q.item_1$, $p.item_2 = q.item_2$, ..., $p.item_{k-2} = q.item_{k-2}$).

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 $Supp_{min}$. 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 $Supp_{min} = 0{,}5$. 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 <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:

$$<\text{T.TID}, \{c \in C_k \mid c \in T\}>.$$

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 сиквенциального 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.
- Сиквенциальный анализ заключается в поиске частых последовательностей. Основным отличием задачи сиквенциального 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 годов. 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).

Кластеризация отличается от классификации тем, что для проведения анализа не требуется иметь выделенную целевую переменную, с этой точки зрения 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 задачей (descriptive).

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

Большое достоинство кластерного анализа в том, что он позволяет производить 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.

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, четко количественно очерченные объекты, а с другой — объекты, имеющие вероятностное или нечеткое описание.

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

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 отдель-

ными 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 о ирисах, с которыми в середине 30-х годов прошлого столетия работал известный статист 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.

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 σ — величина, определяющая меру близости для включения объектов в один кластер; $d(i_j, i_p)$ — мера близости между объектами, называемая расстоянием.

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

Неотрицательное значение $d(i_j, i_p)$ называется расстоянием между элементами i_j и i_p, если выполняются следующие условия:

а) $d(i_j, i_p) \geq 0$, для всех i_j и i_p;

б) $d(i_j, i_p) = 0$, тогда и только тогда, когда $i_j = i_p$;

в) $d(i_j, i_p) = d(i_j, i_p)$;

г) $d(i_j, i_p) \leq d(i_j, i_r) + d(i_r, i_p)$.

Если расстояние $d(i_j, i_p)$ меньше некоторого значения σ, 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 ниже 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;
- $x_i = (x_{i1}, \ldots, x_{im}) \in X_Q$, $i = \overline{1, Q}$ — 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*):

$$d_2(x_i, x_j) = \sqrt{\sum_{t=1}^{m}(x_{it} - x_{jt})^2}\ . \tag{7.1}$$

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

$$d_H(x_i, x_j) = \sum_{t=1}^{m} | x_{it} - x_{jt} |\ . \tag{7.2}$$

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

$$d_\infty(x_i, x_j) = \max_{1 \le t \le m} | x_{it} - x_{jt} |\ . \tag{7.3}$$

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:

$$d_M(x_i, x_j) = (x_i - x_j)S^{-1}(x_i - x_j)^t\ . \tag{7.4}$$

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.

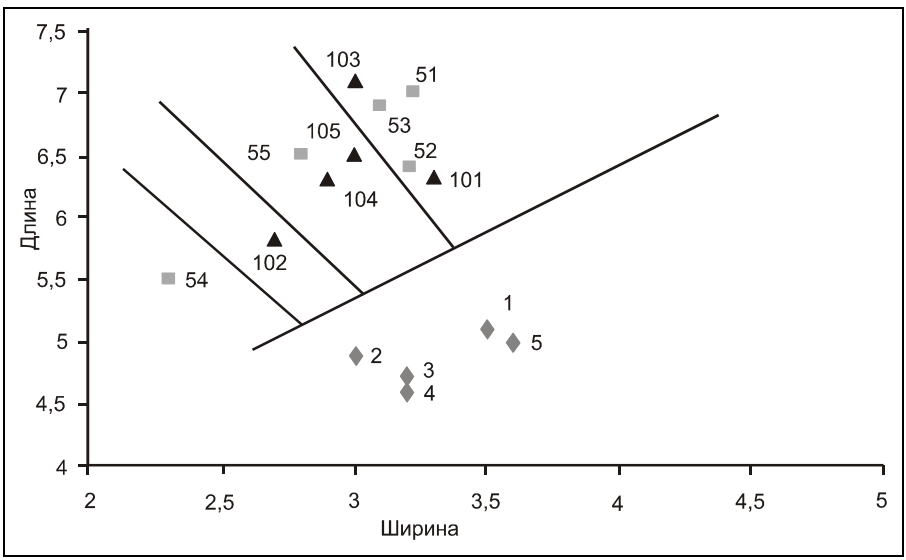

Рис. 7.1. Разделение ирисов на кластеры линиями

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.

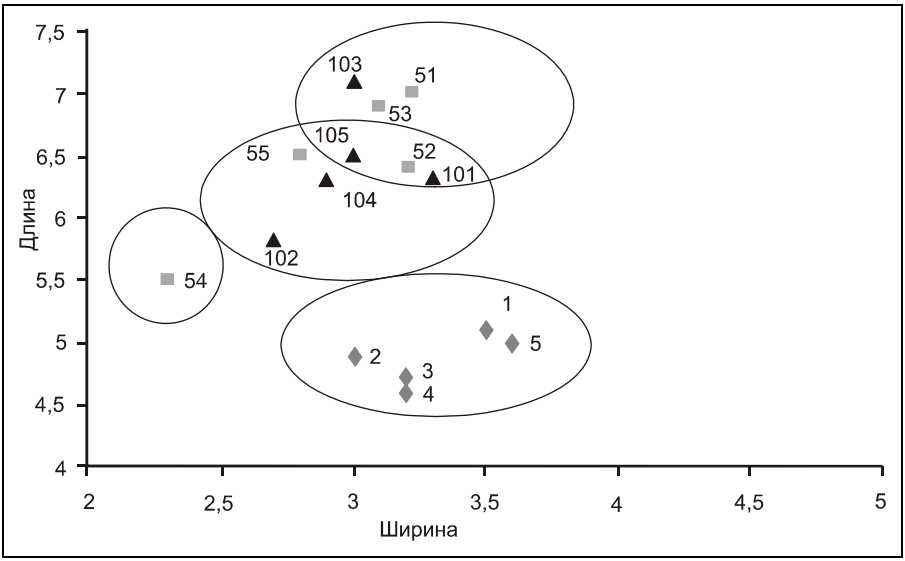

Рис. 7.2. Разделение ирисов на кластеры с использованием Венских диаграмм

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.

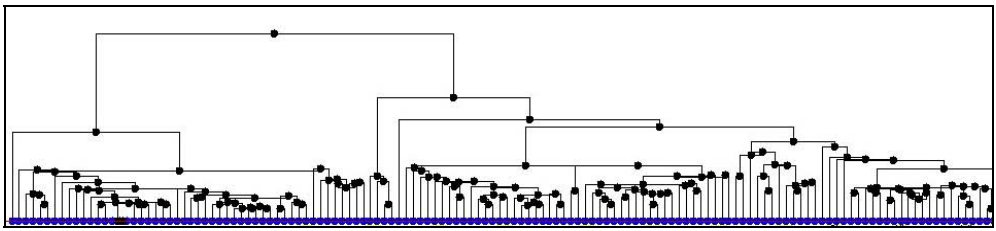

Рис. 7.3. Дендограмма, построенная для данных из табл. 7.1

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 *дендограммами* (dendrograms). Этот термин подчеркивает древовидную структуру диаграмм (от греч. *dendron* — дерево).

Существует много способов построения дендограмм. Для примера с ирисами дендограмма 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 признаков, 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:

а) на агломеративные, характеризуемые последовательным объединением исходных элементов и соответствующим уменьшением числа кластеров (построение кластеров снизу вверх);

б) 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:

$$c_1 = \{i_1\},\ c_2 = \{i_2\},\ \ldots,\ c_m = \{i_m\}.$$

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:

$$c_1 = \{i_1\},\ c_2=\{i_2\},\ \ldots,\ c_p = \{i_p,\ i_q\},\ \ldots,\ c_m = \{i_m\}.$$

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 α_p, α_q, β и γ.

В табл. 7.2 приведены коэффициенты пересчета расстояний между кластерами α_p, α_q, β и γ.

Таблица 7.2

Название метода	α_p	α_q	β	γ
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
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 году.

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 предыдущем уровне. Выбор расщепляемого кластера может выполняться поразному.

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. 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.

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.

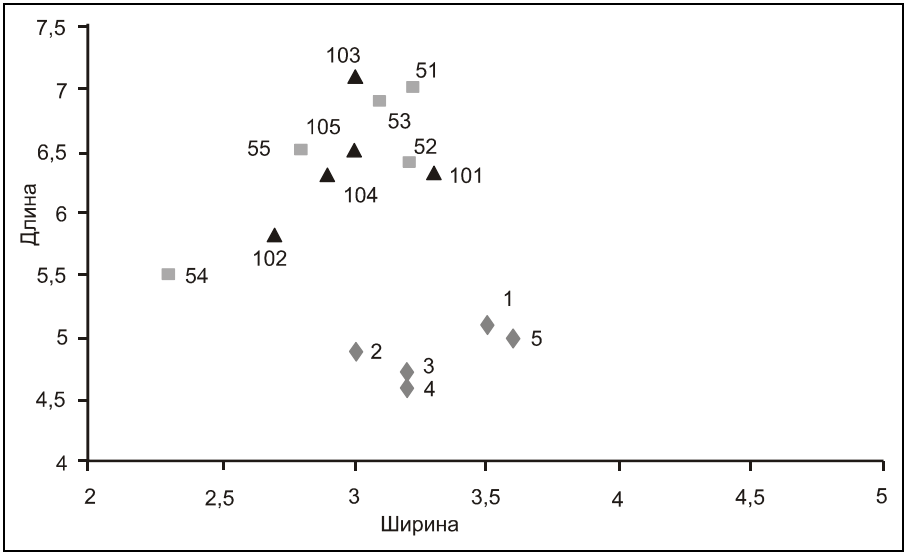

Рис. 7.4. Представление ирисов в двумерном пространстве

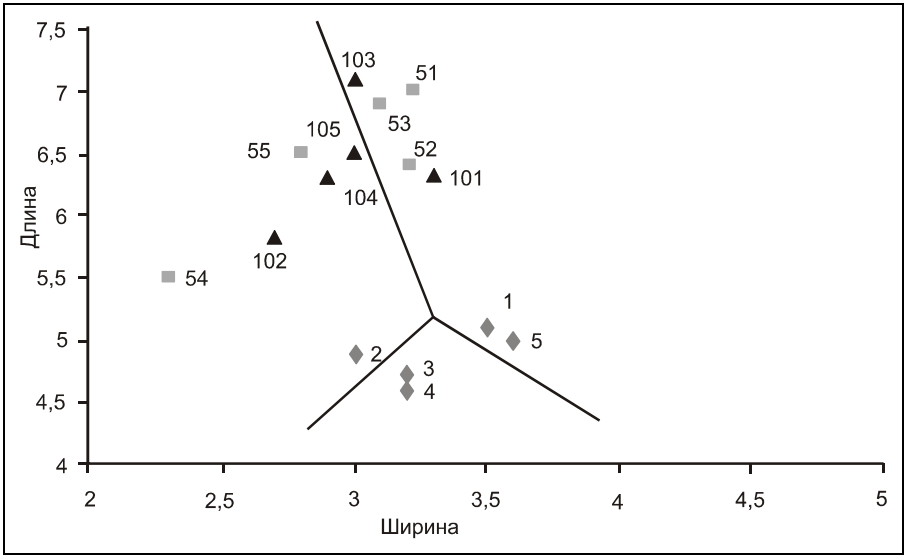

Рис. 7.5. Начальное разбиение ирисов на три кластера

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 преды-

дущих. 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.

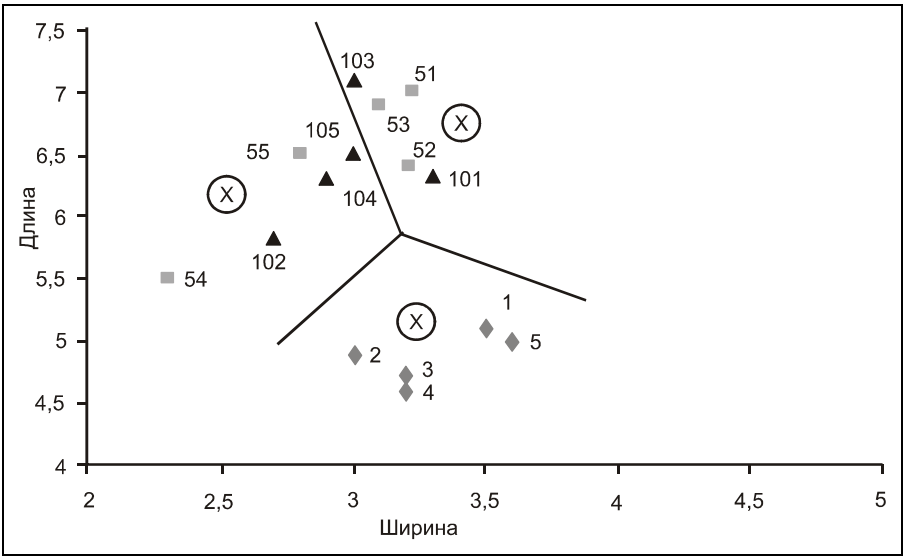

Рис. 7.6. Второе разбиение множества ирисов на кластеры

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 среднее значение этой переменной, на диапазон ее изменения (разность между максимальным и минимальным значениями переменной), или на ее стандарт-

ное 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.

В заключение 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 рассмотрение 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 δ (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 δ.

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:

$$u_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_A^2(m_j, c^{(i)})}{d_A^2(m_j, c^{(k)})} \right)^{\frac{1}{w-1}}} \text{ для всех } 1 \le i \le c, \quad 1 \le j \le d\,, \tag{7.19}$$

учитывая ограничения из формулы (7.16).

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.

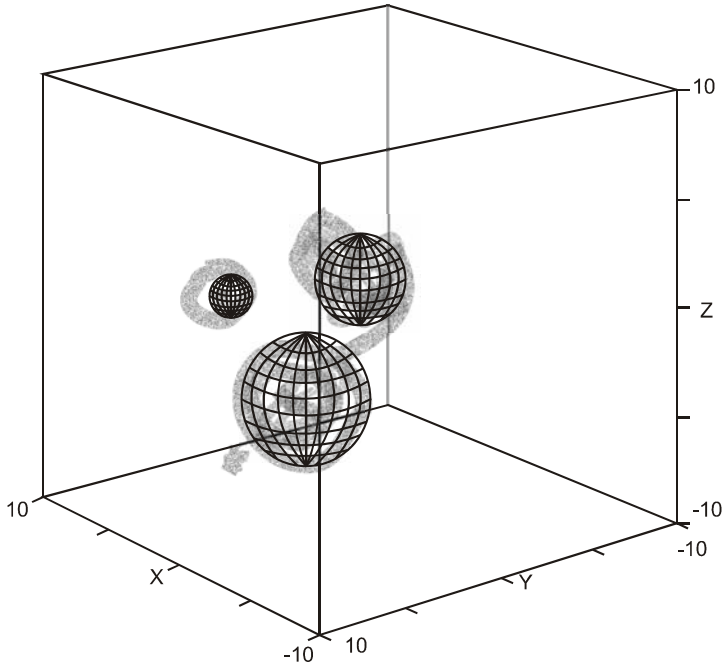

Рис. 7.7. Форма кластеров в алгоритме Fuzzy C-Means

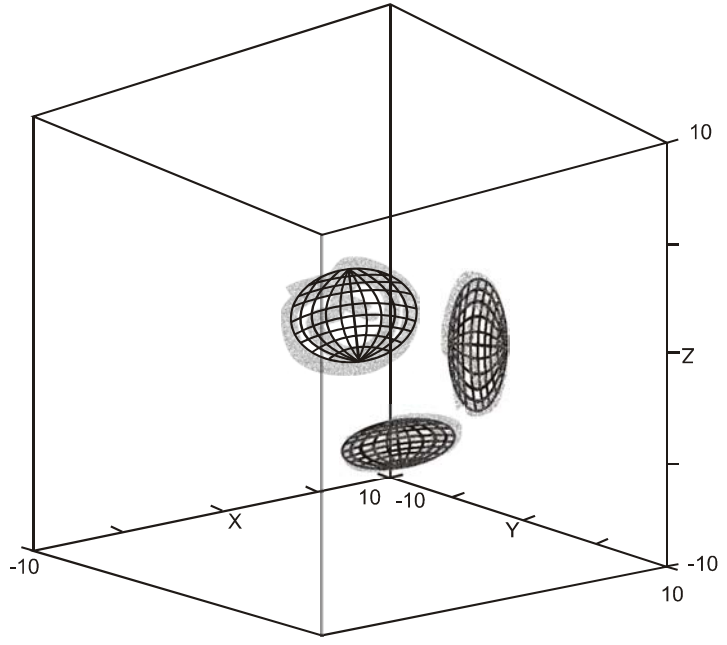

Рис. 7.8. Форма кластера в алгоритме кластеризации по Гюстафсону-Кесселю

Кластер характеризуется не только своим центром, но и ковариационной матрицей:

$$F^{(i)} = \frac{\sum_{j=1}^{d} (u_{ij})^w (m_j - c^{(i)})(m_j - c^{(i)})^t}{\sum_{j=1}^{d} (u_{ij})^w}, \qquad (7.20)$$

где λ_{ik} обозначает k-е собственное значение матрицы $F^{(i)}$, а Φ_{ik} — k-й единичный собственный вектор $F^{(i)}$. Собственные значения λ_{ik} упорядочены в порядке убывания. Тогда собственные векторы $\Phi_{i1}...\Phi_{i(n-1)}$ охватывают линейное подпространство i-го кластера, а Φ_{in} является нормалью к этому линейному подпространству. Все изложенное иллюстрирует рис. 7.9.

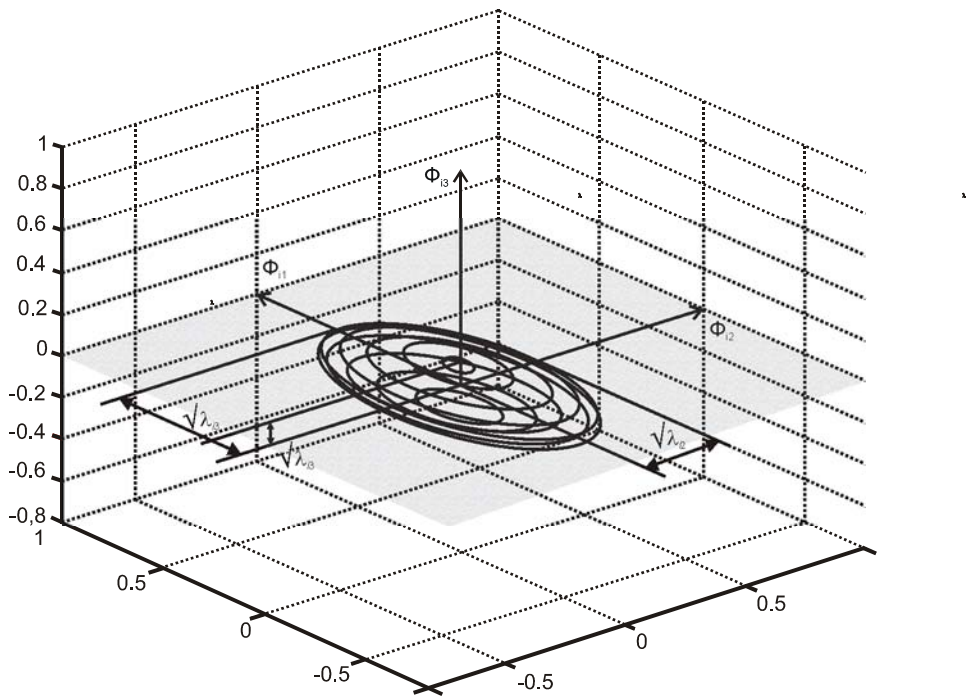

Рис. 7.9. Геометрическая иллюстрация к алгоритму кластеризации по Гюстафсону-Кесселю

Данный алгоритм использует свою нормирующую матрицу для вычисления расстояния. Выражения для вычисления выглядят следующим образом:

$$d^2_{A^{(i)}} = \left(c^{(i)} - m_j\right)^t A^{(i)} \left(c^{(i)} - m_j\right), \qquad (7.21)$$

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 оче-

видным 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 распределения 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 ка-

ких 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. Кластеризация данных при помощи нечетких отношений

7.4.1. Анализ свойств нечетких бинарных отношений применительно к анализу данных

Отношения и свойства отношений

Рассмотрим набор непустых множеств X_1, ..., X_k.

Определение 1 — *нечеткое k-арное гетерогенное отношение R* есть нечеткое подмножество декартова произведения $X_1 \times ... \times X_k$. Другими словами, $R \in F(X_1 \times ... \times X_k)$.

Определение 2 — *нечеткое k-арное гомогенное отношение R* есть нечеткое k-арное гетерогенное отношение при $X_1 = ... = X_k = X$. Другими словами, $R \in F(X^k)$.

Определение 3 — *нечеткое гетерогенное (гомогенное) бинарное отношение R* называется бинарным при условии, что $k = 2$. Нечеткое гетерогенное бинарное отношение $R \in F(X_1 \times X_2)$. Нечеткое гомогенное бинарное отношение $R \in F(X^2)$.

Замечание 1. Важным отличием нечетких отношений от классических теоретико-множественных отношений является определение характеристической функции $\mu_X(x)$ множеств. В случае классической теории множеств характеристическая функция принимает одно из двух допустимых дискретных значений, смысл которых в идентификации принадлежности/непринадлежности данного элемента универсального множества множеству X. Теория не-

четких множеств дает обобщение характеристической функции множества, которая носит название функции принадлежности. В теории нечетких множеств элемент универсального множества принадлежит заданному множеству X с определенной степенью принадлежности, принимающей значения из интервала [0, 1]. Обобщая изложенное, можно записать:

классическая теория множеств	теория нечетких множеств
$\mu_X(x): U \to \{0, 1\}; \quad X \subseteq U$	$\mu_X(x): U \to [0, 1]; \quad X \subseteq U$.
$\mu_X(x) = \begin{cases} 1, & \text{если } x \in X \\ 0, & \text{если } x \notin X \end{cases}$	Чем больше $\mu_X(x)$, тем больше степень принадлежности элемента x нечеткому множеству X, и наоборот.

Важным аспектом в теории нечетких множеств является обобщение свойств отношений для случая нечетких множеств. Напомним определения свойств отношений, которые даются в классической теории множеств (обратите внимание, что определения даются для гомогенных бинарных отношений $R \subseteq X^2$).

Определение 4 — *рефлексивность.* Отношение R называется рефлексивным, если для $\forall x \in X$ выполняется $(x, x) \in R$. В этом случае говорят, что R обладает свойством рефлексивности.

Определение 5 — *антирефлексивность.* Отношение R называется антирефлексивным, если $\forall x \in X$ выполняется $(x, x) \notin R$. В этом случае говорят, что R обладает свойством антирефлексивным.

Замечание 2. Хотя свойства рефлексивности и антирефлексивности обладают взаимоисключающим характером, т. е. отношение не может быть одновременно рефлексивным и антирефлексивным, тем не менее, отношение может одновременно не обладать ни одним из указанных свойств. В этом случае $\exists x_1 \neq x_2 \in X$ такие, что $(x_1, x_1) \in R$ и $(x_2, x_2) \notin R$.

Определение 6 — *симметричность.* Отношение R называется симметричным, если при условии $(x_1, x_2) \in R$ выполняется $(x_2, x_1) \in R$ для $\forall x_1, x_2 \in X$. В этом случае говорят, что отношение R обладает свойством симметричности.

Определение 7 — *антисимметричность.* Отношение R называется антисимметричным, если при условии $(x_1, x_2) \in R$ и $(x_2, x_1) \in R$ выполняется $x_1 = x_2$ для $\forall x_1, x_2 \in X$. В этом случае говорят, что отношение R обладает свойством антисимметричности.

Замечание 3. Хотя свойства симметричности и антисимметричности обладают взаимоисключающим характером, т. е. отношение не может быть одно-

временно симметричным и антисимметричным, тем не менее, отношение может одновременно не обладать ни одним из указанных свойств.

Определение 8 — *транзитивность.* Отношение R называют транзитивным, если для $\forall x_1, x_2, x_3 \in X$ при $(x_1, x_2) \in R$ и $(x_2, x_3) \in R$ выполняется $(x_1, x_3) \in R$. В этом случае говорят, что отношение R обладает свойством транзитивности.

Определение 9 — *толерантность.* Отношение R называют отношением толерантности, если оно рефлексивно и симметрично.

Определение 10 — *эквивалентность.* Отношение R называется отношением эквивалентности, если оно рефлексивно, симметрично и транзитивно.

Определение 11 — *частичный порядок.* Отношение R называется отношением частичного порядка, если оно рефлексивно, антисимметрично и транзитивно.

Определение 12 — *полный порядок.* Отношение R называется отношением полного порядка, если оно является отношением частичного порядка и при этом $(x_1, x_2) \in R$ либо $(x_2, x_1) \in R$ для $\forall x_1, x_2 \in X$.

Свойства нечетких отношений.

Обобщим данные свойства гомогенных бинарных отношений для случая нечетких отношений, т. е. $R \in F(x^2)$.

Определение 13 — *нечеткая рефлексивность и антирефлексивность*.

• **Определение 13.1** — *квазирефлексивность*. Нечеткое отношение R называют квазирефлексивным, если для $\forall x \in X$ выполняется $\mu_R((x,x)) > 0$. В этом случае говорят, что R обладает свойством квазирефлексивности.

• **Определение 13.2** — α-*рефлексивность*. Назовем нечеткое отношение α-рефлексивным, если $\mu_R((x,x)) \geq \alpha, \quad \alpha > 0$.

• **Определение 13.3** — *равномерная* α-*рефлексивность*. Равномерным α-рефлексивным нечетким отношением будем называть такое нечеткое отношение, для которого $\mu_R((x,x)) = \alpha, \quad \alpha > 0 \quad \forall x \in X$.

• **Определение 13.4** — *рефлексивность*. Если $\mu_R((x,x)) = 1$ нечеткое отношение R будем называть четко рефлексивным, или просто рефлексивным $\forall x \in X$.

• **Определение 13.5** — *антирефлексивность*. Если $\mu_R((x,x)) = 0 \quad \forall x \in X$ нечеткое отношение R будем называть четко антирефлексивным, или просто антирефлексивным.

Замечание 4. Исходя из подобного определения нечеткого отношения рефлексивности можно записать следующее выражение, описывающее взаимосвязь свойств рефлексивности и антирефлексивности нечетких отношений на множестве X:

$$\mu_X^{\text{рефлексивность}} + \mu_X^{\text{антирефлексивность}} = 1. \qquad (7.31)$$

То есть, являясь в какой-то степени рефлексивным, отношение в то же время является и антирефлексивным, и наоборот.

Определение 14 — *нечеткая симметричность и антисимметричность*.

• **Определение 14.1** — *квазисимметричность*. Нечеткое отношение R называют квазисимметричным, если $\forall x_1, x_2 \in X$ при $\mu_R((x_1, x_2)) > 0$ выполняется $\mu_R((x_2, x_1)) > 0$.

• **Определение 14.2** — *нормальная квазисимметричность*. Нечеткое отношение R называется нормальным квазисимметричным отношением, если $\forall x_1, x_2 \in X$ при $\mu_R((x_1, x_2)) > 0$ выполняется $\mu_R((x_1, x_2)) = \mu_R((x_2, x_1)) > 0$.

• **Определение 14.3** — *α-квазисимметричность*. Нечеткое отношение R называют α-квазисимметричным, если $\forall x_1, x_2 \in X$ при $\mu_R((x_1, x_2)) \geq \alpha$ выполняется $\mu_R((x_2, x_1)) \geq \alpha$ и $\alpha > 0$.

• **Определение 14.4** — *нормальная α-симметричность*. Нечеткое отношение R называют нормальным α-симметричным нечетким отношением, если $\forall x_1, x_2 \in X$ при $\mu_R((x_1, x_2)) \geq \alpha$ выполняется $\mu_R((x_1, x_2)) = \mu_R((x_2, x_1)) \geq \alpha$ и $\alpha > 0$.

• **Определение 14.5** — *равномерная нормальная α-симметричность*. Нечеткое отношение R называют равномерным нормальным α-симметричным нечетким отношением, или просто α-симметричным нечетким отношением, если $\forall x_1, x_2 \in X$ при $\mu_R((x_1, x_2)) = \alpha$ выполняется $\mu_R((x_1, x_2)) = \mu_R((x_2, x_1)) = \alpha$ и $\alpha > 0$.

• **Определение 14.6** — *симметричность*. Нечеткое отношение R называют четко симметричным, или просто симметричным, если при $\mu_R((x_1, x_2)) = 1$ выполняется $\mu_R((x_2, x_1)) = 1 \ \ \forall x_1, x_2 \in X$.

• **Определение 14.7** — *антисимметричность*. Нечеткое отношение R называется нечетким антисимметричным отношением, если при $\mu_R((x_1, x_2)) > 0$ и $\mu_R((x_2, x_1)) > 0 \ \ x_1 = x_2 \ \ \forall x_1, x_2 \in X$.

Замечание 5. Исходя из подобного определения нечеткого отношения рефлексивности можно записать следующее выражение, описывающее взаимосвязь свойств рефлексивности и антирефлексивности нечетких отношений на множестве X:

$$\mu_X^{\text{симметричность}} + \mu_X^{\text{антисимметричность}} = 1. \qquad (7.32)$$

То есть, являясь в какой-то степени симметричным, отношение в то же время является и антисимметричным, и наоборот.

Замечание 6. (о нечетком отношении толерантности). Поскольку отношение толерантности должно включать в себя нечеткое отношение симметричности и нечеткое отношение рефлексивности, возможны различные варианты отношения толерантности — от минимально заданного до варианта, существующего в классической теории множеств. Учитывая определения 13 и 14, можно построить целое семейство нечетких отношений толерантности, которые будут получаться не только комбинированным включением в определение различных вариантов свойств рефлексивности и симметричности, но и взаимным отношением параметров, используемых в соответствующих определениях. В данном семействе будут присутствовать отношения толерантности различной степени четкости, тем не менее, с точки зрения данной работы, особый интерес будет представлять следующий вид нечеткого отношения толерантности.

Определение 15 — *нечеткая* α*-толерантность*. Нечеткое отношение R, обладающее свойствами четкой рефлексивности и нормальной α-симметричности, назовем нечетким отношением α-толерантности.

Напомним свойства отношения нечеткой α-толерантности, которые получаются из свойств отношений четкой рефлексивности и нечеткой нормальной α-симметричности:

$$\mu_R((x,x)) = 1 \quad \forall x \in X .$$

При условии $\mu_R((x_1,x_2)) \geq \alpha$ выполняется $\mu_R((x_1,x_2)) = \mu_R((x_2,x_1)) \geq \alpha$ при $\alpha > 0$ и $\forall x_1, x_2 \in X$.

Замечание 7. Поскольку отношение толерантности имеет смысл нетранзитивного подобия, оно является особенно ценным в задаче первичного анализа данных, т. к., с одной стороны, устанавливаются степени схожести данных, а с другой — не требуется транзитивность, которая превращает отношение толерантности в отношение эквивалентности, что в случае нечеткого анализа накладывает слишком жесткие условия на взаимосвязь данных, о которых по постановке задачи ничего неизвестно. Выбор именно такого варианта нечет-

кого отношения толерантности обусловлен следующими объективными допущениями:

- каждый экземпляр исследуемых данных полностью подобен самому себе;
- степень подобия элемента *a* элементу *b* равна степени подобия элемента *b* элементу *a*.

Покажем, как построить подобное отношение на множестве исследуемых данных. Для начала определимся с понятием входных данных, которое используется в поставленной задаче.

Сравнение данных

Определение 16 — *образец данных*. *n*-атрибутным образцом данных называется вектор, состоящий из *n* элементов, называемых также атрибутами, представленных вещественными значениями. Другое название для образца данных — точка данных, или вектор данных.

Будем оценивать степень подобия образцов данных понятием расстояния между ними *d*.

Пусть дано *Q* образцов данных (множество X_Q), имеющих по *n*-атрибутов. Построим, используя расстояние *d*, семейство нечетких множеств, которые можно назвать "точки, схожие с точкой *q*". Степень принадлежности элементов данных множеств будет показывать, насколько образец данных "схож" с предъявляемым образцом данных.

Определение 17 — *мера сходства по расстоянию*. Мерой сходства по расстоянию с образцом данных x_0 назовем функцию $\mu_{x_0} : X \to [0, 1] \quad x_0 \in X$, которая определяется по формуле:

$$\mu_{x_0}(x) = 1 - \frac{d(x_0, x)}{K}, \tag{7.33}$$

где *K* — постоянный коэффициент, выбираемый в соответствии с ограничениями на область значений функции $\mu_{x_0}(x)$.

Замечание 8. Из определения меры сходства по расстоянию следует, что μ_{x_0} — линейно убывающая по расстоянию *d* функция, а $\mu_{x_0}(x_0) = 1$.

В дальнейшем для краткости будем называть функцию μ_{x_0} мерой сходства.

Определение 18 — *нормальной мерой сходства* по расстоянию с образцом данных x_0 назовем такую меру, которая достигает своих граничных значений на множестве *X*.

С учетом определений 17 и 18 функции принадлежности вводимых в рассмотрение нечетких множеств можно называть мерами сходства соответствующих точек данных.

Очень важное значение при определении данных нечетких множеств имеет конструктивная процедура определения меры сходства. Даже при помощи одного понятия расстояния между образцами данных можно по-разному определить указанную меру сходства, по-разному выбирая коэффициент K из определения 17. Учитывая это, определения 16 и 17 и замечание 8 можно представить по крайней мере два способа конструктивного определения значений функции принадлежности:

$$1) \quad \mu_{x_q}(x_i) = 1 - \frac{d(x_q, x_i)}{\max\limits_{k \in [1,Q]} (d(x_q, x_k))}, \tag{7.34}$$

и

$$2) \quad \mu_{x_q}(x_i) = 1 - \frac{d(x_q, x_i)}{\max\limits_{\substack{k \in [1,Q] \\ q \in [1,Q]}} (d(x_q, x_k))}. \tag{7.35}$$

Обе формулы удовлетворяют определению меры сходства, причем первая из них в соответствии с определением 18 является нормальной мерой сходства. Для дальнейшего анализа потребуется именно нормальная мера сходства. Ее достоинства заключаются в гарантии, что для каждого образца данных x_i существует по крайней мере один образец данных, который абсолютно схож с x_i (мера сходства при этом достигает значения 1). Таким образцом является сам образец данных x_i. Одновременно с этим для каждого образца данных x_i существует по крайней мере один образец данных, максимально отличающийся от x_i (мера сходства при этом достигает значения 0). С другой стороны, данная мера сходства не учитывает расстояния между остальными точками (расстояния, отличные от расстояний $d(x_q, x_k)$, q — заданная точка, $k = \overline{1,Q}$), и в этом смысле ее можно назвать относительной.

Вторая мера сходства сравнивает образцы данных относительно двух взаимно удаленных точек, что усложняет интерпретацию значений данной меры сходства, когда необходимо узнать, какая точка является наиболее близкой к данной, а какая наиболее удаленной.

Пример выбора нормирующего коэффициента K для случаев 1 и 2 при построении нечеткого "множества точек, близких к x_1" показан на рис. 7.10. Для случая 1 выбирается коэффициент K, а для случая 2 — K'.

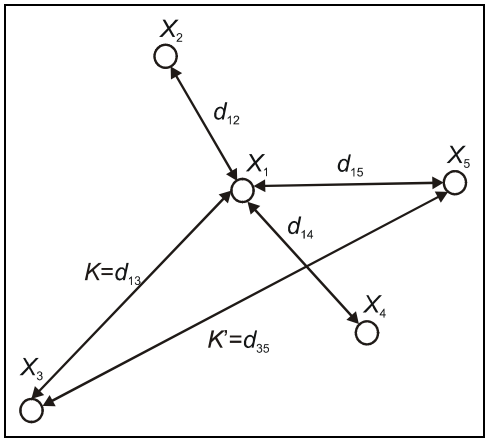

Рис. 7.10. Выбор нормирующего коэффициента для определения меры сходства по расстоянию

Подведем итоги:

- определен способ построения Q нечетких множеств, имеющих название "множество точек, близких к точке q";
- в каждом множестве есть по крайней мере один элемент, обладающий нулевой степенью принадлежности данному множеству, и по крайней мере один элемент, обладающий единичной степенью принадлежности данному множеству (соответствующая точка q);
- функции принадлежности построенных нечетких множеств являются нормальными мерами сходства относительно соответствующих образцов данных.

Отношение α-толерантности

На основании полученной меры сходства предлагается построить семейство α-толерантных отношений, смысл которых заключается в степени схожести двух образцов данных относительно некоторого заданного образца. Для упрощения дальнейшего изложения введем следующее определение.

Определение 19 — *относительной мерой сходства двух образцов данных* относительно образца данных x_0 будем называть функцию $\xi_{x_0} : X^2 \to [0, 1]$ $x_0 \in X$, которая определяется как:

$$\xi_{x0}(x_1, x_2) = 1 - |\mu_{x0}(x_1) - \mu_{x0}(x_2)|, \tag{7.36}$$

где $\mu_{x_0}(x)$ — нормальная мера сходства.

Легко показать, что каждое отношение из семейства отношений $\xi_{x_1},...,\xi_{x_Q}$ является α-толерантным.

Теперь покажем, как построить бинарное нечеткое отношение меры сходства образцов данных на множестве X. Для этого запишем нечеткое правило: если $\xi_{x_1}(a,b)$ и ... и $\xi_{x_Q}(a,b)$, то $\xi(a,b)$. Данное правило формализует следующее очевидное высказывание: "если два образца сходны относительно x_1 и ... и сходны относительно X_Q, то два образца данных сходны относительно множества X". Исходя из изложенного, можно ввести следующее определение.

Определение 20 — *мерой сходства образцов данных на множестве* X будем называть функцию $\xi : X^2 \to [0, 1]$, которая определяется как

$$\xi(a,b) = T(\xi_{x_1}(a,b),\ ...,\ \xi_{x_Q}(a,b)), \tag{7.37}$$

где T — t-норма, $\xi_{x_i}(a,b)$ — относительная мера сходства, $x_i \in X$, $i = \overline{1,Q}$, $a,b \in X$. Отношение, порождаемое функцией ξ, назовем отношением сходства образцов данных на множестве X.

Исходя из определения 20 отношение, порождаемое мерой сходства образцов данных на множестве X, также является α-толерантным.

Подведем итоги:

- построено нечеткое отношение α-толерантности, порождаемое мерой сходства из определения 20, которое объективным образом показывает сходство между парами объектов из множества X;
- матрица данного отношения сама по себе несет информацию о взаимосвязи данных и может являться самостоятельным результатом анализа данных;
- задавшись определенным коэффициентом α, отражающим допущение о том, какие образцы данных можно считать сходными, можно перейти к четкому отношению толерантности.

7.4.2. Отношение α-квазиэквивалентности

В классической теории множеств при помощи отношения толерантности можно построить классы эквивалентности на множестве образцов данных X и построить покрытие этого множества. Для этого необходимо найти совокупности транзитивно зависимых образцов данных на отношении толерантности.

Для случая нечетких отношений определить свойство транзитивности, в силу недискретной характеристической функции отношения, можно различными

способами, которые бы учитывали желаемую степень нечеткости проявления свойства транзитивности.

Определение 21 — *квазитранзитивность*. Нечеткое отношение R называют транзитивным, если $\forall x_1, x_2, x_3 \in X$ при $\mu_R((x_1, x_2)) > 0$ и $\mu_R((x_2, x_3)) > 0$ выполняется $\mu_R((x_1, x_3)) > 0$.

Определение 22 — γ-*квазитранзитивность*. Нечеткое отношение R называют γ-квазитранзитивным, если при $\mu_R((x_1, x_2)) \geq \gamma$ и $\mu_R((x_2, x_3)) \geq \gamma$ выполняется $\mu_R((x_1, x_3)) \geq \gamma$ $\forall x_1, x_2, x_3 \in X$ и $\gamma > 0$.

Замечание 9. Если при $\mu_R((x_1, x_2)) = 1$ и $\mu_R((x_2, x_3)) = 1$ выполняется $\mu_R((x_1, x_3)) = 1$, то получаем свойство транзитивности в своем классическом определении.

Определение 23 — *T-транзитивность*. Нечеткое отношение R называют T-транзитивным, если $\forall x_1, x_2, x_3 \in X$ выполняется

$$\mu_R((x_1, x_3)) = T(\mu_R((x_1, x_2)), \mu_R((x_2, x_3))), \text{ где } T \text{ — } t\text{-норма}.$$

Пример — MIN-транзитивность. Нечеткое отношение R называют MIN-транзитивным, если $\forall x_1, x_2, x_3 \in X$ выполняется

$$\mu_R((x_1, x_3)) = \min(\mu_R((x_1, x_2)), \mu_R((x_2, x_3))).$$

Замечание 10. (о нечетком отношении эквивалентности). Как и в случае определения нечеткого отношения толерантности (замечание 6), определение нечеткого отношения эквивалентности может быть получено путем определения семейства нечетких отношений эквивалентности, каждый экземпляр в котором представлен определенной комбинацией из трех составляющих нечеткого отношения эквивалентности — нечеткой симметричности (определения 13.1—13.5), нечеткой рефлексивности (определения 14.1—14.7) и нечеткой транзитивности (определения 21—23). Если на всем полученном ранее нечетком отношении α-толерантности, которое строится на всем множестве, попытаться построить транзитивные связи, пользуясь определением T-транзитивности, то можно потерять большое количество информации о классовых взаимосвязях данных, которые, в общем случае, можно наблюдать в матрице нечеткого отношения α-толерантности. Поэтому в данной работе важным является следующее определение.

Определение 24 — *нечеткая* α-*квазиэквивалентность*. Нечеткое отношение R будем называть α-квазиэквивалентным, если оно обладает свойствами четкой рефлексивности, нормальной α-симметричности и α-квазитранзитивности.

Замечание 11. (о связи отношений эквивалентности и нечеткой α-квазиэквивалентности). От отношения нечеткой α-квазиэквивалентности можно перейти к отношению эквивалентности в классическом смысле, если определить некоторый уровень эквивалентности α и все элементы отношения, имеющие степень принадлежности ниже этого уровня, приравнять нулю, а остальные — единице.

Определение 25 — *транзитивное замыкание нечеткого отношения R*, определенного на множестве *X*, будем называть нечеткое отношение, получаемое вычислением следующего выражения:

$$\hat{R} = \bigcup_{i=1}^{|X|} R^i \ . \tag{7.38}$$

Для указания правил вычисления транзитивного замыкания необходимо ввести ряд определений.

Определение 26 — *треугольная норма*. Отображение $T : [0, 1]\times[0, 1]\to[0, 1]$ есть треугольная норма (*t*-норма), если оно симметрично, ассоциативно, неубывающее по каждому аргументу и $T(a,1) = a \quad \forall a \in [0, 1]$. Другими словами, любая *t*-норма *T* удовлетворяет свойствам:

- $T(x, y) = T(y, x)$ — симметричность;
- $T(x, T(y, z)) = T(T(x, y), z)$ — ассоциативность;
- $T(x, y) \le T(x', y')$ при $x \le x'$ и $y \le y'$ — монотонность;
- $T(x,1) = x \quad \forall x \in [0,1]$.

Кроме того, существует еще одно важное свойство треугольной нормы, а именно $T(x, y) \le \min\{x, y\}$.

Определение 27 — *пересечение по t-норме*. Пусть T — треугольная норма. Пересечение по *T*-норме определяется как $(A \cap B)(t) = T(A(t), B(t)) \quad \forall t \in X$.

Определение 28 — *треугольная конорма*. Отображение

$$S : [0, 1]\times[0, 1] \to [0, 1]$$

есть треугольная конорма (*T*-конорма), если оно симметрично, ассоциативно, неубывающее по каждому аргументу и $S(a,0) = a, \quad \forall a \in [0, 1]$. Другими словами, любая *T*-конорма *S* удовлетворяет свойствам:

- $S(x, y) = S(y, x)$ — симметричность;
- $S(x, S(y, z)) = S(S(x, y), z)$ — ассоциативность;

- $S(x,y) \le S(x',y')$ при $x \le x'$ и $y \le y'$ — монотонность;
- $S(x,0) = x \quad \forall x \in [0,\ 1]$.

Кроме того, существует еще одно важное свойство треугольной конормы, а именно $S(x,y) \ge \max\{x,y\}$.

Определение 29 — *объединение по Т-конорме*. Пусть S — треугольная конорма. Объединение по T-конорме определяется как $(A \cup B)(t) = S(A(t), B(t))$ $\forall t \in X$.

Замечание 12. (Вычисление транзитивного замыкания нечеткого отношения.) Операции в формуле (7.38) для вычисления транзитивного замыкания определяются в соответствии с определениями операций над нечеткими множествами и операций композиции нечетких отношений, а именно:

$$A = \{a_{ik}\}_{\substack{i=1\ldots N \\ k=1\ldots Q}},\ B = \{b_{kj}\}_{\substack{j=1\ldots M \\ k=1\ldots Q}},\ A \circ B = \{c_{ij}\}_{\substack{i=1\ldots N \\ j=1\ldots M}}; \tag{7.39}$$

$$c_{ij} = S(T(a_{i1}, b_{1j}),\ \ldots,\ T(a_{iQ}, b_{Qj})) = \overset{Q}{\underset{k=1}{S}} T(a_{ik}, b_{kj}), \tag{7.40}$$

где S — T-конорма, а T — T-норма;

$$R^n = \underbrace{R \circ \ldots \circ R}_{n} = R^{n-1} \circ R, \tag{7.41}$$

где $R^1 = R$;

$$A \cup B = S(A, B), \tag{7.42}$$

где S — T-конорма.

■ **ТЕОРЕМА 1.** Если отношение R — отношение α-толерантности, то справедливо следующее утверждение: $R \subseteq R^2 \subseteq \ldots \subseteq R^n \subseteq \ldots$.

ДОКАЗАТЕЛЬСТВО

Для доказательства данной теоремы достаточно показать, что $R^k \subseteq R^{k+1}$ $\forall k \ge 1$. Запись $R^k \subseteq R^{k+1}$ означает, что $\mu_{R^k}(x,y) \le \mu_{R^{k+1}}(x,y)$. Проверим, что $R \subseteq R^2$. На основании замечания 12 и формулы (7.41) получим:

$$r_{ij}^2 = S(T(r_{i1}, r_{1j}), \ldots, T(r_{iQ}, r_{Qj})) = \overset{Q}{\underset{k=1}{S}} T(r_{ik}, r_{kj}).$$

Известно, что $\underset{i}{T}(x_i) \le \min_i\{x_i\}$, а $\underset{i}{S}(x_i) \ge \max_i\{x_i\}$. Рассмотрим $T_k = T(r_{ik}, r_{kj})$. При $k = i$ или $k = j$ в силу четкой рефлексивности и нормальной α-сим-

метричности $T_k \le \min\{r_{ii}, r_{ij}\} = \min\{r_{ij}, r_{jj}\} = \min\{1, r_{ij}\} = r_{ij}$, при $k \ne i$ и $k \ne j$ имеем $T_k \le \min\{r_{ik}, r_{kj}\}$. Тогда $\overset{Q}{\underset{k=1}{S}} T(r_{ik}, r_{kj}) \ge \max\{r_{ij}, \max\limits_{\substack{k \ne i \\ k \ne j}}\{T_k\}\}$. Рассмотрим две возможные ситуации:

1) $r_{ij} \ge \max\limits_{\substack{k \ne i \\ k \ne j}}\{T_k\}$. В этом случае $\max\{r_{ij}, \max\limits_{\substack{k \ne i \\ k \ne j}}\{T_k\}\} = r_{ij}$;

2) $r_{ij} < \max\limits_{\substack{k \ne i \\ k \ne j}}\{T_k\} = T_{k\max}$. В этом случае $\max\{r_{ij}, \max\limits_{\substack{k \ne i \\ k \ne j}}\{T_k\}\} = T_{k\max} > r_{ij}$.

Обобщая ситуации 1 и 2, получаем

$$r_{ij}^2 = \overset{Q}{\underset{k=1}{S}} T(r_{ik}, r_{kj}) \ge \max\{r_{ij}, \max_{\substack{k \ne i \\ k \ne j}}\{T_k\}\} \ge r_{ij} \quad \forall i, j$$

или, другими словами, $\mu_R(x, y) \le \mu_{R^2}(x, y)$, что означает $R \subseteq R^2$.

Аналогично можно показать, что $R^2 \subseteq R^3$, ..., $R^k \subseteq R^{k+1}$, а значит, можно утверждать, что $R \subseteq R^2 \subseteq ... \subseteq R^n \subseteq ...$ для отношения α-толерантности.

Теорема доказана.

Следствие 1. Отношение R^i при условии, что отношение R — отношение α-толерантности, также является отношением α-толерантности.

Следствие 2. Для отношения R^2 справедливо: если $r_{ik} \ge \alpha$ и $r_{kj} \ge \alpha$, то $r_{ij} \ge \alpha \quad \forall k = \overline{1, |X|}$. Также справедливо обобщенное утверждение для R^i: если $r_{qk_1} \ge \alpha, r_{k_1k_2} \ge \alpha, ..., r_{k_{i-1}p} \ge \alpha$, то справедливо $r_{qp} \ge \alpha \quad \forall k_1, ..., k_{i-1} = \overline{1, |X|}$. Таким образом, очевидно, если R — отношение α-толерантности, то $r_{ij}^{|X|} \ge \alpha$ $\forall i, j = \overline{1, |X|}$.

■ **ТЕОРЕМА 2.** Транзитивное замыкание $\hat{R} = \bigcup\limits_{i=1}^{|X|} R^i$, вычисляемое как наименьшая верхняя граница объединения отношений R^i, для отношения α-толерантности R на множестве X равно отношению $R^{|X|}$.

ДОКАЗАТЕЛЬСТВО

По определению транзитивного замыкания и учитывая замечание 12 имеем $\hat{R} = \bigcup\limits_{i=1}^{|X|} R^i = \overset{|X|}{\underset{i=1}{S}} R^i$. При вычислении транзитивного замыкания как наименьшей

верхней границы отношений получим $\hat{R} = \bigcup_{i=1}^{|X|} R^i = \underset{i=1}{\overset{|X|}{S}} R^i = \sup\left(\underset{i=1}{\overset{|X|}{S}} R^i\right)$. Из свойств *T*-конормы известно, что $\underset{i}{S}(x_i) \geq \max\limits_i\{x_i\}$. Значит, $\sup\limits_i(\underset{i}{S}(x_i)) = \max\limits_i\{x_i\}$. Таким образом, имеем $\hat{R} = \bigcup_{i=1}^{|X|} R^i = \underset{i=1}{\overset{|X|}{S}} R^i = \sup\left(\underset{i=1}{\overset{|X|}{S}} R^i\right) = \max\left\{\underset{i=1}{\overset{|X|}{S}} R^i\right\}$. По теореме 1 $R \subseteq R^2 \subseteq ... \subseteq R^{|X|}$, а это означает, что $\max\left\{\underset{i=1}{\overset{|X|}{S}} R^i\right\} = R^{|X|}$. Таким образом, получаем $\hat{R} = \bigcup_{i=1}^{|X|} R^i = \underset{i=1}{\overset{|X|}{S}} R^i = \sup\left(\underset{i=1}{\overset{|X|}{S}} R^i\right) = \max\left\{\underset{i=1}{\overset{|X|}{S}} R^i\right\} = R^{|X|}$, что и требовалось доказать.

Теорема доказана.

■ **ТЕОРЕМА 3.** Объединение отношений α-толерантности есть также отношение α-толерантности.

ДОКАЗАТЕЛЬСТВО

Пусть R_1 и R_2 — отношения α-толерантности. Покажем, что $R = R_1 \cup R_2$ есть также отношение α-толерантности. По определению объединения $R = R_1 \cup R_2 = S(R_1, R_2) \geq \max\{R_1, R_2\}$.

Покажем, что отношение *R* обладает свойством четкой рефлексивности. Из свойства четкой рефлексивности, которым по определению обладает отношение α-толерантности, имеем

$$\mu_R(x,x) \geq \max\{\mu_{R_1}(x,x), \mu_{R_2}(x,x)\} = \max\{1,1\} = 1,$$

но $\mu_R(x,x) \leq 1$ по определению функции принадлежности, значит, $\mu_R(x,x) = 1$. Следовательно, отношение *R* обладает свойством четкой рефлексивности.

Покажем, что отношение *R* обладает свойством нормальной α-симметричности. Для отношений R_1 и R_2 по определению α-толерантности имеем: при $\mu_{R_1}(x,y) \geq \alpha$ выполняется $\mu_{R_1}(y,x) = \mu_{R_1}(x,y) \geq \alpha$, а при $\mu_{R_2}(x,y) \geq \alpha$ выполняется $\mu_{R_2}(y,x) = \mu_{R_2}(x,y) \geq \alpha \quad \forall x, y \in X$. Таким образом, используя свойства конормы, имеем: при $\mu_{R_1}(x,y) \geq \alpha$ или $\mu_{R_2}(x,y) \geq \alpha$ выполняется $\mu_R(x,y) \geq \max\{\mu_{R_1}(x,y), \mu_{R_2}(x,y)\} \geq \alpha$ и $\mu_R(y,x) \geq \max\{\mu_{R_1}(y,x), \mu_{R_2}(y,x)\} \geq \alpha$, т. е. отношение *R* обладает свойством α-квазисимметричности. Далее, при фиксации любой *T*-конормы *S*:

$$\mu_R(x,y) = S(\mu_{R_1}(x,y), \mu_{R_2}(x,y)) = S(\mu_{R_1}(y,x), \mu_{R_2}(y,x)) = \mu_R(y,x),$$

а с учетом доказанного свойства α-квазисимметричности $\mu_R(x,y) = \mu_R(y,x) \geq \alpha$, т. е. отношение R обладает свойством нормальной α-симметричности. Поскольку T-конорма выбиралась произвольно, то полученный результат справедлив для любой T-конормы, используемой при объединении множеств. Таким образом, получаем:

- $\mu_R(x,x) = 1$ для $\forall x \in X$ — свойство четкой рефлексивности;
- $\mu_R(x,y) = \mu_R(x,y) \geq \alpha \quad \forall x, y \in X$ — свойство нормальной α-симметричности.

Значит, отношение R есть отношение α-толерантности по определению, что и требовалось доказать.

Теорема доказана.

Следствие 1. Учитывая следствие 1 и теорему 3, можно заключить, что транзитивное замыкание отношения α-толерантности также является отношением α-толерантности.

■ **УТВЕРЖДЕНИЕ 1.** Транзитивное замыкание отношения α-толерантности порождает отношение α-квазиэквивалентности на множестве X.

ДОКАЗАТЕЛЬСТВО

Выпишем свойства α-толерантности:

- $\mu_R((x,x)) = 1 \quad \forall x \in X$;
- при условии $\mu_R((x_1,x_2)) \geq \alpha$ выполняется $\mu_R((x_1,x_2)) = \mu_R((x_2,x_1)) \geq \alpha$ при $\alpha > 0$ и $\forall x_1, x_2 \in X$.

Покажем свойства α-квазиэквивалентности:

- $\mu_R((x,x)) = 1 \quad \forall x \in X$;
- при условии $\mu_R((x_1,x_2)) \geq \alpha$ выполняется $\mu_R((x_1,x_2)) = \mu_R((x_2,x_1)) \geq \alpha$ при $\alpha > 0$ и $\forall x_1, x_2 \in X$;
- при $\mu_R((x_1,x_2)) \geq \alpha$ и $\mu_R((x_2,x_3)) \geq \alpha$ выполняется $\mu_R((x_1,x_3)) \geq \alpha$ $\forall x_1, x_2, x_3 \in X$ и $\alpha > 0$.

Из следствия 3 имеем, что транзитивное замыкание отношения α-толерантности есть также отношение α-толерантности. Таким образом, первые два свойства, которыми обладает отношение α-квазиэквивалентности, соблюдаются. Необходимо доказать, что транзитивное замыкание отношения α-толерантности обладает к тому же свойством α-квазитранзитивности.

Из следствия 2 $r_{ij}^{|X|} \geq \alpha \quad \forall i, j = \overline{1,|X|}$, т. е. выполняется и $\mu(x_i, x_k) \geq \alpha$, $\mu(x_k, x_j) \geq \alpha$, $\mu(x_i, x_j) \geq \alpha$. Значит, транзитивное замыкание отношения α-толерантности обладает свойством α-квазитранзитивности.

Обобщая изложенное, получается, что транзитивное замыкание отношения α-толерантности есть отношение α-квазиэквивалентности. Что и требовалось доказать.

Утверждение доказано.

Определение 30 — *порогом α-квазиэквивалентности* на множестве X называется число $\alpha_{\inf}$, при котором все элементы множества X являются α-квазиэквивалентными, а при любом $\alpha > \alpha_{\inf}$ данное утверждение не соблюдается. Другими словами, порог α-квазиэквивалентности есть наибольшая нижняя граница значений степеней принадлежности отношения α-квазиэквивалентности.

Определение 31 — уровень α-квазиэквивалентности. Уровнем α-квазиэквивалентности называется число $\alpha_L \geq \alpha_{\inf}$.

В классической теории множеств доказана следующая теорема.

■ **ТЕОРЕМА 4** (без доказательства). Отношение эквивалентности R разбивает множество X на попарно непересекающиеся классы эквивалентных элементов таким образом, что каждый элемент X принадлежит точно одному классу эквивалентности.

■ **УТВЕРЖДЕНИЕ 2.** Задание уровня α-квазиэквивалентности порождает разбиение множества X на классы эквивалентных элементов таким образом, что каждый элемент X принадлежит точно одному классу эквивалентности.

ДОКАЗАТЕЛЬСТВО

По замечанию 11, задав уровень α-квазиэквивалентности, можно перейти к четкому отношению эквивалентности, а из теоремы 4 следует, что данное отношение порождает разбиение множества X на классы эквивалентных элементов таким образом, что каждый элемент X принадлежит точно одному классу эквивалентности. Что и требовалось доказать.

Утверждение доказано.

Определение 32 — *шкалой отношения α-квазиэквивалентности* называется минимальный набор уровней эквивалентности $\alpha_{L_i} \in [\alpha_{\inf}, 1)$, каждый из которых порождает отношение эквивалентности $R(\alpha_{L_i})$, такое, что $R(\alpha_{L_i}) \neq R(\alpha_{L_j}) \quad \forall i \neq j$.

Построение шкалы отношения α-квазиэквивалентности как алгоритм анализа данных

В предыдущем разделе была представлена теоретическая база для исследования взаимосвязей образцов данных. Рассмотрим обобщенный алгоритм анализа данных с использованием отношения α-квазиэквивалентности и шкалы данного отношения.

Дано: множество образцов данных $X = \{x_i\}_{i=1}^{Q}$, где $x_i = (x_{i1},...,x_{in})$, $x_{ij} \in R$, n — размерность образцов данных, $Q = |X|$ — мощность множества X.

Найти шкалу отношения α-квазиэквивалентности на множестве X.

Алгоритм

Перед описанием алгоритма необходимо договориться об используемых в нем T-норме и T-конорме. Будем использовать MIN-норму и MAX-конорму.

1. Построить для каждого образца данных $x_i = (x_{i1},...,x_{in})$ нормальную меру сходства (определение 18) по формуле (7.34):

$$\mu_{x_q}(x_i) = 1 - \frac{d(x_q, x_i)}{\max\limits_{k \in [1,Q]}(d(x_q, x_k))},\ q,i = \overline{1,Q},\ d \text{ — расстояние по Евклиду.}$$

2. Построить относительно каждого образца данных на основании нормальной меры сходства относительную меру сходства для пар образцов данных (определение 19) по формуле (7.36):

$$\xi_{x_q}(x_i, x_j) = 1 - |\mu_{x_q}(x_i) - \mu_{x_q}(x_j)|,\ i,j,q = \overline{1,Q}.$$

3. Построить меру сходства образцов данных на множестве X (определение 20) по формуле (7.37):

$$\xi(a,b) = T(\xi_{x_1}(a,b),...,\xi_{x_Q}(a,b)) = \min_{i=\overline{1,Q}} \xi_{x_i}(a,b),\ a,b \in X.$$

   Полученное отношение является отношением α-толерантности на множестве X.

4. Построить транзитивное замыкание отношения меры сходства образцов данных на множестве X, используя определение 25, замечание 12, теорему 2 и договоренности об используемых в алгоритме T-норме и T-конорме. То есть каждый элемент построенного отношения будет иметь степень принадлежности, вычисляемую по следующему алгоритму:

$$R_\xi^1 = R_\xi.$$

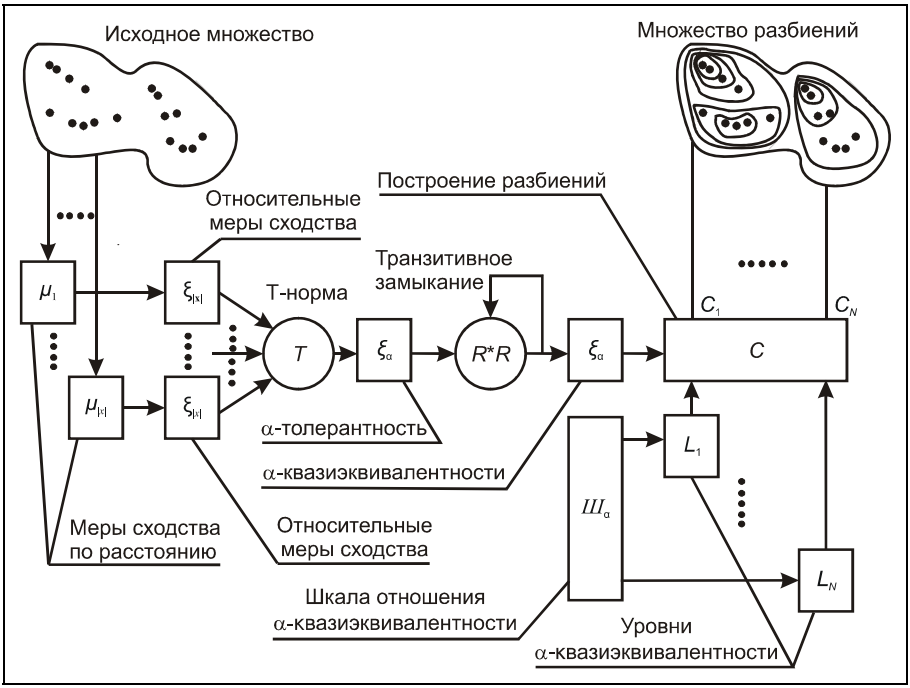

Рис. 7.11. Этапы выполнения алгоритма

По $q = \overline{2,Q}$ цикл

$$R_\xi^q = R_\xi^{q-1} \circ R_\xi$$

$$R_\xi^{|X|} = R_\xi^Q,\ r_{ij}^{|X|} = r_{ij}^Q$$

По утверждению 2, построенное отношение $R_\xi^{|X|}$ есть отношение α-квазиэквивалентности.

Построим для отношения α-квазиэквивалентности $R_\xi^{|X|}$ шкалу α-квазиэквивалентности как множество различных элементов отношения $R_\xi^{|X|}$.

Конец алгоритма.

Данный алгоритм укрупненно проиллюстрирован на рис. 7.11.

Об использовании шкалы α-квазиэквивалентности для анализа данных

Шкала α-квазиэквивалентности, построенная при помощи описанного алгоритма и представленная в виде упорядоченной по возрастанию последова-

тельности, порождает семейство отношений эквивалентности. Каждое из данных отношений соответствует своему уровню α-квазиэквивалентности, причем важным свойством данных отношений эквивалентности является то, что выбор каждого последующего уровня α-квазиэквивалентности (и соответствующего отношения эквивалентности) порождает более детальное разбиение множества X. Это "более детальное разбиение" получается разбиением классов эквивалентности, полученных при использовании предыдущего уровня α-квазиэквивалентности.

С другой стороны, чем меньше уровень α-квазиэквивалентности, тем меньше классов эквивалентности получается, и наоборот. Чем больше разность между соседними уровнями α-квазиэквивалентности, тем лучше обособлены кластеры точек.

Проиллюстрируем данные выводы примерами.

Примеры анализа данных при помощи шкалы α-квазиэквивалентности

В примерах используются двумерные образцы данных для достижения большей наглядности результатов анализа.

ПРИМЕР 1. Типичный набор данных для кластеризации.

Этот пример призван показать, что предложенный механизм анализа данных справляется с задачами, на которые ориентированы алгоритмы нечеткой кластеризации, подразумевающие некоторую заданную форму кластеров.

Дано: множество точек $X = \{x_1, ..., x_{18}\}$ (табл. 7.3).

Таблица 7.3

x_1	(0,11; 1)	x_7	(0,5; 3)	x_{13}	(0,35; 2,7)
x_2	(0,09; 1,1)	x_8	(0,4; 3)	x_{14}	(3,4; 0)
x_3	(0,1; 0,9)	x_9	(0,42; 2,5)	x_{15}	(3,6; 0)
x_4	(0,12; 0,9)	x_{10}	(0,48; 2,5)	x_{16}	(3,6; 1)
x_5	(0,4; 2)	x_{11}	(0,45; 2,8)	x_{17}	(3,4; 1)
x_6	(0,5; 2)	x_{12}	(0,45; 2,2)	x_{18}	(3,5; 0,5)

Графически образцы данных из множества X можно представить точками на плоскости (рис. 7.12).

Покажем, как шкала α-квазиэквивалентности разбивает множество X на классы эквивалентности.

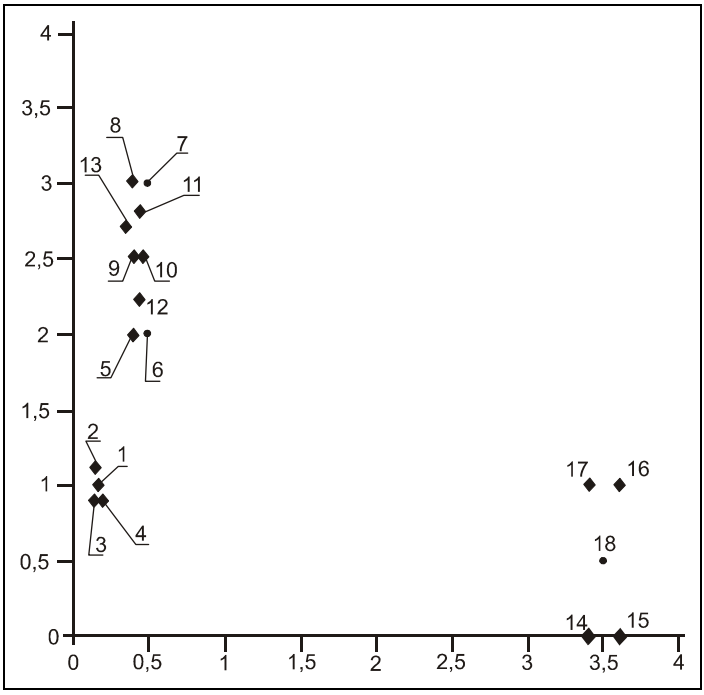

Рис. 7.12. Входные данные для примера 1

Матрица евклидовых расстояний между точками будет выглядеть следующим образом (табл. 7.4).

Матрица отношения сходства образцов данных на множестве X выглядит следующим образом (табл. 7.5).

Матрица отношения α-квазиэквивалентности выглядит следующим образом (табл. 7.6).

На основании последней таблицы получаем шкалу отношения α-квазиэквивалентности и соответствующие классы эквивалентности (табл. 7.7).

Данный набор разбиений в соответствии с уровнями отношения α-квазиэквивалентности удобно проиллюстрировать на рис. 7.13.

ПРИМЕР 2. Частный случай взаимного расположения данных — данные, равномерно расположенные на отрезке прямой.

Этот пример иллюстрирует свойство шкалы α-квазиэквивалентности, которое заключается в том, что если расстояние между соседними точками постоянно, то возможны лишь два уровня шкалы α-квазиэквивалентности, больший из которых равен 1. При выборе меньшего из них получается

Таблица 7.4

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_1	0	0,102	0,1	0,1	1,041	1,073	2,038	2,021	1,532	1,545	1,832	1,247	1,717	3,439	3,63	3,49	3,29	3,427
x_2	0,102	0	0,2	0,202	0,952	0,989	1,944	1,925	1,438	1,453	1,738	1,157	1,621	3,488	3,678	3,511	3,312	3,462
x_3	0,1	0,2	0	0,02	1,14	1,17	2,138	2,121	1,632	1,645	1,932	1,346	1,817	3,421	3,614	3,501	3,302	3,423
x_4	0,1	0,202	0,02	0	1,135	1,164	2,134	2,119	1,628	1,64	1,928	1,341	1,815	3,401	3,594	3,481	3,282	3,404
x_5	1,041	0,952	1,14	1,135	0	0,1	1,005	1	0,5	0,506	0,802	0,206	0,702	3,606	3,774	3,353	3,162	3,444
x_6	1,073	0,989	1,17	1,164	0,1	0	1	1,005	0,506	0,5	0,802	0,206	0,716	3,523	3,689	3,257	3,068	3,354
x_7	2,038	1,944	2,138	2,134	1,005	1	0	0,1	0,506	0,5	0,206	0,802	0,335	4,173	4,314	3,689	3,523	3,905
x_8	2,021	1,925	2,121	2,119	1	1,005	0,1	0	0,5	0,506	0,206	0,802	0,304	4,243	4,386	3,774	3,606	3,982
x_9	1,532	1,438	1,632	1,628	0,5	0,506	0,506	0,5	0	0,06	0,301	0,301	0,212	3,89	4,045	3,516	3,336	3,672
x_{10}	1,545	1,453	1,645	1,64	0,506	0,5	0,5	0,506	0,06	0	0,301	0,301	0,239	3,844	3,998	3,462	3,283	3,622
x_{11}	1,832	1,738	1,932	1,928	0,802	0,802	0,206	0,206	0,301	0,301	0	0,6	0,141	4,067	4,215	3,628	3,456	3,82
x_{12}	1,247	1,157	1,346	1,341	0,206	0,206	0,802	0,802	0,301	0,301	0,6	0	0,51	3,68	3,842	3,371	3,185	3,492
x_{13}	1,717	1,621	1,817	1,815	0,702	0,716	0,335	0,304	0,212	0,239	0,141	0,51	0	4,073	4,225	3,668	3,492	3,842
x_{14}	3,439	3,488	3,421	3,401	3,606	3,523	4,173	4,243	3,89	3,844	4,067	3,68	4,073	0	0,2	1,02	1	0,51
x_{15}	3,63	3,678	3,614	3,594	3,774	3,689	4,314	4,386	4,045	3,998	4,215	3,842	4,225	0,2	0	1	1,02	0,51
x_{16}	3,49	3,511	3,501	3,481	3,353	3,257	3,689	3,774	3,516	3,462	3,628	3,371	3,668	1,02	1	0	0,2	0,51
x_{17}	3,29	3,312	3,302	3,282	3,162	3,068	3,523	3,606	3,336	3,283	3,456	3,185	3,492	1	1,02	0,2	0	0,51
x_{18}	3,427	3,462	3,423	3,404	3,444	3,354	3,905	3,982	3,672	3,622	3,82	3,492	3,842	0,51	0,51	0,51	0,51	0

Таблица 7.5

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_1	1	0,972	0,972	0,972	0,712	0,704	0,434	0,439	0,575	0,572	0,491	0,655	0,523	0,053	0	0,039	0,088	0,056
x_2	0,972	1	0,945	0,944	0,74	0,731	0,463	0,467	0,603	0,6	0,52	0,683	0,551	0,052	0	0,045	0,082	0,059
x_3	0,972	0,945	1	0,994	0,684	0,676	0,408	0,413	0,548	0,545	0,465	0,627	0,497	0,053	0	0,031	0,084	0,053
x_4	0,972	0,944	0,994	1	0,684	0,676	0,406	0,411	0,547	0,544	0,464	0,627	0,495	0,054	0	0,031	0,087	0,053
x_5	0,712	0,74	0,684	0,684	1	0,973	0,722	0,726	0,863	0,86	0,779	0,943	0,811	0,045	0	0,112	0,123	0,087
x_6	0,704	0,731	0,676	0,676	0,973	1	0,729	0,728	0,863	0,864	0,783	0,944	0,806	0,045	0	0,117	0,149	0,091
x_7	0,434	0,463	0,408	0,406	0,722	0,729	1	0,977	0,859	0,863	0,943	0,779	0,911	0,017	0	0,022	0,023	0,019
x_8	0,439	0,467	0,413	0,411	0,726	0,728	0,977	1	0,863	0,863	0,945	0,783	0,915	0	0	0	0	0
x_9	0,575	0,603	0,548	0,547	0,863	0,863	0,859	0,863	1	0,985	0,916	0,919	0,943	0,038	0	0,068	0,075	0,078
x_{10}	0,572	0,6	0,545	0,544	0,86	0,864	0,863	0,863	0,985	1	0,918	0,917	0,94	0,039	0	0,083	0,09	0,09
x_{11}	0,491	0,52	0,465	0,464	0,779	0,783	0,943	0,945	0,916	0,918	1	0,837	0,966	0,035	0	0,039	0,042	0,041
x_{12}	0,655	0,683	0,627	0,627	0,943	0,944	0,779	0,783	0,919	0,917	0,837	1	0,862	0,042	0	0,107	0,117	0,091
x_{13}	0,523	0,551	0,497	0,495	0,811	0,806	0,911	0,915	0,943	0,94	0,966	0,862	1	0,036	0	0,028	0,032	0,035
x_{14}	0,053	0,052	0,053	0,054	0,045	0,045	0,017	0	0,038	0,039	0,035	0,042	0,036	1	0,946	0,73	0,723	0,864
x_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0,946	1	0,735	0,717	0,859
x_{16}	0,039	0,045	0,031	0,031	0,112	0,117	0,022	0	0,068	0,083	0,039	0,107	0,028	0,73	0,735	1	0,944	0,865
x_{17}	0,088	0,082	0,084	0,087	0,123	0,149	0,023	0	0,075	0,09	0,042	0,117	0,032	0,723	0,717	0,944	1	0,859
x_{18}	0,056	0,059	0,053	0,053	0,087	0,091	0,019	0	0,078	0,09	0,041	0,091	0,035	0,864	0,859	0,865	0,859	1

Таблица 7.6

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_1	1	0,972	0,972	0,972	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,149	0,149	0,149	0,149	0,149
x_2	0,972	1	0,972	0,972	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,149	0,149	0,149	0,149	0,149
x_3	0,972	0,972	1	0,994	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,149	0,149	0,149	0,149	0,149
x_4	0,972	0,972	0,994	1	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,149	0,149	0,149	0,149	0,149
x_5	0,74	0,74	0,74	0,74	1	0,973	0,919	0,919	0,919	0,919	0,919	0,944	0,919	0,149	0,149	0,149	0,149	0,149
x_6	0,74	0,74	0,74	0,74	0,973	1	0,919	0,919	0,919	0,919	0,919	0,944	0,919	0,149	0,149	0,149	0,149	0,149
x_7	0,74	0,74	0,74	0,74	0,919	0,919	1	0,977	0,943	0,943	0,945	0,919	0,945	0,149	0,149	0,149	0,149	0,149
x_8	0,74	0,74	0,74	0,74	0,919	0,919	0,977	1	0,943	0,943	0,945	0,919	0,945	0,149	0,149	0,149	0,149	0,149
x_9	0,74	0,74	0,74	0,74	0,919	0,919	0,943	0,943	1	0,985	0,943	0,919	0,943	0,149	0,149	0,149	0,149	0,149
x_{10}	0,74	0,74	0,74	0,74	0,919	0,919	0,943	0,943	0,985	1	0,943	0,919	0,943	0,149	0,149	0,149	0,149	0,149
x_{11}	0,74	0,74	0,74	0,74	0,919	0,919	0,945	0,945	0,943	0,943	1	0,919	0,966	0,149	0,149	0,149	0,149	0,149
x_{12}	0,74	0,74	0,74	0,74	0,944	0,944	0,919	0,919	0,919	0,919	0,919	1	0,919	0,149	0,149	0,149	0,149	0,149
x_{13}	0,74	0,74	0,74	0,74	0,919	0,919	0,945	0,945	0,943	0,943	0,966	0,919	1	0,149	0,149	0,149	0,149	0,149
x_{14}	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	1	0,946	0,864	0,864	0,864
x_{15}	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,946	1	0,864	0,864	0,864
x_{16}	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,864	0,864	1	0,944	0,865
x_{17}	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,864	0,864	0,944	1	0,865
x_{18}	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,149	0,864	0,864	0,865	0,865	1

Таблица 7.7

Уровень отношения α-квазиэквивалентности / (Количество классов эквивалентности)	**Разбиение множества X**
0,149208513 / (1)	$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}\}$
0,739911062 / (2)	$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}$, $\{x_{14}, x_{15}, x_{16}, x_{17}, x_{18}\}$
0,864071258 / (3)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}$, $\{x_{14}, x_{15}, x_{16}, x_{17}, x_{18}\}$
0,864876253 / (4)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}, x_{17}, x_{18}\}$
0,918625598 / (5)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}, x_{17}\}$, $\{x_{18}\}$
0,943203673 / (6)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_{12}\}$, $\{x_7, x_8, x_9, x_{10}, x_{11}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}, x_{17}\}$, $\{x_{18}\}$
0,944118841 / (7)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6, x_{12}\}$, $\{x_9, x_{10}\}$, $\{x_7, x_8, x_{11}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}, x_{17}\}$, $\{x_{18}\}$
0,944383730 / (8)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_9, x_{10}\}$, $\{x_{12}\}$, $\{x_7, x_8, x_{11}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}, x_{17}\}$, $\{x_{18}\}$
0,944858491 / (9)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_9, x_{10}\}$, $\{x_{12}\}$, $\{x_7, x_8, x_{11}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,946234261 / (10)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{12}\}$, $\{x_{11}, x_{13}\}$, $\{x_{14}, x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,966444559 / (11)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{12}\}$, $\{x_{11}, x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,971695967 / (12)	$\{x_1, x_2, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,972190761 / (13)	$\{x_2\}$, $\{x_1, x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,972893656 / (14)	$\{x_1\}$, $\{x_2\}$, $\{x_3, x_4\}$, $\{x_5, x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,976819286 / (15)	$\{x_1\}$, $\{x_2\}$, $\{x_3, x_4\}$, $\{x_5\}$, $\{x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,984992682 / (16)	$\{x_1\}$, $\{x_2\}$, $\{x_3, x_4\}$, $\{x_5\}$, $\{x_6\}$, $\{x_7\}$, $\{x_8\}$, $\{x_9, x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
0,994435937 / (17)	$\{x_1\}$, $\{x_2\}$, $\{x_3, x_4\}$, $\{x_5\}$, $\{x_6\}$, $\{x_7\}$, $\{x_8\}$, $\{x_9\}$, $\{x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$
1 / (18)	$\{x_1\}$, $\{x_2\}$, $\{x_3\}$, $\{x_4\}$, $\{x_5\}$, $\{x_6\}$, $\{x_7\}$, $\{x_8\}$, $\{x_9\}$, $\{x_{10}\}$, $\{x_{11}\}$, $\{x_{12}\}$, $\{x_{13}\}$, $\{x_{14}\}$, $\{x_{15}\}$, $\{x_{16}\}$, $\{x_{17}\}$, $\{x_{18}\}$

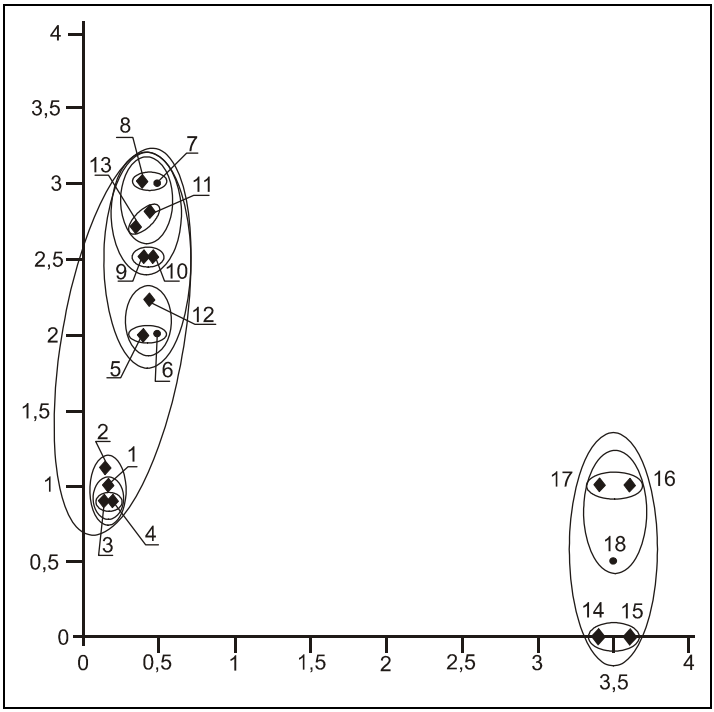

Рис. 7.13. Иллюстрация разбиения множества X
на шкале отношения α-квазиэквивалентности для примера 1

универсальное отношение, т. е. соответствующее отношение эквивалентности порождает разбиение, состоящее лишь из одного класса эквивалентности, равного исходному множеству. При выборе большего из них (равного 1) множество X разбивается на классы эквивалентности, каждый из которых представлен единственным образцом данных. Этот тезис соответствует интуитивному пониманию сходства между равномерно отстоящими друг от друга образцами данных. Они либо принадлежат одному классу, либо каждый из образцов данных представляет свой собственный класс.

Дано: множество точек $X = \{x_1, ..., x_{10}\}$ (табл. 7.8).

Иллюстрация взаимного размещения данных представлена на рис. 7.14.

Таблица 7.8

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

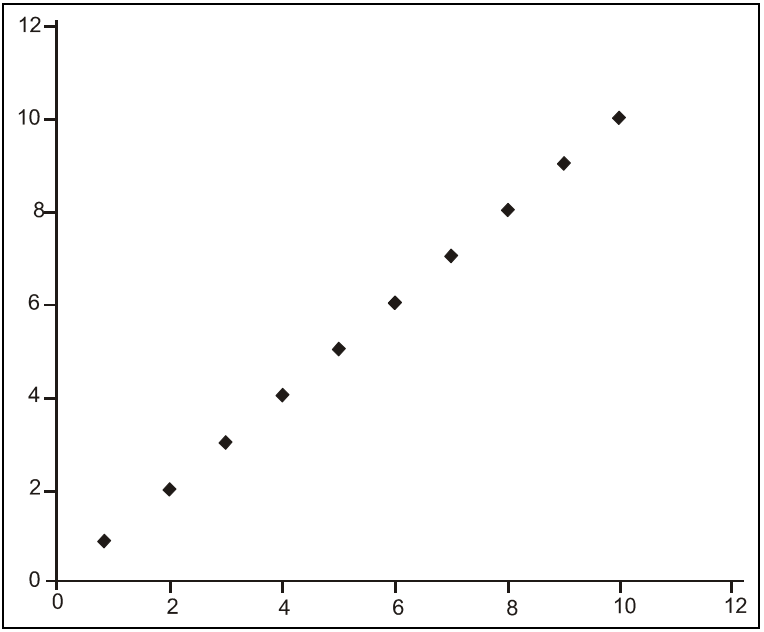

Рис. 7.14. Множество входных данных для примера 2

Матрица отношения сходства образцов данных на множестве X выглядит следующим образом (табл. 7.9).

Таблица 7.9

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	1	0,8	0,6	0,4	0,2	0	0	0	0	0
x_2	0,8	1	0,8	0,6	0,4	0,2	0,167	0,143	0,125	0
x_3	0,6	0,8	1	0,8	0,6	0,4	0,333	0,286	0,143	0
x_4	0,4	0,6	0,8	1	0,8	0,6	0,5	0,333	0,167	0
x_5	0,2	0,4	0,6	0,8	1	0,8	0,6	0,4	0,2	0
x_6	0	0,2	0,4	0,6	0,8	1	0,8	0,6	0,4	0,2
x_7	0	0,167	0,333	0,5	0,6	0,8	1	0,8	0,6	0,4
x_8	0	0,143	0,286	0,333	0,4	0,6	0,8	1	0,8	0,6
x_9	0	0,125	0,143	0,167	0,2	0,4	0,6	0,8	1	0,8
x_{10}	0	0	0	0	0	0,2	0,4	0,6	0,8	1

Матрица транзитивного замыкания отношения сходства данных на множестве X (матрица отношения α-квазиэквивалентности) выглядит следующим образом (табл. 7.10).

Таблица 7.10

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	1	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
x_2	0,8	1	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
x_3	0,8	0,8	1	0,8	0,8	0,8	0,8	0,8	0,8	0,8
x_4	0,8	0,8	0,8	1	0,8	0,8	0,8	0,8	0,8	0,8
x_5	0,8	0,8	0,8	0,8	1	0,8	0,8	0,8	0,8	0,8
x_6	0,8	0,8	0,8	0,8	0,8	1	0,8	0,8	0,8	0,8
x_7	0,8	0,8	0,8	0,8	0,8	0,8	1	0,8	0,8	0,8
x_8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1	0,8	0,8
x_9	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1	0,8
x_{10}	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1

Разбиение множества X показано в табл. 7.11.

Таблица 7.11

Уровень отношения α-квазиэквивалентности / (Количество классов эквивалентности)	**Разбиение множества X**
0,8 / (1)	$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$
1 / (10)	$\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}$

Разбиения, соответствующие различным уровням α-квазиэквивалентности, показаны на рис. 7.15.

ПРИМЕР 3. Частный случай взаимного расположения данных — данные, расположенные на двух вложенных окружностях.

Этот пример призван проиллюстрировать те возможности предложенного подхода к кластеризации данных, что недоступны другим алгоритмам кластеризации, в основе которых положено разбиение множества образцов

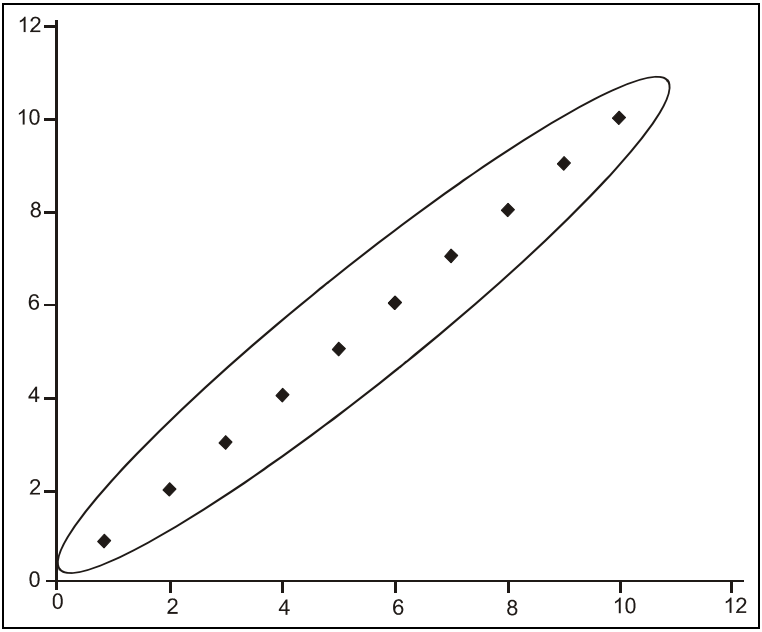

Рис. 7.15. Иллюстрация разбиения множества *X* на шкале отношения α-квазиэквивалентности для примера 2

данных на аппроксимации кластеров конкретными фигурами (сферами, эллипсоидами). Этот пример иллюстрирует уже упомянутую ранее проблему кластеризации данных, представляющих собой две вложенные окружности. Отмечалось, что алгоритмы, подобные Fuzzy C-Means, не справляются с данной задачей. Покажем, что при использовании следующего за пороговым уровня α-квазиэквивалентности из шкалы отношения α-квазиэквивалентности можно получить требуемое разбиение.

Дано: $X = \{x_1, ..., x_{40}\}$ (табл. 7.12).

Входные данные иллюстрируются на рис. 7.16.

Откажемся в данном примере от демонстрации матриц отношений, используемых при построении шкалы α-квазиэквивалентности. Покажем лишь собственно шкалу и соответствующие разбиения исходного множества образцов данных (табл. 7.13).

Как и было заявлено, при использовании следующего за пороговым уровня α-квазиэквивалентности из шкалы отношения α-квазиэквивалентности получается разбиение множества образцов данных на две окружности.

Таблица 7.12

x_1	0	1
x_2	0,30901699437494742	0,95105651629515357
x_3	0,58778525229247312	0,80901699437494742
x_4	0,80901699437494742	0,58778525229247312
x_5	0,95105651629515357	0,30901699437494742
x_6	1	0
x_7	0,95105651629515357	–0,30901699437494742
x_8	0,80901699437494742	–0,58778525229247312
x_9	0,58778525229247312	–0,80901699437494742
x_{10}	0,30901699437494742	–0,95105651629515357
x_{11}	0	–1
x_{12}	–0,30901699437494742	–0,95105651629515357
x_{13}	–0,58778525229247312	–0,80901699437494742
x_{14}	–0,80901699437494742	–0,58778525229247312
x_{15}	–0,95105651629515357	–0,30901699437494742
x_{16}	–1	0
x_{17}	–0,95105651629515357	0,30901699437494742
x_{18}	–0,80901699437494742	0,58778525229247312
x_{19}	–0,58778525229247312	0,80901699437494742
x_{20}	–0,30901699437494742	0,95105651629515357
x_{21}	0	0,5
x_{22}	0,15450849718747371	0,47552825814757678
x_{23}	0,29389262614623656	0,40450849718747371
x_{24}	0,40450849718747371	0,29389262614623656
x_{25}	0,47552825814757678	0,15450849718747371
x_{26}	0,5	0
x_{27}	0,47552825814757678	–0,15450849718747371
x_{28}	0,40450849718747371	–0,29389262614623656
x_{29}	0,29389262614623656	–0,40450849718747371
x_{30}	0,15450849718747371	–0,47552825814757678
x_{31}	0	–0,5
x_{32}	–0,15450849718747371	–0,47552825814757678
x_{33}	–0,29389262614623656	–0,40450849718747371
x_{34}	–0,40450849718747371	–0,29389262614623656
x_{35}	–0,47552825814757678	–0,15450849718747371
x_{36}	–0,5	0
x_{37}	–0,47552825814757678	0,15450849718747371
x_{38}	–0,40450849718747371	0,29389262614623656
x_{39}	–0,29389262614623656	0,40450849718747371
x_{40}	–0,15450849718747371	0,47552825814757678

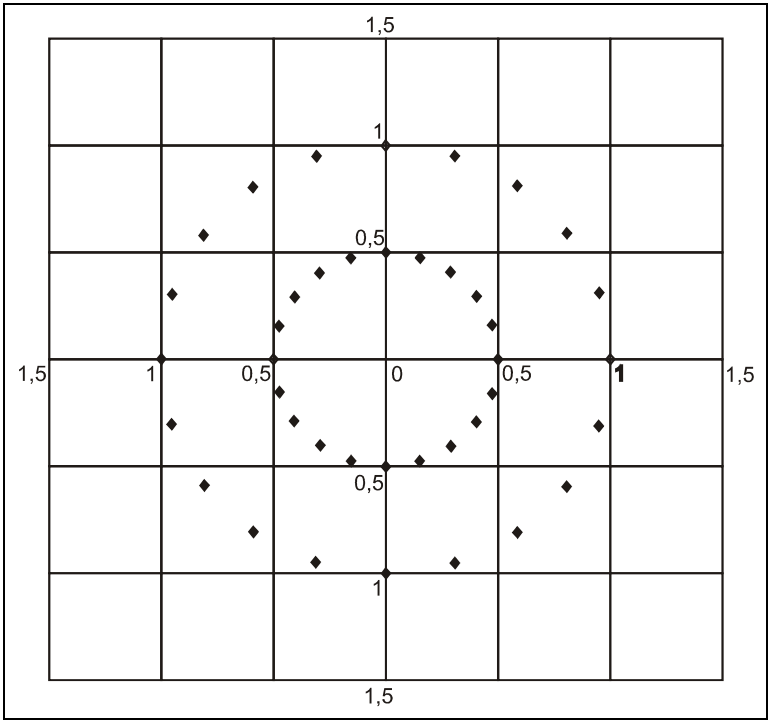

Рис. 7.16. Входные данные для примера 3

Таблица 7.13

Уровень отношения α-квазиэквивалентности / (Количество классов эквивалентности, примечание)	**Разбиение множества *X***
0,66667 / (1, "универсальный" класс)	$\{x_1, ..., x_{40}\}$
0,84357 / (2, "разбиение по окружностям")	$\{x_1, ..., x_{20}\}$, $\{x_{21}, ..., x_{40}\}$
0,89571 / (21, "разбиение на внутреннюю окружность и классы — по одной точке в каждом")	$\{x_1\}, ..., \{x_{20}\}$, $\{x_{21}, ..., x_{40}\}$
1 / (40, "разбиение на классы — по одной точке в каждом")	$\{x_1\}, ..., \{x_{40}\}$

Обобщая приведенные примеры, можно отметить, что при использовании следующего за пороговым уровня отношения α-квазиэквивалентности получается разбиение на наиболее крупные и наиболее различающиеся классы эквивалентности (случай разбиения по порогу уровня отношения α-квазиэквивалентности не принимается во внимание, т. к. при его использовании на

множестве X порождается универсальное отношение и, следовательно, разбиение состоит из одного универсального класса эквивалентности, равного множеству X).

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 дентограмма — отображение последовательного процесса кластеризации.
- Базовые методы кластеризации делятся на иерархические и неиерархические. Первые строят дентограммы 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.
- Построено отношение α-толерантности на множестве образцов данных. Предложенная конструктивная процедура построения данного отношения отражает интуитивное понимание того, как можно сравнить два образца данных в рамках предложенного множества образцов. Указанное отношение получается из следующего правила: "если два образца данных сходны относительно каждого из образцов данных входного множества, то их можно считать сходными относительно всего множества образцов данных".
- Предлагается процедура получения этого отношения из относительных мер сходства образцов данных, которые в свою очередь получаются на основании нормальной меры сходства и понятия расстояния между образцами данных.
- Построено отношение α-квазиэквивалентности на множестве образцов данных. Получение данного отношения основано на построении транзитивных связей на отношении α-толерантности. Математическая процедура, положенная в основу вычисления данного отношения (прежде всего

методика вычисления транзитивного замыкания на отношении α-толерантности), подразумевает под собой следующую семантику: "степень сходства двух образцов данных в отношении α-квазиэквивалентности может увеличиться по отношению к степени сходства этих образцов в отношении α-толерантности, если между ними найдется такая последовательность образцов данных, степень сходства между соседними образцами данных в которой превышает исходную степень сходства между заданными образцами на всей последовательности образцов данных". Таким образом, отношение α-квазиэквивалентности является непосредственным инструментом кластеризации данных. Данное отношение может быть легко преобразовано в отношение эквивалентности в классическом смысле, а это позволяет разбивать множество образцов данных на классы эквивалентности, что является, по сути, основным результатом решения задачи кластеризации.

- В соответствии с постановкой задачи кластеризации ее решение в рамках предложенного подхода выглядит следующим образом:
  - **способ сравнения данных** — в данном издании предложено несколько мер сходства, обладающих различной степенью объективности по отношению к сравниваемым объектам. В приведенном далее списке каждый последующий из способов сравнения данных основывается на предыдущем и является более подходящим для непосредственного решения задачи кластеризации;
  - **сравнение по расстоянию между образцами данных** (в примерах, приведенных в данном издании, используется евклидово расстояние) — данный способ сравнения можно охарактеризовать как попарное сравнение образцов данных в терминах тех единиц измерения, в которых задаются атрибуты образцов данных;
  - **сравнение при помощи семейства нормальных мер сходства** — результаты сравнения при помощи данного способа являются безразмерными числовыми значениями в диапазоне от 0 до единицы и могут интерпретироваться как нечеткие множества образцов данных, близких к каждому из образцов данных. Этот способ сравнения можно охарактеризовать как поочередное сравнение всех образцов данных с каждым из образцов;
  - **сравнение при помощи семейства относительных мер сходства** — результатом сравнения при помощи данного способа является семейство нечетких отношений α-толерантности, каждое из которых имеет смысл попарного сравнения образцов данных относительно заданного образца. Результаты сравнения выражаются безразмерной величиной в диапазоне от 0 до 1. Чем больше значение данной величины, тем более

схожи элементы. Математическая интерпретация относительной меры сходства — степень принадлежности каждой пары образцов данных к соответствующему отношению сходства;

- **сравнение образцов данных при помощи отношения α-толерантности на множестве образцов данных** — первый в данном списке способ "объективного" сравнения образцов данных, т. к. учитывает схожесть любых двух образцов данных относительно всех остальных образцов. Этот способ сравнения данных еще нельзя непосредственно использовать для их кластеризации, т. к. в отношении α-толерантности отсутствует свойство транзитивности, т. е. отсутствует возможность группового сравнения данных. Результатом сравнения двух образцов данных является безразмерная величина в диапазоне от нуля до единицы. Математическая интерпретация данной меры сходства — степень принадлежности каждой пары образцов данных к отношению α-толерантности;
- **сравнение при помощи отношения α-квазиэквивалентности и шкалы отношения α-квазиэквивалентности** — это тот способ сравнения образцов данных, который непосредственно используется для кластеризации данных. В отличие от предыдущего данный способ сравнения является наиболее "объективным", т. к. в отношении α-квазиэквивалентности присутствует один из нечетких аналогов свойства транзитивности (отношении α-квазитранзитивности), что позволяет учитывать межгрупповое сходство данных. Используя минимально полный набор уровней отношения α-квазиэквивалентности, можно породить целое семейство отношений эквивалентности в классическом смысле, при помощи которых строится набор разбиений множества образцов данных на классы эквивалентности. Результатом сравнения является безразмерная величина в диапазоне от 0 до 1. Математическая интерпретация данной меры сходства — степень принадлежности каждой пары образцов данных к отношению α-квазиэквивалентности;
- **способ кластеризации** — способом кластеризации в данном издании в соответствии с выбранным подходом является применение семейства отношений эквивалентности. Каждое из этих отношений получается при помощи перехода от отношения α-квазиэквивалентности к отношению эквивалентности в классическом смысле использованием соответствующего уровня отношения α-квазиэквивалентности из шкалы отношения α-квазиэквивалентности. Те образцы данных, которые в соответствии с отношением α-квазиэквивалентности имеют сходство, превышающее указанный уровень, являются эквивалентными, остальные — неэквивалентными;

- **разбиение данных по кластерам** — разбиение по кластерам не является однозначным. Это очевидно из допущения о нечеткой взаимосвязи данных. Тем не менее количество разбиений в соответствии с данным подходом является конечным и определяется мощностью шкалы отношения α-квазиэквивалентности. Каждое конкретное разбиение по кластерам соответствует разбиению множества образцов данных на классы эквивалентности при данном уровне α-квазиэквивалентности.

❒ Более кратко решение задачи кластеризации выглядит так:

- **способ сравнения** — отношение α-квазиэквивалентности и его шкала;
- **способ кластеризации** — применение семейства отношений эквивалентности, получаемого из отношения α-квазиэквивалентности и шкалы отношения α-квазиэквивалентности;
- **разбиение множества данных на кластеры** — набор разбиений множества образцов данных на классы эквивалентности на шкале отношения α-квазиэквивалентности.

ГЛАВА 8

Стандарты Data Mining

8.1. 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. 8.1.

8.2. Стандарт CWM

8.2.1. 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

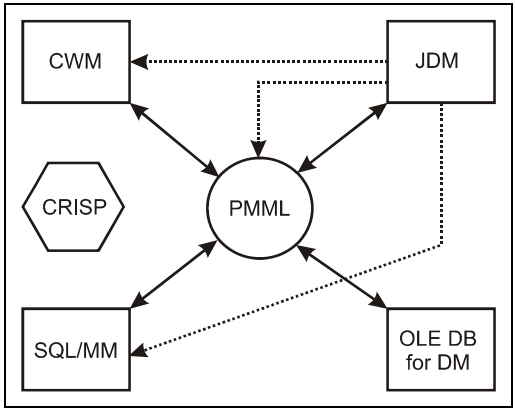

Рис. 8.1. Отношения между основными стандартами Data Mining

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 1.0.

8.2.2. 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. 8.2).

Рис. 8.2. Структура и состав CWM

Объектное ядро (ObjectModel) 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.

8.2.3. 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. 8.3):

- `MiningModel` — представляет Mining модель;
- `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).

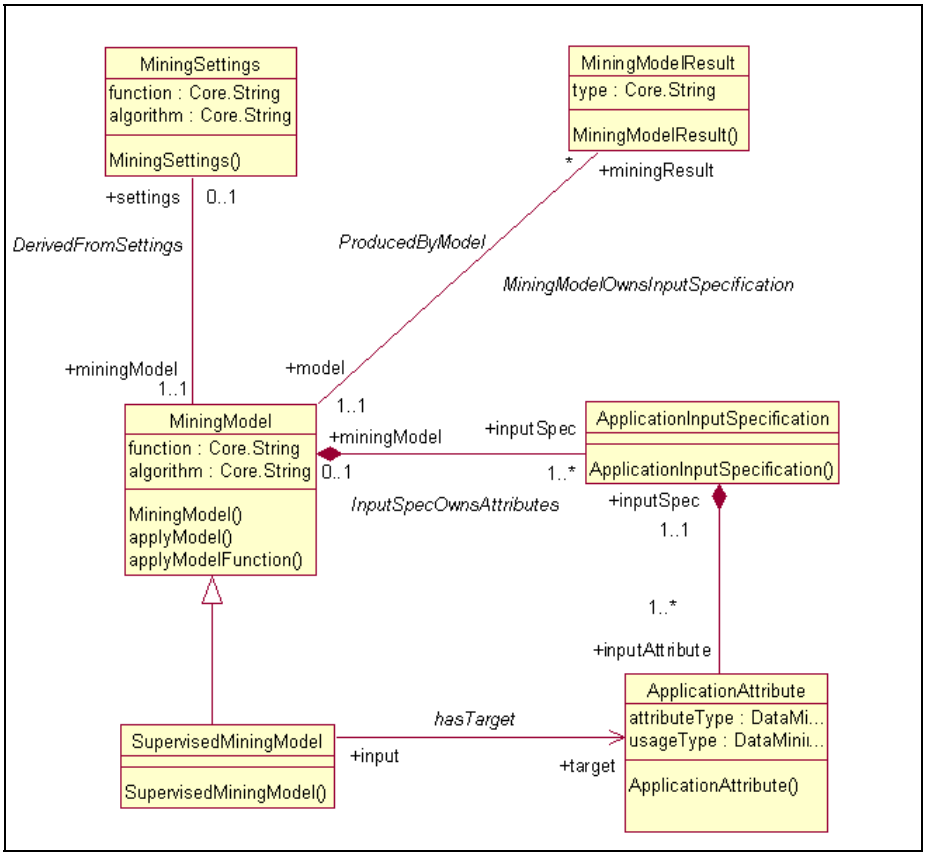

Рис. 8.3. Диаграмма классов метамодели Model

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. 8.4):

- `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.

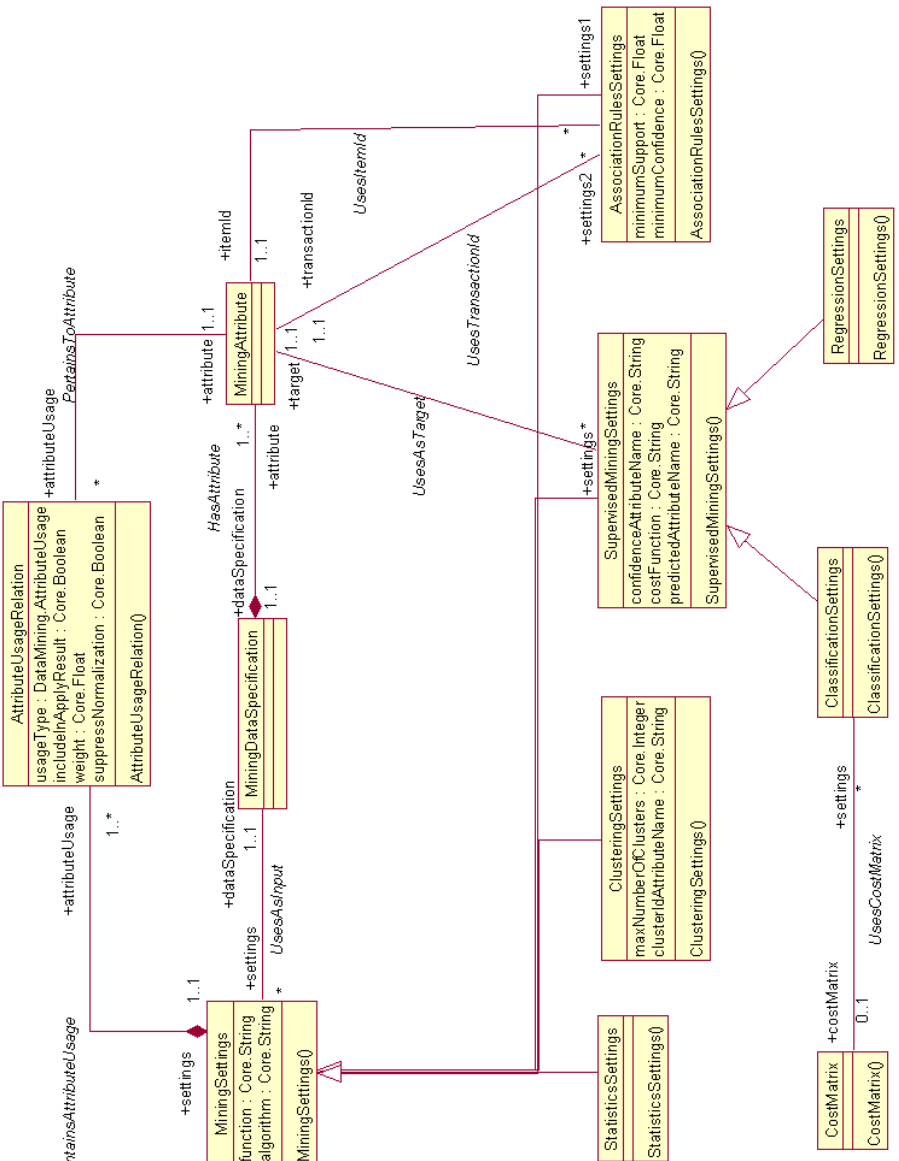

Рис. 8.4. Диаграмма классов метамодели Settings

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. 8.5):

- `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.

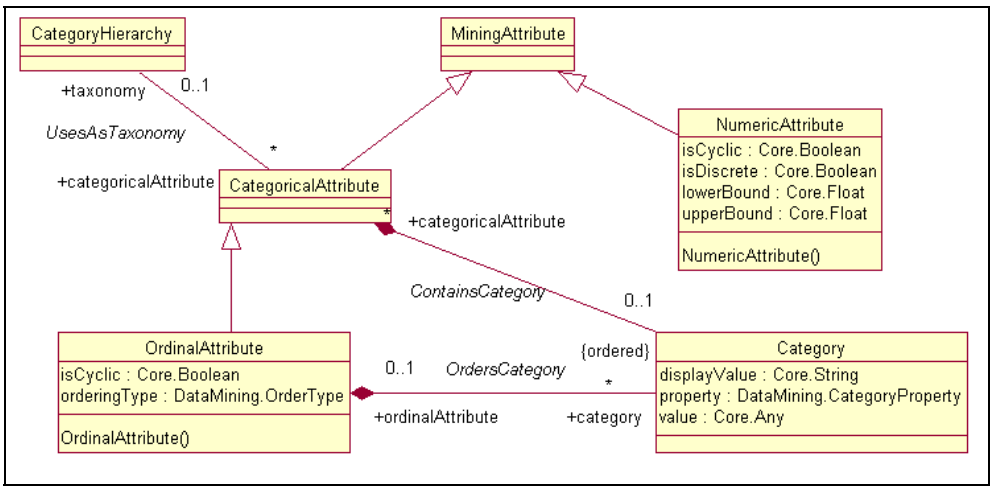

Рис. 8.5. Диаграмма классов метамодели Attributes

8.3. Стандарт CRISP

8.3.1. 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.

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.

8.3.2. Структура стандарта CRISP

Стандарт CRISP-DM описывается в терминах иерархической модели процесса (рис. 8.6). 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.

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.

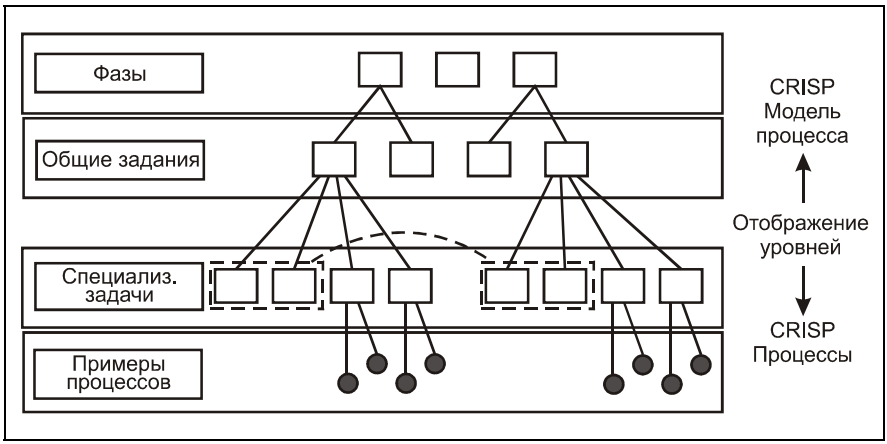

Рис. 8.6. Четыре уровня методологии CRISP

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. 8.7 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.

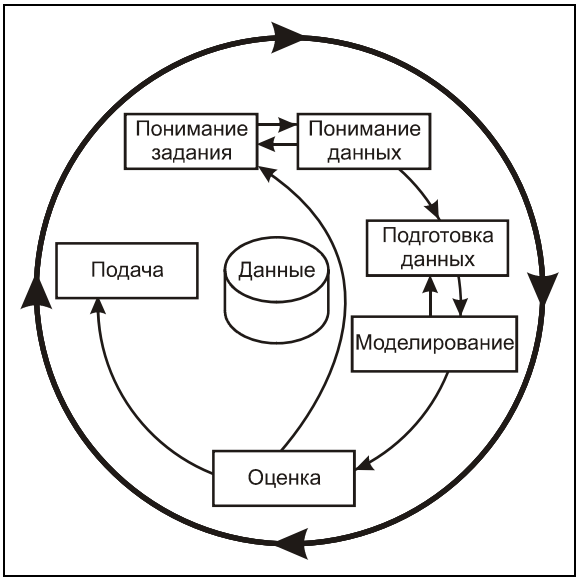

Рис. 8.7. Жизненый цикл процесса Data Mining согласно методологии CRISP

8.3.3. 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, некоторые 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 (удаления 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.

8.4. 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 — достаточно молодой стандарт (в настоящее время опубликована

версия 2.0) и нуждается в дальнейшем совершенствовании. Однако можно смело утверждать, что в настоящее 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 2.0.

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"?>
   <!DOCTYPE PMML
             PUBLIC "PMML 2.0"
             "http://www.dmg.org/v2-0/pmml_v2_0.dtd">
   <PMML  version="2.0">
   ...
   </PMML>
```

Тег `DOCTYPE` в PMML-документе не обязателен.

Структура тега `PMML` имеет следующий вид:

```
<!ENTITY % A-PMML-MODEL '(TreeModel |
                          NeuralNetwork |
                          ClusteringModel |
                          RegressionModel |
                          GeneralRegressionModel |
                          NaiveBayesModels |
                          AssociationModel |
                          SequenceModel)' >

<!ELEMENT PMML ( Header,
                 MiningBuildTask?,
                 DataDictionary,
                 TransformationDictionary?,
                 (%A-PMML-MODEL;)*,
                 Extension*)>
<!ATTLIST PMML
     version    CDATA    #REQUIRED   >

<!ELEMENT MiningBuildTask (Extension*) >
```

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:

```
<!ELEMENT Header(Application?,Annotation*,Timestamp?)>
<!ATTLIST Header
       copyright           CDATA              #REQUIRED
       description         CDATA              #IMPLIED
>
<!ELEMENT Application (EMPTY)>
<!ATTLIST Application
       name                CDATA              #REQUIRED
       version             CDATA               #IMPLIED
>
<!ELEMENT Annotation (#PCDATA)>
<!ELEMENT Timestamp  (#PCDATA)>
```

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 2.0 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:

```
<!ENTITY         % FIELD-NAME                "CDATA" >

<!ELEMENT DataDictionary (Extension*, DataField+, Taxonomy*) >
<!ATTLIST DataDictionary
     numberOfFields    %INT-NUMBER;  #IMPLIED
>

<!ELEMENT DataField  ( Extension*, (Interval*| Value*)) >
<!ATTLIST DataField
     name         %FIELD-NAME;                 #REQUIRED
     displayName  CDATA                        #IMPLIED
     optype       (categorical|ordinal|continuous)   #REQUIRED
     taxonomy     CDATA                        #IMPLIED
     isCyclic     ( 0 | 1)   "0"
>
```

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:

```
<!ELEMENT TransformationDictionary ( Extension*, DerivedField*) >

<!ENTITY  % EXPRESSION "( Constant | FieldRef |
                          NormContinuous | NormDiscrete |
                          Discretize |
                          MapValues |
                          Aggregate)" >
```

```
<!ELEMENT DerivedField (Extension*, %EXPRESSION;)>
<!ATTLIST DerivedField
     name         %FIELD-NAME;  #IMPLIED
     displayName  CDATA         #IMPLIED
>
```

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.
- 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.
- 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.
- 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).
- 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`.

- 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.
- Последовательность (SequenceModel) — данная модель состоит из последовательности объектов, идентифицируемых первичным ключом ("Primary Key"), некоторые результаты которых описываются вторичным ключом ("Secondary Key"). Каждый результат состоит из набора упорядоченных элементов. Поле "порядок" ("Order Field") описывает порядок элементов в результате.

Для всех моделей структура их описания в PMML-документе имеет следующий вид:

```
<ENTITY % MINING-FUNCTION '(associationRules |
                            sequences |
                            classification |
                            regression |
                            clustering)' >

<!ELEMENT XModel (Extension*, MiningSchema,
                  ModelStats?, ..., Extension*) >
   <!ATTLIST XModel
        modelName           CDATA                #IMPLIED
        functionName        %MINING-FUNCTION;    #REQUIRED
        algorithmName       CDATA                #IMPLIED
   >

   <!ELEMENT MiningSchema   (MiningField+) >
   <!ELEMENT ModelStats     (UnivariateStats+) >
```

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.

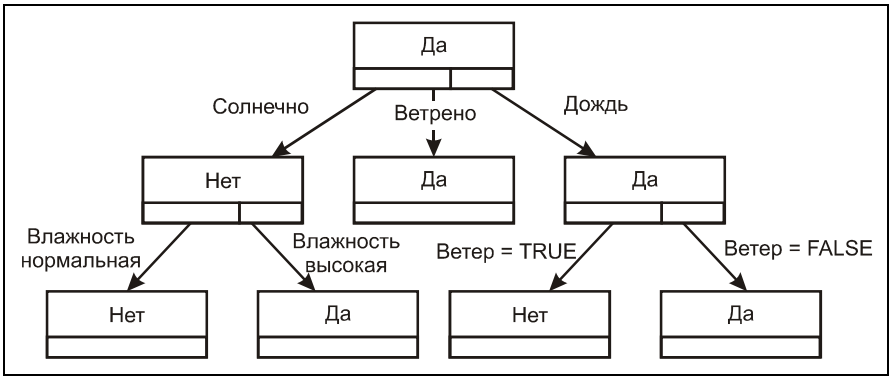

Рис. 8.8. Пример дерева решений

В заключение приведем пример модели в формате PMML для дерева решения, представленного на рис. 8.8.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PMML PUBLIC "pmml20.dtd" "pmml20.dtd">
<PMML version="2.0">
    <Header copyright="Copyright (c) 2003 prudsys AG"
description="Xelopes mining model. See www.zsoft.ru or www.prudsys.com">
        <Application version="1.1" name="Xelopes" />
        <Timestamp>2003-12-07 12:57:09 MSK</Timestamp>
    </Header>
    <DataDictionary numberOfFields="5">
        <DataField displayName="outlook" name="outlook" isCyclic="0"
optype="categorical">
            <Value displayValue="sunny" property="valid" value="sunny" />
            <Value displayValue="overcast" property="valid"
value="overcast" />
            <Value displayValue="rainy" property="valid" value="rainy" />
        </DataField>
        <DataField displayName="temperature" name="temperature"
isCyclic="0" optype="categorical">
            <Value displayValue="hot" property="valid" value="hot" />
            <Value displayValue="mild" property="valid" value="mild" />
            <Value displayValue="cool" property="valid" value="cool" />
        </DataField>
        <DataField displayName="humidity" name="humidity" isCyclic="0"
optype="categorical">
            <Value displayValue="high" property="valid" value="high" />
            <Value displayValue="normal" property="valid" value="normal"
/>
```

```
        </DataField>
        <DataField displayName="windy" name="windy" isCyclic="0"
optype="categorical">
            <Value displayValue="TRUE" property="valid" value="TRUE" />
            <Value displayValue="FALSE" property="valid" value="FALSE" />
        </DataField>
        <DataField displayName="play" name="play" isCyclic="0"
optype="categorical">
            <Value displayValue="yes" property="valid" value="yes" />
            <Value displayValue="no" property="valid" value="no" />
        </DataField>
    </DataDictionary>
    <TreeModel modelName="Decision Tree Model"
splitCharacteristic="multiSplit" algorithmName="decisionTree"
functionName="classification">
        <MiningSchema>
            <MiningField missingValueTreatment="asMode" name="outlook"
missingValueReplacement="sunny" outliers="asMissingValues"
usageType="active" />
            <MiningField missingValueTreatment="asMode"
name="temperature" missingValueReplacement="mild"
outliers="asMissingValues" usageType="active" />
            <MiningField missingValueTreatment="asMode" name="humidity"
missingValueReplacement="high" outliers="asMissingValues"
usageType="active" />
            <MiningField missingValueTreatment="asMode" name="windy"
missingValueReplacement="FALSE" outliers="asMissingValues"
usageType="active" />
            <MiningField missingValueTreatment="asMode" name="play"
missingValueReplacement="yes" outliers="asMissingValues"
usageType="predicted" />
        </MiningSchema>
        <Node recordCount="14.0" score="yes">
            <True />
            <ScoreDistribution recordCount="9.0" value="yes" />
            <ScoreDistribution recordCount="5.0" value="no" />
            <Node recordCount="5.0" score="no">
                <SimplePredicate operator="equal" value="sunny"
field="outlook" />
                <ScoreDistribution recordCount="2.0" value="yes" />
                <ScoreDistribution recordCount="3.0" value="no" />
                <Node recordCount="3.0" score="no">
                    <SimplePredicate operator="equal" value="high"
field="humidity" />
                    <ScoreDistribution recordCount="0.0" value="yes" />
                    <ScoreDistribution recordCount="3.0" value="no" />
```

```
                </Node>
                <Node recordCount="2.0" score="yes">
                    <SimplePredicate operator="equal" value="normal"
field="humidity" />
                    <ScoreDistribution recordCount="2.0" value="yes" />
                    <ScoreDistribution recordCount="0.0" value="no" />
                </Node>
            </Node>
            <Node recordCount="4.0" score="yes">
                <SimplePredicate operator="equal" value="overcast"
field="outlook" />
                <ScoreDistribution recordCount="4.0" value="yes" />
                <ScoreDistribution recordCount="0.0" value="no" />
            </Node>
            <Node recordCount="5.0" score="yes">
                <SimplePredicate operator="equal" value="rainy"
field="outlook" />
                <ScoreDistribution recordCount="3.0" value="yes" />
                <ScoreDistribution recordCount="2.0" value="no" />
                <Node recordCount="2.0" score="no">
                    <SimplePredicate operator="equal" value="TRUE"
field="windy" />
                    <ScoreDistribution recordCount="0.0" value="yes" />
                    <ScoreDistribution recordCount="2.0" value="no" />
                </Node>
                <Node recordCount="3.0" score="yes">
                    <SimplePredicate operator="equal" value="FALSE"
field="windy" />
                    <ScoreDistribution recordCount="3.0" value="yes" />
                    <ScoreDistribution recordCount="0.0" value="no" />
                </Node>
            </Node>
        </Node>
    </TreeModel>
</PMML>
```

8.5. Другие стандарты Data Mining

8.5.1. 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 Data Mining алгоритмам. 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;
- на Rule — для модели правил;

- на Clustering — для кластерной модели;
- на 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.

8.5.2. Стандарт OLE DB для Data Mining

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 фирмы Microsoft и включен в SQL Server 2000 Analysis Services.

Интерфейс OLE DB for Data Mining 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 2000, 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 OLE DB for Data Mining 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 OLE DB for Data Mining обеспечивают доступ к службам исследования данных, соответствующим мастерам и приложениям независимых производителей. Помимо спецификации OLE DB for Data Mining к числу ключевых элементов среды Data Mining относятся модель исследования данных DSO и процессор Data Mining Engine, который

включает как алгоритм деревьев принятия решений, так и алгоритм кластерного анализа. Analysis Services и любые процессы компаний, совместимые с OLE DB for Data Mining, 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 OLE DB for Data Mining interface, 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 OLE DB for Data Mining.

Наконец, выпустив спецификацию OLE DB for Data Mining, корпорация Microsoft предоставила независимым компаниям возможность применять интерфейсы COM, входящие в состав OLE DB for Data Mining, 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 OLE DB for Data Mining. 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.

8.5.3. Стандарт JDMAPI

Стандарт JDMAPI (Java Data Mining API) в настоящее время разрабатывается группой JSR 73 и должен быть реализован в ближайшее время. 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.

JDMAPI будет основываться на объектно-ориентированной концептуальной модели Data Mining. Он будет совместим со стандартами OMG CWM, SQL/MM for Data Mining и DMG's PMML. Он будет поддерживать четыре концептуальные области, описанные в CWM: настройки, модели, трансформация и результаты. Объектная модель будет обеспечивать уровень ядра сервисов и интерфейсов, которые будут доступны всем клиентам. Клиенты кодируются в соответствии с одинаковыми интерфейсами и семантикой.

Отдельному разработчику объектной модели нет необходимости поддерживать все интерфейсы и сервисы, описанные в JDMAPI. Каждый разработчик сам решает, как реализовать JDMAPI. Некоторые могут реализовывать JDMAPI как "родной API" в своих продуктах, другие — разрабатывать драйверы/адаптеры, которые будут служить в качестве посредников между уровнем ядра JDMAPI и продуктами других разработчиков. Стандарт JDMAPI не предписывает какой-либо определенной стратегии реализации.

Планируется, что JDMAPI будет реализована для платформы J2EE.

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 JDMAPI будет представлять собой спецификацию API-функций для построения моделей Data Mining, извлечения знаний, их использование, а также создание, хранение, доступ и сохранение данных и метаданных, поддерживающих результаты Data Mining и выбор трансформации данных.

ГЛАВА 9

Библиотека Xelopes

9.1. 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. 9.1 представлены основные уровни MDA, отражающие его идеологию.

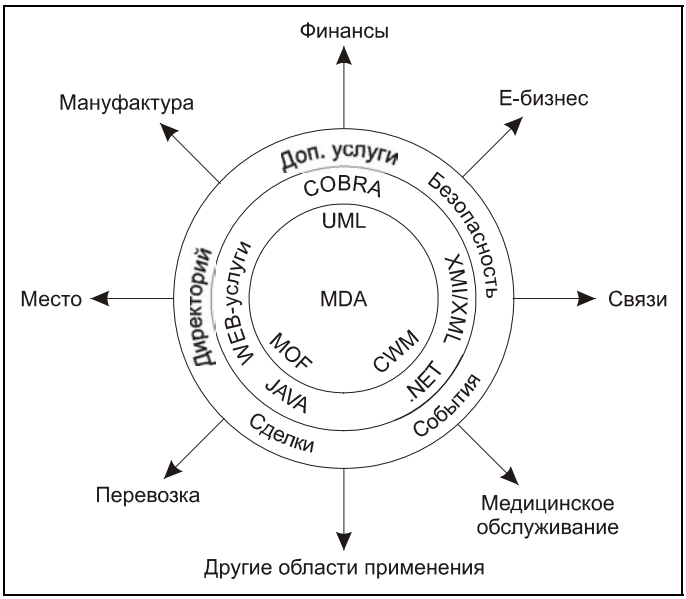

Рис. 9.1. Уровни стандарта MDA

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. 9.2).

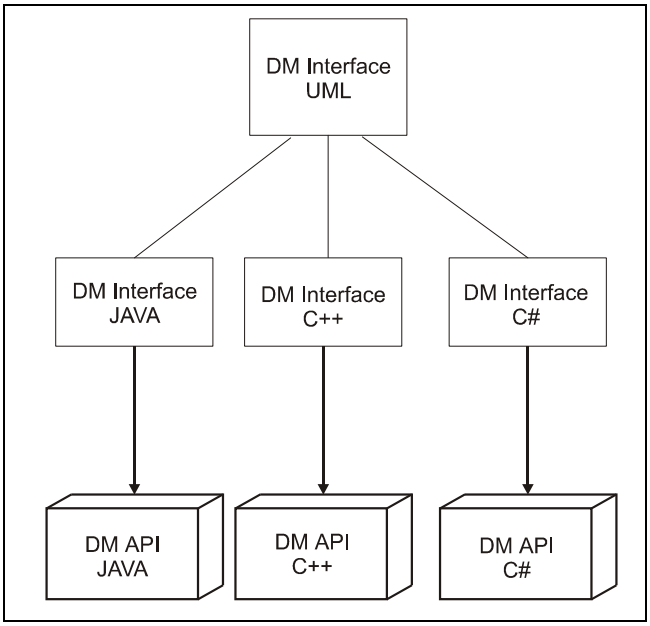

Рис. 9.2. Схема реализации ядра библиотеки Xelopes на трех языках

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 Data Mining-алгоритмам;
- Algoritms-диаграмма, описывающая процесс создания Data Mining-моделей, т. е. алгоритмов 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.

9.2. Диаграмма Model

9.2.1. 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 с классом `SupervisedMiningModel` описываются расширения класса `MiningModel` — моделей, получаемых в результате других методов Data Mining, а именно (рис. 9.3):

- `StatisticsMiningModel` — статистическая модель;
- `AssociationRulesMiningModel` — модель правил связи;
- `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.

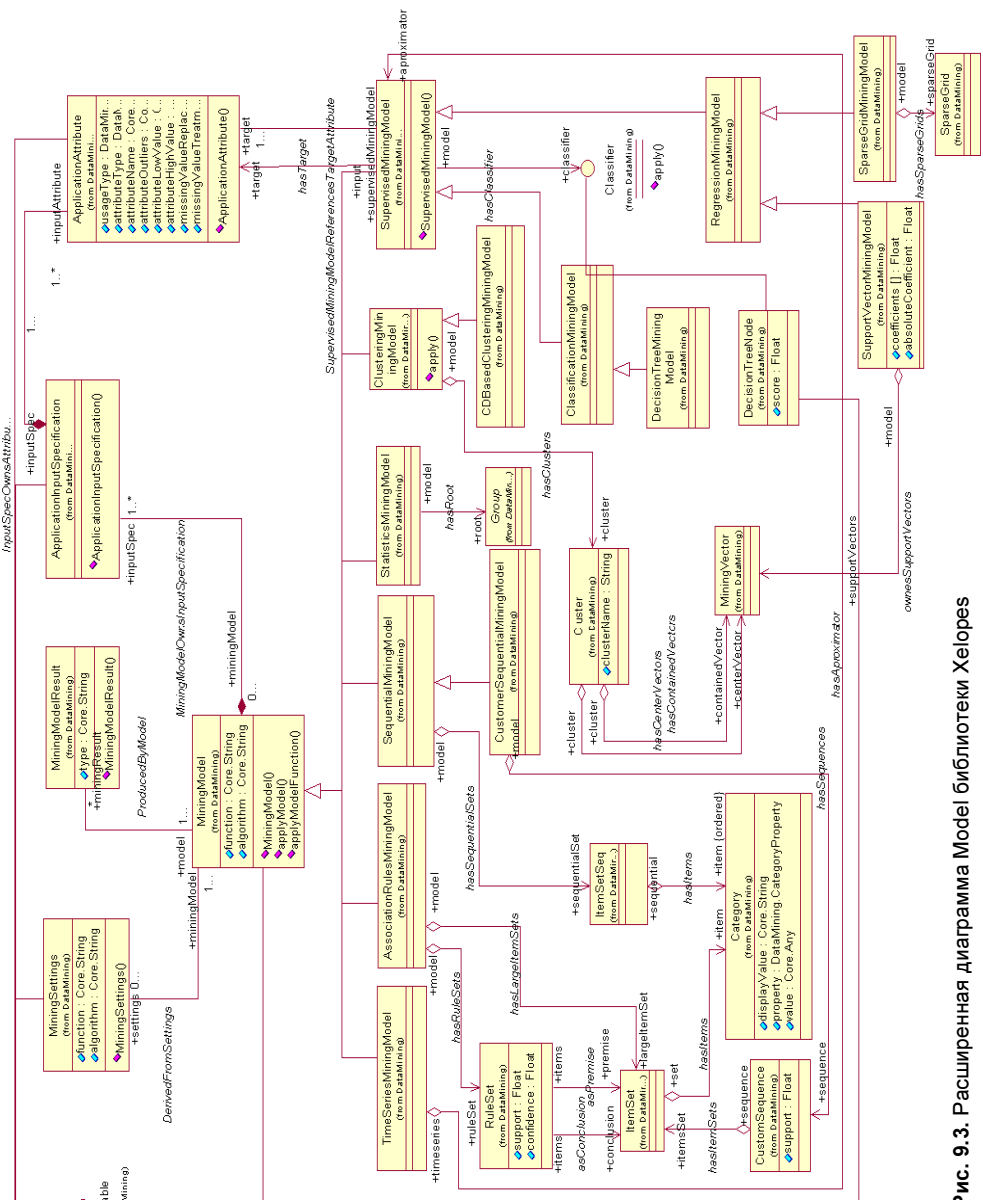

Рис. 9.3. Расширенная диаграмма Model библиотеки Xelopes

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.

9.2.2. 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 реализован всеми основ-

ными элементами данных Xelopes, такими как `MiningAttribute`, `MiningVector`, `MiningInputStream`, `ItemSet` и `RuleSet`.

9.2.3. 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`.

9.3. Диаграмма Settings

9.3.1. 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. 9.4. 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

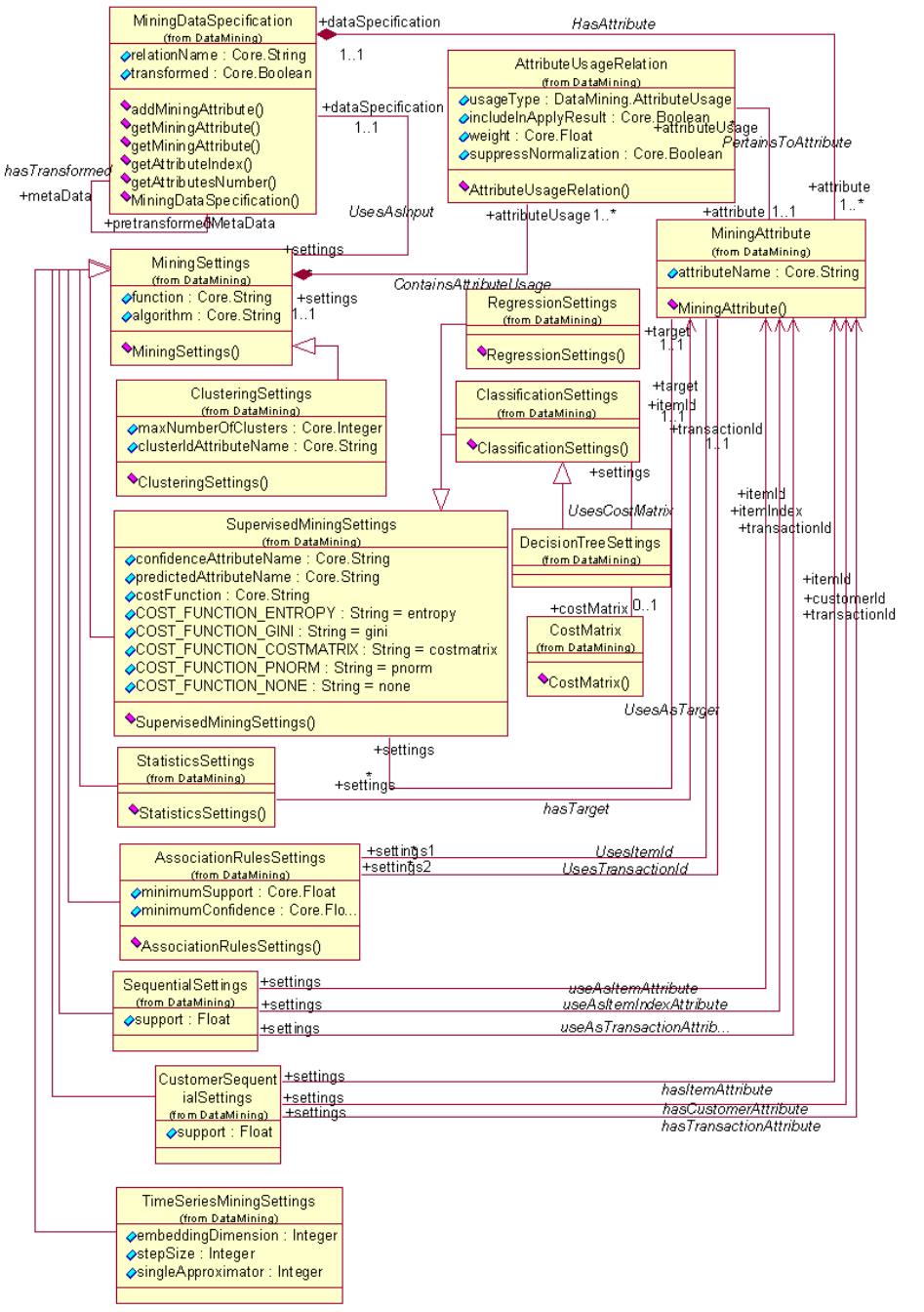

Рис. 9.4. Расширенная диаграмма пакета Settings библиотеки Xelopes

размера временного шага, singleApproximator для использования модели одиночной или множественной регрессии для предсказаний.

9.3.2. 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 булевская 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.

9.4. Диаграмма Attribute

9.4.1. Классы пакета Attribute

Диаграмма Attribute в Xelopes (рис. 9.5) 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 булевскую 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.

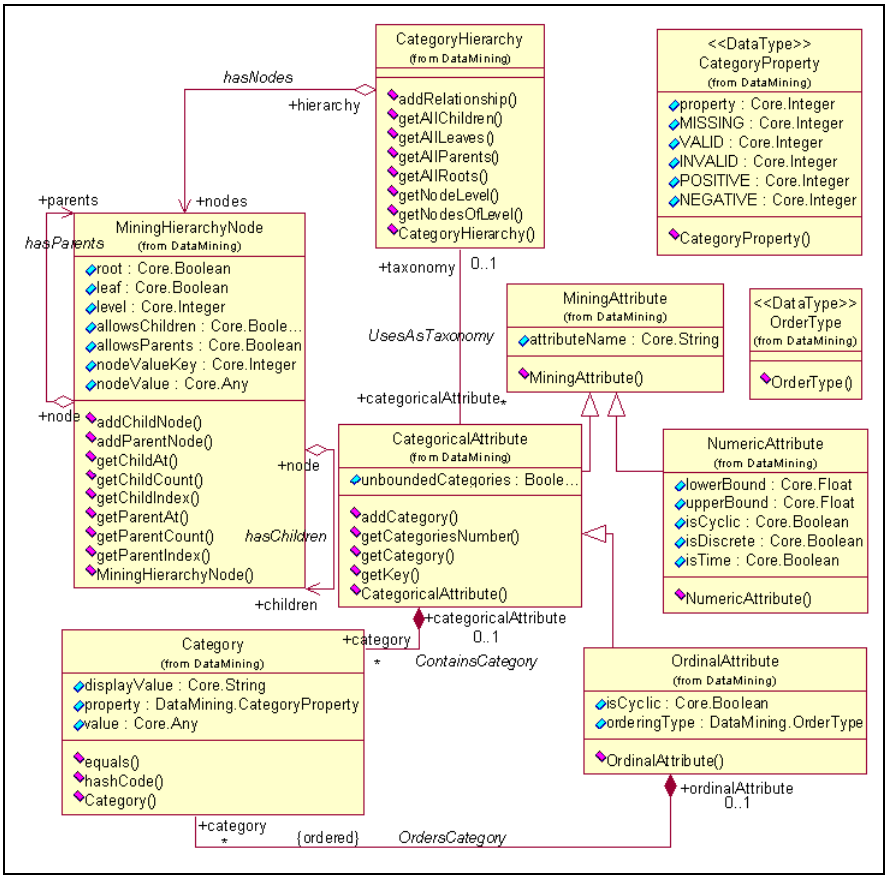

Рис. 9.5. Расширенная диаграмма Attribute Xelopes

9.4.2. 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. 9.6. В примере в правой части рисунка представление уровней, очевидно, не имеет смысла.

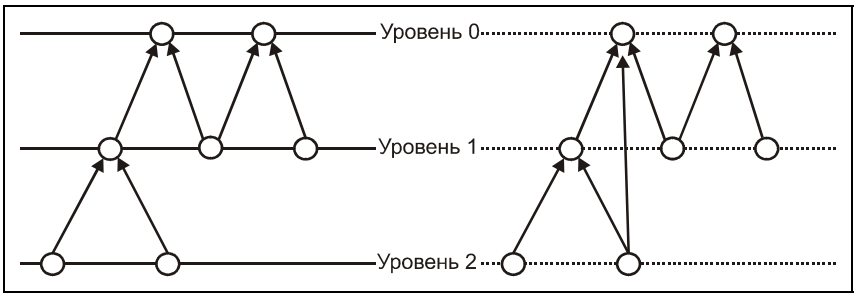

Рис. 9.6. Примеры многоуровневых (слева) и немногоуровневых (справа) DAG

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.

9.5. Диаграмма Algorithms

Диаграмма Algorithms содержит все классы, которые необходимы для выполнения алгоритмов Data Mining.

9.5.1. 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. 9.7.

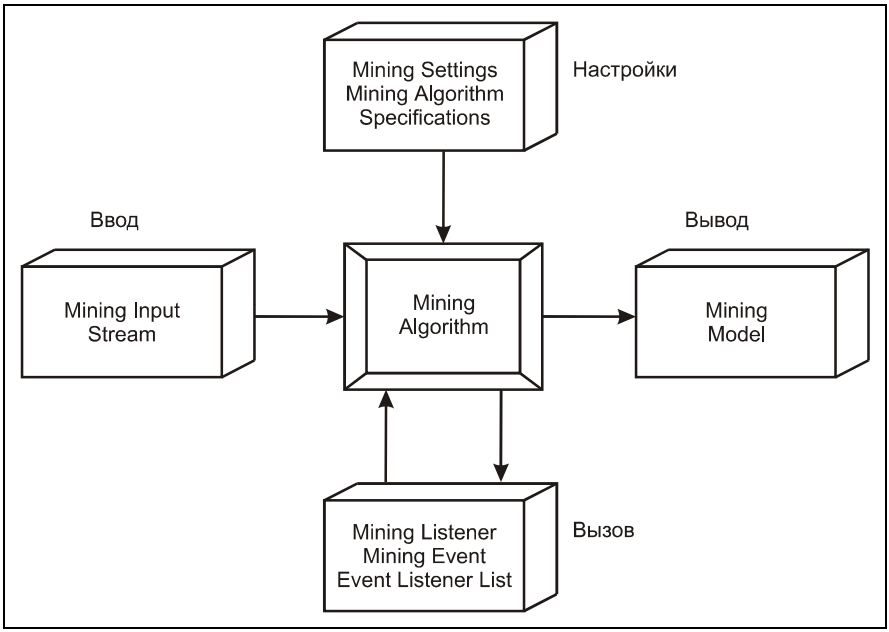

Рис. 9.7. Главные интерфейсы алгоритма Data Mining в Xelopes

9.5.2. 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.

9.5.3. Расширение класса *MiningAlgorithm*

По аналогии с диаграммами модели и настроек, MiningAlgorithm является суперклассом для основных типов алгоритмов Data Mining (рис. 9.8):

- 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.

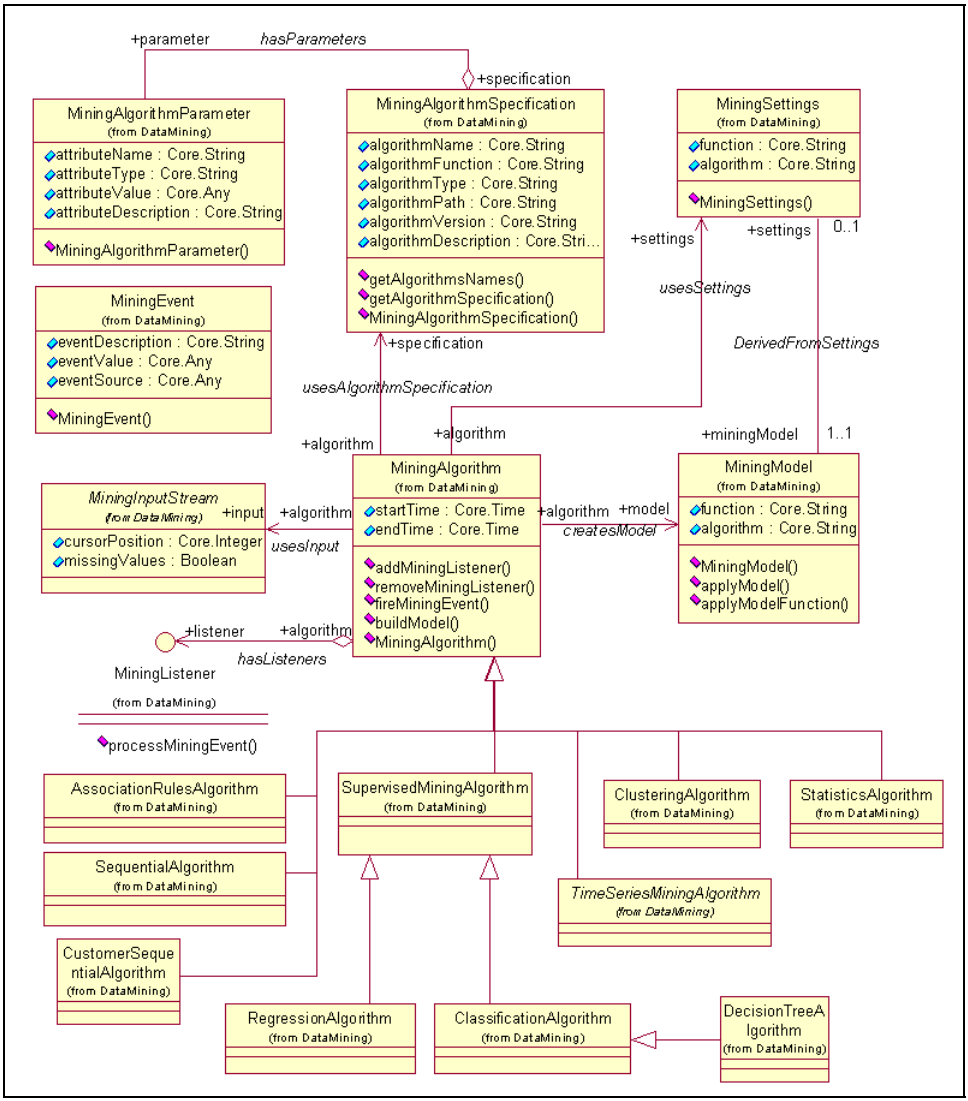

Рис. 9.8. Диаграмма Algorithms Xelopes

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`.

9.5.4. 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.

9.5.5. 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.

9.6. Диаграмма DataAccess

Диаграмма DataAccess (рис. 9.9) содержит все классы, необходимые для доступа к данным из алгоритмов.

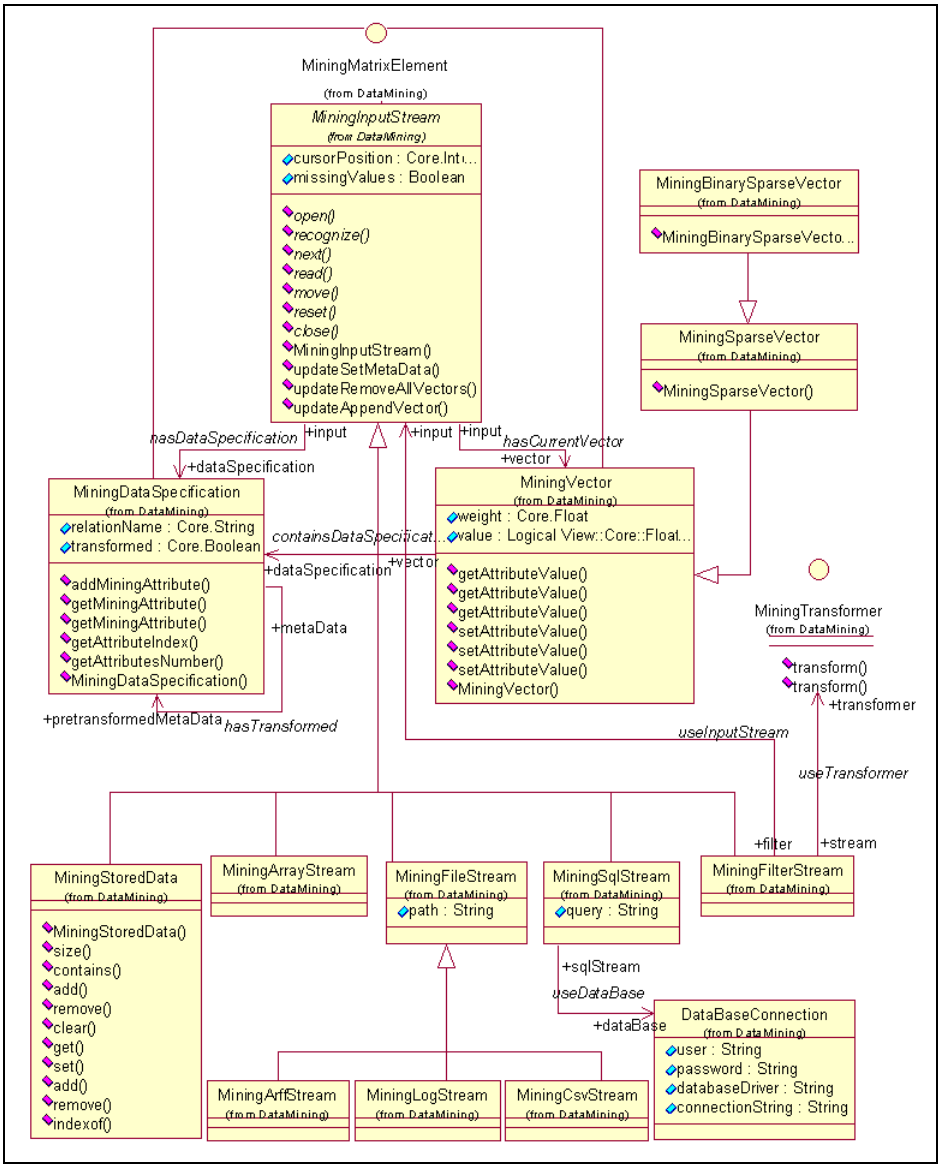

Рис. 9.9. Диаграмма классов пакета DataAccess

9.6.1. 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 нескольким реляционными таблицами или с текстовыми файлами. В связи с этим невозможно использовать стандартные классы, описанные в CWM для работы хранилищ данных.

9.6.2. 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.

9.6.3. 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`.

9.6.4. 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.

9.7. 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. 9.10). 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. 9.11, используют их.

Стратегия пошагового преобразования предполагает, что любое преобразование исходных данных можно представить как композицию *n* последова-

тельных 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.

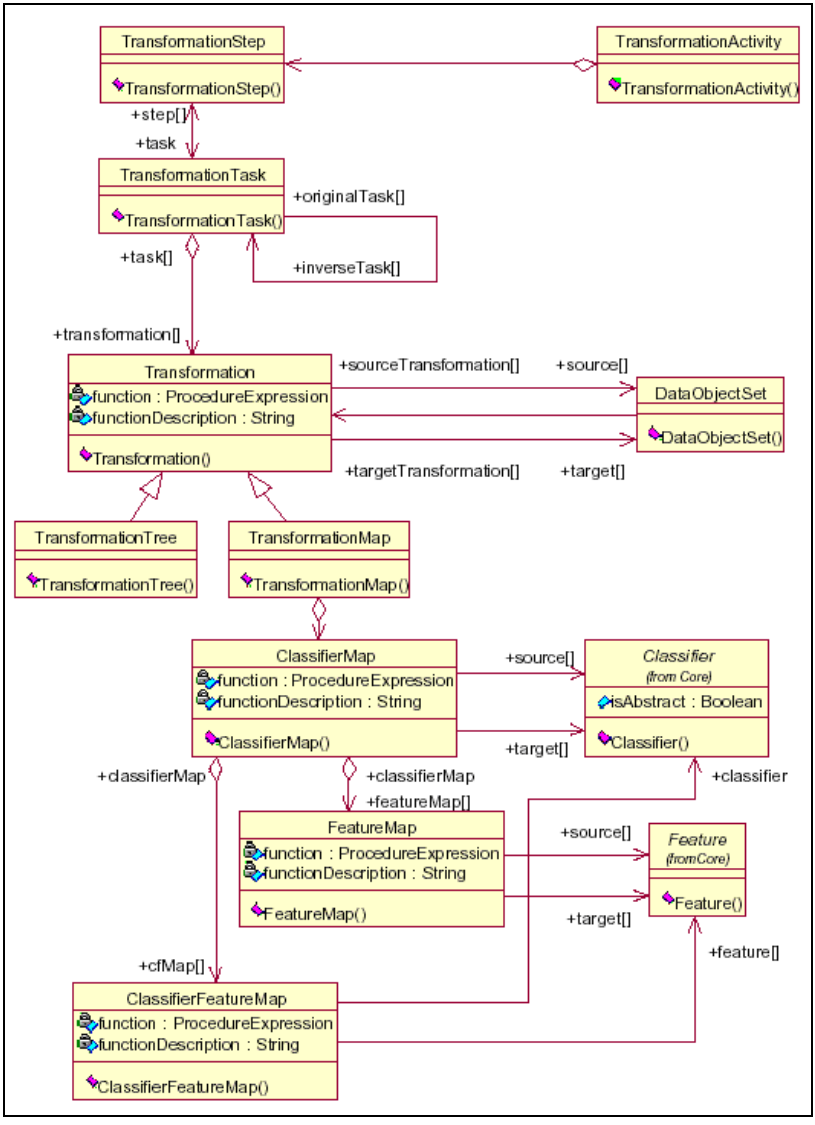

Рис. 9.10. Диаграмма классов пакета Transformation стандарта CWM

В пакете Transformation (рис. 9.12) библиотеки Xelopes для представления шагов и задач преобразования существуют классы `MiningTransformationStep`

и `MiningTransformationTask`. Все шаги, выполняемые по преобразованию, объединяются в классе `TransformationActivity` представляющего собой всю деятельность, связанную с преобразованием.

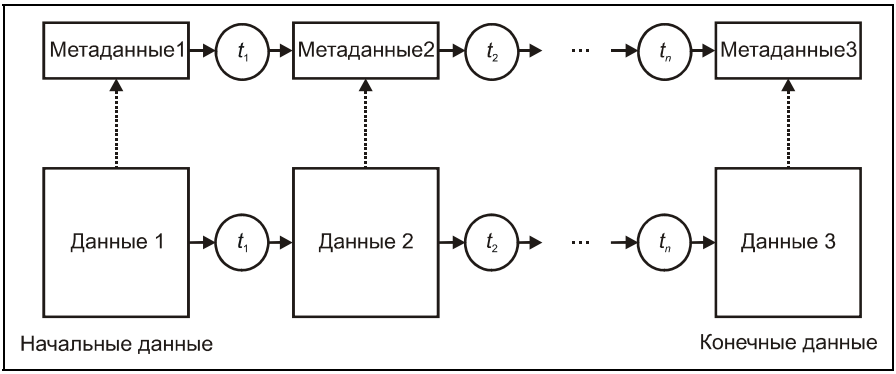

Рис. 9.11. Концепция пошагового преобразования данных

9.8. Примеры использования библиотеки Xelopes

9.8.1. 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");
```

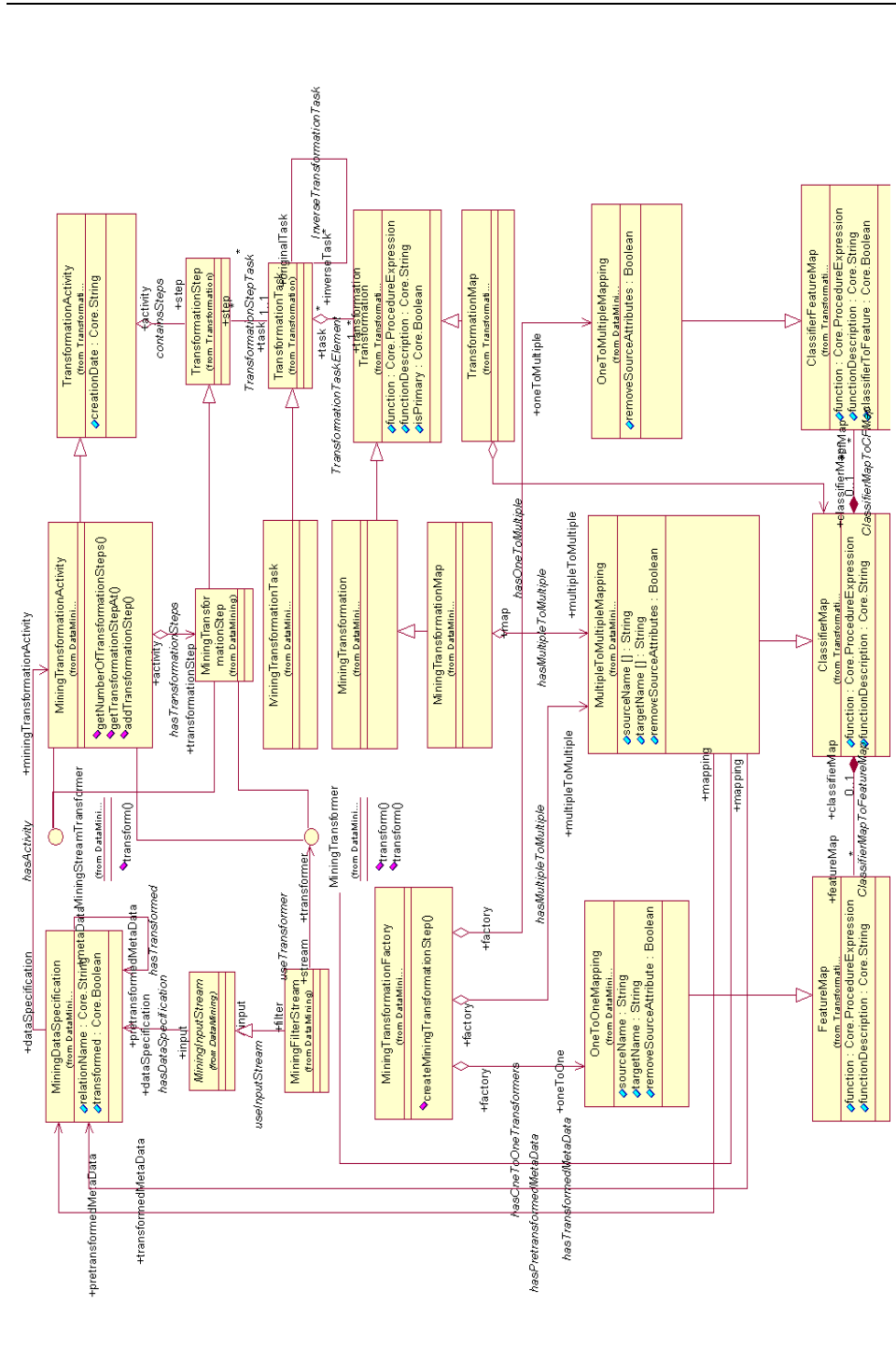

Рис. 9.12. Диаграмма классов пакета Transformation библиотеки Xelopes

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 зависят 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.

9.8.2. 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 ниже 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
```

9.8.3. Решение задачи кластеризации

Рассмотрим решение задачи кластеризации на примере сегментации потребителей. Исходные данные для этой задачи приведены в табл. 9.1 и представляют собой информацию о клиентах туристического агентства.

Таблица 9.1

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 9.1 *(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	17870
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.

9.8.4. Решение задачи классификации

Решим задачу классификацию для данных (табл. 9.2), 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 9.2

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 9.2 *(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. 9.13.

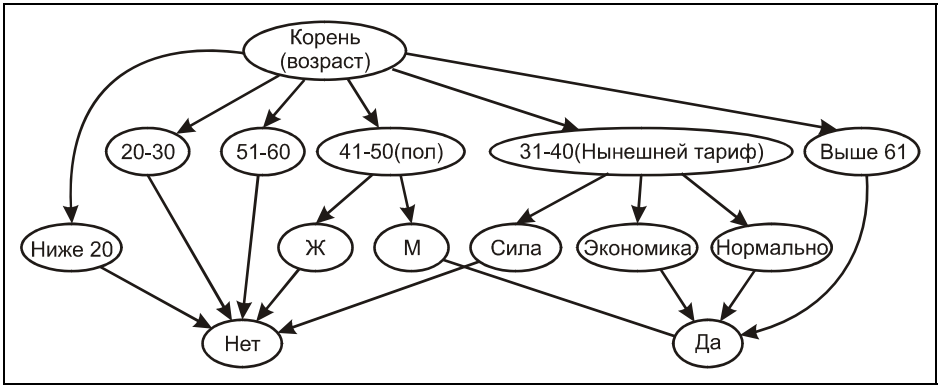

Рис. 9.13. Дерево решений, построенное алгоритмом ID3

Решим задачу для новых данных, представленных в табл. 9.3.

Таблица 9.3

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.

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.

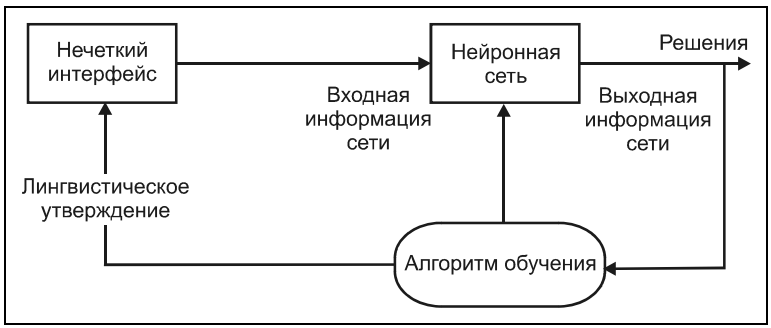

Рис. П1.1. Первая модель нечеткой нейронной системы

Многоуровневая нейронная сеть запускает интерфейсный механизм нечеткой логики.

Основные обрабатываемые элементы нейронной сети называют искусственными нейронами, или просто нейронами. Сигнал с нейронных входов *xj* считается однонаправленным, направление обозначено стрелкой, то же касается нейронного выходного сигнала.

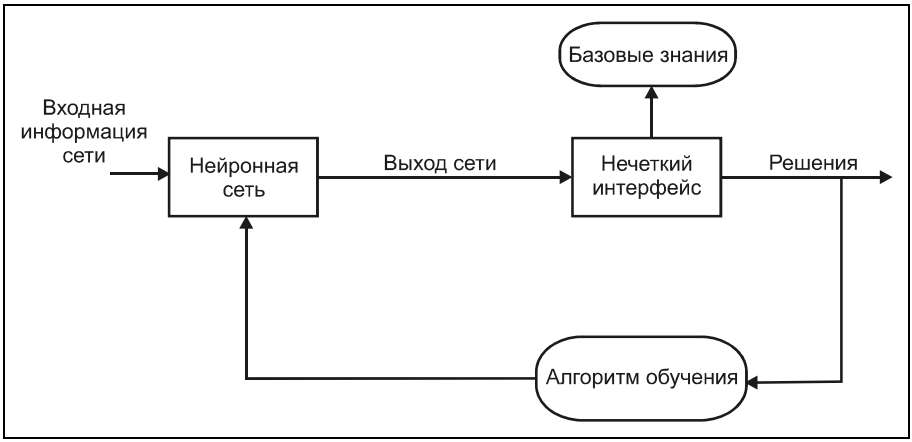

Рис. П1.2. Вторая модель нечеткой нейронной системы

Простая нейронная сеть представлена на рис. П1.3. Все сигналы и веса задаются вещественными числами.

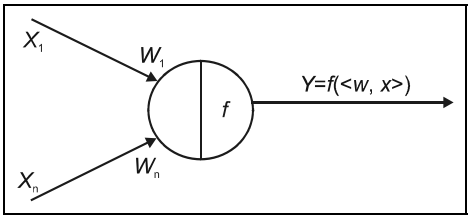

Рис. П1.3. Простая нейронная сеть

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. 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 на то, что все входные, выходные параметры и веса гибридной нейронной сети представляют собой вещественные числа из интервала [0, 1].

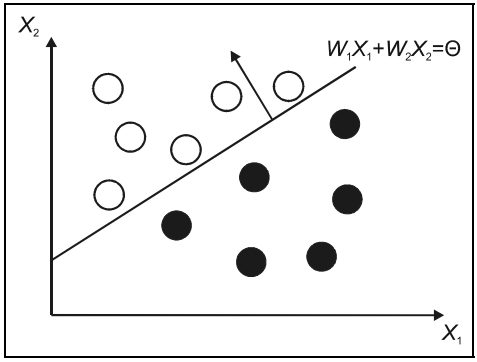

Рис. П1.4. Передаточная функция гибридной нейронной сети

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:

$$y = \max\{w_1 x_1, w_2 x_2\}.$$

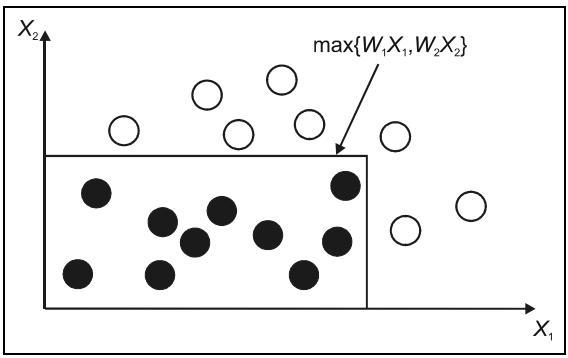

Рис. П1.5. Передаточная функция нечеткого нейрона ИЛИ

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 η — 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

...

$$\Re_n\text{: if } x \text{ is } A_n \text{ and } y \text{ is } B_n \text{ then } z \text{ is } C_n$$
$$\text{fact: } x = x_0 \text{ and } y = y_0$$
$$\text{consequence: } z \text{ is } C$$

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 и Такаги используют следующие правила:

$$\Re_1\text{: if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z_1 = a_1 x + b_1 y;$$
$$\Re_2\text{: if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z_2 = a_2 x + b_2 y.$$

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 β_1 и β_2 — нормированные значения α_1 и α_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 Sugeno на следующем примере:

$$\text{if } x \text{ is SMALL and } y \text{ is BIG then } o = x - y;$$
$$\text{if } x \text{ is BIG and } y \text{ is SMALL then } o = x + y;$$
$$\text{if } x \text{ is BIG and } y \text{ is BIG then } o = x - 2y,$$

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:

$$o_2 = x_0 + y_0 = 3 + 3 = 6.$$

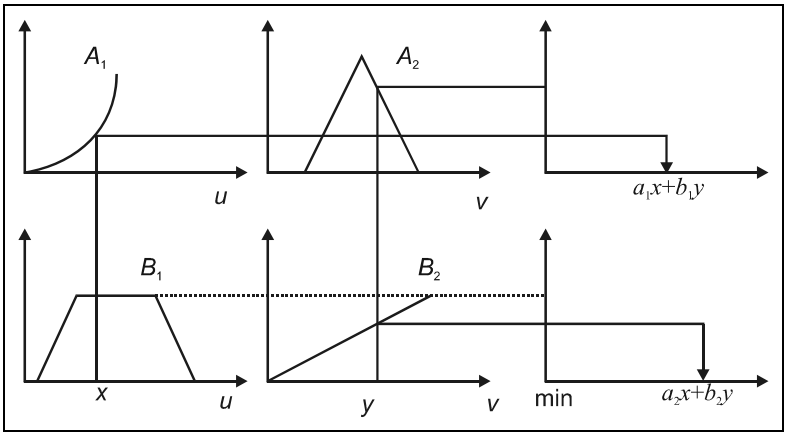

Рис. П1.6. Выходная информация правила

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.

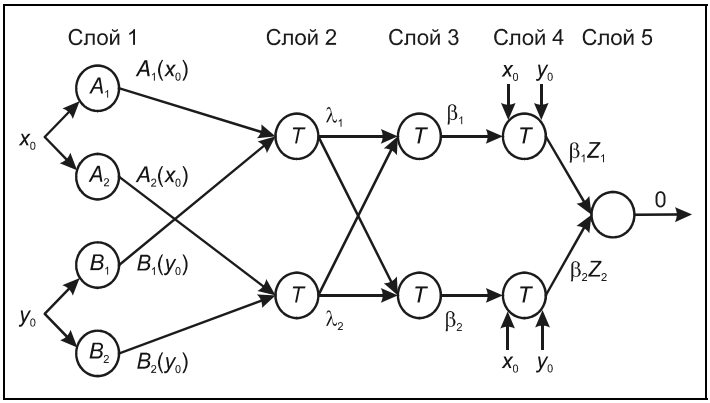

Рис. П1.7. Гибридная нейронная сеть

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 осуществляется операция 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 границу:

$$\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}.$$

- **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 пра-вила:

$$\beta_2 z_2 = \beta_2(a_2 x_0 + b_2 y_0).$$

- **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.

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 $k = 1, \ldots, K$, т. е. имеется следующая обучающая последовательность:

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}.$$

Для моделирования неизвестного отображения f применим упрощенное нечеткое правило IF-THEN следующего вида:

$$\Re_i\text{: if } x_1 \text{ is } A_{i1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{in} \text{ then o} = z_i$$

$i = 1, \ldots, m$, 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, что firing-уровень i-го правила, обозначенный через α_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.

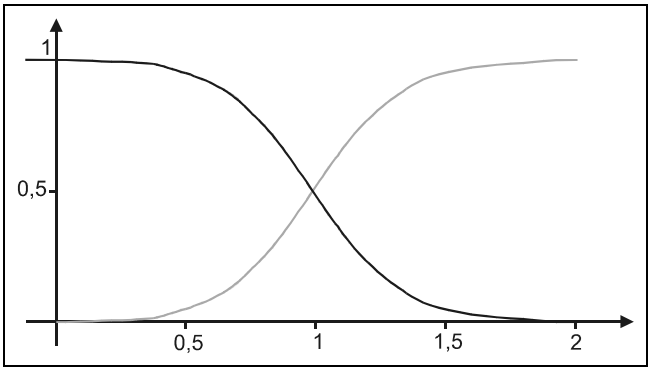

Рис. П1.8. Симметричная функция принадлежности

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 выше 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:

$$o = \frac{A_1(x) z_1 + A_2(x) z_2}{A_1(x) + A_2(x)}.$$

Весовые коэффициенты определяются следующим образом:

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E}{\partial z_1} = z_1(t) - \eta (o - y) A_1(x),$$

$$z_2(t+1) = z_2(t) - \eta \frac{\partial E}{\partial z_2} = z_2(t) - \eta (o - y) A_2(x^k),$$

$$a(t+1) = a(t) - \eta \frac{\partial E(a,b)}{\partial a},$$

$$b(t+1) = b(t) - \eta \frac{\partial E(a,b)}{\partial b},$$

где

$$\frac{\partial E(a,b)}{\partial a} = (o - y) \frac{\partial o^k}{\partial a} = (o - y) \frac{\partial}{\partial a} \left[z_1 A_1(x) + z_2 A_2(x) \right] =$$

$$= (o - y) \frac{\partial}{\partial a} \left[z_1 A_1(x) + z_2 (1 - A_1(x)) \right] = (o - y)(z_1 - z_2) \frac{\partial A_1(x)}{\partial a} =$$

$$= (o - y)(z_1 - z_2) b\, A_1(x) A_2(x);$$

$$\frac{\partial E(a,b)}{\partial b} = (o - y)(z_1 - z_2) \frac{\partial A_1(x)}{\partial b} = -(o - y)(z_1 - z_2)(x - a) A_1(x) A_2(x).$$

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.

if x_1 is SMALL and x_2 is SMALL then o = $ax_1 - bx_2$;
if x_1 is SMALL and x_2 is BIG then o = $ax_1 + bx_2$;
if x_1 is BIG and x_2 is SMALL then o = $bx_1 + ax_2$;
if x_1 is BIG and x_2 is BIG then o = $bx_1 - ax_2$,

где функции принадлежности нечетких чисел SMALL и BIG задаются так:

$$\mathrm{SMALL}(\upsilon) = \begin{cases} 1 - \dfrac{\upsilon}{2}, & \text{if } 0 \le \upsilon \le 2 \\ 0 & \text{otherwise;} \end{cases},$$

$$\mathrm{BIG}(\upsilon) = \begin{cases} 1 - \dfrac{2 - \upsilon}{2}, & \text{if } 0 \le \upsilon \le 2 \\ 0 & \text{otherwise,} \end{cases},$$

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:

α_1 = SMALL(1) $\wedge$ SMALL(1) = 0,5;
α_2 = SMALL(1) $\wedge$ BIG(1) = 0,5;
α_3 = BIG(1) $\wedge$ SMALL(1) = 0,5;
α_4 = BIG(1) $\wedge$ 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 = SMALL, A_2 = MEDIUM, а A_3 = BIG имеют треугольную форму функций (см. рис. 4.16, *гл. 4*):

$$A_1(\upsilon) = \begin{cases} 1, & \text{if } \upsilon \le c_1 \\ \dfrac{c_2 - x}{c_2 - c_1}, & \text{if } c_1 \le \upsilon \le c_2 \\ 0 & \text{othewise,} \end{cases},$$

$$A_2(\upsilon) = \begin{cases} \dfrac{x - c_1}{c_2 - c_1}, & \text{if } c_1 \le \upsilon \le c_2 \\ \dfrac{c_3 - x}{c_3 - c_2}, & \text{if } c_2 \le \upsilon \le c_3 \\ 0 & \text{othewise,} \end{cases},$$

$$A_3(u)=\begin{cases}1, & \text{if } u \ge c_3\\ \dfrac{x-c_2}{c_3-c_2}, & \text{if } c_2 \le u \le c_3\\ 0 & \text{othewise.}\end{cases}.$$

Примените метод наискорейшего спуска для настройки исходных параметров $\{c_1, c_2, c_3\}$ и консеквентные параметры $\{y_1, y_2, y_3\}$.

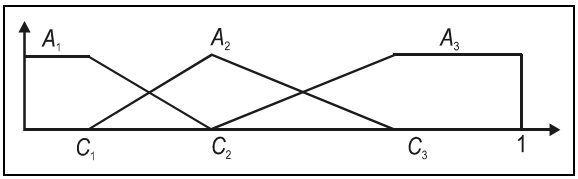

Рис. П1.9. Лингвистические термины A_1, A_2 и A_3

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 α_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.

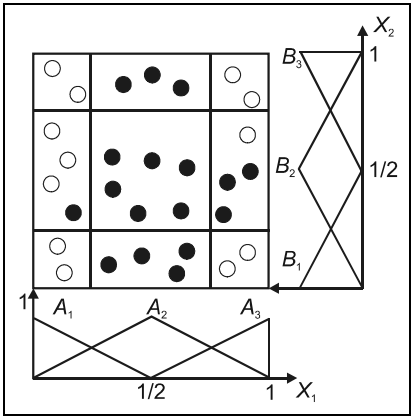

Рис. П1.10. Исходное нечеткое деление с 9 нечеткими подпространствами и 2 неправильно классифицированными моделями.
Закрашенные и пустые кружки отображают данные модели класса 1 и класса 2 соответственно

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:

$\mathfrak{R}_1$: if x_1 is SMALL and x_2 is BIG then $x_p = (x_1, x_2)$ belongs to Class C_1;

$\mathfrak{R}_2$: if x_1 is SMALL and x_2 is MEDIUM then $x_p = (x_1, x_2)$ belongs to Class C_1;

$\mathfrak{R}_3$: if x_1 is SMALL and x_2 is SMALL then $x_p = (x_1, x_2)$ belongs to Class C_1;

$\mathfrak{R}_4$: if x_1 is BIG and x_2 is SMALL then $x_p = (x_1, x_2)$ belongs to Class C_1;

$\mathfrak{R}_5$: if x_1 is BIG and x_2 is BIG then $x_p = (x_1, x_2)$ belongs to Class C_1;

$\mathfrak{R}_6$: 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.

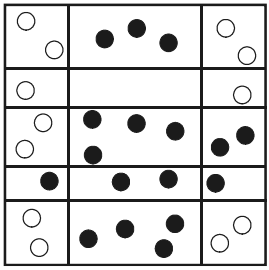

Рис. П1.11. Пример нечетких делений

Sun и Jang 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.

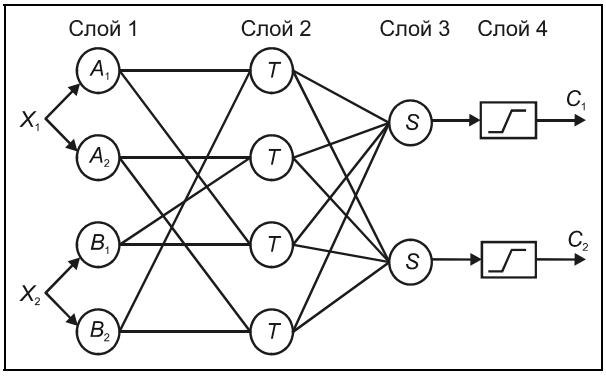

Рис. П1.12. Архитектура нечеткого классификатора

- **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 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 *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.

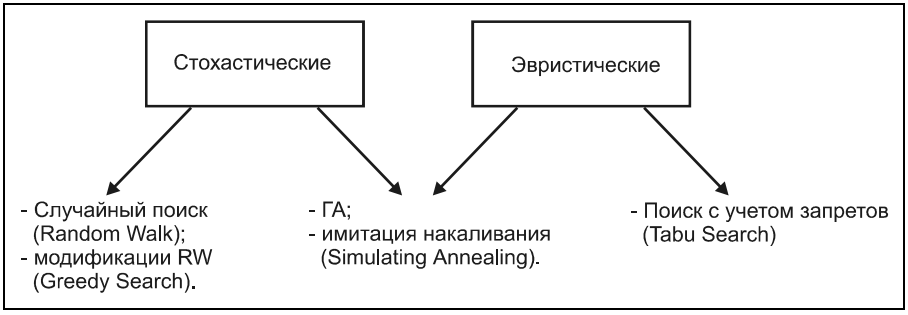

Рис. П2.1. Классификация эвристических и стохастических алгоритмов

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).

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.

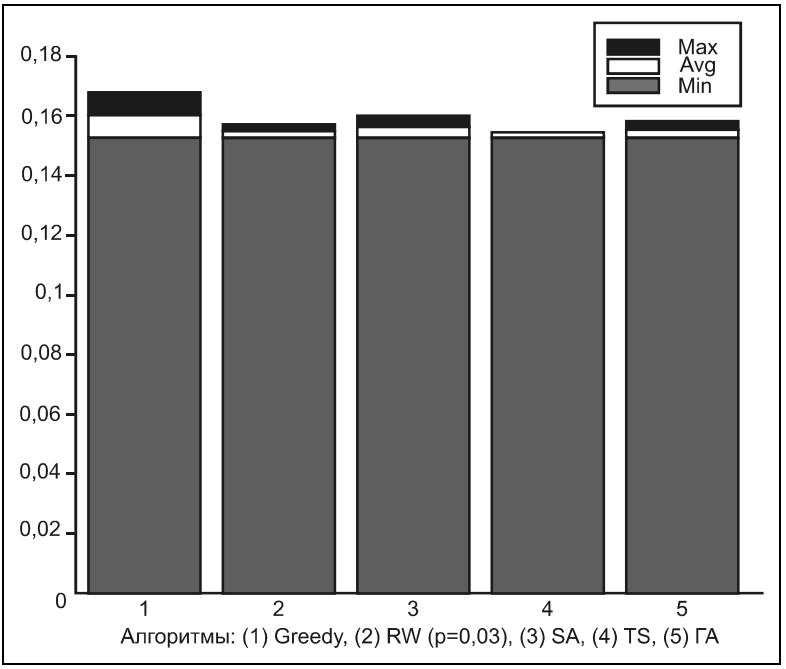

Рис. П2.2. График зависимостей качества решений задачи от числа шагов выполнения алгоритмов

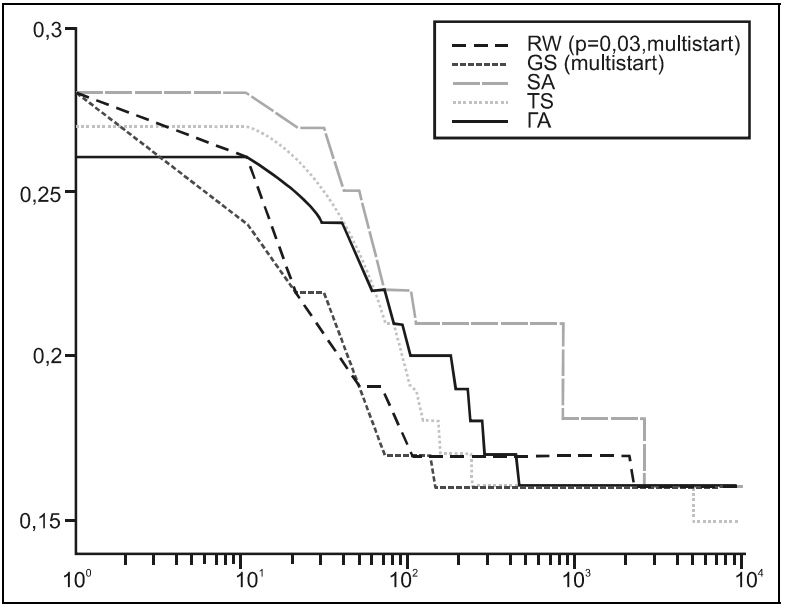

Рис. П2.3. Динамика получения результата

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 выше 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 λ.
Шаг 2. Вычисление пригодности $F(x_i)$ $i = 1, \ldots, \lambda$.
Шаг 3. Селекция (отбор) $\mu < \lambda$ лучших индивидов.
Шаг 4. Создание λ / μ потомков каждого из μ 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 произвести 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, …, SP_N.
Шаг 2. Формирование структуры подпопуляций.
Шаг 3. Для SPi , i = 1, …, 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.

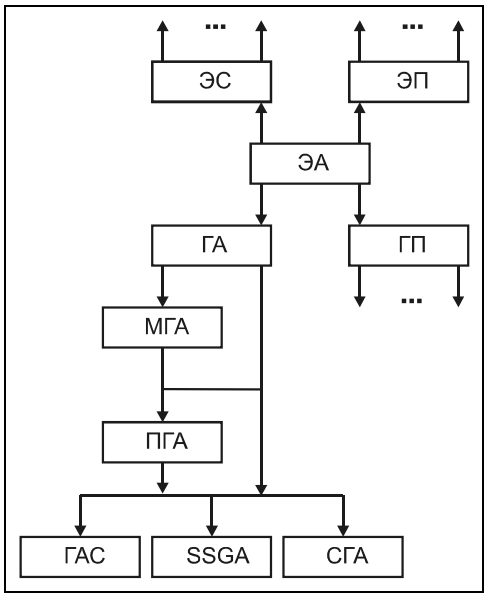

Рис. П2.11. Классификация эволюционных алгоритмов

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 это 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];

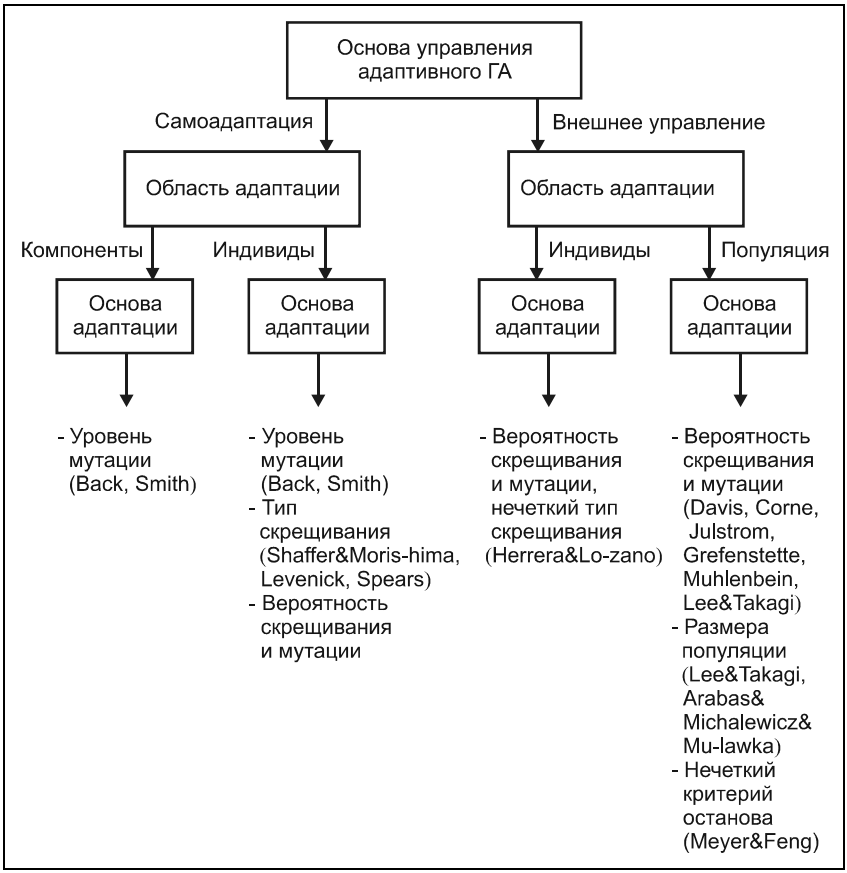

Рис. П2.12. Классификация генетических алгоритмов

- 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 су-

ществовать 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 заключаются 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).

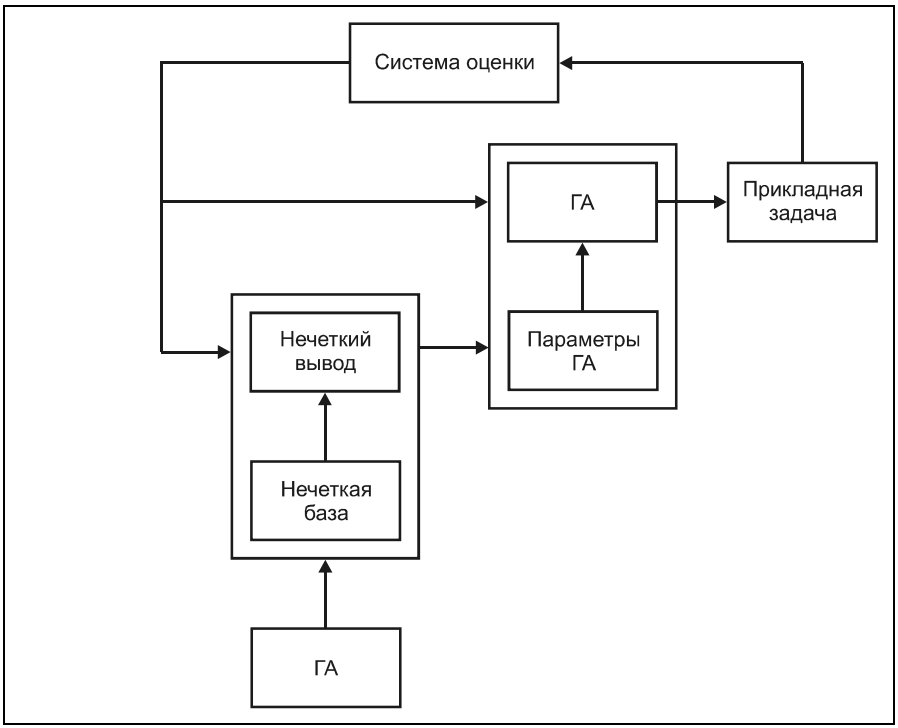

Рис. П2.13. Схема организации ГА с функцией адаптации (обратной связи)

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.

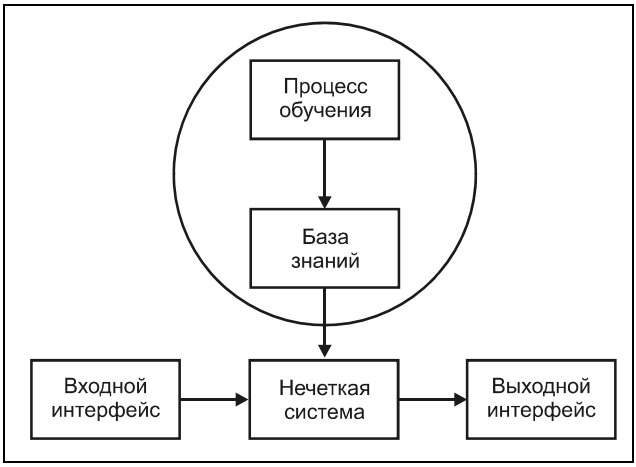

Рис. П2.14. Использование ГА для оптимизации задач

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

   $$ED = (d_{\text{ср}} - d_{\min}) / (d_{\max} - d_{\min}),$$

   где $d_{\text{ср}}$, $d_{\min}$, $d_{\max}$ — среднее, минимальное и максимальные значения отклонений эффективности индивидов Ci от лучшего значения в популяции P размером N:

   $$d_{\text{ср}} = (1/N) \sum_{i=1}^{N} d\,(C_{\text{best}}, Ci),$$

   $$d_{\max} = \max\{d(C_{\text{best}}, Ci)|, Ci \in P\},$$

   $$d_{\min} = \min\{d(C_{\text{best}}, Ci)|, Ci \in P\}.$$

   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 — $PDM1 = f_{\text{best}} / f_{\text{ср}}$.

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 δp_e и $\delta\eta_{\min}$, вероятности скрещивания p_e и интенсивности 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.

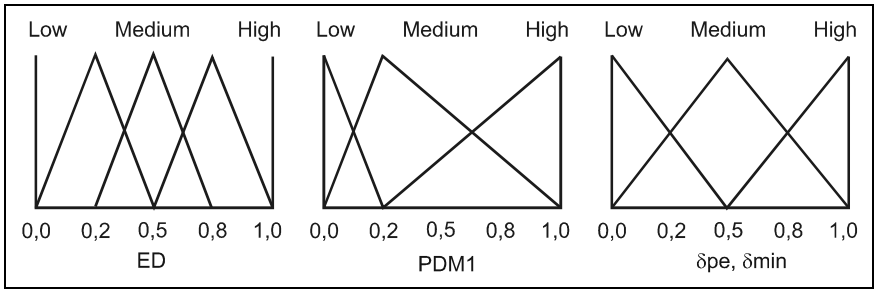

Рис. П2.15. Лингвистические термы

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].

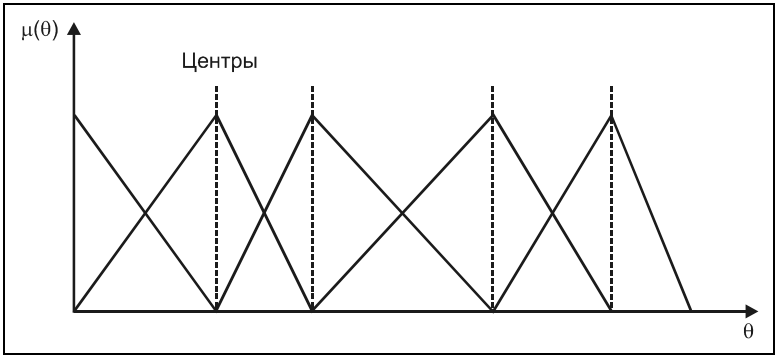

Рис. П2.16. Центры функции принадлежности

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

P_e	**PDM1**		
ED	Low	Medium	High
Low	Small	Small	Medium
Medium	Big	Big	Medium
High	Big	Big	Medim
η_{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 использованием 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 используется 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-v2-0.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 Data Mining алгоритмов;
- № 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**. Ниже указан состав прилагаемой библиотеки:

- javadoc — документация к Java-версии библиотеки Xelopes;
- XELOPES_Java.zip — библиотека Xelopes;
- XelopesGuiLast.zip — графический интерфейс к библиотеке Xelopes;
- Xelopes1.1_Doc.pdf — документация к библиотеке Xelopes.

Полное описание структуры диска приведено в табл. П3.1.

Таблица П3.1

Папка	Содержание
\Acrobat	Дистрибутив Acrobat Reader для чтения файлов в PDF-формате
\Drivers	Драйверы и библиотеки, необходимые для работы GUI-интерфейса библиотеки Xelopes
\JDK 1.4	Дистрибутив Java версии 1.4
\Labs	Лабораторный практикум по алгоритмам Data Mining
\Standard	Стандарты Data Mining
\Xelopes	Свободно распространяемая библиотека алгоритмов Data Mining от компании Prudsys
\Diagrams	UML-диаграммы классов стандарта CWM и библиотеки Xelopes

Для инсталляции библиотеки Xelopes и GUI-интерфейса к ней необходимо выполнить следующую последовательность действий:

1. Проверить наличие следующих программ, установленных в системе:
   - Borland JBuilder (версии 7 и выше);
   - j2sdk1.4.2_01;
   - j2re1.4.2_01;
   - DirectX (версии 8 и выше).
2. Скопировать каталоги javadoc, jgrash-2.1-java1, XELOPES UML Sheme, XelopesGuiLast на жесткий диск.
3. Открыть каталог jgrash-2.1-java1, в котором находятся три подкаталога: ChartText, jgrash-2.1-java1.4, jgrash-examples-2.0 и приложение java3d-1_3_1-beta-windows-i586-directx-sdk. Запустить это приложение.
4. Откроется окно установки программы Java 3D 1.3.1-beta (DirectX) SDK, для продолжения нажать кнопку **Next**.
5. Ознакомиться с лицензионным соглашением на программу Java 3D 1.3.1-beta (DirectX) SDK, нажать кнопку **Yes**.
6. Прочитать текст, содержащий информацию о том, какие минимальные требования необходимы для установки программы Java 3D 1.3.1-beta (DirectX) SDK и какие ошибки могут возникнуть при установке, нажать кнопку **Next**.
7. Выбрать каталог для установки программы, нажать кнопку **Yes**.

Программа установки дальше сама проведет инсталляцию.

Для запуска и настройки GUI-интерфейса необходимо выполнить следующие шаги:

1. Открыть каталог XelopesGuiLast и запустите файл XelopesGui.jpx, программа Borland JBuilder откроет этот файл;
2. В Borland JBuilder на панели задач выполнить **Project | Project Properties**.
3. В диалоговом окне **Project Properties** на вкладке **Paths** в поле **JDK** установить JDK-a java 1.4.2_01-XXX, в случае, если поле не содержит данную установку, нажать **Select a JDK** и **NEW...**
4. В диалоговом окне **New JDK Wizard** открыть поле **Existing JDK home path**, в появившемся окне указать папку j2sdk1.4.2_01 и нажать кнопку **OK**.

5. Подождать...
6. В поле **Name for this JDK** диалогового окна **New JDK Wizard** появится название JDK-a java 1.4.2_01-XXX. Нажать кнопку **OK**.
7. В диалоговом окне **Select a JDK** выбрать JDK-a java 1.4.2_01-XXX, затем нажать **OK**.
8. В диалоговом окне **Project Properties** нажать **OK**.
9. Откомпилировать проект (комбинацией клавиш <Ctrl>+<F9>) и запустить его на выполнение (клавишей <F9>).

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. http://intsys.msu.ru.

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. D.H. Wolpert, W.G. Macready. 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. W. H. Inmon 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 Lad-

kranbang, 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, Tel. +49.531.391--3155, Fax: +49.531.391-5936, D.Nauck@tu-bs.de, www.cs.tu-bs.de/~nauck.
32. Clustering Methods. Applications of Multivariate Statistical Analysis. Jiangsheng Yu, ©Institute of Computational Linguistics, Peking University, Beijing, 100871, yujs@pku.edu.cn, http://icl.pku.edu.cn/yujs.
33. Fuzzy Clustering. Hard-c-Means, Fuzzy-c-Means, Gustafson-Kessel. Olaf Wolkenhauer, Control Systems Centre, o.wolkenhauer@umist.ac.uk, www.csc.umist.ac.uk/people/wolkenhauer.htm.
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, ichi@ie.osakafu-u.ac.jp, 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, ichi@ie.osakafu-u.ac.jp, 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, lzhang@cs.unr.edu, 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.

Предметный указатель

1

1-rule 100
1R-алгоритм 100

C

Cluster 150
Conference on Data Systems Languages 17

D

Data Base Task Group 17
Data Mart 32
Data Mining 16, 67
◊ churn detection 77
◊ descriptive model 81
◊ fraud detection 76, 77, 80
◊ java 59, 210, 237
◊ machine learning 69
◊ predictive model 80
◊ SQL Server 2000 235
◊ supervised learning 69
◊ unsupervised learning 69
◊ web usage mining 76
◊ базовые методы 83
◊ биоинформатика 78
◊ генетические алгоритмы (ГА) 86
◊ задачи 68
◊ машинное обучение 69
◊ нейронные сети 88
◊ нечеткая логика 83
◊ обучение без учителя 69
◊ обучение с учителем 69
◊ описательные модели 81
◊ очистка данных 42
◊ практическое применение 76
◊ предсказательные модели 80
◊ сиквенциальный анализ 77, 132
◊ стандарт CRISP 218
◊ стандарт CWM 209
◊ стандарт JDMAPI 237
◊ стандарт OLE DB 235
◊ стандарт PMML 225
◊ стандарт SQL/MM 233
Dendrograms 158

E

ETL 37
Executive Information Systems 21

H

Hard-c-means 162

I

Information Management System 17
Integrated Database Management System 17

M

Multi-dimensional conceptual view 49
◊ dimensions 50
◊ drill up 52
◊ measures 50
◊ rotate 51
◊ slice 51

N

Neighborhood Search 303

O

OLAP 16, 53
◊ HOLAP 65
◊ MOLAP 59
◊ ROLAP 62
◊ архитектура системы 58
◊ схема "звезда" 62
◊ схема "снежинка" 62
OLE DB 209, 235, 243
OLTP 16, 21

S

SQL 16, 61, 143, 243
Support Vector Machines (SVM) 121

А

Агломеративные алгоритмы 159
Агрегированные данные 35
Алгоритм:
◊ Apriori 141
◊ AprioriTid 146
◊ C4.5 110
◊ ID3 107
◊ *k*-means 162
◊ Naive Bayes 101
◊ покрытия 112
Ассоциативные правила:
◊ достоверность (confidence) 139
◊ поддержка (support) 139
◊ улучшение (improvement) 140

Б

Базовый генетический алгоритм 304
Библиотека Xelopes 241

В

Витрина данных (ВД) 32

Д

Дендограмма 158
Деревья решений 97
Дивизимные алгоритмы 161

И

Информационные потоки 37
Информационные системы руководства 21

К

Карта Кохонена 124
Классификационные правила 97
Кластер 150
Кластеризация по Гюстафсону-Кесселю 168
Концепция Захмана 36

М

Математическая функция 99
Меры близости 154
◊ Евклидово расстояние 155
◊ пиковое расстояние 155
◊ расстояние Махаланобиса 155
◊ расстояние по Хеммингу 155
◊ расстояние Чебышева 155
Метаданные 36
Метод поиска соседей 303
Механизм управления транзакциями 20
Многомерная модель данных 49
◊ вращение 51
◊ консолидация 52
◊ мера 50
◊ оси измерений 50
◊ срез 51
Многомерная модель данных и гиперкуб 50

Н

Нейронная сеть:
◊ метод рассуждений Sugeno 281
◊ нечеткий нейрон 277
Неоднородная функция распределения вероятности (НФРВ) 311
Нечеткие бинарные отношения 174

О

Оперативные источники данных (ОИД) 27
Очистка данных 39

П

Правила Кодда 17, 53
Профайлинг данных 42
Процесс переноса данных 37

Р

Реляционная БД 17

С

СППР 13, 22
СУБД 16

Т

Теория нормализации БД 19
Тест FASMI 57
Транзакция 20

Х

Хранилище данных (ХД) 27
◊ концепция 45
Хэш-дерево 145

Э

Эволюционные алгоритмы 304
Эвристические методы 300