数据挖掘：概念与技术

韩家炜

Data Mining: Concepts and Techniques

J. Han and M. Kamber

Morgan Kaufmann

2000

目录

第一章 引言

本书是一个导论，介绍什么是数据挖掘，什么是数据库中知识发现。书中的材料从数据库角度提供，特别强调发现隐藏在大型数据集中有趣数据模式的数据挖掘基本概念和技术。所讨论的实现方法主要面向可规模化的、有效的数据挖掘工具开发。本章，你将学习数据挖掘如何成为数据库技术自然进化的一部分，为什么数据挖掘是重要的，以及如何定义数据挖掘。你将学习数据挖掘系统的一般结构，并考察挖掘的数据种类，可以发现的数据类型，以及什么样的模式提供有用的知识。除学习数据挖掘系统的分类之外，你将看到建立未来的数据挖掘工具所面临的挑战性问题。

1.1 什么激发数据挖掘？为什么它是重要的？

需要是发明之母。

近年来，数据挖掘引起了信息产业界的极大关注，其主要原因是存在大量数据，可以广泛使用，并且迫切需要将这些数据转换成有用的信息和知识。获取的信息和知识可以广泛用于各种应用，包括商务管理、生产控制、市场分析、工程设计和科学探索等。

数据挖掘是信息技术自然进化的结果。进化过程的见证是数据库工业界开发以下功能（图 1.1）：数据收集和数据库创建，数据管理（包括数据存储和提取，数据库事务处理），以及数据分析与理解（涉及数据仓库和数据挖掘）。例如，数据收集和数据库创建机制的早期开发已成为稍后数据存储和提取、查询和事务处理有效机制开发的必备基础。随着提供查询和事务处理的大量数据库系统广泛付诸实践，数据分析和理解自然成为下一个目标。

自 60 年代以来，数据库和信息技术已经系统地从原始的文件处理进化到复杂的、功能强大的数据库系统。自 70 年代以来，数据库系统的研究和开发已经从层次和网状数据库发展到开发关系数据库系统（数据存放在关系表结构中；见 1.3.1 小节）、数据建模工具、索引和数据组织技术。此外，用户通过查询语言、用户界面、优化的查询处理和事务管理，可以方便、灵活地访问数据。**联机事务处理(OLTP)**将查询看作只读事务，对于关系技术的发展和广泛地将关系技术作为大量数据的有效存储、提取和管理的主要工具作出了重要贡献。

自 80 年代中期以来，数据库技术的特点是广泛接受关系技术，研究和开发新的、功能强大的数据库系统。这些使用了先进的数据模型，如扩充关系、面向对象、对象-关系和演绎模型。包括空间的、时间的、多媒体的、主动的和科学的数据库、知识库、办公信息库在内的面向应用的数据库系统百花齐放。涉及分布性、多样性和数据共享问题被广泛研究。异种数据库和基于 Internet 的全球信息系统，如 WWW 也已出现，并成为信息工业的生力军。

在过去的三十年中，计算机硬件稳定的、令人吃惊的进步导致了功能强大的计算机、数据收集设备和存储介质的大量供应。这些技术大大推动了数据库和信息产业的发展，使得大量数据库和信息存储用于事务管理、信息提取和数据分析。

现在，数据可以存放在不同类型的数据库中。最近出现的一种数据库结构是**数据仓库**（1.3.2 小节）。这是一种多个异种数据源在单个站点以统一的模式组织的存储，以支持管理决策。数据仓库技术包括数据清理、数据集成和**联机分析处理**（**OLAP**）。OLAP 是一种分析技术，具有汇总、合并和聚集功能，以及从不同的角度观察信息的能力。尽管 OLAP 工具支持多维分析和决策，对于深层次的分析，如数据分类、聚类和数据随时间变化的特征，仍然需要其它分析工具。

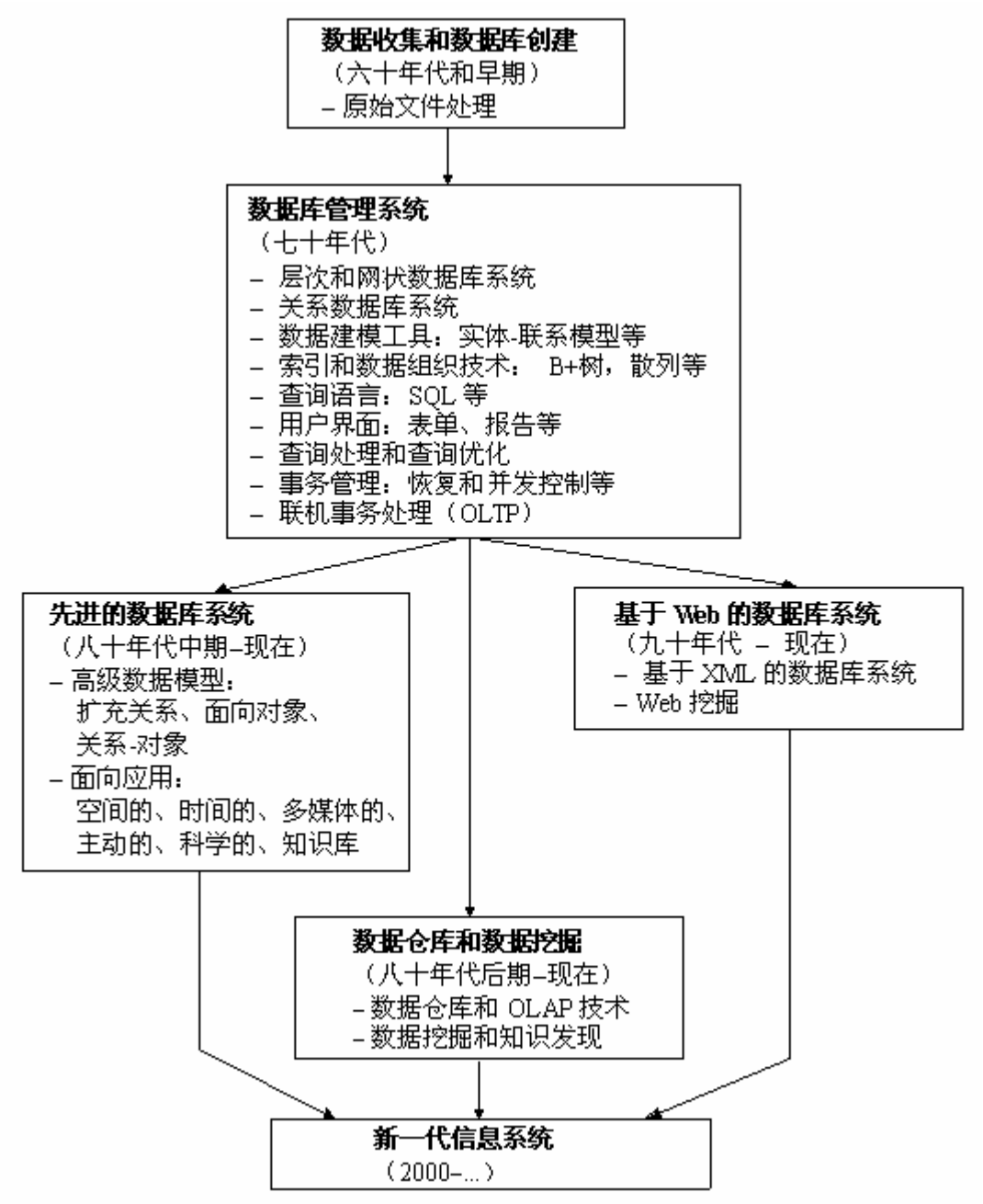


图 1.1：数据库技术的进化

数据丰富，伴随着对强有力的数据分析工具的需求，被描述为“数据丰富，但信息贫乏”。快速增长的海量数据收集、存放在大型和大量数据库中，没有强有力的工具，理解它们已经远远超出了人的能力（图 1.2）。结果，收集在大型数据库中的数据变成了“数据坟墓”——难得再访问的数据档案。这样，重要的决定常常不是基于数据库中信息丰富的数据，而是基于决策者的直观，因为决策者缺乏从海量数据中提取有价值知识的工具。此外，考虑当前的专家系统技术。通常，这种系统依赖用户或领域专家人工地将知识输入知识库。不幸的是，这一过程常常有偏差和错误，并且耗时、费用高。数据挖掘工具进行数据分析，可以发现重要的数据模式，对商务决策、知识库、科学和医学研究作出了巨大贡献。数据和信息之间的鸿沟要求系统地开发数据挖掘工具，将数据坟墓转换成知识“金块”。

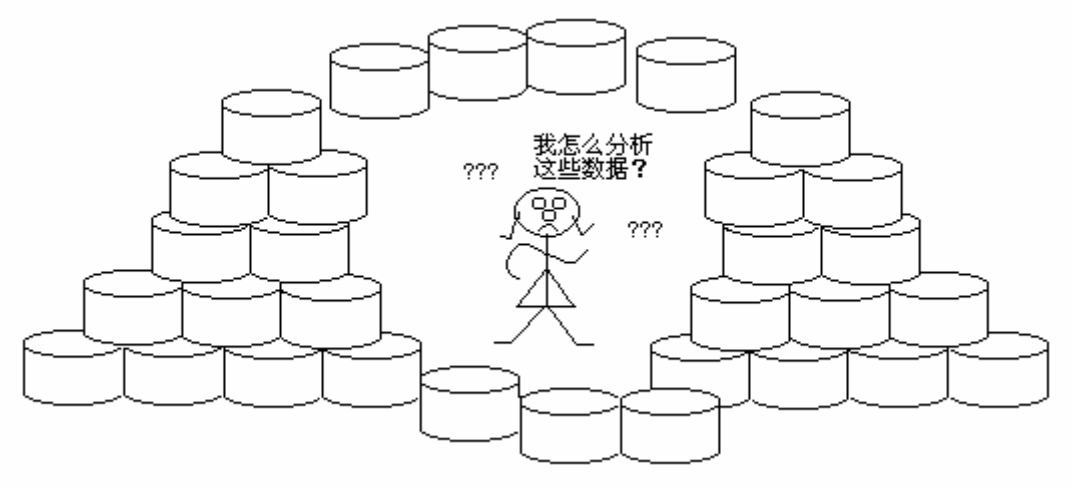


图 1.2　我们数据丰富，知识贫乏

1.2 什么是数据挖掘？

简单地说，**数据挖掘**是从大量数据中提取或“挖掘”知识。该术语实际上有点用词不当。注意，从矿石或砂子挖掘黄金称作黄金挖掘，而不是砂石挖掘。这样，数据挖掘应当更正确地命名为“从数据中挖掘知识”，不幸的是它有点长。“知识挖掘”是一个短术语，可能不能强调从大量数据中挖掘。毕竟，挖掘是一个很生动的术语，它抓住了从大量的、未加工的材料中发现少量金块这一过程的特点（图 1.3）。这样，这种用词不当携带了“数据”和“挖掘”，成了流行的选择。还有一些术语，具有和数据挖掘类似，但稍有不同的含义，如**数据库中知识挖掘**、**知识提取**、**数据/模式分析**、**数据考古**和**数据捕捞**。

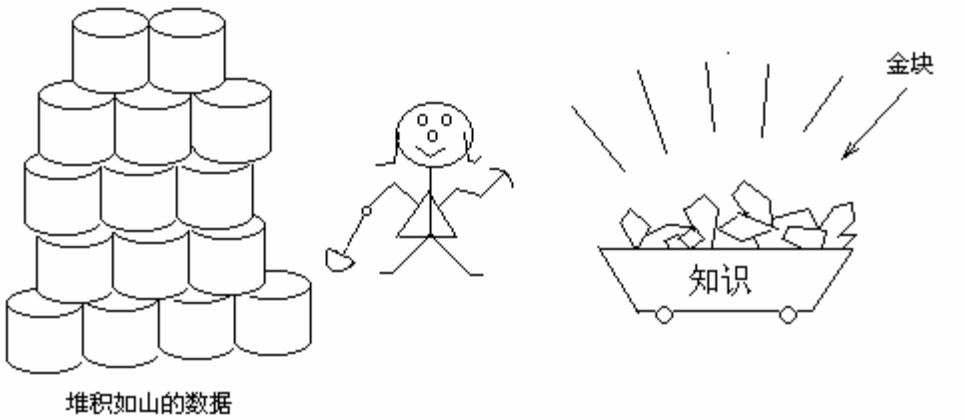


图 1.3 数据挖掘：在你的数据中搜索知识（有趣的模式）

许多人把数据挖掘视为另一个常用的术语“**数据库中知识发现**”或 **KDD** 的同义词。而另一些人只是把数据挖掘视为数据库中知识发现过程的一个基本步骤。知识发现过程如图 1.4 所示，由以下步骤组成：

1. **数据清理**（消除噪音或不一致数据）
2. **数据集成**（多种数据源可以组合在一起）[1]
3. **数据选择**（从数据库中提取与分析任务相关的数据）
4. **数据变换**（数据变换或统一成适合挖掘的形式；如，通过汇总或聚集操作）[2]

[1] 信息产业界的一个流行趋势是将数据清理和数据集成作为预处理步骤执行，结果数据存放在数据仓库中。
[2] 有时，数据变换和数据统一在数据选择过程之前进行，特别是在数据仓库情况下。

5. **数据挖掘**（基本步骤，使用智能方法提取数据模式）
6. **模式评估**（根据某种兴趣度度量，识别提供知识的真正有趣的模式；1.5 节）
7. **知识表示**（使用可视化和知识表示技术，向用户提供挖掘的知识）。

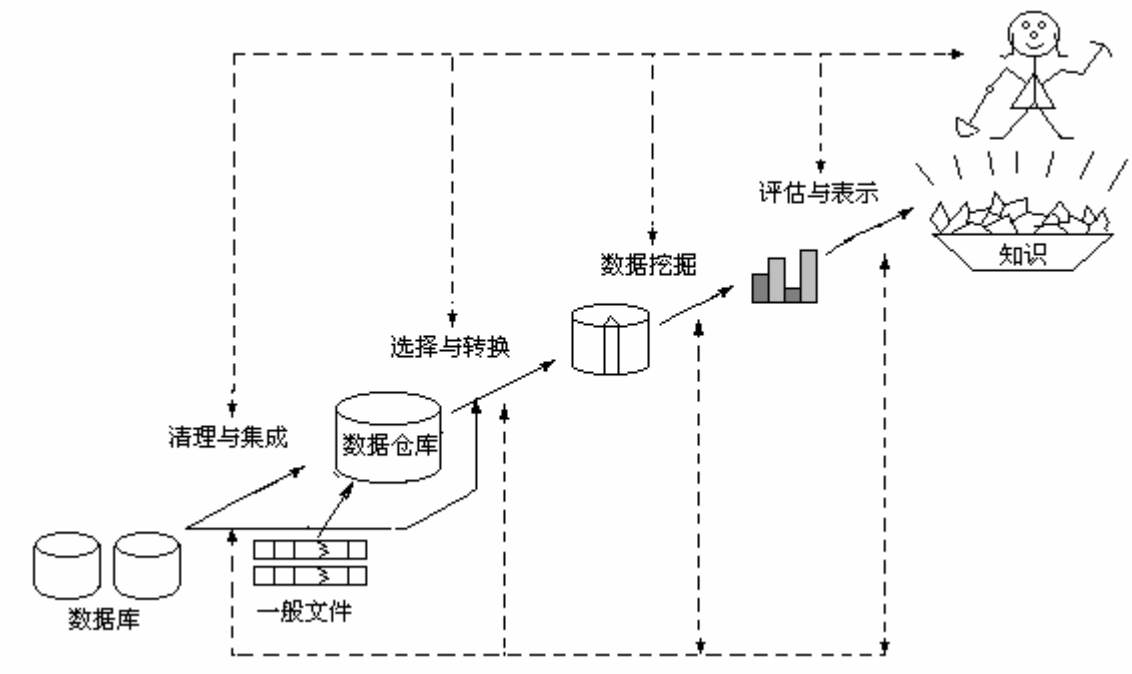


图 1.4：数据挖掘视为知识发现过程的一个步骤

数据挖掘步骤可以与用户或知识库交互。有趣的模式提供给用户，或作为新的知识存放在知识库中。注意，根据这种观点，数据挖掘只是整个过程中的一步，尽管是最重要的一步，因为它发现隐藏的模式。

我们同意数据挖掘是知识发现过程的一个步骤。然而，在工业界、媒体和数据库研究界，“数据挖掘”比较长的术语“数据库中知识发现”更流行。因此，在本书中，我们选用术语数据挖掘。我们采用数据挖掘的广义观点：数据挖掘是从存放在数据库、数据仓库或其它信息库中的大量数据挖掘有趣知识的过程。

基于这种观点，典型的数据挖掘系统具有以下主要成分（图 1.5）：

- **数据库、数据仓库、或其它信息库**：这是一个或一组数据库、数据仓库、展开的表、或其它类型的信息库。可以在数据上进行数据清理和集成。
- **数据库或数据仓库服务器**：根据用户的数据挖掘请求，数据库或数据仓库服务器负责提取相关数据。
- **知识库**：这是领域知识，用于指导搜索，或评估结果模式的兴趣度。这种知识可能包括**概念分层**，用于将属性或属性值组织成不同的抽象层。用户确信方面的知识也可以包含在内。可以使用这种知识，根据非期望性评估模式的兴趣度。领域知识的其它例子有兴趣度限制或阈值和元数据（例如，描述来自多个异种数据源的数据）。
- **数据挖掘引擎**：这是数据挖掘系统基本的部分，由一组功能模块组成，用于特征、关联、分类、聚类分析、演变和偏差分析。
- **模式评估模块**：通常，该部分使用兴趣度度量（1.5 节），并与挖掘模块交互，以便将搜索聚焦在有趣的模式上。它可能使用兴趣度阈值过滤发现的模式。模式评估模块也可以与挖掘模块集成在一起，这依赖于所用的数据挖掘方法的实现。对于有效的数据挖掘，建议尽可能地将模式评估推进到挖掘过程之中，以便将搜索限制在有兴趣的模式上。

- **图形用户界面**：该模块在用户和挖掘系统之间通讯，允许用户与系统交互，指定数据挖掘查询或任务，提供信息、帮助搜索聚焦，根据数据挖掘的中间结果进行探索式数据挖掘。此外，该成分还允许用户浏览数据库和数据仓库模式或数据结构，评估挖掘的模式，以不同的形式对模式可视化。

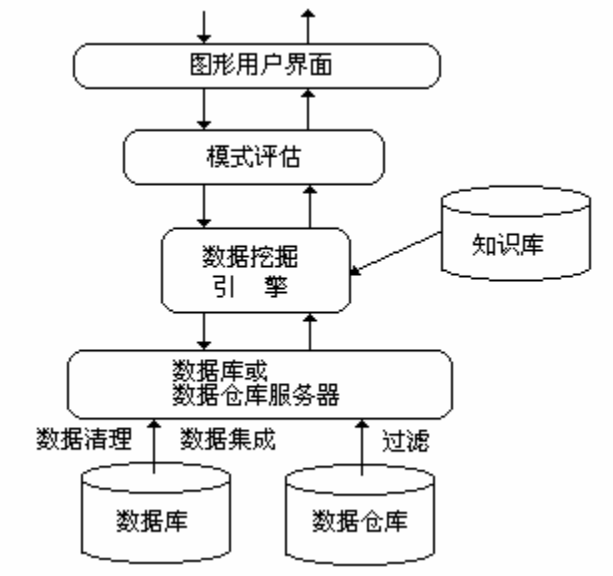


图 1.5：典型的数据挖掘系统结构

从数据仓库观点，数据挖掘可以看作联机分析处理（OLAP）的高级阶段。然而，通过结合更高级的数据理解技术，数据挖掘比数据仓库的汇总型分析处理走得更远。

尽管市场上已有许多“数据挖掘系统”，但是并非所有的都能进行真正的数据挖掘。不能处理大量数据的数据分析系统，最多称作机器学习系统、统计数据分析工具或实验系统原型。一个系统只能够进行数据或信息提取，包括在大型数据库找出聚集值或回答演绎查询，应当归类为数据库系统，或信息提取系统，或演绎数据库系统。

数据挖掘涉及多学科技术的集成，包括数据库技术、统计、机器学习、高性能计算、模式识别、神经网络、数据可视化、信息提取、图象与信号处理和空间数据分析。在本书讨论数据挖掘时，我们采用数据库观点。即，着重强调大型数据库中有效的和可规模化的数据挖掘技术。一个算法是**可规模化的**，如果给定内存和磁盘空间等可利用的系统资源，其运行时间应当随数据库大小线性增加。通过数据挖掘，可以从数据库提取有趣的知识、规律、或高层信息，并可以从不同角度观察或浏览。发现的知识可以用于决策、过程控制、信息管理、查询处理、等等。因此，数据挖掘被信息产业界认为是数据库系统最重要的前沿之一，是信息产业最有前途的交叉学科。

1.3 数据挖掘——在何种数据上进行？

本节，我们考察可以进行挖掘的各种数据存储。原则上讲，数据挖掘可以在任何类型的信息存储上进行。这包括关系数据库、数据仓库、事务数据库、先进的数据库系统、展平的文件和 WWW。先进的数据库系统包括面向对象和对象-关系数据库；面向特殊应用的数据库，如空间数据库、时间序列数据库、文本数据库和多媒体数据库。挖掘的挑战和技术可能因存储系统而异。

尽管本书假定读者具有信息系统的基本知识，我们还是对以上提到的主要数据存储系统做简要介绍。本节，我们还介绍编造的 AllElectronics 商店，它在本书各处用来解释概念。

1.3.1 关系数据库

数据库系统，也称**数据库管理系统**（**DBMS**），由一组内部相关的数据，称作数据库，和一组管理和存取数据的软件程序组成。软件程序涉及如下机制：数据库结构定义，数据存储，并行、共享或分布的数据访问，面对系统瘫痪或未授权的访问，确保数据的一致性和安全性。

关系数据库是**表**的集合，每个表都赋予一个唯一的名字。每个表包含一组**属性**（列或字段），并通常存放大量**元组**（记录或行）。关系中的每个元组代表一个被唯一关键字标识的对象，并被一组属性值描述。语义数据模型，如**实体-联系**（**ER**）数据模型，将数据库作为一组实体和它们之间的联系进行建模。通常为关系数据库构造 ER 模型。

考虑下面的例子。

例 1.1 AllElectronics 公司由下列关系表描述：*customer, item, employee* 和 *branch*。这些表的片段在图 1.6 中给出。

- 关系 *customer* 由一组属性，包括顾客的唯一标识号(*cust_ID*)，顾客的姓名、地址、年龄、职业、年收入、信誉信息、分类等。

customer

cast_ID	name	address	age	income	credit_info	...
C1	Smith,Sandy	4563 E.Hastings,Burnaby, BC,V5A 4S9, Canada ...	21	$27000	1	... ...

items

item_ID	name	brand	category	type	price	place_made	supplier	cost
13 18 ...	high_res TV mutidisc CDplay	Toshiba Sanyo ...	high resolution mutidisc ...	TV CD player ...	$988.00 $369.00 ...	Japan Japan ...	Niko X Music Front ...	$600.00 $120.00 ...

employee

empl_ID	name	category	group	salary	commission
E35 ...	Jones,Jane ...	home entertainment ...	manager ...	$18,000 ...	2% ..

branch

branch_ID	name	address
B1 ...	City Square ...	369 Cambie St.,Vancouver,BC V5L 3A2,Canada ...

purchases

trans_ID	cast_ID	empl_ID	date	time	method_paid	amount
T100 ...	C1 ...	E55 ...	09/21/98 ...	15:45 ...	Visa ...	$1357.00

items_sold

trans_ID	item_ID	qty
T100 T100 ...	13 18 ...	1 2 ...

works_at

empl_ID	brabch_ID

E55 ...	B1 ...

图 1.6: AllElectronics 关系数据库的关系片段

- 类似地，关系 *employee, branch* 和 *item* 的每一个都包含一组属性，描述它们的性质。
- 表也用于表示多个关系表之间的联系。对于我们的例子，包括 *purchase*（顾客购买商品，创建一个由雇员处理的销售事务）和 *work_at*（雇员在的一个分店工作）。□

关系数据可以通过数据库查询访问。数据库查询使用如 SQL 这样的关系查询语言，或借助于图形用户界面书写。在后者，用户可以使用菜单指定包含在查询中的属性和属性上的限制。一个给定的查询被转换成一系列关系操作，如连接、选择和投影，并被优化，以便有效地处理。查询可以提取数据的一个指定的子集。假定你的工作是分析 AllElectronics 的数据。通过使用关系查询，你可以提这样的问题：“显式一个上个季度销售的商品的列表”。关系查询语言也可以包含聚集函数，如 sum, avg（平均）,count, max（最大）和 min（最小）。这些使得你可以问“给我显式上个月的总销售，按分店分组”，或“多少销售事务出现在 12 月份？”，或“哪一位销售人员的销售额最高？”。

当数据挖掘用于关系数据库时，你可以进一步搜索趋势或数据模式。例如，数据挖掘系统可以分析顾客数据，根据顾客的收入、年龄和以前的信誉信息预测新顾客的信誉风险。数据挖掘系统也可以检测偏差，如，与以前的年份相比，哪种商品的销售出人预料。这种偏差可以进一步考察（例如，包装是否有变化，或价格是否大幅度提高？）。

关系数据库是数据挖掘的最流行的、最丰富的数据源，因此它是我们数据挖掘研究的主要数据形式。

1.3.2 数据仓库

假定 AllElectronics 是一个成功的跨国公司，分部遍及世界。每个分部有自己的一组数据库。AllElectronics 的总裁要你提供公司第三季度每种商品、每个分部的销售分析。这是一个困难的任务，特别是当相关数据散布在多个数据库，物理地存放在许多站点时。

如果 AllElectronics 有一个数据仓库，该任务将是容易的。数据仓库是一个从多个数据源收集的信息存储，存放在一个一致的模式下，并通常驻留在单个站点。数据仓库通过数据清理、数据变换、数据集成、数据装入和定期数据刷新构造。该过程在第 2、3 章详细研究。图 1.7 给出了 AllElectronics 的数据仓库的基本结构。

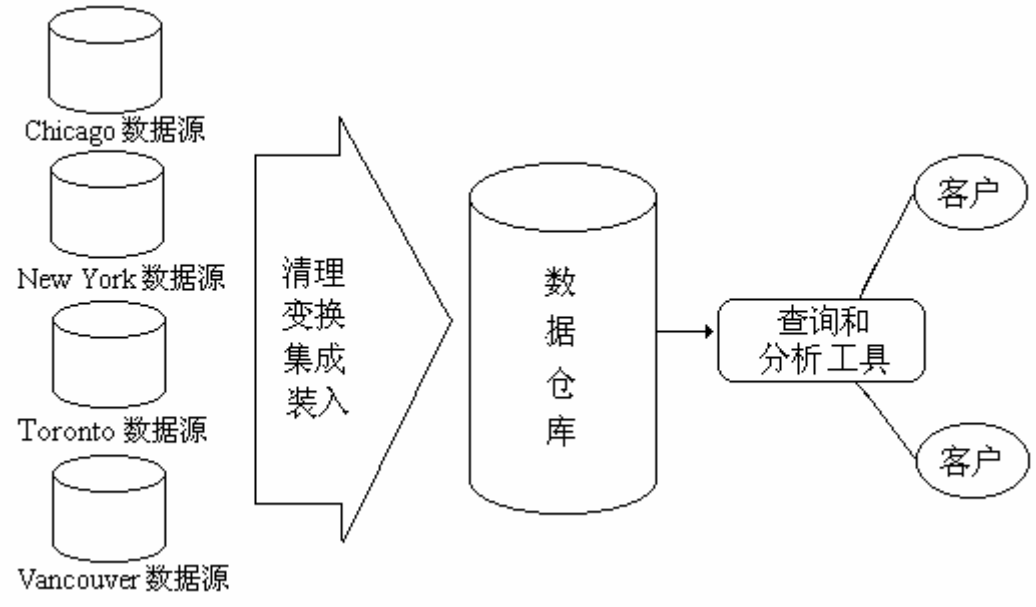


图 1.7：*AllElectronics* 典型的数据仓库结构

为便于制定决策，数据仓库中的数据围绕诸如顾客、商品、供应商和活动等主题组织。数据存储，从历史的角度（如过去的 5-10 年）提供信息，并且是汇总的。例如，数据仓库不是存放每个销售事务的细节，而是存放每个商店，或（汇总到较高层次）每个销售地区每类商品的销售事务汇总。

通常，数据仓库用多维数据库结构建模。其中，每个**维**对应于模式中一个或一组属性，每个**单元**存放聚集度量，如 *count* 或 *sales_amount*。数据仓库的实际物理结构可以是关系数据存储或**多维数据方**。它提供数据的多维视图，并允许快速访问预计算的和汇总的数据。

例 1.2 AllElectronics 的汇总销售数据数据方在图 1.8(a)中。该数据方有三个维：*address*（城市值），*time*（季度值 *Q1, Q2, Q3, Q4*）和 *item*（商品类型值：家庭娱乐、计算机、电话、安全）。存放在方体的每个单元中的聚集值是 *sales_amount*（单位：$1000）。例如，安全系统第一季度在 Vancouver 的总销售为$400,000，存放在单元<Vancouver,Q1, 安全>中。其它方体可以用于存放每个维上的聚集和，对应于使用不同的 SQL 分组得到的聚集值（例如，每个城市和季度，或每个季度和商品，或每单个维的总销售量）。□

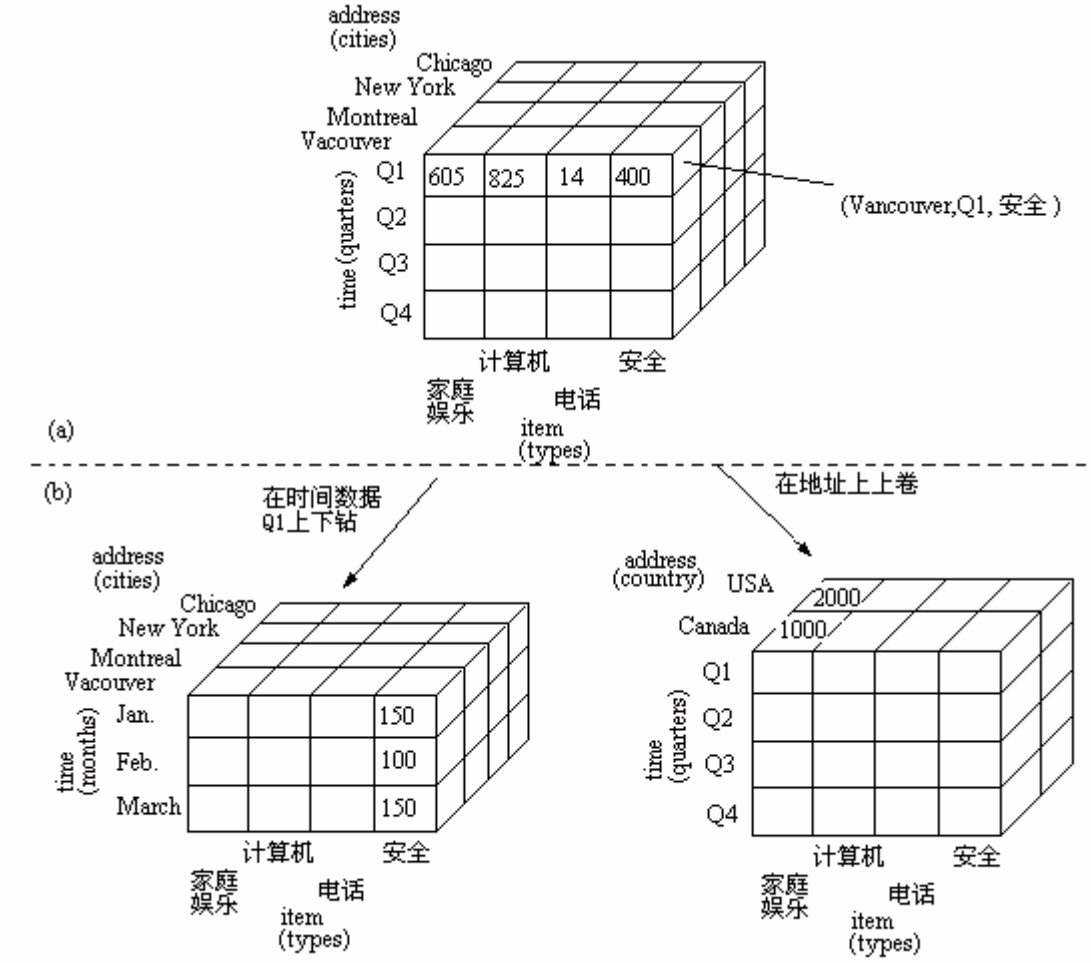


图 1.8　一个通常用于数据仓库多维数据方，(a) 展示 *AllElectronics* 的汇总数据　(b) 展示数据方(a)上的下钻与上卷结果。为便于观察，只给出部分单元值

你可能会问：“我还听说过数据集市。数据仓库和数据集市的区别是什么？”数据仓库收集了整个组织的主题信息，因此，它是企业范围的。另一方面，**数据集市**是数据仓库的一个部门子集。它聚焦在选定的主题上，是部门范围的。

通过提供多维数据视图和汇总数据的预计算，数据仓库非常适合**联机分析处理**（**OLAP**）。OLAP 操作使用数据的领域背景知识，允许在不同的抽象层提供数据。这些操作适合不同的用户。OLAP 操作的例子包括**下钻**和**上卷**，它们允许用户在不同的汇总级别观察数据，如图 1.8(b)所示。例如，可以对按季度汇总的销售数据下钻，观察按月汇总的数据。类似地，可以对按城市汇总的销售数据上卷，观察按国家汇总的数据。

尽管数据仓库工具对于支持数据分析是有帮助的，但是仍需要更多的数据挖掘工具，以便进行更深入的自动分析。数据仓库技术在第 2 章详细讨论。

1.3.3 事务数据库

一般地，**事务数据库**由一个文件组成，其中每个记录代表一个事务。通常，一个事务包含一个唯一的事务标识号(*trans_ID*)，和一个组成事务的项的列表（如，在商店购买的商品）。事务数据库可能有一些与之相关联的附加表，包含关于销售的其它信息，如事务的日期、顾客的 *ID* 号、销售者的 *ID* 号、销售分店，等等。

例 1.3 事务可以存放在表中，每个事务一个记录。AllElectronics 的事务数据库的片段在图 1.9 中给出。从关系数据库的观点，图 1.9 的销售表是一个嵌套的关系，因为属性"*list of item_ID*"包含 *item* 的集合。由于大部分关系数据库系统不支持嵌套关系结构，事务数据库通常存放在一个类似于图 1.9 中的表格式的展平的文件中，或展开到类似于图 1.6 的 *items_sold* 表的标准关系中。□

sales

trans_ID	list of item_ID
T100	I1,I3,I8,I16
...	...

图 1.9：*AllElectronics* 销售事务数据库的片段

作为 AllElectronics 数据库的分析者，你想问"显示 Sandy Smith 购买的所有商品"或"有多少事务包含商品号 I3？"。回答这种查询可能需要扫描整个事务数据库。

假定你想更深地挖掘数据，问"哪些商品适合一块销售？"这种"购物篮分析"使你能够将商品捆绑成组，作为一种扩大销售的策略。例如，给定打印机与计算机经常一起销售的知识，你可以向购买选定计算机的顾客提供对一种很贵的打印机打折，希望销售更多较贵的打印机。常规的数据提取系统不能回答上面这种查询。然而，通过识别频繁一块销售的商品，事务数据的数据挖掘系统可以做到。

1.3.4 高级数据库系统和高级数据库应用

关系数据库系统广泛地用于商务应用。随着数据库技术的发展，各种先进的数据库系统已经出现并在开发中，以适应新的数据库应用需要。

新的数据库应用包括处理空间数据（如地图）、工程设计数据（如建筑设计、系统部件、集成电路）、超文本和多媒体数据（包括文本、图象和声音数据）、时间相关的数据（如历史数据或股票交换数据）和万维网（Internet 使得巨大的、广泛分布的信息存储可以利用）。这些应用需要有效的数据结构和可规模化的方法，处理复杂的对象结构、变长记录、半结构化或无结构的数据，文本和多媒体数据，以及具有复杂结构和动态变化的数据库模式。

为响应这些需求，开发了先进的数据库系统和面向特殊应用的数据库系统。这些包括面向对象和对象-关系数据库系统、空间数据库系统、时间和时间序列数据库系统、异种和遗产数据库系统、基于万维网的全球信息系统。

虽然这样的数据库或信息存储需要复杂的机制，以便有效地存储、提取和更新大量复杂的数据，它们也为数据挖掘提供了肥沃的土壤，提出了挑战性的研究和实现问题。本节，我们将介绍上面列举的每种高级数据库系统。

面向对象数据库

面向对象数据库基于面向对象程序设计范例。用一般术语，每个实体被看作一个对象。对于 AllElectronics 例子，对象可以是每个雇员、顾客、商品。涉及一个对象的数据和代码封装在一个单元中。每个对象关联：

- 一个**变量**集，它描述数据。这对应于实体-联系和关系模型的属性。
- 一个**消息**集，对象可以使用它们与其它对象，或与数据库系统的其它部分通讯。

- 一个**方法**集，其中每个方法存放实现一个消息的代码。一旦收到消息，方法就返回一个响应值。例如，消息 *get_photo*(*employee*)的方法将提取并返回给定雇员对象的照片。

共享公共特性集的对象可以归入一个**对象类**。每个对象都是其对象类的**实例**。对象类可以组成类/子类层次结构，使得每个类代表该类对象共有的特性。例如，类 *employee* 可以包含变量 *name*, *address* 和 *birthdate*。假定类 *sales_person* 是 *employee* 的子类。一个 *sales_person* 对象将继承属于其超类 *employee* 的所有变量。此外，它还具有作为一个销售员特有的所有变量（如，*commission*）。这种类继承特性有利于信息共享。

对象-关系数据库

对象-关系数据库基于对象-关系数据模型构造。该模型通过提供处理复杂对象的丰富数据类型和对象定位，扩充关系模型。此外，它还包含关系查询语言的特殊构造，以便管理增加的数据类型。通过增加处理复杂数据类型、类层次结构和如上所述的对象继承，对象-关系模型扩充了基本关系模型。对象-关系数据库在工业和应用正日趋流行。

在面向对象和对象-关系系统中的数据挖掘具有某些类似性。与关系数据挖掘相比，需要开发新的技术，处理复杂对象结构、复杂数据类型、类和子类层次结构、特性继承以及方法和过程。

空间数据库

空间数据库包含涉及空间的信息。这种数据库包括地理（地图）数据库、VLSI 芯片设计数据库、医疗和卫星图象数据库。空间数据可能以**光栅格式**提供，由 *n* 维位图或象素图构成。例如，一个 2 维卫星图象可以用光栅数据表示，每个象素存放一个给定区域的降雨量。地图也可以用**向量格式**提供，其中，路、桥、建筑物和湖泊可以用诸如点、线、多边形和这些形状形成的分化和网络等基本地理结构表示。

地理数据库有大量应用，包括从森林和生态规划，到提供关于电话和电缆、管道和下水系统位置在内的公共信息服务。此外，地理数据库还用于车辆导航和分流系统。例如，一个用于出租车的系统可以存储一个城市的地图，提供关于单行道、交通拥挤时从区域 *A* 到区域 *B* 的建议路径、饭店和医院的位置、以及每个司机的当前位置等信息。

你可能会问：“空间数据库上可以进行何种数据挖掘？”数据挖掘可以发现描述座落在特定类型地点（如，公园）的房屋特征。其它模式可能描述不同海拔高度山区的气候，或根据城市离主要公路的距离描述都市贫困率的变化趋势。此外，可以构造“空间数据方”，将数据组织到多维结构和层次中，OLAP 操作（如，下钻和上卷）可以在其上进行。

时间数据库和时间序列数据库

时间数据库和时间序列数据库都存放与时间有关的数据。**时间数据库**通常存放包含时间相关属性的数据。这些属性可能涉及若干时间标签，每个都具有不同的语义。**时间序列数据库**存放随时间变化的值序列，如，收集的股票交易数据。

数据挖掘技术可以用来发现数据库中对象演变特征或对象变化趋势。这些信息对于决策和规划是有用的。例如，银行数据的挖掘可能有助于根据顾客的流量安排银行出纳员。可以挖掘股票交易数据，发现可能帮助你制订投资策略的趋势（例如，何时是购买 AllElectronics 的股票的最佳时机？）。通常，这种分析需要定义时间的多粒度。例如，时间可以按财政年、学年或日历年分解。年可以进一步分解成季度或月。

文本数据库和多媒体数据库

文本数据库是包含对象文字描述的数据库。通常，这种词描述不是简单的关键词，而是长句子或短文，如产品介绍、错误或故障报告、警告信息、汇总报告、笔记或其它文档。文本数据库可能是高度非规格化的（如，万维网上的网页）。有些文本数据库可能是半结构化的（如 email 消息和一些 HTML/XML 网页），而其它的可能是良结构化的（如图书馆数据库）。通常，具有很好结构的文本数据库可以使用关系数据库系统实现。

“文本数据库上的数据挖掘可以发现什么？”说到底，可以发现对象类的一般描述，以及关键词或内容的关联和文本对象的聚类行为。为做到这一点，需要将标准的数据挖掘技术与信息提取技术和文本数据特有的层次构造（如字典和辞典），以及面向学科的（如化学、医学、法律或经济）术语分类系统集成在一起。

多媒体数据库存放图象、音频和视频数据。它们用于基于图内容的提取、声音传递、录像点播、万维网和识别口语命令的基于语音的用户界面等方面。多媒体数据库必须支持大对象，因为象视频这样的数据对象可能需要数十亿字节的存储。还需要特殊的存储和检索技术，因为视频和音频数据需要以稳定的、预先确定的速率实时检索，防止图象或声音间断和系统缓冲区溢出。这种数据称为**连续媒体数据**。

对于多媒体数据库挖掘，需要将存储和检索技术与标准的数据挖掘方法集成在一起。有前途的方法包括构造多媒体数据方、多媒体数据的多特征提取和基于相似的模式匹配。

异种数据库和遗产数据库

异种数据库由一组互连的、自治的成员数据库组成。这些成员相互通讯，以便交换信息和回答查询。一个成员数据库中的对象可能与其它成员数据库中的对象很不相同，使得很难将它们的语义吸收进一个整体的异种数据库中。

许多企业需要遗产数据库，作为信息技术长时间开发（包括使用不同的硬件和操作系统）的结果。**遗产数据库**是一组异种数据库，它将不同的数据系统组合在一起。这些数据系统如关系或对象-关系数据库、层次数据库、网状数据库、电子表格、多媒体数据库或文件系统。遗产数据库中的异种数据库可以通过网内或网间计算机网络连接。

这种数据库的信息交换是困难的，因为需要考虑发散的语义，制定从一种表示到另一种表示的精确转换规则。例如，考虑不同学校之间学生学业情况数据交换问题。每个学校可能有自己的计算机系统和课程与评分体系。一所大学可能采用学季系统（每学期三个月——译注），开三门数据库课程，并按由 A+到 F 评定成绩；而另一所可能采用学期系统，开两门数据库课程，并按由 1 到 10 评定成绩。很难制定这两所大学的课程-成绩转换精确的规则，使得信息交换很困难。通过将给定的数据转换到较高的、更一般的概念层（对于学生成绩，如不及格、良好或优秀），数据挖掘技术可以对此问题提供有趣的解，使得数据交换可以更容易地进行。

万维网

万维网和与之关联的分布信息服务（如，美国在线，Yahoo!, Alta Vista, Prodigy）提供了丰富的、世界范围的联机信息服务；这里，数据对象被链接在一起，便于交互访问。用户通过链接，从一个对象到另一个，寻找有趣的信息。这种系统对数据挖掘提供了大量机会和挑战。例如，理解用户的访问模式不仅能够帮助改进系统设计（通过提供高度相关的对象间的有效访问），而且还可以引导更好的市场决策（例如，通过在频繁访问的文档上布置广告，或提供更好的顾客/用户分类和行为分析）。在这种分布式信息环境下，捕获用户访问模式称作**挖掘路径遍历模式**。

尽管网页看上去好看并且信息丰富，但它们实际上是非结构化的并且缺乏预定义的模式、类型和格式。这样，对于系统地进行信息提取和数据挖掘，计算机很难理解各种网页的语义并把它们以有组织的形式结构化。提供基于关键字的搜索服务，而不理解特定网页的上下文，只能给用户提供有限的帮助。例如，基于单个关键字的网搜索可能返回数以百计的指针，指向包含该关键字的网页，而其中大部分与用户期望的查找无关。数据挖掘可以提供比网搜索服务更多的帮助吗？数据挖掘能够帮助我们学习网上信息的一般分布、网页特征和不同网页之间的关联吗？能够帮助我们找到特定主题的权威网页吗？这些问题对高级的数据挖掘提出了新的挑战。

1.4 数据挖掘功能——可以挖掘什么类型的模式？

我们已经观察了可以进行数据挖掘的各种数据存储和数据库系统。现在，让我们考察可以挖掘的数据模式。

数据挖掘功能用于指定数据挖掘任务中要找的模式类型。一般地，数据挖掘任务可以分两类：**描述**和**预测**。描述性挖掘任务刻划数据库中数据的一般特性。预测性挖掘任务在当前数据上进行推断，以进行预测。

在某些情况下，用户不知道他们的数据中什么类型的模式是有趣的，因此可能想并行地搜索多种不同的模式。这样，重要的是，数据挖掘系统要能够挖掘多种类型的模式，以适应不同的用户需求或不同的应用。此外，数据挖掘系统应当能够发现各种粒度（即，不同的抽象层）的模式。数据

挖掘系统应当允许用户给出提示，指导或聚焦有趣模式的搜索。由于有些模式并非对数据库中的所有数据都成立，通常每个被发现的模式带上一个确定性或“可信性”度量。

数据挖掘功能以及它们可以发现的模式类型介绍如下。

1.4.1 概念/类描述：特征和区分

数据可以与类或概念相关联。例如，在 AllElectronics 商店，销售的商品类包括计算机和打印机，顾客概念包括 *bigSpenders* 和 *budgetSpenders*。用汇总的、简洁的、精确的方式描述每个类和概念可能是有用的。这种类或概念的描述称为**类/概念描述**。这种描述可以通过下述方法得到（1）数据特征化，一般地汇总所研究类（通常称为**目标类**）的数据，或（2）数据区分，将目标类与一个或多个比较类（通常称为**对比类**）进行比较，或（3）数据特征化和比较。

数据特征是目标类数据的一般特征或特性的汇总。通常，用户指定类的数据通过数据库查询收集。例如，为研究上一年销售增加 10%的软件产品的特征，可以通过执行一个 SQL 查询收集关于这些产品的数据。

有许多有效的方法，将数据特征化和汇总。例如，基于数据方的 OLAP 上卷操作（1.3.2 小节）可以用来执行用户控制的、沿着指定维的数据汇总。该过程将在第 2 章介绍数据仓库时进一步详细讨论。面向属性的归纳技术可以用来进行数据的泛化和特征化，而不必一步步地与用户交互。这一技术将在第 5 章讨论。

数据特征的输出可以用多种形式提供。包括**饼图**、**条图**、**曲线**、**多维数据方**和包括交叉表在内的**多维表**。结果描述也可以用**泛化关系**或规则（称作**特征规则**）形式提供。这些不同的输出形式和它们的转换在第 5 章讨论。

例 1.4 数据挖掘系统应当能够产生一年之内在 AllElectronics 花费$1000 以上的顾客汇总特征的描述。结果可能是顾客的一般轮廓，如年龄在 40-50、有工作、有很好的信誉度。系统将允许用户在任意维下钻，如在 *occupation* 下钻，以便根据他们的职业来观察这些顾客。□

数据区分是将目标类对象的一般特性与一个或多个对比类对象的一般特性比较。目标类和对比类由用户指定，而对应的数据通过数据库查询提取。例如，你可能希望将上一年销售增加 10%的软件产品与同一时期销售至少下降 30%的那些进行比较。用于数据区分的方法与用于数据特征的那些类似。

“区分描述如何输出？”输出的形式类似于特征描述，但区分描述应当包括比较度量，帮助区分目标类和对比类。用规则表示的区分描述称为**区分规则**。用户应当能够对特征和区分描述的输出进行操作。

例 1.5 数据挖掘系统应当能够比较两组 AllElectronics 顾客，如定期（每月多于 2 次）购买计算机产品的顾客和偶尔（即，每年少于 3 次）购买这种产品的顾客。结果描述可能是一般的比较轮廓，如经常购买这种产品的顾客 80%在 20-40 岁之间，受过大学教育；而不经常购买这种产品的顾客 60%或者太老，或者太年青，没有大学学位。沿着维下钻，如沿 *occupation* 维，或添加新的维，如 *income_level*，可以帮助发现两类之间的更多区分特性。□

概念描述，包括特征和区分，是第 5 章的主题。

1.4.2 关联分析

“什么是关联分析？”**关联分析**发现关联规则，这些规则展示属性-值频繁地在给定数据集中一起出现的条件。关联分析广泛用于购物篮或事务数据分析。

更形式地，**关联规则**是形如 $X \Rightarrow Y$，即”$A_1 \wedge ... \wedge A_m \Rightarrow B_1 \wedge ... \wedge B_n$”的规则；其中，$A_i$ ($i \in \{1,...,m\}$)，B_j ($j \in \{1,...,n\}$)是属性-值对。关联规则解释为“满足 X 中条件的数据库元组多半也满足 Y 中条件”。

例 1.6 给定 AllElectronics 关系数据库，一个数据挖掘系统可能发现如下形式的规则

$$age(X,"20-29") \wedge income(X,"20-30K") \Rightarrow buys(X,"CD\_player")$$
$$[support = 2\%, confidence = 60\%]$$

其中，*X* 是变量，代表顾客。该规则是说，所研究的 AllElectronics 顾客 2%（**支持度**）在 20-29 岁，年收入 20-29K，并且在 AllElectronics 购买 CD 机。这个年龄和收入组的顾客购买 CD 机的可能性有 60%（**置信度**或可信性）。

注意，这是一个以上属性之间（即 *age*, *income* 和 *buys*）的关联。采用多维数据库使用的术语，每个属性称为一个维，上面的规则可以称作**多维关联规则**。

假定作为 AllElectronics 的市场部经理，你想知道在一个事务中，哪些商品经常一块购买。这种规则的一个例子是

$$contains(T,"computer") \Rightarrow contains(T,"software")$$
[*support = 1%,confidence* =50%]

该规则是说，如果事务 *T* 包含"*computer*"，则它也包含"*software*"的可能性有 50%，并且所有事务的 1%包含二者。这个规则涉及单个重复的属性或谓词（即，*contains*）。包含单个谓词的关联规则称作**单维关联规则**。去掉谓词符号，上面的规则可以简单地写成 *computer* ⇒ *software*[1%,50%]。□

近年来，已经提出了许多有效的关联规则挖掘算法。关联规则挖掘在第 6 章详细讨论。

1.4.3 分类和预测

分类是这样的过程，它找描述或识别数据类或概念的**模型**(或函数)，以便能够使用模型预测类标号未知的对象。导出模型是基于对**训练数据**集（即，其类标号已知的数据对象）的分析。

"如何提供导出模型？"导出模式可以用多种形式表示，如分类（IF-THEN）规则、判定树、数学公式、或神经网络。**判定树**是一个类似于流程图的结构，每个结点代表一个属性值上的测试，每个分枝代表测试的一个输出，树叶代表类或类分布。判定树容易转换成分类规则。当用于分类时，**神经网络**是一组类似于神经元的处理单元，单元之间加权连接。

分类可以用来预测数据对象的类标号。然而，在某些应用中，人们可能希望预测某些遗漏的或不知道的数据值，而不是类标号。当被预测的值是数值数据时，通常称之为**预测**。尽管预测可以涉及数据值预测和类标号预测，通常预测限于值预测，并因此不同于分类。预测也包含基于可用数据的分布趋势识别。

相关分析可能需要在分类和预测之前进行，它试图识别对于分类和预测无用的属性。这些属性应当排除。

例 1.7 假定作为 AllElectronics 的销售经理，你想根据对销售活动的反映，对商店的商品集合分成三大类：好的反映，中等反映和差的反映。你想根据商品的描述特性，如 *price, brand, place_made* 和 *category*，对这三类的每一种导出模型。结果分类将最大限度地区别每一个类，提供有组织的数据集视图。假定结果分类用判定树的形式表示。例如，判定树可能把 *price* 看作最能区分三个类的因素。该树可能揭示，在 *price* 之后，帮助进一步区分每类对象的其它特性包括 *brand* 和 *place_made*。这样的判定树可以帮助你理解给定销售活动的影响，并帮助你设计未来更有效的销售活动。□

第 7 章将详细讨论分类和预测。

1.4.4 聚类分析

"何为聚类分析？"与分类和预测不同，**聚类**分析数据对象，而不考虑已知的类标号。一般地，训练数据中不提供类标号，因为不知道从何开始。聚类可以产生这种标号。对象根据最大化类内的相似性、最小化类间的相似性的原则进行聚类或分组。即，对象的聚类这样形成，使得在一个聚类中的对象具有很高的相似性，而与其它聚类中的对象很不相似。所形成的每个聚类可以看作一个对象类，由它可以导出规则。聚类也便于**分类编制**，将观察组织成类分层结构，类似的事件组织在一起。

例 1.8 聚类分析可以在 AllElectronics 的顾客数据上进行，识别顾客的同类子群。这些聚类可以表示每个购物目标群。图 1.10 展示一个城市内顾客的 2-D 图。数据点的三个聚类是显而易见的。□

聚类分析形成第 8 章的主题。

1.4.5 局外者分析

数据库中可能包含一些数据对象，它们与数据的一般行为或模型不一致。这些数据对象是**局外者**。大部分数据挖掘方法将局外者视为噪音或例外而丢弃。然而，在一些应用中（如，欺骗检测），罕见的事件可能比正规出现的那些更有趣。局外者数据分析称作**局外者挖掘**。

局外者可以使用统计试验检测。它假定一个数据分布或概率模型，并使用距离度量，到其它聚类的距离很大的对象被视为局外者。基于偏差的方法通过考察一群对象主要特征上的差别识别局外者，而不是使用统计或距离度量。

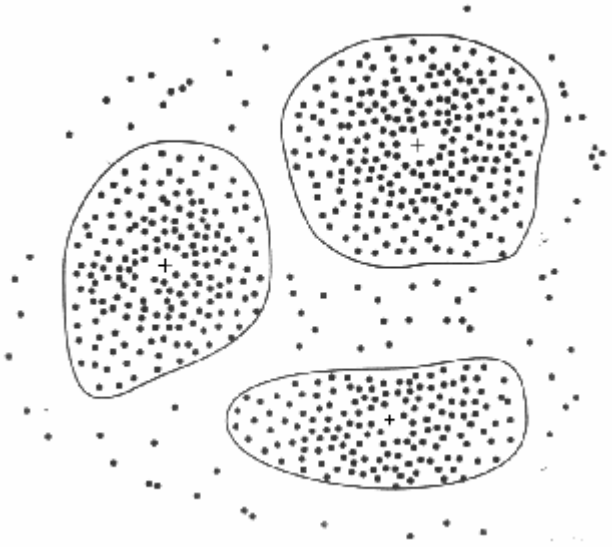


图 1.10　关于一个城市内顾客的 2-D 图，显示了 3 个聚类，每个聚类的“中心”用“+”标记

例 1.9　局外者分析可以发现信用卡欺骗。通过检测一个给定帐号与正常的付费相比，付款数额特别大来发现信用卡欺骗性使用。局外者值还可以通过购物地点和类型，或购物频率来检测。□

局外者分析也在第 8 章讨论。

1.4.6 演变分析

数据**演变分析**描述行为随时间变化的对象的规律或趋势，并对其建模。尽管这可能包括时间相关数据的特征、区分、关联、分类或聚类，这类分析的不同特点包括时间序列数据分析、序列或周期模式匹配和基于类似性的数据分析。

例 1.10　假定你有纽约股票交易所过去几年的主要股票市场（时间序列）数据，并希望调查高科技工业公司股份。股票数据挖掘研究可以识别整个股票市场和特定的公司的股票演变规律。这种规律可以帮助预测股票市场价格的未来走向，帮助你对股票投资作出决策。□

数据演变分析将在第 9 章进一步讨论。

1.5 所有模式都是有趣的吗？

数据挖掘系统具有产生数以千计，甚至数以万计模式或规则的潜在能力。

你可能会问："所有模式都是有趣的吗？"答案是否定的。实际上，对于给定的用户，在可能产生的模式中，只有一小部分是他感兴趣的。

这对数据挖掘系统提出了一系列的问题。你可能会想："什么样的模式是有趣的？数据挖掘系统能够产生所有有趣的模式吗？数据挖掘系统能够仅产生有趣的模式吗？"

对于第一个问题，一个模式是有趣的，如果（1）它易于被人理解，（2）在某种程度上，对于新的或测试数据是有效的，（3）是潜在有用的，（4）是新颖的。如果一个模式符合用户确信的某种假设，它也是有趣的。有趣的模式表示**知识**。

存在一些**模式兴趣度的客观度量**。这些基于所发现模式的结构和关于它们的统计。对于形如 $X \Rightarrow Y$ 的关联规则，一种客观度量是规则的**支持度**。规则的支持度表示满足规则的样本百分比。支持度是概率 $P(X \cup Y)$，其中，$X \cup Y$ 表示同时包含 X 和 Y 的事务；即，项集 X 和 Y 的并。关联规则的另一种客观度量是**置信度**。置信度是条件概率 $P(Y \mid X)$；即，包含 X 的事务也包含 Y 的概率。更形式地，支持度和置信度定义为

$$support(X \Rightarrow Y) = P(X \cup Y)$$
$$confidence(X \Rightarrow Y) = P(Y \mid X)$$

一般地，每个兴趣度度量都与一个阈值相关联，该阈值可以由用户控制。例如，不满足置信度阈值 50%的规则可以认为是无趣的。低于阈值的规则可能反映噪音、例外，或少数情况，可能不太有价值。

尽管客观度量可以帮助识别有趣的模式，但是仅有这些还不够，还要结合反映特定用户需要和兴趣的主观度量。例如，对于市场经理，描述频繁在 AllElectronics 购物的顾客特性的模式应当是有趣的；但对于研究同一数据库，分析雇员业绩模式的分析者，它可能不是有趣的。此外，有些根据客观标准有趣的模式可能反映一般知识，因而实际上并不令人感兴趣。**主观兴趣度度量**基于用户对数据的确信。这种度量发现模式是有趣的，如果它们是**出乎意料的**（根据用户的确信），或者提供用户可以采取行动的策略信息。在后一种情况下，这样的模式称为**可行动的**。意料中的模式也可能是有趣的，如果它们证实了用户希望验证的假设，或与用户的预感相似。

第二个问题——"数据挖掘系统能够产生所有有趣的模式吗？"——涉及数据挖掘算法的**完全性**。期望数据挖掘系统产生所有可能的模式是不现实的和低效的。实际上，应当根据用户提供的限制和兴趣度对搜索聚焦。对于某些数据挖掘任务，这通常能够确保算法的完全性。关联规则挖掘就是一个例子，那里，使用限制和兴趣度度量可以确保挖掘的完全性。所涉及的方法细节将在第 6 章详细考察。

最后，第三个问题——"数据挖掘系统能够仅产生有趣的模式吗？"是数据挖掘的优化问题。对于数据挖掘系统，仅产生有趣的模式是非常期望的。这对于用户和数据挖掘系统是非常有效的，因为这样就不需要搜索所产生的模式，以便识别真正有趣的模式。在这方面已经有了进展。然而，在数据挖掘中，这种优化仍然是个挑战。

为了有效地发现对于给定用户有价值的模式，兴趣度度量是必需的。这种度量可以在数据挖掘步之后使用，根据它们的兴趣度评估所发现的模式，过滤掉不感兴趣的那些。更重要的是这种度量可以用来指导和限制发现过程，剪去模式空间中不满足预先设定的兴趣度限制的子集，改善搜索性能。

对于每类可挖掘的模式，评估兴趣度和使用它们改善数据挖掘的有效性的方法将在全书加以讨论。

1.6 数据挖掘系统的分类

数据挖掘是一个交叉科学领域，受多个学科影响（见图 1.11），包括数据库系统、统计、机器学习、可视化和信息科学。此外，依赖于所用的数据挖掘方法，可以使用其它学科的技术，如神经网络、模糊/粗糙集理论、知识表示、归纳逻辑程序设计、或高性能计算。依赖于所挖掘的数据类型或给定的数据挖掘应用，数据挖掘系统也可能集成空间数据分析、信息提取、模式识别、图象分析、信号处理、计算机图形学、Web 技术、经济、或心理学领域的技术。

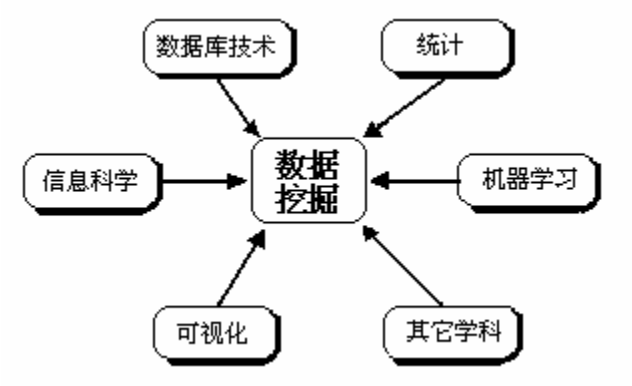


图 1.11：数据挖掘受多学科的影响

由于数据挖掘源于多个学科，因此数据挖掘研究就产生了大量的、各种不同类型数据挖掘系统。这样，就需要对数据挖掘系统给出一个清楚的分类。这种分类可以帮助用户区分数据挖掘系统，确定最适合其需要的数据挖掘系统。根据不同的标准，数据挖掘系统可以分类如下：

根据挖掘的数据库类型分类：数据挖掘系统可以根据挖掘的数据库类型分类。数据库系统本身可以根据不同的标准（如数据模型，或数据或所涉及的应用类型）分类，每一类可能需要自己的数据挖掘技术。这样，数据挖掘系统就可以相应分类。

例如，如果根据数据模型分类，我们可以有关系的、事务的、面向对象的、对象-关系的、或数据仓库的数据挖掘系统。如果根据所处理的数据的特定类型分类，我们有空间的、时间序列的、文本的、或多媒体的数据挖掘系统，或 WWW 数据挖掘系统。

根据挖掘的知识类型分类：数据挖掘系统可以根据所挖掘的知识类型分类。即，根据数据挖掘的功能，如特征、区分、关联、聚类、局外者、趋势和演化分析、偏差分析、类似性分析等分类。一个全面的数据挖掘系统应当提供多种和/或集成的数据挖掘功能。

此外，数据挖掘系统可以根据所挖掘的知识的粒度或抽象层进行区分，包括泛化知识（在高抽象层），原始层知识（在原始数据层），或多层知识（考虑若干抽象层）。一个先进的数据挖掘系统应当支持多抽象层的知识发现。

数据挖掘系统还可以分类为挖掘数据规律（通常出现的模式）和数据反规律（如例外或局外者）。一般地，概念描述、关联分析、分类、预测和聚类挖掘数据规律，将局外者作为噪音排除。这些方法也能帮助检测局外者。

根据所用的技术分类：数据挖掘系统也可以根据所用的数据挖掘技术分类。这些技术可以根据用户交互程度（例如，自动系统、交互探查系统、查询驱动系统），或所用的数据分析方法（例如，面向数据库或数据仓库的技术，机器学习、统计、可视化、模式识别、神经网络等等）描述。复杂的数据挖掘系统通常采用多种数据挖掘技术，或采用有效的、集成的技术，结合一些方法的优点。

根据应用分类：数据挖掘系统可以根据其应用分类。例如，可能有些数据挖掘系统特别适合财政、电讯、DNA、股票市场、e_mail 等等。不同的应用通常需要集成对于该应用特别有效的方法。因此，普通的、全能的数据挖掘系统可能并不适合特定领域的挖掘任务。

本书的第 5 章至第 8 章根据所挖掘的知识类型组织。在第 9 章，我们讨论在各种先进的数据库系统上，复杂的数据类型的挖掘。第 10 章讨论一些数据挖掘应用。

1.7 数据挖掘的主要问题

本书强调数据挖掘的主要问题，考虑挖掘技术、用户界面、性能和各种数据类型。这些问题介绍如下：

数据挖掘技术和用户界面问题：这反映所挖掘的知识类型、在多粒度上挖掘知识的能力、领域知识的使用、特定的挖掘和知识显示。

- 在数据库中挖掘不同类型的知识：由于不同的用户可能对不同类型的知识感兴趣，数据挖掘系统应当覆盖广谱的数据分析和知识发现任务，包括数据特征、区分、关联、聚类、趋势、偏差分析和类似性分析。这些任务可能以不同的方式使用相同的数据库，并需要开发大量数据挖掘技术。
- 多个抽象层的交互知识挖掘：由于很难准确地知道能够在数据库中发现什么，数据挖掘过程应当是交互的。对于包含大量数据的数据库，应当使用适当的选样技术，进行交互式数据探查。交互式挖掘允许用户聚焦搜索模式，根据返回的结果提出和精炼数据挖掘请求。特殊地，类似于 OLAP 在数据方上做的那样，应当通过交互地在数据空间和知识空间下钻、上卷和转轴，挖掘知识。用这种方法，用户可以与数据挖掘系统交互，以不同的粒度和从不同的角度观察数据和发现模式。
- 结合背景知识：可以使用背景知识或关于所研究领域的信息来指导发现过程，并使得发现的模式以简洁的形式，在不同的抽象层表示。关于数据库的领域知识，如完整性限制和演绎规则，可以帮助聚焦和加快数据挖掘过程，或评估发现的模式的兴趣度。
- 数据挖掘查询语言和特定的数据挖掘：关系查询语言（如 SQL）允许用户提出特定的数据提取查询。类似地，需要开发高级**数据挖掘查询语言**，使得用户通过说明分析任务的相关数据集、领域知识、所挖掘的数据类型、被发现的模式必须满足的条件和兴趣度限制，描述特定的数据挖掘任务。这种语言应当与数据库或数据仓库查询语言集成，并且对于有效的、灵活的数据挖掘是优化的。
- 数据挖掘结果的表示和显示：发现的知识应当用高级语言、可视化表示形式、或其它表示形式表示，使得知识易于理解，能够直接被人使用。如果数据挖掘系统是交互的，这一点尤为重要。这要求系统采用有表达能力的知识表示技术，如树、表、图、图表、交叉表、矩阵或曲线。
- 处理噪音和不完全数据：存放在数据库中数据可能反映噪音、例外情况、或不完全的数据对象。这些对象可能搞乱分析过程，导致数据与所构造的知识模型过分适应。其结果是，所发现的模式的精确性可能很差。需要处理数据噪音的数据清理方法和数据分析方法，以及发现和分析例外情况的局外者挖掘方法。
- 模式评估——兴趣度问题：数据挖掘系统可能发现数以千计的模式。对于给定的用户，许多模式不是有趣的，它们表示平凡知识或缺乏新颖性。关于开发模式兴趣度的评估技术，特别是关于给定用户类，基于用户的信赖或期望，评估模式价值的主观度量，仍然存在一些挑战。使用兴趣度度量，指导发现过程和压缩搜索空间，是又一个活跃的研究领域。

性能问题：这包括数据挖掘算法的有效性、可规模性和并行处理。

- 数据挖掘算法的有效性和可规模性：为了有效地从数据库中大量数据提取信息，数据挖掘算法必须是有效的和可规模化的。换一句话说，对于大型数据库，数据挖掘算法的运行时间必须是可预计的和可接受的。从数据库角度，有效性和可规模性是数据挖掘系统实现的关键问题。上面讨论的挖掘技术和用户交互的大多数问题，也必须考虑有效性和可规模性。
- 并行、分布和增量挖掘算法：许多数据库的大容量、数据的广泛分布和一些数据挖掘算法的计算复杂性是促使开发**并行和分布式数据挖掘算法**的因素。这些算法将数据划分成部分，这些部分可以并行处理，然后合并每部分的结果。此外，有些数据挖掘过程的高花费导致了对**增量**数据挖掘算法的需要。增量算法与数据库更新结合在一起，而不必重新挖掘全部数据。这种算法渐增地进行知识更新，修正和加强先前业已发现的知识。

关于数据库类型的多样性问题：

- 关系的和复杂的数据类型的处理：由于关系数据库和数据仓库已经广泛使用，对它们开发有效的数据挖掘系统是重要的。然而，其它数据库可能包含复杂的数据对象、超文本和多媒体数据、空间数据、时间数据、或事务数据。由于数据类型的多样性和数据挖掘的目标不同，指望一个

系统挖掘所有类型的数据是不现实的。为挖掘特定类型的数据，应当构造特定的数据挖掘系统。这样，对于不同类型的数据，我们可能有不同的数据挖掘系统。

- 由异种数据库和全球信息系统挖掘信息：局域和广域（如 Internet）计算机网络连接了许多数据源，形成了大的、分布的和异种的数据库。从具有不同数据语义的结构的、半结构的、和无结构的不同数据源发现知识，对数据挖掘提出了巨大挑战。数据挖掘可以帮助发现多个异种数据库中的数据规律，这些规律多半难以被简单的查询系统发现，并可以改进异种数据库信息交换和协同操作的性能。Web 挖掘发现关于 Web 连接、Web 使用和 Web 动态情况的有趣知识，已经成为数据挖掘的一个非常具有挑战性的领域。

以上问题是数据挖掘技术未来发展的的主要需求和挑战。在近来的数据挖掘研究和开发中，一些挑战业已受到关注，并已成为必备的，而另一些仍处于研究阶段。然而，这些问题将继续刺激进一步的研究和改进。涉及数据挖掘应用、保密性和社会影响的问题将在本书的最后一章，第 10 章讨论。

1.8 总结

- **数据库技术**已经从原始的数据处理，发展到开发具有查询和事务处理能力的数据库管理系统。进一步的发展导致越来越需要有效的数据分析和数据理解工具。这种需求是各种应用收集的数据爆炸性增长的必然结果；这些应用包括商务和管理、行政管理、科学和工程、环境控制。
- **数据挖掘**是从大量数据中发现有趣模式，这些数据可以存放在数据库、数据仓库或其它信息存储中。这是一个年青的跨学科领域，源于诸如数据库系统、数据仓库、统计、机器学习、数据可视频化、信息提取和高性能计算。其它有贡献的领域包括神经网络、模式识别、空间数据分析、图象数据库、信号处理和一些应用领域，包括商务、经济和生物信息学。
- **知识发现**过程包括数据清理、数据集成、数据变换、数据挖掘、模式评估和知识表示。
- 数据模式可以从不同类型的**数据库**挖掘；如关系数据库，数据仓库，事务的、对象-关系的和面向对象的数据库。有趣的数据模式也可以从其它类型的**信息存储**中提取，包括空间的、时间相关的、文本的、多媒体的和遗产数据库，以及万维网。
- **数据仓库**是一种数据的长期存储，这些数据来自多数据源，是有组织的，以便支持管理决策。这些数据在一种一致的模式下存放，并且通常是汇总的。数据仓库提供一些数据分析能力，称作 **OLAP**（**联机分析处理**）。
- **数据挖掘功能**包括发现概念/类描述、关联、分类、预测、聚类、趋势分析、偏差分析和类似性分析。特征和区分是数据汇总的形式。
- 模式提供**知识**，如果它易于被人理解、在某种程度上对于测试数据是有效的、潜在有用的、新颖的，或者它验证了用户关注的某种预感。**模式兴趣度**度量，无论是客观的还是主观的，都可以用来指导发现过程。
- **数据挖掘系统**可以根据所挖掘的数据库类型、所挖掘的知识类型、或所使用的技术加以分类。
- 大型数据库中有效的数据挖掘对于研究者和开发者提出了大量需求和巨大的挑战。问题涉及数据挖掘技术、用户交互、性能和可规模性、以及大量不同数据类型的处理。其它问题包括数据挖掘的应用开发和它们的社会影响。

习题

1.1 什么是数据挖掘？在你的回答中，强调以下问题：
(a) 它是又一个骗局吗？
(b) 它是一种从数据库、统计和机器学习发展的技术的简单转换吗？
(c) 解释数据库技术发展如何导致数据挖掘。
(d) 当把数据挖掘看作知识发现过程时，描述数据挖掘所涉及的步骤。

1.2 给出一个例子，其中数据挖掘对于商务的成功是至关重要的。该商务需要什么数据挖掘功能？它们能够由数据查询处理或简单的统计分析来实现吗？

1.3 假定你是 *Big-University* 的软件工程师，任务是设计一个数据挖掘系统，分析学校课程数据库。该数据库包括如下信息：每个学生的姓名、地址和状态（例如，本科生或研究生）、所修课程，以及他们的 GPA。描述你要选取的结构。该结构的每个成分的作用是什么？

1.4 数据仓库和数据库有何不同？它们有哪些相似之处？

1.5 简述以下高级数据库系统和应用：面向对象数据库、空间数据库、文本数据库、多媒体数据库、万维网。

1.6 定义下列数据挖掘功能：特征、区分、关联、分类、预测、聚类、演变分析。使用你熟悉的生活中的数据库，给出每种数据挖掘功能的例子。

1.7 区分和分类的差别是什么？特征和聚类的差别是什么？分类和预测呢？对于每一对任务，它们有何相似之处？

1.8 根据你的观察，描述一个可能的知识类型，它需要由数据挖掘方法发现，但本章未列出。它需要一种不同于本章列举的数据挖掘技术吗？

1.9 描述关于数据挖掘技术和用户交互问题的三个数据挖掘挑战。

1.10 描述关于性能问题的两个数据挖掘挑战。

文献注释

Piatetsky-Shapiro 和 Frawley 编辑的书 *Knowledge Discovery in Databases* [PSF91]是数据库中知识发现早期研究论文的汇集。Fayyad, Piatetsky-Shapiro, Smyth 和 Uthurusamy 编辑的书 *Advance in Knowledge Discovery and Data Mining*[FPS+96]是一本知识发现和数据挖掘研究成果的很好的汇集。其它数据挖掘书籍包括 Weiss 和 Indurkhya 的 *Predictive Data Mining* [WI98]，Michalski, Brakto 和 Kubat 的 *Machine Learning and Data Mining: Methods and Applications* [MBK98]，Westphal 和 Blaxton 的 *Data Mining Solutions: Methoda and Tools for Solving Real-World Problems* [WB98]，Berry 和 Linoff 的 *The Art and Science of Customer Relationship Management* [BL99]，Berson, Smith 和 Thearling 的 *Building Data Mining Applications for CRM* [BST99]，和 Groth 的 *Data Mining: Building Competitive Advantadge* [Gro99]。还有一些书包含知识发现特定方面应用的论文，如 Ziarko 编辑的 *Rough Sets, Fuzzy Sets and Knowledge Discovery* [Zia94]，以及一些数据挖掘指南手册，如 ACM 出版社出版的 *Tutorial Notes of the 1999 International Conference on Knowledge Discovery and Data Mining*（KDD'99）。

KDD Nuggets 是一个包含知识发现和数据挖掘有关信息的定期的、免费的电子通讯。投稿可以连同描述主题行（和 URL）用电子邮件发往 editor@kdnuggets.com。关于订阅的信息可以在 http://www.kdnuggets.com/news/subscribe.html 找到。自 1991 年以来，*KDD Nuggets* 已被 Piatetsky_Shapiro 调整。位于 http://www.kdnuggets.com/的 Internet 的站点 *Knowledge Discovery Mine* 包含大量关于 KDD 的信息。

数据挖掘界于 1995 年开始了它的第一届次知识发现与数据挖掘国际学术会议 [FU96]。该会议是由 1989 至 1994 年举行的四次数据库中知识发现国际研讨会[PS89，PS91a，FU93，FU94]发展起来的。数据挖掘研究界于 1998 年建立起一个新的学术组织 ACM-SIGKDD，ACM 下的数据库中知

识发现专业组。1999 年 ACM-SIGKDD 组织了第五届知识发现与数据挖掘国际学术会议（KDD'99）。专题杂志 *Data Mining and Knowledge Discovery* 自 1997 年起由 Kluwers 出版社出版。ACM-SIGKDD 还出版一种季刊电子通讯 *SIGKDD Explorations*，SIGKDD 成员可以使用。还有一些其它国际或地区性数据挖掘会议，如，知识发现与数据挖掘太平洋亚洲会议（PAKDD），数据库中知识发现原理与实践欧洲会议（PKDD），和数据仓库与知识发现国际会议（DaWaK）。

数据挖掘研究还在出版的书籍、会议、以及数据库、统计、机器学习和数据可视化杂志上发表。源于这些的文献列举如下。

数据库系统的流行教科书包括 Ullamn 的 *Principles of Database and Knowledege-Base Systems, Vol.* 1 [Ull88]，Elmasri 和 Navathe 的 *Fundamentals of Database Systems,2nd ed.[*EN94]，Silberschatz, Korth 和 Sudarshan 的 *Database System Concepts* [SK97]，Ullman 和 Widom 的 *A First Course in Database Systems* [UW97]，和 Ramakrishnan 和 Gehrke 的 *Database Management Systems, 3rd ed.* [RG00]。数据库系统的文章汇集见 Stonebraker 和 Hellerstein 编辑的 *Readings in Database Systems* [SH98]。关于数据库系统的成就与研究挑战的回顾与讨论在 Stonebraker, Agrawal, Dayal 等[SAD+93]和 Silberschatz, Stonebraker 和 Ullman [SSU96]中找到。

在过去的几年中，许多关于数据仓库技术、系统和应用的书籍已经出版。如，Kimball 的 *The Data Warehouse Toolkit* [Kim96]，Inmon 的 *Building the Data Warehouse* [Inm96]，Thomsen 的 *OLAP Solutions: Building Mulyidimensional Information Systems* [Tho97]。Chaudhuri 和 Dayal [CD97]给出了数据仓库技术的全面回顾。

涉及数据挖掘和数据仓库的研究结果已在许多数据库国际学术会议论文集发表，包括 ACM-SIGMOD 数据管理国际会议（SIGMOD），超大型数据库国际会议（VLDB），ACM-SIGMOD-SIGART 数据库原理研讨会（PODS），数据工程国际会议（ICDE），扩展数据库技术国际会议（EDBT），数据库理论国际会议（ICDT），信息与知识管理国际会议（CIKM），数据库与专家系统应用国际会议（DEXA），和数据库系统高级应用国际会议（DASFAA）。数据挖掘研究也发表在主要数据库杂志上，包括 IEEE 知识与数据工程汇刊（TKDE），ACM 数据库系统汇刊（TODS），ACM 杂志（JACM），信息系统，VLDB 杂志，数据与知识工程，和智能信息系统国际杂志（JIIS）。

有许多教材涵盖了统计分析的不同主题，如，Devore 的 *Probability and Statistics for Engineering and Science, 4th ed.* [Dev95]，Neter, Kutner, Nachtsheim 和 Wasserman 的 *Applied Linear Statistical Models, 4th ed.* [NKNW96]，Dobson 的 *An Introduction to Generalized Linear models* [Dob90]，Shumway 的 *Applied Statistical Time Series Analysis 3rd ed.* [Shu88]，和 Johnson 和 Wichern 的 *Applied Multivariate Statistical Analysis, 3rd ed.[*JW92]

统计研究发表在一些主要的统计会议上，包括联合统计会议，皇家统计会国际会议，界面研讨会：计算科学与统计。其它刊物源包括皇家统计会杂志，统计年鉴，美国统计学会杂志，*Technometrics*，和 *Biometrika*。

机器学习方面的教材和书籍包括 Michalski 等编辑的 Machine Learning, An Artifical Intelligence Approach, Vols. 1-4 [MCM83，MCM86，KM90，MT94]，Quinlan 的 C4.5: Programs for Machine Learning [Qui93]，Langley 的 Elements of Machine Learning [Lan96]，和 Mitchell 的 Machine Learning [M97]。Weiss 和 Kulikowski 的书 Cmputer Systemthat Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems [WK91]比较了若干不同领域的分类和比较方法。一本编辑的机器学习论文汇集见 Shavlik 和 Dietterich 的 Readingd in Machine Learning [SD90]。

机器学习研究发表在一些大型机器学习和人工智能会议论文集上，包括机器学习国际会议（ML），ACM 计算学习理论会议（COLT），人工智能国际联合会议（IJCAI），和美国人工智能学会会议（AAAI）。其它出版源包括主要的机器学习、人工智能和知识系统杂志，其中，有些上面已经提到。其余的包括机器学习（ML），人工智能杂志（AI），认知科学。从统计模式识别角度的分类回顾可以在 Duda 和 Hart [DH73]中找到。

数据可视化技术的先驱者工作在 Tufte 的 *The Visual Display of Quantitative Information* [T83]和 *Envisioning Information* [Tuf90]，以及 Bertin 的 *Graphics and Graphic Information Processing* [Ber81]中介绍。Keim 的 *Visual Techniques for Exploring Databases* [Kei97]给出了数据挖掘可视化的指南。可视化主要的会议和研讨会包括 ACM 计算机系统中人的因素（CHI），可视化，以及信息可视化

国际研讨会。可视化研究也发表在可视化和计算机图形学汇刊，计算和图形统计杂志，以及 IEEE 计算机图形学及其应用。

第二章 数据仓库和数据挖掘的 OLAP 技术

构造数据仓库涉及数据清理和数据集成，可以看作数据挖掘的一个重要预处理步骤。此外，数据仓库提供联机分析处理(OLAP)工具，用于各种粒度的多维数据分析，有利于有效的数据挖掘。进一步讲，许多其它数据挖掘功能，如分类、预测、关联、聚集，都可以与 OLAP 操作集成，以加强多个抽象层上的交互知识挖掘。因此，数据仓库已经成为数据分析和联机数据分析处理日趋重要的平台，并将为数据挖掘提供有效的平台。在系统地介绍数据挖掘技术之前，我们概括地介绍数据仓库技术。对于理解数据挖掘技术，这种概述是必要的。

本章，你将学习数据仓库和 OLAP 技术使用的基本概念、一般结构和主要实现技术，以及它们与数据挖掘的联系。

2.1 什么是数据仓库?

数据仓库为商务运作提供结构与工具，以便系统地组织、理解和使用数据进行决策。大量组织机构已经发现，在当今这个充满竞争、快速发展的世界，数据仓库是一个有价值的工具。在过去的几年中，许多公司已花费数百万美元，建立企业范围的数据仓库。许多人感到，随着工业竞争的加剧，数据仓库成了必备的最新营销武器——通过更多地了解客户需求而保住客户的途径。

“那么”，你可能会充满神秘地问，“到底什么是数据仓库？”数据仓库已被多种方式定义，使得很难严格地定义它。宽松地讲，数据仓库是一个数据库，它与组织机构的操作数据库分别维护。数据仓库系统允许将各种应用系统集成在一起，为统一的历史数据分析提供坚实的平台，对信息处理提供支持。

按照 W. H. Inmon，一位数据仓库系统构造方面的领头建筑师的说法，“**数据仓库**是一个**面向主题的、集成的、时变的、非易失的**数据集合，支持管理决策制定”[Inm96]。这个简短、全面的定义指出了数据仓库的主要特征。四个关键词，面向主题的、集成的、时变的、非易失的，将数据仓库与其它数据存储系统（如，关系数据库系统、事务处理系统、和文件系统）相区别。让我们进一步看看这些关键特征。

- **面向主题的**：数据仓库围绕一些主题，如顾客、供应商、产品和销售组织。数据仓库关注决策者的数据建模与分析，而不是构造组织机构的日常操作和事务处理。因此，数据仓库排除对于决策无用的数据，提供特定主题的简明视图。
- **集成的**：通常，构造数据仓库是将多个异种数据源，如关系数据库、一般文件和联机事务处理记录，集成在一起。使用数据清理和数据集成技术，确保命名约定、编码结构、属性度量的一致性。
- **时变的**：数据存储从历史的角度（例如，过去 5-10 年）提供信息。数据仓库中的关键结构，隐式或显式地包含时间元素。
- **非易失的**：数据仓库总是物理地分离存放数据；这些数据源于操作环境下的应用数据。由于这种分离，数据仓库不需要事务处理、恢复和并行控制机制。通常，它只需要两种数据访问：数据的初始化装入和数据访问。

概言之，数据仓库是一种语义上一致的数据存储，它充当决策支持数据模型的物理实现，并存放企业决策所需信息。数据仓库也常常被看作一种体系结构，通过将异种数据源中的数据集成在一起而构造，支持结构化和启发式查询、分析报告和决策制定。

“好”，你现在问，“那么，什么是建立数据仓库(data warehousing)？”

根据上面的讨论，我们把建立数据仓库看作构造和使用数据仓库的过程。数据仓库的构造需要数据集成、数据清理、和数据统一。利用数据仓库常常需要一些决策支持技术。这使得“知识工人”（例如，经理、分析人员和主管）能够使用数据仓库，快捷、方便地得到数据的总体视图，根据数据仓库中的信息作出准确的决策。有些作者使用术语“建立数据仓库”表示构造数据仓库的过程，而用术语“**仓库 DBMS**”表示管理和使用数据仓库。我们将不区分二者。

“组织机构如何使用数据仓库中的信息？”许多组织机构正在使用这些信息支持商务决策活动，包括（1）增加顾客关注，包括分析顾客购买模式（如，喜爱买什么、购买时间、预算周期、消费习惯）；（2）根据季度、年、地区的营销情况比较，重新配置产品和管理投资，调整生产策略；（3）分析运作和查找利润源；（4）管理顾客关系、进行环境调整、管理合股人的资产开销。

从异种数据库集成的角度看，数据仓库也是十分有用的。许多组织收集了形形色色数据，并由多个异种的、自治的、分布的数据源维护大型数据库。集成这些数据，并提供简便、有效的访问是非常希望的，并且也是一种挑战。数据库工业界和研究界都正朝着实现这一目标竭尽全力。

对于异种数据库的集成，传统的数据库做法是：在多个异种数据库上，建立一个**包装程序**和一个**集成程序**（或**仲裁程序**）。这方面的例子包括 IBM 的数据连接程序 (Data Joiner) 和 Informix 的数据刀(DataBlade)。当一个查询提交客户站点，首先使用元数据字典对查询进行转换，将它转换成相应异种站点上的查询。然后，将这些查询映射和发送到局部查询处理器。由不同站点返回的结果被集成为全局回答。这种**查询驱动的方法**需要复杂的信息过滤和集成处理，并且与局部数据源上的处理竞争资源。这种方法是低效的，并且对于频繁的查询，特别是需要聚集操作的查询，开销很大。

对于异种数据库集成的传统方法，数据仓库提供了一个有趣的替代方案。数据仓库使用**更新驱动的方法**，而不是查询驱动的方法。这种方法将来自多个异种源的信息预先集成，并存储在数据仓库中，供直接查询和分析。与联机事务处理数据库不同，数据仓库不包含最近的信息。然而，数据仓库为集成的异种数据库系统带来了高性能，因为数据被拷贝、预处理、集成、注释、汇总，并重新组织到一个语义一致的数据存储中。在数据仓库中进行的查询处理并不影响在局部源上进行的处理。此外，数据仓库存储并集成历史信息，支持复杂的多维查询。这样，建立数据仓库在工业界已非常流行。

2.2.1 操作数据库系统与数据仓库的区别

由于大多数人都熟悉商品关系数据库系统，将数据仓库与之比较，就容易理解什么是数据仓库。

联机操作数据库系统的主要任务是执行联机事务和查询处理。这种系统称为**联机事务处理（OLTP）**系统。它们涵盖了一个组织的大部分日常操作，如购买、库存、制造、银行、工资、注册、记帐等。另一方面，数据仓库系统在数据分析和决策方面为用户或“知识工人”提供服务。这种系统可以用不同的格式组织和提供数据，以便满足不同用户的形形色色需求。这种系统称为**联机分析处理（OLAP）**系统。

OLTP 和 OLAP 的主要区别概述如下。

- **用户和系统的面向性：**OLTP 是面向顾客的，用于办事员、客户、和信息技术专业人员的事务和查询处理。OLAP 是面向市场的，用于知识工人（包括经理、主管、和分析人员）的数据分析。
- **数据内容：**OLTP 系统管理当前数据。通常，这种数据太琐碎，难以方便地用于决策。OLAP 系统管理大量历史数据，提供汇总和聚集机制，并在不同的粒度级别上存储和管理信息。这些特点使得数据容易用于见多识广的决策。
- **数据库设计：**通常，OLTP 系统采用实体-联系（ER）模型和面向应用的数据库设计。而 OLAP 系统通常采用星形或雪花模型(2.2.2 小节讨论）和面向主题的数据库设计。
- **视图：**OLTP 系统主要关注一个企业或部门内部的当前数据，而不涉及历史数据或不同组织的数据。相比之下，由于组织的变化，OLAP 系统常常跨越数据库模式的多个版本。OLAP 系统也处

理来自不同组织的信息，由多个数据存储集成的信息。由于数据量巨大，OLAP 数据也存放在多个存储介质上。

- **访问模式：**OLTP 系统的访问主要由短的、原子事务组成。这种系统需要并行控制和恢复机制。然而，对 OLAP 系统的访问大部分是只读操作（由于大部分数据仓库存放历史数据，而不是当前数据），尽管许多可能是复杂的查询。

OLTP 和 OLAP 的其它区别包括数据库大小、操作的频繁程度、性能度量等。这些都概括在表 2.1 中。

表 2.1: OLTP 系统和 OLAP 系统的比较

特性	OLTP	OLAP
特征	操作处理	信息处理
面向	事务	分析
用户	办事员、DBA、数据库专业人员	知识工人（如经理、主管、分析员）
功能	日常操作	长期信息需求，决策支持
DB 设计	基于 E-R，面向应用	星形/雪花，面向主题
数据	当前的；确保最新	历史的；跨时间维护
汇总	原始的，高度详细	汇总的，统一的
视图	详细，一般关系	汇总的，多维的
工作单位	短的、简单事务	复杂查询
存取	读/写	大多为读
关注	数据进入	信息输出
操作	主关键字上索引/散列	大量扫描
访问记录数量	数十个	数百万
用户数	数千	数百
DB 规模	100MB 到 GB	100GB 到 TB
优先	高性能，高可用性	高灵活性，端点用户自治
度量	事务吞吐量	查询吞吐量，响应时间

2.1.2 但是，为什么需要一个分离的数据仓库

“既然操作数据库存放了大量数据”，你注意到，“为什么不直接在这种数据库上进行联机分析处理，而是另外花费时间和资源去构造一个分离的数据仓库？”分离的主要原因是提高两个系统的性能。操作数据库是为已知的任务和负载设计的，如使用主关键字索引和散列，检索特定的记录，和优化“罐装的”查询。另一方面，数据仓库的查询通常是复杂的，涉及大量数据在汇总级的计算，可能需要特殊的数据组织、存取方法和基于多维视图的实现方法。在操作数据库上处理 OLAP 查询，可能会大大降低操作任务的性能。

此外，操作数据库支持多事务的并行处理，需要加锁和日志等并行控制和恢复机制，以确保一致性和事务的强健性。通常，OLAP 查询只需要对数据记录进行只读访问，以进行汇总和聚集。如果将并行控制和恢复机制用于这种 OLAP 操作，就会危害并行事务的运行，从而大大降低 OLTP 系统的吞吐量。

最后，数据仓库与操作数据库分离是由于这两种系统中数据的结构、内容和用法都不相同。决策支持需要历史数据，而操作数据库一般不维护历史数据。在这种情况下，操作数据库中的数据尽管很丰富，但对于决策，常常还是远远不够的。决策支持需要将来自异种源的数据统一（如，聚集和汇总），产生高质量的、纯净的和集成的数据。相比之下，操作数据库只维护详细的原始数据（如事务），这些数据在进行分析之前需要统一。由于两个系统提供很不相同的功能，需要不同类型的数据，因此需要维护分离的数据库。然而，许多关系数据库管理系统卖主正开始优化这种系统，使之支持 OLAP 查询。随着这一趋势的继续，OLTP 和 OLAP 系统之间的分离可望消失。

2.2 多维数据模型

数据仓库和 OLAP 工具基于**多维数据模型**。该模型将数据看作**数据方**形式。本节，你将学习数据方如何对 *n*-维（*n-D*）数据建模。你还将学习概念分层，以及如何在基本 OLAP 操作中使用它们，在多个抽象层上进行交互式挖掘。

2.2.1 由表和电子数据表到数据方

“什么是数据方？”数据方允许以多维对数据建模和观察。它由维和事实定义。

一般地，**维**是透视或关于一个组织想要记录的实体。例如，AllElectronics 可能创建一个数据仓库 *sales*，记录商店的销售，涉及维 *time, item, branch*, 和 *location*。这些维使得商店能够记录商品的月销售，销售商品的分店和地点。每一个维都有一个表与之相关联。该表称为**维表**，它进一步描述维。例如，*item* 的维表可以包含属性 *item_name, branch,* 和 *type*。维表可以由用户或专家设定，或者根据数据分布自动产生和调整。

通常，多维数据模型围绕中心主题（例如，*sales*）组织。该主题用事实表表示。**事实**是数值度量的。把它们看作数量，是因为我们想根据它们分析维之间的关系。例如，数据仓库 *sales* 的事实包括 *dollars_sold*, *units_sold* 和 *amount_budgeted*。事实表包括事实的名称或度量，以及每个相关维表的关键字。当我们稍后考察多维模式时，你很快就会明白这一切如何工作。

尽管我们经常把数据方看作 3-D 几何结构，在数据仓库中，数据方是 *n*-D 的。为了更好地理解数据方和多维数据模型，让我们由考察 2-D 数据方开始。事实上，它是 AllElectronics 的销售数据表或电子数据表。特殊地，我们将观察 AllElectronics 的销售数据中 Vancouver 每季度销售的商品；这些数据在表 2.2 中。在这个 2-D 表示中，Vancouver 的销售用维 *time*（按季度组织）和维 *item*（按所售商品的类型组织）表示。所显示的事实或度量是 *dollars_sold*（单位：$1000)。

表 2.2: AllElectronics 的销售数据按照维 *time*, *item* 的 2-D 视图。其中销售是取自 *location* = “Vancouver”的所有分店，所显示的度量是 *dollars_sold*

location = “Vancouver”			
	item (type)		
time (quarter)	家庭娱乐 安全	计算机	电话
Q1	605		825
Q2	14	400	
Q3	680		952
Q4	31	512	
	812		1023
	30	501	
	927		1038
	38	580	

现在，假定我们想以三维角度观察销售数据。例如，我们想根据 *time*, *item*, 和 *location* 观察数据。*location* 是城市 Chicago, New York, Toronto 和 Vancouver。3-D 数据如表 2.3 所示。表 2.3 的 3-D 数据表以 2-D 数据表序列的形式表示。概念上讲，我们也可以以 3-D 数据方的形式表示这些数据，如图 2.1 所示。

表 2.3 Allelectronics 销售数据的 3-D 视图, 根据 *time*, *item*, 和 *location*, 所显示的度量是 *dollars_sold* （单位：$1000）

	location= “Chicago”				**location**=“New York”				l**ocation**=“Toronto”				**location**=“Vancouver”			
	item				**item**				**item**				**item**			
time	家庭娱乐	计算机	电话	安全	家庭娱乐	计算机	电话		家庭娱乐	计算机	电话		家庭娱乐	计算机	电话	安全

Q1	854	882	89	1087	968	38	819	746	43	605	825	14
Q2	623			872			591			400		
Q3	943	890	64	1130	1024	41	894	769	52	680	952	31
Q4	698			925			682			512		
	1032	924	59	1034	1048	45	940	795	58	812	1023	30
	789			1002			728			501		
	1129	992	63	1142	1091	54	978	864	59	927	1038	38
	870			984			784			580		

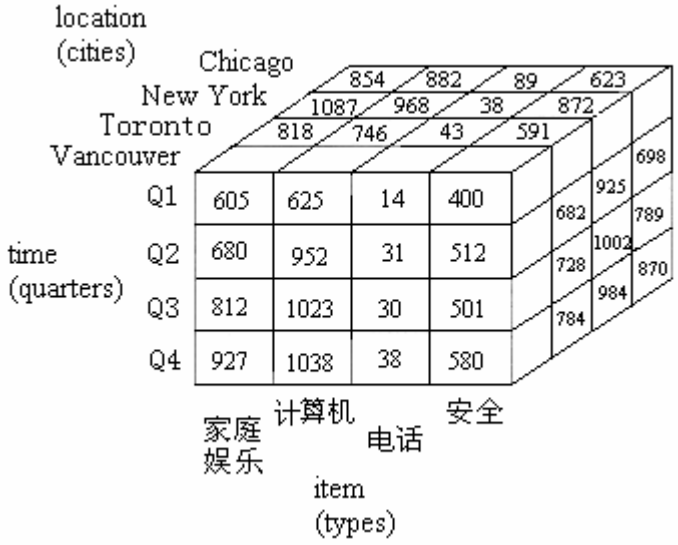


图 2.1：表 2.3 数据的 3-D 数据方表示，维是 time,item 和 location，所显示的度量为 dollars_sold（单位：$1000）

现在，假定我们想从四维的角度观察销售数据，附加一维，如 *supplier*。观察 4-D 事物变得麻烦。然而，我们可以把 4-D 方看成 3-D 方的序列，如图 2.2 所示。如果我们按这种方法继续下去，我们可以把任意 *n*-D 数据方显示成(*n*-1)-D 数据方的序列。数据方是对多维数据存储的一种比喻，这种数据的实际物理存储不同于它的逻辑表示。重要的是，数据方是 *n* 维的，而不限于 3-D。

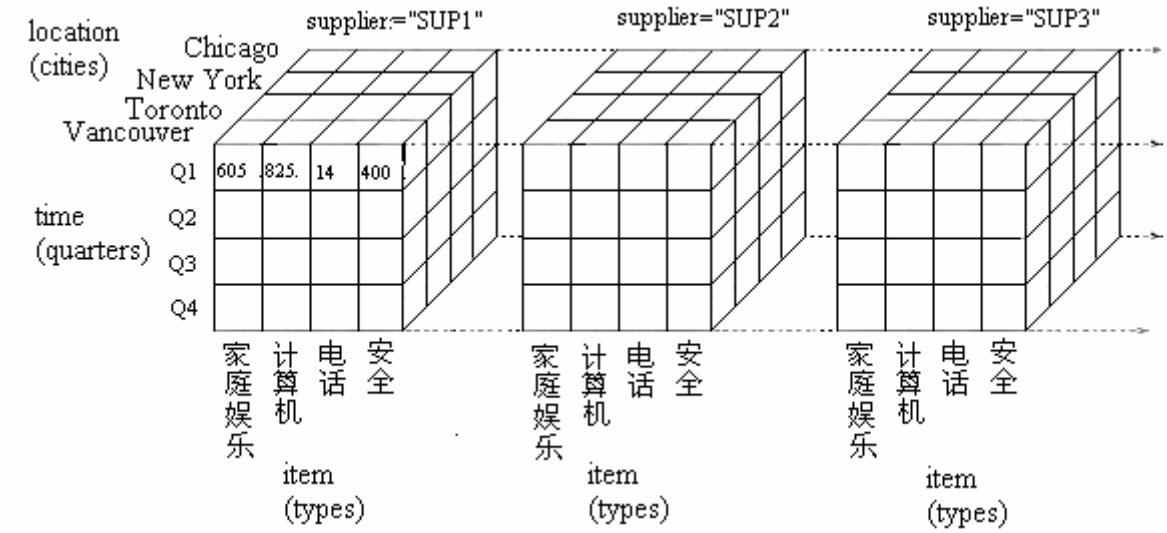


图 2.2　销售数据的 4-D 数据方表示，维是和 *time,item,location* 和 *supplier*，所显示的度量为 *dollars_sold*（单位：$1000）

上面的表显示不同汇总级的数据。在数据仓库研究界，上面所示的每个数据方称作一个方体。**给定一个维的集合，我们可以构造方体的**格，**每个在不同的汇总级或** group by[3]**（即，根据维的不同**

[3] 注意，在本书中，查询语言关键字用黑体。

子集）显示数据。方体的格称作数据方。图 2.3 给出形成维 *time, item, location* 和 *supplierr* 的数据方的方体格。

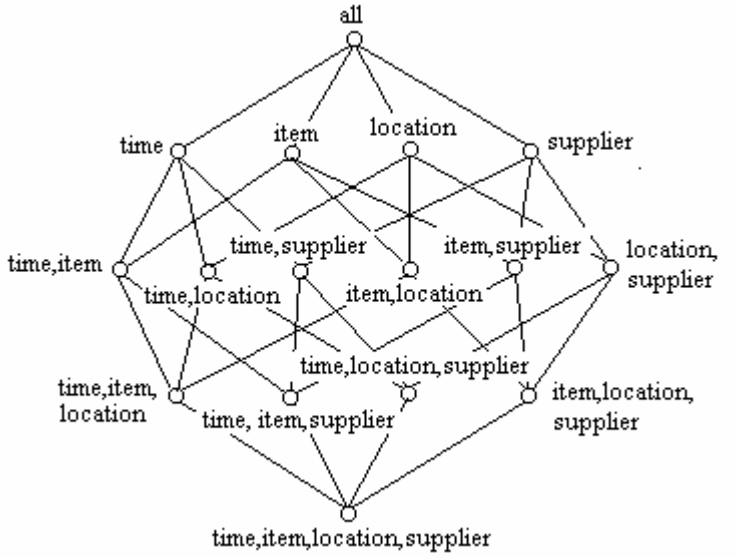


图 2.3　方体格，形成维 *time, item, location* 和 *supplierr* 的 4-D 数据方。每个方体代表一个不同的汇总

存放最低层汇总的方体称为**基本方体**。例如，图 2.2 中的 4-D 方体是给定维 *time, item, location* 和 *supplier* 的基本方体。图 2.1 是 *time, item* 和 *location* 的 3-D 方体（非基本的），对所有的供应商汇总。0-D 方体存放最高层的汇总，称作**顶点方体**。在我们的例子中，这是总销售 *dollars_sold*，在所有的四个维上汇总。顶点方体通常用 **all** 标记。

2.2.2 星形、雪花和事实星座：多维数据库模式

实体-联系数据模型广泛用于关系数据库设计。在那里，数据库模式由实体的集合和它们之间的联系组成。这种数据模型适用于联机事务处理。然而，数据仓库需要简明的、面向主题的模式，便于联机数据分析。

最流行的数据仓库数据模型是**多维数据模型**。这种模型可以以**星形模式**、**雪花模式**、或**事实星座模式**形式存在。让我们看看这些模式。

星形模式：最常见的模型范例星形模式；其中数据仓库包括（1）一个大的、包含大批数据、不含冗余的中心表（**事实表**）；（2）一组小的附属表（**维表**），每维一个。这种模式图很象星星爆发，维表围绕中心表显示在射线上。

例 2.1 作为一个例子，Allelectronics 的星形模式如图 2.4 所示。*sales* 有四个维，分别是 *time, item, branch* 和 *location*。该模式包含一个中心事实表 *sales*，它包含四个维的关键字和两个度量 *dollars_sold* 和 *units_sold*。为尽量减小事实表的尺寸，维标识符（如，*time_key* 和 *item_key*）是系统产生的标识符。□

注意：在星形模式中，每维只用一个表表示，每个表包含一组属性。例如，*location* 维表包含属性集{*location_key, street, city, province_or_state, country*}。这一限制可能造成某些冗余。例如，“Vancouver”和“Victoria”都是加拿大不列颠哥伦比亚省的城市。*location* 维表中这些城市实体的属性 *province_or_state*，*country* 之间都会有些冗余，即，(..., Vancouver, British Columbia, Canada), (...,Victoria, British Columbia, Canada)。此外，一个维表中的属性可能形成一个层次（全序）或格（偏序）。

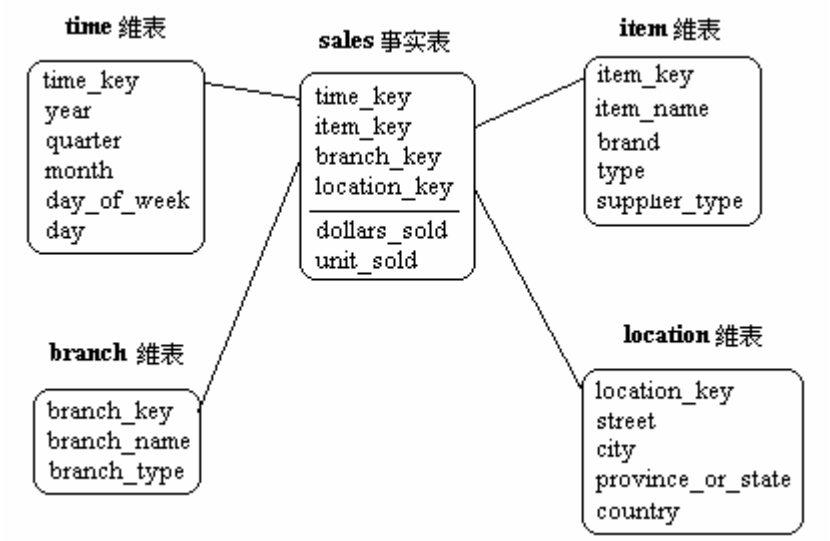


图 2.4： *Sales* 数据仓库的星形模式

雪花模式：雪花模式是星型模式的变种，其中某些维表是规范化的，因而把数据进一步分解到附加的表中。结果，模式图形成类似于雪花的形状。

雪花模式和星形模式的主要不同在于，雪花模式的维表可能是规范化形式，以便减少冗余。这种表易于维护，并节省存储空间，因为当维结构作为列包含在内时，大维表可能非常大。然而，与巨大的事实表相比，这种空间的节省可以忽略。此外，由于执行查询需要更多的连接操作，雪花结构可能降低浏览的性能。这样，系统的性能可能相对受到影响。因此，在数据仓库设计中，雪花模式不如星形模式流行。

例 2.2 作为一个例子，Allelectronics 的 *sales* 的雪花模式在图 2.5 给出。这里，*sales* 事实表与图 2.4 的星形模式相同。两个模式的主要不同是维表。星形模式中的 *item* 的单个维表在雪花模式中被规范化，导致新的 *item* 表和 *supplier* 表。例如，现在 *item* 维表包含属性 *item_key, item_name, brand, type* 和 *supplier_key*，*supplier_key* 连接到 *supplier* 维表。而 *supplier* 维表包含信息 *supplier_key* 和 *supplier_type*。类似地，星形模式中 *location* 的单个维表被规范化成两个表：新的 *location* 和 *city*。新的 *location* 表中的 *location_key* 现在连接到 *city* 维。注意，如果愿意的话，图 2.5 雪花模式中的 *province_or_state* 和 *country* 还可以进一步规范化。□

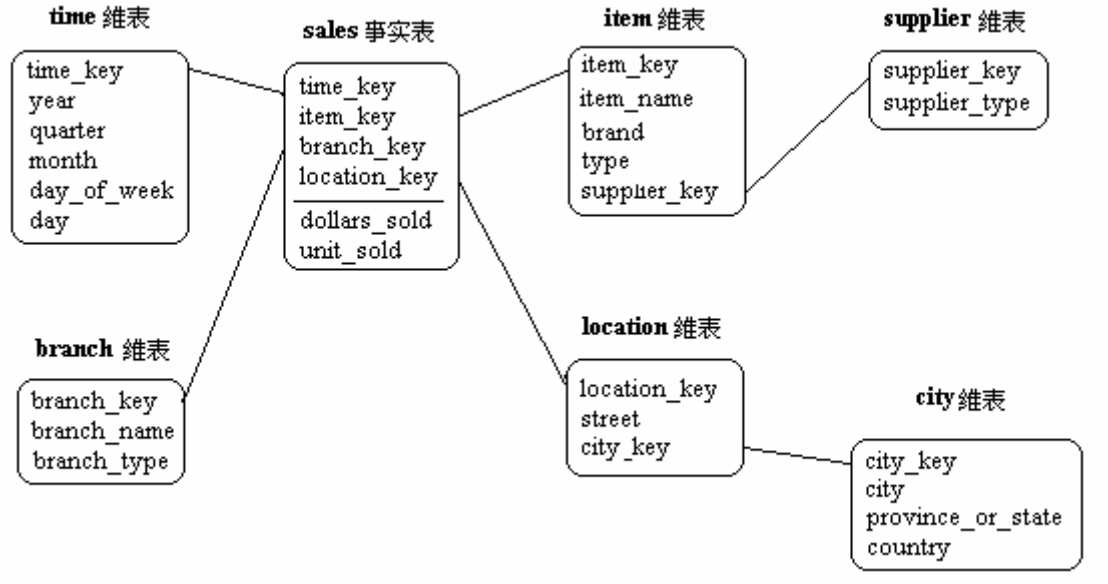


图 2.5： sales 数据仓库的雪花模式

事实星座：复杂的应用可能需要多个事实表共享维表。这种模式可以看作星形模式集，因此称为**星系模式**，或**事实星座**。

例 2.3 一个事实星座的例子在图 2.6 中给出。该模式说明了两个事实表，*sales* 和 *shipping*。*sales* 表的定义与星形模式（图 2.4）相同。*shipping* 表有五个维或关键字：*item_key, time_key, shipper_key, from_location* 和 *to_location*；两个度量：*dollars_cost* 和 *units_shipped*。事实星座模式允许事实表共享维表。例如，*sales* 和 *shipping* 事实表共享维表 *time, item* 和 *location*。□

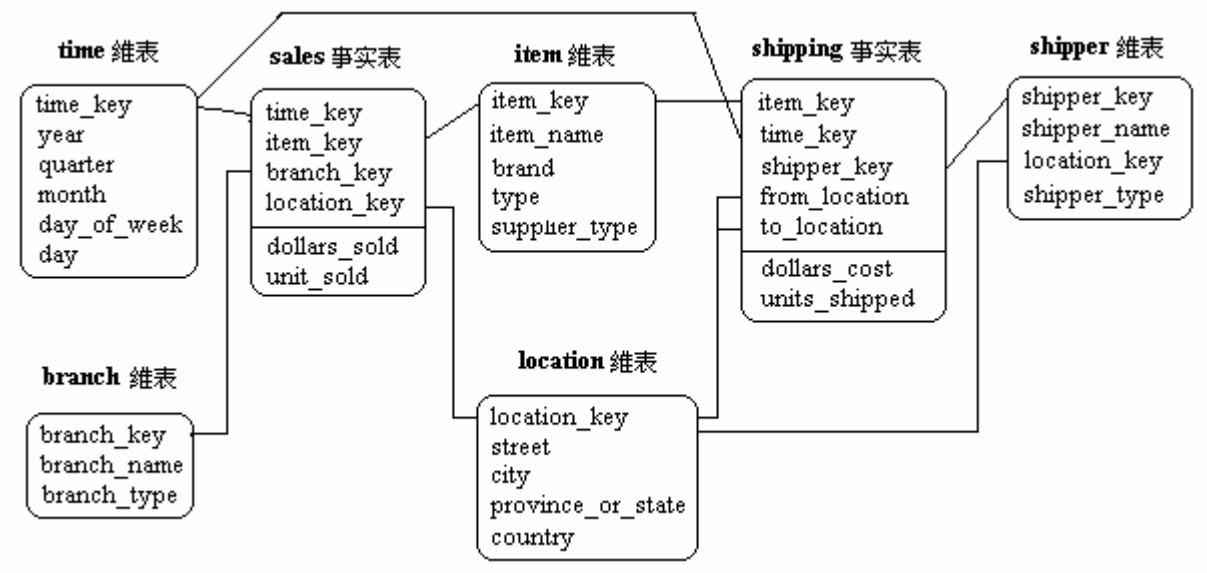


图 2.6：*sales* 和 *shipping* 数据仓库的事实星座模式

在数据仓库中，数据仓库和数据集市是有区别的。数据仓库收集了关于整个组织的主题（如顾客、商品、销售、资产和人员）信息，因此是企业范围的。对于数据仓库，通常使用事实星座模式，因为它能对多个相关的主题建模。另一方面，**数据集市**是数据仓库的一个部门子集，它针对选定的主题，因此是部门范围的。对于数据集市，流行星形或雪花模式，因为它们都适合对单个主题建模，尽管星形模式更流行、更有效。

2.2.3 定义星形、雪花和事实星座的例子

“我怎样对我的数据定义多维模式？”正象关系数据库查询语言 SQL 可以用于说明关系查询一样，**数据挖掘查询语言**可以用于说明数据挖掘任务。特殊地，我们考察一种基于 SQL 的数据挖掘查询语言 **DMQL**。DMQL 包括定义数据仓库和数据集市的语言原语。说明其它数据挖掘任务的原语，如挖掘概念/类描述、关联、分类等，将在第 4 章介绍。

数据仓库和数据集市可以使用两种原语定义：一种是方定义，一种是维定义。方定义语句具有如下语法形式：

define cube <cube_name> [<dimension_list>] : <measure_list>

维定义语句具有如下语法形式：

define dimension <dimension_name> **as** (<attribute_or_subdimension_list>)

让我们看一些例子，看看如何使用 DMQL 定义例 2.1 到 2.3 的星形、雪花和星座模式。

例 2.4 例 2.1 和图 2.4 定义的星形模式用 DMQL 定义如下：

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), units_sold = count(*)
define dimension time **as** (time_key, day, day_of_week, month, quarter, year)
define dimension item **as** (item_key, item_name, brand, type, supplier_type)
define dimension branch **as** (branch_key, branch_name, branch_type)

define dimension location **as** (location_key, street, city, province_or_state, country)

define cube 语句定义一个方，叫做 *sales_star*，它对应于例 2.1 的中心表 *sales* 事实表。该命令说明维表的关键字和两个度量，*dollars_sold* 和 *units_sold*。数据方具有四个维，分别为 *time*, *item*, *branch* 和 *location*。一个 **define dimension** 语句定义一个维。□

例 2.5 例 2.2 和图 2.5 定义的雪花模式用 DMQL 定义如下：

define cube sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), units_sold = count(*)
define dimension time **as** (time_key, day, day_of_week, month, quarter, year)
define dimension item **as** (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
define dimension branch **as** (branch_key, branch_name, branch_type)
define dimension location **as** (location_key, street, city(city_key, city, province_or_state, country))

该定义类似于 *sales_star*（例 2.4），不同的是这里 *item* 和 *location* 维表是规范化的。例如，在 *sales_snowflake* 数据方中，*sales_star* 数据方的 *item* 维被规范化成两个维表，*item* 和 *supplier*。注意：*supplier* 的维定义在 *item* 的定义中说明。用这种方式定义 *supplier*，隐式地在 *item* 的定义中创建了一个 *supplier_key*。类似地，在 *sales_snowflake* 数据方中，*sales_star* 数据方的 *location* 维被规范化成两个维表，*location* 和 *city*。*city* 的维定义在 *location* 的定义中说明。用这种方式，*city_key* 在 *location* 的定义中隐式地创建。□

例 2.6 例 2.3 和图 2.6 定义的星座模式用 DMQL 定义如下：

define cube sales [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), units_sold = count(*)
define dimension time **as** (time_key, day, day_of_week, month, quarter, year)
define dimension item **as** (item_key, item_name, brand, type, supplier_type)
define dimension branch **as** (branch_key, branch_name, branch_type)
define dimension location **as** (location_key, street, city, province_or_state, country)

define cube shipping [time, item, shipper, from_location, to_location]:
dollars_sold = sum(cost_in_dollars), units_sipped = count(*)
define dimension time **as** time **in cube** sales
define dimension item **as i**tem **in cube** sales
define dimension shipper **as** (shipper_key, shipper_name, location **as** location **in cube** sales,
shipper_type)
define dimension from_location **as** location **in cube** sales
define dimension to_location **as** location **in cube** sales

define cube 语句用于定义数据方 *sales* 和 *shipping*，分别对应于例 2.3 模式的两个事实表。注意，数据方 *sales* 的 *time*, *item* 和 *location* 维可以与数据方 *shpping* 共享。例如，*time* 维，在定义数据方 *shipping* 语句之下，用“**define dimension** time **as** time **in cube** sales”说明。□

2.2.3 度量：它们的分类和计算

“如何计算度量？”为回答这个问题，我们首先看看如何对度量分类。注意，数据方空间的多维点由维-值对定义。例如, <*time* = “Q1”, *location* = “Vancouver”, *item* = “computer”>。数据方**度量**是一个数值函数，该函数可以对数据方的每一个点求值。通过对给定点的各维-值对聚集数据，计算该点的度量值。稍后，我们看一些具体的例子。

度量可以根据其所用的聚集函数分成三类：

分布的：一个聚集函数是分布的，如果它能以如下分布方式进行计算：设数据被划分为 *n* 个集合，函数在每一部分上的计算得到一个聚集值。如果将函数用于 *n* 个聚集值得到的结果，与将函数用于所有数据得到的结果一样，则该函数可以用分布方式计算。例如，count()可以这样计算：首先将数据方分割成子方的集合，对每个子方计算 count()，然后对这些子方得到的计数求和。因此，count()是分布聚集函数。同理，sum(), min()和 max()是分布聚集函数。一个度量是分布的，如果它可以用分布聚集函数得到。

代数的：一个聚集函数是代数的，如果它能够由一个具有 *M*（其中，*M* 是一个整数界）个参数的代数函数计算，而每个参数都可以用一个分布聚集函数求得。例如，avg()可以由 sum()/count()计算，其中 sum()和 count()是分布聚集函数。类似地，可以表明 min_N(), max_N()和 standard_deviation**()** 是代数聚集函数。一个度量是代数的，如果它可以用代数聚集函数得到。

整体的：一个聚集函数是整体的，如果描述它的子聚集所需的存储没有一个常数界。即，不存在一个具有 *M* 个(其中，*M* 是常数)参数的代数函数进行这一计算。整体函数的常见例子包括 median(), mode()（即，最常出现的项），和 rank()。一个度量是整体的，如果它可以用整体聚集函数得到。

大部分数据方应用需要有效地计算分布的和代数的度量。对于这些，存在许多有效的技术。相比之下，有效地计算整体度量是很困难的。然而，对于有些整体函数的近似计算，有效的技术是存在的。例如，有些技术可以以满意的结果估计大数据集的中值，而不是精确地计算 median()。在许多情况下，这些技术足以克服整体函数有效计算的困难。

例 2.7 许多数据方度量可以用关系的聚集操作计算。在图 2.4 中，我们看到了 AllElectronics 的 *sales* 星形模式，它包含两个度量，*dollars_sold* 和 *units_sold*。在例 2.4 中，我们用 DMQL 命令定义了对应于该模式的 *sales_star* 数据方。“但是，如何解释这些命令，以产生特定的数据方？”

设定义 AllElectronics 的关系数据库模式如下：

```
time(time_key, day, day_of_week, month, quarter, year)
item(item_key, item_name, branch, type)
branch(branch_key, branch_name, branch_type)
location(location_key, street, city, province_or_state, country)
sales(time_key, item_key, branch_key, location_key, number_of_units_sold, price)
```

例 2.4 中 DMQL 说明被翻译成如下 SQL 查询，这些查询产生所需要的 *sales_star* 数据方。这里，聚集函数 sum 用于计算 *dollars_sold* 和 *units_sold*。

```
select s.time_key, s.item_key, s.branch_key, s.location_key,
     sum(s.number_of_units_sold*s.price), sum(s.number_of units_sold)
from time t, item i, branch b, location l, sales s
where s.time_key=t.time_key and s.item_key=i.item_key
     and s.branch_key=b.branch_key and s.location_key=l.location_key
group by s.time_key,s.item_key,s.branch_key,s.location_key
```

以上查询创建的方是 *sales_star* 数据方的**基本方体**。它包含数据方定义中说明的所有维，其中每个维的粒度在**连接键**层。连接键是连接事实表和维表的关键字。与基本方体关联的事实表称为**基本事实表**。

改变 **group by** 子句，可以产生 *sales_star* 数据方的其它方体。例如，我们可以按 *t.month*，而不是按 *s.time_key* 分组，这将按月分组求和，得到度量。去掉“**group by** s.branch_key”，也可以得到较高层的方体（其中，销售将对所有分店求和，而不再按分店）。假定我们修改以上 SQL 查询，去掉所有的 **group by** 子句。这将得到给定数据的 *dollars_sold* 的总和，*units_sold* 的全部计数。这个零维方体称为 *sales_star* 数据方的顶点方体。此外，其它方体可以通过对基本方体进行选择和/或投影产生。按照这种办法，可以把数据方看作由方体的格组成，每个方体对应于给定数据在不同层次上的汇总。□

当前，数据方技术大多限制多维数据库的度量为数值数据。然而，度量也可以用于其它数据类型，如空间、多媒体、和文本数据。这些技术将在第 9 章讨论。

2.2.5 引入概念分层

“什么是概念分层？”一个概念分层定义一个映射序列，将低层概念到更一般的高层概念。考虑维 *location* 的概念分层。*location* 的城市值包括 vancouver, Toronto, New York 和 Chicago。然而，每个城市可以映射到它所属的省或州。例如，Vancouver 可以映射到 British Columbia，而 Chicago 映射到 Illinois。这些省和州依次可以映射到它所属的国家，如加拿大或美国。这些映射形成 *location* 维的概念分层，将低层概念（如，城市）映射到更一般的较高层概念（如，国家）。上面介绍的概念分层如图 2.7 所示。

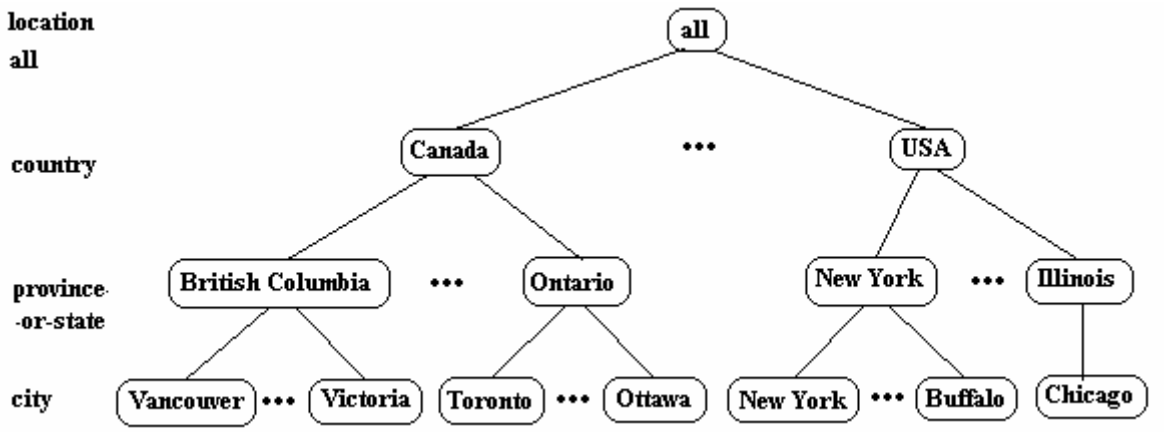


图 2.7　location 维的一个概念分层。由于版面限制，并非所有结点都在图中显示（在结点之间用“...”指出）

许多概念分层隐含在数据库模式中。例如，假定 *location* 维由属性 *number, street, city, province_or_state, zipcode* 和 country 定义。这些属性按一个全序相关，形成一个层次，如“*city* < *province_or_state* < *country*”。该层次如图 2.8(a)所示。维的属性也可以组织成偏序，形成一个**格**。例如，维 *time* 基于属性 *day, week, month, quarter* 和 *year* 就是一个偏序“*day* < {*month* < *quarter; week*} < *year*”[4]。该格结构如图 2.8(b)所示。概念分层为数据库模式中属性的全序或偏序称作**模式分层**。许多应用共有的概念分层，如时间的概念分层，可以在数据挖掘系统中预定义。数据挖掘系统应当为用户提供灵活性，允许用户根据具体的应用剪裁预定义的分层。例如，用户可能想定义财政年由 4 月 1 日开始，而学年由 7 月 1 日开始。

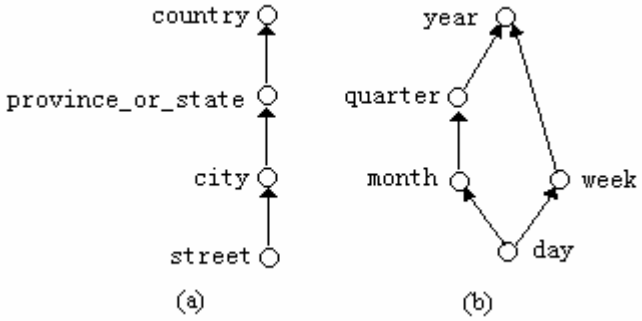


图 2.8:　数据仓库中属性的层次结构和格结构：*location* 的分层；*time* 的格

概念分层也可以通过将给定维或属性的值离散化或分组来定义，产生**集合分组分层**。可以在值组间定义全序或偏序。集合分组概念分层的一个例子如图 2.9 所示关于维 *price* 的集合分组概念分层。其中，区间($*X*...$*Y*]表示由$*X*（不包括）到$*Y*（包括）。

对于一个给定的属性或维，根据不同的用户视图，可能有多个概念分层。例如，用户可能愿意用 *inepensive, moderately_priced* 和 *expensive* 来组织 *price*。

概念分层可以由系统用户、领域专家、知识工程师人工地提供，也可以根据数据分布的统计分析自动地产生。概念分层的自动产生在第 3 章介绍。概念分层的进一步讨论在第 4 章。

正如我们在下一小节将看到的，概念分层允许我们在各种抽象级处理数据。

[4] 由于周(week)通常跨月(month)，常常不把它视为月的低层抽象。然而，常常把它视为年(year)的低层抽象，因为一年大约包含 52 周。

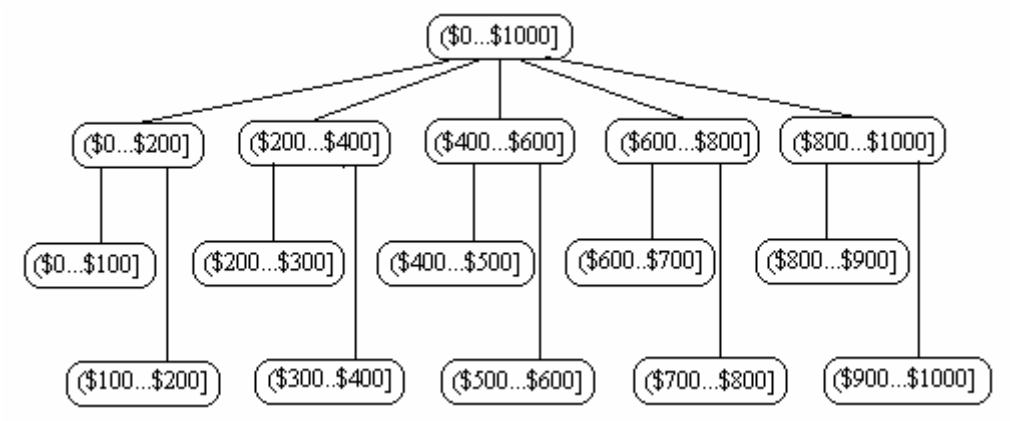


图 2.9: 属性 *price* 的概念分层

2.2.6 多维数据模型上的 OLAP 操作

“在 OLAP 中，如何使用概念分层？”在多维数据模型中，数据组织成多维，每维包含由概念分层定义的多个抽象层。这种组织为用户从不同角度观察数据提供了灵活性。有一些 OLAP 数据方操作用来物化这些不同视图，允许交互查询和分析手头数据。因此 ，OLAP 为交互数据分析提供了友好的环境。

例 2.8 让我们看看一些典型的多维数据 OLAP 操作。所描述的每种操作都图示在图 2.10 中。图的中心是 AllElectronics 的 *sales* 数据方。该数据方包含 *location, time, item* 维；其中，*location* 按 *city* 值聚集，*time* 按 *quarter* 值聚集，而 *item* 按 *types* 聚集。为便于解释，我们称该数据方为中心数据方。所显示的度量是 *dollars-sold*（单位：$1000）。（为提高可读性，只显示一些方体单元值。）所考察的数据是 Vancouver, Toronto, New York 和 Chicago 的数据。

上卷：上卷操作（有些人称之为“上钻”操作）或者通过沿概念分层向上攀升，或者通过维归约，在数据方上进行聚集。图 2.10 图示了在图 2.7 给出的 *location* 维层次向上攀升，在中心数据方执行上卷操作的结果。分层被定义为全序 *street* < *city* < *province_or_state* < *country*。所展示的上卷操作沿 *location* 的分层，由 *city* 层向上到 *country* 层聚集数据。换一句话说，结果数据方按 *country*，而不是按 *city* 对数据分组。

当用维归约进行上卷时，一个或多个维由给定的数据方删除。例如，考虑只包含两维 *location* 和 *time* 的数据方 *sales*。上卷可以删除 *time* 维，导致整个销售按地点，而不是按地点和时间聚集。

下钻：下钻是上卷的逆操作，它由不太详细的数据到更详细的数据。下钻可以通过沿维的概念分层向下或引入新的维来实现。图 2.10 图示了沿着 *day* < *month* < *quarter* < *year* 定义的 *time* 维的概念分层向下，在中心数据方执行下钻操作的结果。这里，下钻由 *time* 维的分层向下，由 *quarter* 层到更详细的 *month* 层。结果数据方详细地列出每月的总销售，而不是按季度求和。

由于下钻操作对给定数据添加更多细节，它也可以通过添加新的维到数据方来实现。例如，可以通过引入一个的维，如 *customer_type*，在图 2.10 中心表的数据方上执行下钻操作。

切片和切块：切片操作在给定的数据方的一个维上进行选择，导致一个子方。图 2.10 图示了一个对维 *time* 的切片操作，它对中心数据方使用条件 *time* = "*Q1*" 选择销售数据。切块操作通过对两个或多个维执行选择，定义子方。图 2.10 图示了一个切块操作，它涉及三个维，根据如下条件对中心表切块：(*location* = "Montreal" or 'Vancouver") and (*time* = "*Q1*" or "*Q2*") and (*item* = "home entertainment" or "computer")。

转轴：转轴（又称**旋转**）是一种目视操作，它转动数据的视角，提供数据的替代表示。图 2.10 给出一个转轴操作，这里 *item* 和 *location* 在一个 2-D 切片上转动。其它例子包括转动 3-D 数据方 ，或将一个 3-D 立方转换成 2-D 平面序列。

其它 OLAP 操作：有些 OLAP 还提供其它操作。例如，drill_across 执行涉及多个事实表的查询；drill_through 操作使用关系 SQL 机制，钻到数据方的底层，到后端关系表。

其它 OLAP 操作可能包括列出表中最高或最低的 *N* 项，以及计算移动平均值、增长率、利润、内部返回率、贬值、流通转换、和统计功能。□

OLAP 提供了分析建模机制，包括推导比率、变差等，以及计算跨越多维度量的计算引擎。它能在每一粒度级和在所有维的交叉产生汇总、聚集、分层。OLAP 也支持预报、趋势分析和统计分析函数模型。在这种意义下，OLAP 引擎是一种强有力的数据分析工具。

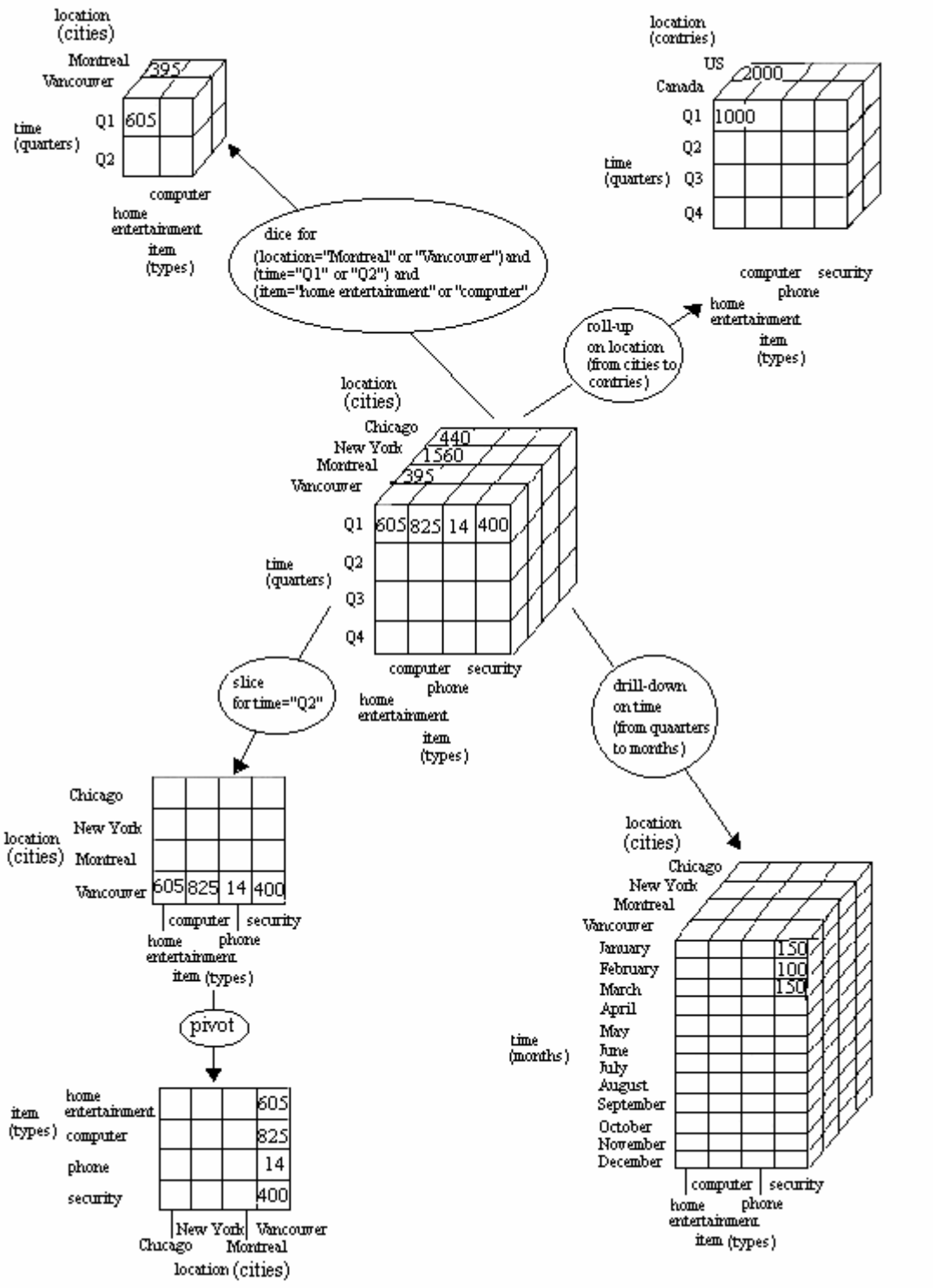


图 2.10　多维数据上 OLAP 操作的典型例子

OLAP 系统与统计数据库

OLAP 的许多特征，如使用多维数据模型和概念分层、与维关联的度量、上卷和下钻概念，也存在于统计数据库（SDB）的早期工作中。统计数据库是一种用于支持统计应用的数据库系统。这两种类型的系统之间的类似性很少讨论，主要是由于它们使用了不同的术语，并有不同的应用领域。

然而，OLAP 和 SDB 也有显著的差别。SDB 趋向于关注社会经济应用，而 OLAP 旨在商务应用。概念分层的保密性问题是 SDB 关注的主要问题。例如，给定汇总的社会经济数据，允许用户观察对应的低层数据是有争议的。最后，不象 SDB，OLAP 需要有效地处理海量数据。

2.2.7 查询多维数据库的星形网查询模型

多维数据库查询可以基于**星形网模型**。星形网模型由从中心点发出的射线组成，其中每一条射线代表一个维概念分层。概念分层上的每个“抽象级”称为一个**脚印**，代表诸如上卷、下钻等 OLAP 操作可用的粒度。

例 2.9 AllElectronics 数据仓库的一个星形网查询模型如图 2.11 所示。该星形网由四条射线组成，分别代表属性 *location, customer, item* 和 *time* 的维层次结构。每条线由一些脚印组成，代表该维的抽象级。例如，*time* 线有四个脚印：“*day*”，“*month*”，“*quarter*”和“*year*”。一个概念分层可以涉及单个属性（象 *time* 分层中的 *date*），或若干属性（例如，概念分层 *location* 涉及属性 *street, city, province_or_state* 和 *country*）。为了考察 AllElectronics 的商品销售，可以沿着 *time* 维上卷，由 *month* 到 *quarter*，或沿着 *location* 维下钻，由 *country* 到 *city*。通过用高层抽象（如 *time* 维的“*year*”）值替换低层抽象（如 *time* 维的“*day*”）值，概念分层可以用于**泛化**数据。通过用低层抽象值替换高层抽象值，概念分层也可以**特化**数据。□

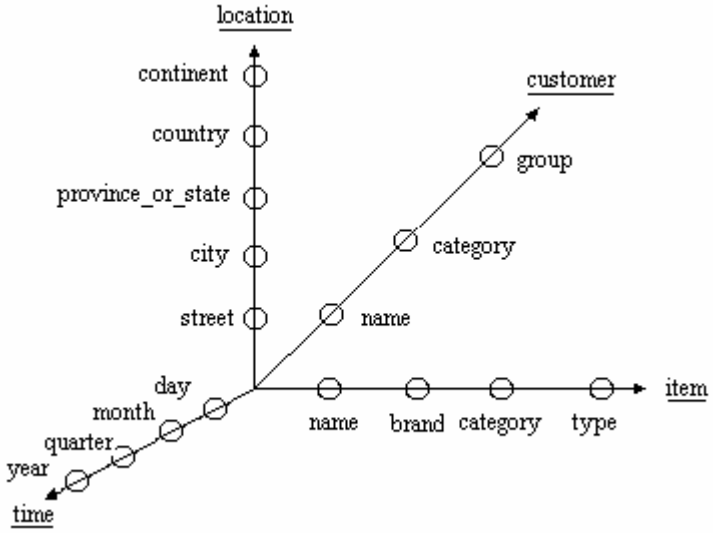


图 2.11：商务查询建模：一个星形网模型

2.3 数据仓库的系统结构

本节，我们讨论数据仓库的结构问题。2.3.1 小节介绍如何设计和构造数据仓库。2.3.2 小节介绍三层数据仓库结构。2.3.3 小节提供用于 OLAP 处理的各种不同类型的仓库服务器。

2.3.1 数据仓库的设计步骤和结构

本小节提供数据仓库设计的一个商务分析框架，同时介绍设计过程所涉及的基本步骤。

数据仓库设计：一个商务分析框架

“数据仓库为商务分析提供了什么？”首先，拥有数据仓库可以提供竞争优势。通过提供相关信息，据此测量性能并作出重要调整，以帮助战胜其它竞争对手。其次，数据仓库可以加强生产能力，因为它能够快速有效地搜集准确描述组织机构的信息。再次，数据仓库促进了与顾客的联系，

因为它跨越所有商务、所有部门、所有市场，提供了顾客和商品的一致视图。最后，通过以一致、可靠的方式长期跟踪趋势、式样、例外，数据仓库可以降低费用。

为建立有效的数据仓库，需要理解和分析商务需求，并构造一个商务分析框架。构造一个大的、复杂的信息系统就象建一个大型、复杂的建筑，业主、设计师、建筑者都有不同的视图。这些观点结合在一起，形成一个复杂的框架，代表自顶向下、商务驱动，或业主的视图，也代表自底向上、建筑者驱动，或信息系统实现者的视图。

关于数据仓库的设计，四种不同的视图必须考虑：自顶向下、数据源、数据仓库、商务查询。

- **自顶向下视图**使得我们可以选择数据仓库所需的相关信息。这些信息能够满足当前和未来商务的需求。
- **数据源视图**揭示被操作数据库系统捕获、存储、和管理的信息。这些信息可能以不同的详细程度和精度建档，存放在由个别数据源表到集成的数据源表中。通常，数据源用传统的数据建模技术，如实体-联系模型或 CASE（计算机辅助软件工程）工具建模。
- **数据仓库视图**包括事实表和维表。它们提供存放在数据仓库内部的信息，包括预先计算的和与计数，以及关于源、日期、原时间等。
- 最后，**商务查询视图**是从最终用户的角度透视数据仓库中的数据。

建立和使用数据仓库是一个复杂的任务，因为它需要商务技巧、技术技巧和程序管理技巧。关于商务技巧，建立数据仓库涉及理解这样一个系统如何存储和管理它的数据；如何构造一个**提取程序**，将数据由操作数据库转换到数据仓库；如何构造一个**仓库刷新软件**，合理地保持数据仓库中的数据相对于操作数据库中数据的当前性。使用数据仓库涉及理解数据的含义，以及理解商务需求并将它转换成数据仓库查询。关于技术技巧，数据分析需要理解如何由定量信息作出估价，以及如何根据数据仓库中的历史信息得到的结论推导事实。这些技巧包括发现模式和趋势，根据历史推断趋势和发现不规则的能力，并根据这种分析提出相应的管理建议。最后，程序管理技巧涉及需要与许多技术人员、经销商、最终用户交往，以便以及时、合算的方式提交结果。

数据仓库的设计过程

“如何设计数据仓库？”数据仓库可以使用自顶向下方法、自底向上方法，或二者结合的混合方法设计。**自顶向下方法**由总体设计和规划开始。当技术成熟并已掌握，对必须解决的商务问题清楚并已很好理解时，这种方法是有用的。**自底向上方法**以实验和原型开始。在商务建模和技术开发的早期阶段，这种方法是有用的。这样可以以相当低的代价前进，在作出重要承诺之前评估技术的利益。在**混合方法**下，一个组织既能利用自顶向下方法的规划的、战略的自然特点，又能保持象自底向上方法一样快速实现和立即应用。

从软件工程的观点，数据仓库的设计和构造包含以下步骤：规划、需求研究、问题分析、仓库设计、数据集成和测试，最后，配置数据仓库。大的软件系统可以用两种方法开发：瀑布式方法和螺旋式方法。**瀑布式方法**在进行下一步之前，每一步都进行结构化和系统的分析，就象瀑布一样，从一级落到下一级。**螺旋式方法**涉及功能渐增的系统的快速产生，相继版本之间的间隔很短。对于数据仓库，特别是对于数据集市的开发，这是一个好的选择，因为其周转时间短，能够快速修改，并且新的设计和技术可以快速接受。

一般地，数据仓库的设计过程包含如下步骤：

1 选取待建模的商务处理，例如，订单、发票、出货、库存、记帐管理、销售、和一般分类帐。如果一个商务过程是有组织的，并涉及多个复杂的对象，应当选用数据仓库模型。然而，如果处理是部门的，并关注某一类商务处理，则应选择数据集市。

2 选取商务处理的粒度。对于处理，该粒度是基本的、在事实表中是数据的原子级。例如，单个事务、一天的快照等。

3 选取用于每个事实表记录的维。典型的维是时间、商品、顾客、供应商、仓库、事务类型和状态。

4 选取将安放在事实表中的度量。典型的度量是可加的数值量，如 *dollars_sold* 和 *units_sold*。

由于数据仓库的构造是一个困难、长期的任务，它的实现范围应当清楚地定义。一个初始的数据仓库的实现目标应当是特定的、可实现、可测量的。这涉及时间和预算的分配，一个组织的哪些子集要建模，选择的数据源数量，提供服务的部门数量和类型。

一旦设计和构造好数据仓库，数据仓库的最初使用包括初始化装入、首次展示规划、培训和定位。平台的升级和管理也要考虑。数据仓库管理包括数据刷新、数据源同步、规划故障恢复、管理

存取控制和安全、管理数据增长、管理数据库性能、以及数据仓库的增强和扩充。范围管理包括控制查询、维、报告的数量和范围，限制数据仓库的大小，或限制进度、预算和资源。

各种数据仓库设计工具都可以使用。**数据仓库开发工具**提供一些操作，定义和编辑元数据库（如模式、脚本或规则），回答查询，输出报告，向或由关系数据库目录传送元数据。**规划与分析工具**研究模式改变的影响，当刷新率或时间窗口改变时对刷新性能的影响。

2.3.2 三层数据仓库结构

“数据仓库的结构是什么样的？”通常，数据仓库采用三层结构，如图 2.12 所示。

1. 底层是数据仓库服务器，它几乎总是一个关系数据库系统。“如何由该层提取数据，创建数据仓库？”使用称作**网间连接程序**的应用程序，由操作数据库和外部数据源（如，由外部咨询者提供的顾客侧面信息）提取数据。网间连接程序由下面的 DBMS 支持，允许客户程序产生 SQL 代码，在服务器上执行。网间连接程序的例子包括 ODBC（开放数据库连接）和微软的 OLE-DB（数据库开放链接和嵌入），JDBC（Java 数据库连接）。
2. 中间层是 **OLAP 服务器**，其典型的实现或者是（1）**关系 OLAP（ROLAP）**模型，即扩充的关系 DBMS，它将多维数据上的操作映射为标准的关系操作；或者是（2）**多维 OLAP（MOLAP）**模型，即特殊的服务器，它直接实现多维数据和操作。OLAP 服务器在 2.3.3 小节讨论。
3. 顶层是**客户**，它包括查询和报告工具、分析工具、和/或数据挖掘工具（例如，趋势分析、预测等）。

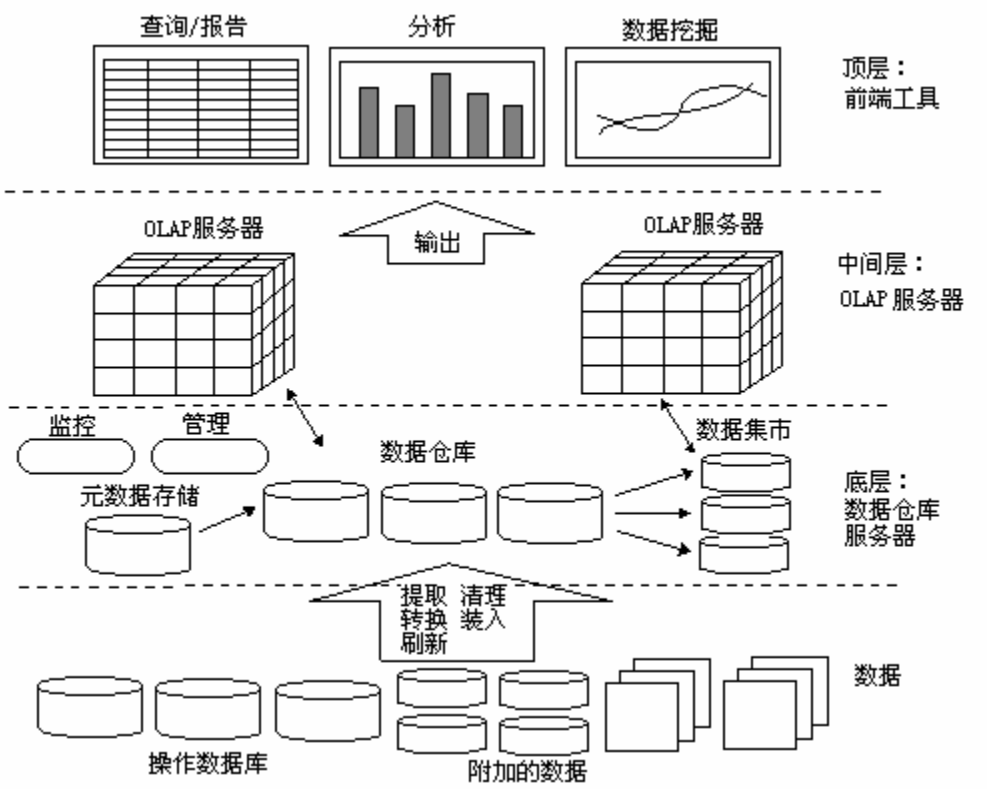


图 2.12：三层数据仓库结构

从结构的角度看，有三种数据仓库模型：企业仓库、数据集市、和虚拟仓库。

企业仓库：企业仓库搜集了关于主题的所有信息，跨越整个组织。它提供企业范围内的数据集成，通常来自一个或多个操作的系统，或外部信息提供者，并且是跨功能的。通常，它包含详细数据和汇总数据，其大小由数千兆字节，到数百千兆字节，数兆兆字节，或更多。企业数据仓库可以在传统的大型机上实现，如 UNIX 超级服务器或并行结构平台。它需要广泛建模，可能需要多年设计和建造。

数据集市：数据集市包含企业范围数据的一个子集，对于特定的用户是有用的。其范围限于选定的主题。例如，一个商场的数据集市可能限定其主题为顾客、商品和销售。包括在数据集市中的数据通常是汇总的。

通常，数据集市可以在低价格的部门服务器上实现，基于 UNIX 或 Windows/NT。实现数据集市的周期一般是数以周计，而不是数以月计或数以年计。然而，如果它们的规划不是企业范围的，从长远讲，可能涉及很复杂的集成。根据数据的来源不同，数据集市分为独立的和依赖的两类。在独立的数据集市中，数据来自一个或多个操作的系统或外部信息提供者，或者来自在一个特定的部门或地域局部产生的数据。依赖的数据集市中的数据直接来自企业数据仓库。

虚拟仓库：虚拟仓库是操作数据库上视图的集合。为了有效地处理查询，只有一些可能的汇总视图被物化。虚拟仓库易于建立，但需要操作数据库服务器具有剩余能力。

自顶向下开发企业仓库是一种系统的解决方法，并能最大限度地减少集成问题。然而，它费用高，需要长时间开发，并且缺乏灵活性，因为整个组织的共同数据模型达到一致是困难的。自底向上设计、开发、配置独立的数据集市方法提供了灵活性、低花费，并能快速回报投资。然而，将分散的数据集市集成，形成一个一致的企业数据仓库时，可能导致问题。

对于开发数据仓库系统，一个推荐的方法是以递增、进化的方式实现数据仓库，如图 2.13 所示。第一，在一个合理短的时间（如，一、两个月）内，定义一个高层次的企业数据模型，在不同的主题和可能的应用之间，提供企业范围的、一致的、集成的数据视图。这个高层模型将大大减少今后的集成问题，尽管在企业数据仓库和部门数据集市的开发中，它还需要进一步提炼。第二，基于上述相同的企业数据模型，可以并行地实现独立的数据集市和企业数据仓库。第三，可以构造分布数据集市，通过网络中心服务器集成不同的数据集市。最后，构造一个**多层数据仓库**，这里，企业仓库是所有仓库数据的唯一管理者，仓库数据分布在一些依赖的数据集市中。

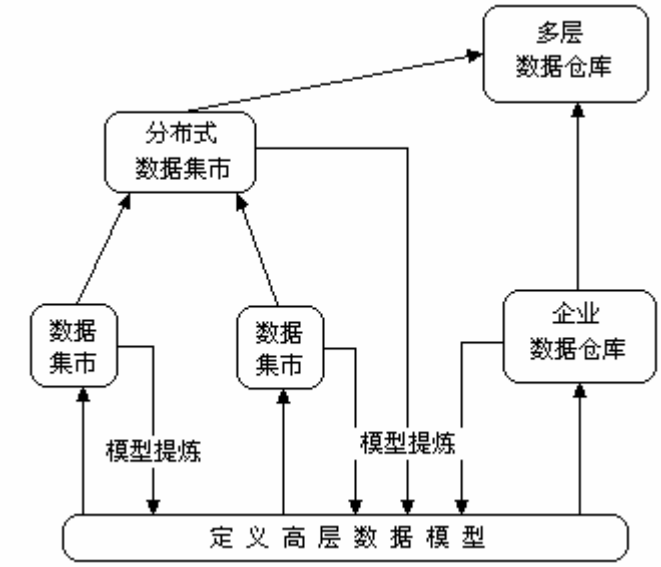


图 2.13　数据仓库开发的推荐方法

2.3.3 OLAP 服务器类型：ROLAP、MOLAP、HOLAP 的比较

“OLAP 服务器的种类有哪些？”逻辑上讲，OLAP 服务器为商务用户提供来自数据仓库或数据集市的多维数据，而不必关心数据如何存放和存放在何处。然而，OLAP 服务器的物理结构和实现必须考虑数据存放问题。OLAP 服务器实现包括：

关系 OLAP（ROLAP）服务器：这是一种中间服务器，介于关系后端服务器和用户前端工具之间。它们使用关系或扩充关系 DBMS 存放并管理数据仓库，而 OLAP 中间件支持其余部分。ROLAP 服务器包括每个 DBMS 后端优化，聚集导航的逻辑实现，附加的工具和服务。看来，ROLAP 技术比 MOLAP 技术具有更大的可规模性。例如，Microstrategy 的 DSS 和 Informix 的 Metacube 都采用

ROLAP 方法[5]。

多维 OLAP（MOLAP）服务器：这些服务器通过基于数组的多维存储引擎，支持数据的多维视图。它们将多维视图直接映射到数据方数组结构。例如，Arbor 的 Essbase 是一个 MOLAP 服务器。使用数据方的优点是能够对预计算的汇总数据快速索引。注意，使用多维数据存储，如果数据集是稀疏的，存储利用率可能很低。在这种情况下，应当使用稀疏矩阵压缩技术（见 2.4 节）。

许多 OLAP 服务器采用两级存储，以便处理稀疏和稠密数据集：稠密子方不变，并作为数组结构存储；而稀疏子方使用压缩技术，从而提高存储利用率。

混合 OLAP（HOLAP）服务器：混合 OLAP 方法结合 ROLAP 和 MOLAP 技术，得益于 ROLAP 较大的可规模性和 MOLAP 的快速计算。例如，HOLAP 服务器允许将大量详细数据存放在关系数据库中，而聚集保持在分离的 MOLAP 存储中。微软的 SQL Server 7.0 OLAP 服务支持混合 OLAP 服务器。

特殊的 SQL 服务器：为了满足在关系数据库中日益增长的 OLAP 处理的需要，一些关系数据库和数据仓库公司（例如 Redbrick）实现了特殊的 SQL 服务器，提供高级查询语言和查询处理，在只读环境下，在星形和雪花模式上支持 SQL 查询。

“那么，数据怎样实际地存放在 ROLAP 和 MOLAP 结构中？”如名称所示，ROLAP 使用关系表存放联机分析处理数据。注意，与基本方体相关联的事实表称为基本事实表。基本事实表存放的数据所处的抽象级由给定的数据方的模式的连接键指出。聚集数据也能存放在事实表中，这种表称为**汇总事实表**。有些汇总事实表既存放基本事实表数据，又存放聚集数据，如例 2.10 所示。也可以对每一抽象级分别使用汇总事实表，只存放聚集数据。

例 2.10 表 2.4 是一个汇总事实表，它既存放基本事实数据，又存放聚集数据。该表的模式是“< *record_identifier(RID), item, ..., day, month, quarter, year, dollars_sold* >”，其中 *day, month, quarter* 和 *year* 定义销售日期。分别考虑 RID 为 1001 和 1002 的元组。它们的数据在基本事实级，销售日期分别是 2000 年 10 月 15 日和 2000 年 10 月 23 日。考虑 RID 为 5001 的元组，它所在的抽象级比 RID 为 1001 和 1002 的元组更一般。这里， *day* 的值被泛化为 all，因此对应的 *time* 值为 2000 年 10 月。这样，显示的 *dollars_sold* 是一个聚集值，代表 2000 年 10 月全月的销售，而不只是 2000 年 10 月 15 日或 2000 年 10 月 23 日的销售。特殊值 all 用于表示汇总数据的子和。□

表 2.4：单个基本和汇总事实表

RID	item	...	day	month	quarter	year	dollars_sold
1001	TV	...	15	10	Q4	2000	250.60
1002	TV	...	23	10	Q4	2000	175.00
...	...	...	...	...	...	...	...
5001	TV	...	all	10	Q4	2000	45,786.08
...	...	...	...	...	...	...	...

MOLAP 使用多维数组结构存放联机分析处理数据。例如，本章介绍的数据方结构就是这种数组结构。

大部分数据仓库系统采用客户-服务器结构。关系数据存储总是驻留在数据仓库/数据集市服务器站点上。多维数据存储可以驻留在数据库服务器站点，或客户站点。

2.4 数据仓库实现

数据仓库包含了海量数据。要求 OLAP 服务器在若干秒内回答决策支持查询。因此，重要的是，数据仓库系统要支持高效的数据方计算技术、存取方法和查询处理技术。“怎么做到这些？”本节，我们考察数据仓库的有效实现方法。

[5] 这些产品的信息分别可以在 *www.microstrategy.com* 和 *www.informix.com* 找到。

2.4.1 数据方的有效计算

多维数据分析的核心是有效地计算多个维集合上的聚集。按 SQL 的术语，这些聚集称为分组。

compute cube 操作及其实现

一种方法是扩充 SQL，使之包含 **compute cube** 操作。**compute cube** 操作在操作指定的维的所有子集上计算聚集。

例 2.11 假定我们想对 AllElectronics 的销售创建一个数据方，包含 *item, city, year* 和 *sales_in_dollars*。你可能想用以下查询分析数据：

- “按 item 和 city 分组，计算销售和。”
- “按 item 分组，计算销售和。”
- “按 city 分组，计算销售和。”

可从该数据方计算的方体或分组的总数是多少？取 *city, item* 和 *year* 三个属性为三个维，*sales_in_dollars* 为度量，可以由该数据方计算的方体总数为 $2^3=8$ 个。可能的分组是{(*city, item, year*), (*city, item*), (*city, year*), (*city*), (*item*), (*year*), () }，其中，()意指按空分组（即，不对任何维分组）。这些分组形成了该数据方的方体格，如图 2.11 所示。基本方体包含所有的维 *city, item* 和 *year*，它可以返回这三维的任意组合。顶点方体，或 0-D 方体表示分组为空的情况，它包含所有销售的总和。□

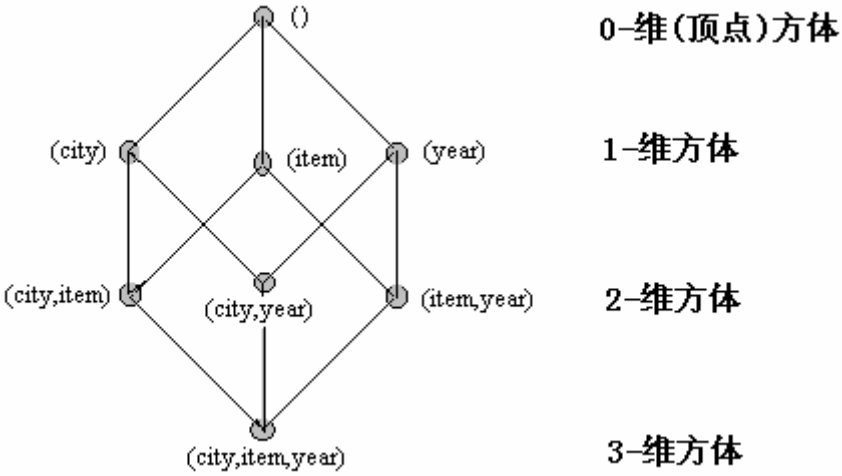


图 2.14：方体格，组成三维数据方，每一个方体代表一个不同的分组；基本方体包含三个维：*city, item* 和 *year*

不包含分组的 SQL 查询，如“计算全部销售的和”，是 0 维操作。包含一个分组的 SQL 查询，如“按 *city* 分组，计算销售和”，是一维操作。在 *n* 维上的一个方操作等价于一组分组语句，每个是 *n* 维的一个子集。因此，方操作是分组操作的 *n* 维泛化。基于 2.2.3 小节介绍的 DMQL 语法，例 2.11 的数据方可以定义为：

define cube sales [item,city,year]: **sum**(sales_in_dollars)

对于 *n* 维方，包括基本方体总共有 2^n 个方体。语句

compute cube sales

显式地告诉系统，计算集合{*item, city, year*}的所有 8 个子集（包括空集合）的销售聚集方体。方体操作首先由 Gray 等提出并研究[GCB+97]。

对于不同的查询，联机分析处理可能需要访问不同的方体。因此，预先计算所有，或者至少一部分方体，看来是个好主意。预计算带来快速的响应时间，并避免一些冗余计算。实际上，如果不是全部，大多数 OLAP 产品都借助于多维聚集的预计算。

然而，如果数据方中所有的方体都预先计算，所需的存储空间可能爆炸，特别是当多个维涉及多个层次时。

“*n* 维数据方有多少个方体？”如果每个维都没有分层，我们上面已看到，*n* 维数据方的方体总数为 2^n。然而，在实践中，许多维确实都有分层。例如，*time* 维通常不只是一层 *year*，而是一个

层次或格，如 *day* < *month* < *quarter* < *year*。对于 *n* 维数据方，可能产生的方体（包括沿着每一维的分层结构攀升，产生的方体）总数是：

$$T = \prod_{i=1}^{n} (L_i + 1)$$

其中，L_i是维 i（除去虚拟的顶层 all，因为泛化到 all 等价于去掉一个维）的层次数。该公式基于这样一个事实：每个维最多只有一个抽象层出现在一个方体中。例如，如果数据方有 10 维，每维有 4 个层次，则可能产生的方体总数将是 $5^{10} \approx 9.8\times10^6$。

现在，你可能已经意识到，预计算并物化由数据方（或由基本方体）可能产生的所有方体是不现实的。如果有很多方体，并且这些方体很大，较合理的选择是部分物化；即，只物化可能产生的方体中的某些。

部分物化：方体的选择计算

给定基本方体，方体的物化有三种选择：（1）不预计算任何“非基本”方体（**不物化**）；（2）预计算所有方体（**全物化**）；（3）在整个可能的方体集中，有选择地物化一个适当的子集（**部分物化**）。第一种选择导致在运行时计算昂贵的多维聚集，这可能很慢。第二种选择可能需要海量存储空间，存放所有预计算的方体。第三种选择在存储空间和响应时间二者提供了很好的折衷。

方体的部分物化应考虑三个因素：（1）确定要物化的方体子集；（2）利用查询处理时物化的方体；（3）在装入和刷新时，有效地更新物化的方体。

物化方体的选择需要考虑工作负荷下的查询，它们的频率，和它们的开销。此外，也要考虑工作负荷的特点，渐进更新的开销和整个存储需求量。选择也必须考虑物理数据库设计的情况，如索引的产生和选择。有些 OLAP 产品采用启发式方法进行方体选择。一种流行的方法是物化这样的方体集，其它经常引用的方体基于它们。

一旦选定的方体被物化，重要的是在查询处理时使用它们。这涉及由大量候选的物化方体中确定相关方体，如何使用物化方体中可用的索引结构，以及如何将 OLAP 操作转换成选定方体上的操作。这些问题将在 2.4.3 小节讨论。

最后，在装入和刷新期间，应当有效地更新物化的方体；应当使用并行机制和渐进更新技术。

数据方计算中多路数组聚集

然而，为了确保快速地联机分析处理，我们可能需要预先计算给定数据方的所有方体。方体可能存放在二级存储器，并在需要时访问它们。因此，重要的是开发一种有效的方法，计算数据方的所有方体。这些方法必须考虑方体计算时可用的主存限制，以及计算所需的时间。为简单起见，我们可以排除沿着每一维的层次结构攀升而产生的方体。

由于关系 OLAP（ROLAP）使用元组和关系表作为它的基本数据结构，而多维 OLAP（MOLAP）使用的基本数据结构是多维数据模型数组，可以预料 ROLAP 和 MOLAP 使用很不相同的方计算技术。

ROALP 方计算使用如下主要优化技术：

- 排序、散列和分组操作用于维属性，以便对相关元组重新排序和分簇。
- 在某些子聚集上分组，作为“部分分组”。这些“部分分组”对于加快其它子聚集的计算是有用的。
- 可以由以前计算的聚集计算新的聚集，而不必由基本事实表计算。

“这些优化技术如何用于 MOLAP？”ROLAP 按值的寻址，通过基于键的寻址搜索访问维值。相比之下，MOALP 使用直接数组寻址，通过位置或对应数组位置的索引访问维值。这样，MOLAP 不能使用 ROLAP 的基于值的重新排序优化技术。因此，应当为 MOLAP 基于数组的方结构开发不同方法，如下所述：

1. 将数组分成块。块（chunk）是一个子方，其大小能够放入方计算时可用的内存。**分块**是一种将 *n* 维数组划分成小的 *n* 维块的方法；其中，每个块作为一个对象存放在磁盘上。块被压缩，以避免空数组单元（即，不含任何有效数据的单元）所导致的空间浪费。对于压缩的稀疏数组结构，可以用“*chunkID + offset*”作为在块内查找单元的寻址机制。这种压缩技术功能强大，足以处理磁盘和内存中的稀疏方。
2. 通过访问方单元（即，访问方单元的值），计算聚集。可以优化访问单元的次序，使得每个单元必须重复访问的次数最小化，从而减少存储访问开销和存储开销。技巧是使用这种定序，使得部分聚集可以同时计算，并避免不必要的单元重新访问。

由于分块技术涉及一些聚集计算的重叠，我们称该技术为数据方计算的**多路数组聚集**。

我们通过一个具体的例子，解释 MOLAP 方结构的这种方法。

例 2.12 考虑一个包含维 A,B,C 的 3-D 数组。该 3-D 数组被划分成小的、基于内存的块。在该例中，数组被划分为 64 块，如图 2.15 所示。维 *A* 组织成 4 个相等部分 a_0，a_1，a_2和 a_3。维 *B* 和 *C* 都类似地组织成 4 部分。块 1,2,...,64 分别对应于子方 $a_0b_0c_0$，$a_1b_0c_0$,...，$a_3b_3c_3$。假定数组的大小对于维 *A*，*B* 和 *C* 分别是 40，400，4000。这样，每个分划，*A*，*B* 和 *C* 的大小分别是 10，100 和 1000。

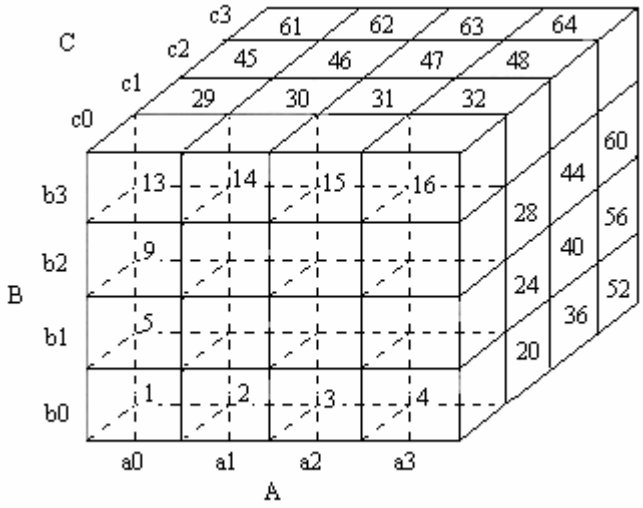


图 2.15：一个 3-D 数组，划分为 64 块

完全物化对应的数据方涉及计算定义该数据方的所有方体。这些方体包括：

- 基本方体，由 *ABC* 定义（其它方体直接或间接地由它计算）。该方体业已计算，并对应于给定的 3-D 数组。
- 2-D 方体，*AB*、*AC* 和 *BC*，分别对应于按 *AB*、*AC* 和 *BC* 分组。这些方体必须计算。
- 1-D 方体，*A*、*B* 和 *C*，分别对应于按 *A*、*B* 和 *C* 分组。这些方体必须计算。
- 0-D（顶点）方体，由 all 定义，对应于按( )分组；即，没有分组。该方体必须计算。

让我们看一看如何用多路数组技术进行这些计算。存在多种可能的次序，将块读入内存，用于计算方体。考虑图 2.15 从 1 到 64 标记的次序。假定我们想计算 *BC* 方体中的 b_0c_0。我们在“块内存”中为该块分配存储空间。通过扫描 *ABC* 中的 1 至 4 块，计算出块 b_0c_0。即，b_0c_0单元在 a_0到 a_3上聚集。块内存可以分给下一个块 b_1c_0，在完成对 *ABC* 的 4 个块，5 到 8 的扫描后，计算出 b_1c_0。如此继续下去，可以计算整个 *BC* 方体。因此，对于 *BC* 中所有块的计算，一次只需一个 *BC* 块在内存。

在 BC 方体的计算中，我们将扫描 64 块中的每一块。“为计算其它方体，如 *AB* 和 *AC*，有没有办法避免重新扫描所有的块？”回答多半是肯定的。这正是多路计算思想的由来。例如，扫描块 1（即 $a_0b_0c_0$）时，（例如，如上所述，为计算 *BC* 中的 2-D 块 b_0c_0），同时计算与 $a_0b_0c_0$有关的所有 2-D 方体。即，扫描 $a_0b_0c_0$时，同时计算三个 2-D 平面 *BC*、*AC* 和 *AB* 上的三个块 b_0c_0，a_0c_0和 a_0b_0上的聚集。换一句话说，当一个 3-D 块在内存时，多路计算向每一个 2-D 平面聚集。

现在，让我们看看，不同的块扫描次序和方体计算对整个数据方的计算效率的影响。注意，维 *A*,*B* 和 *C* 的大小分别为 40,400 和 4000。这样，最大的 2-D 平面是 *BC*（大小为 400×4,000 = 1,600,000）；次大的 2-D 平面是 *AC*（大小为 40×4,000 = 160,000）；*AB* 是最小的 2-D 平面（大小为 40×400 = 16,000）。

假定以所示次序从 1 到 64 扫描块。按这种扫描次序，对于每一次行扫描，可以完全计算最大的 2-D 平面 *BC* 的一块。即，扫描包含块 1 到 4 的行后，b_0c_0完全被聚集；扫描包含块 5 到 8 的行后，b_1c_0完全被聚集；如此等等。相比之下，完全计算次 2-D 大平面 *AC* 上的一块，需要扫描 13 块（给定扫描次序 1 到 64）。例如，扫描块 1，5，9 和 13 后，a_0c_0被完全计算。最后，计算最小的 2-D 平面 *AB* 上的一块需要扫描 49 块。例如，扫描块 1，17，33 和 49 后，a_0b_0被完全聚集。因此，为完成计算，*AB* 需要的扫描块数最多。为了避免将一个 3-D 块多次调入内存，根据 1 到 64 的扫描次序，在块内存中保持所有相关的 2-D 平面所需最小存储为：40×400（用于整个 *AB* 平面）+40×1,000（用于 *AC* 平面的一行）+100×1,000(用于 *BC* 平面的一块) = 16,000+40,000+100,000 = 156,000。

替换地，假定块的扫描次序为 1,17,33,49,5,21,37,53,...。即，假定扫描次序是首先向 *AB* 平面，然后向 *AC* 平面，最后向 *BC* 平面聚集。保持二维平面在块内存的最小内存需求量为：400×4,000（用于整个 *BC* 平面）+40×1,000（用于 *AC* 平面的一行）+10×100(用于 *AB* 平面的一块) = 1,641,000。注意，这是从 1 到 64 扫描次序所需内存的十倍多。

类似地，我们可以算出 1-D 和 0-D 方体多路计算的最小内存需求量。图 2.16 给出最有效的次序和效率最差的次序，都是基于数据方计算的最小内存需求。最有效的块次序是 1 到 64。□

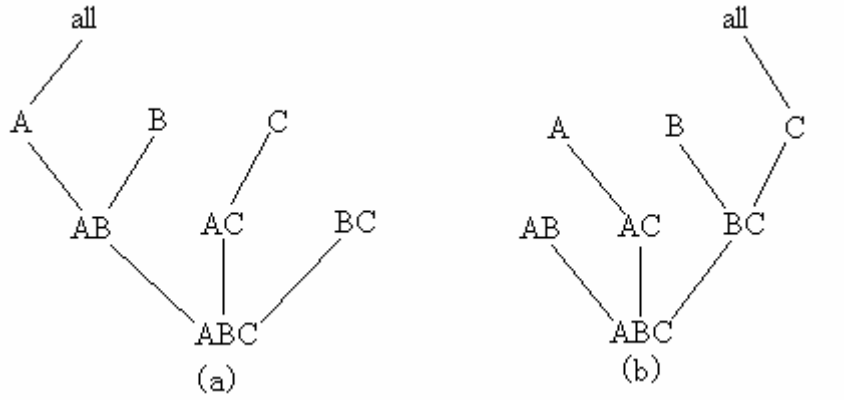


图 2.16　计算例 2.12 三维数据方的多路数组聚集的两种次序：(a)数组聚集最有效的次序(最小内存需求量 156,000 存储单位)；(b) 数组聚集最低效的次序(最小内存需求量 1,641,000 存储单位)

在例 2.12 中，我们假定有足够的内存空间，进行一遍数据方计算（即，由一次扫描所有块计算所有方体）。如果内存空间不够，完成计算将需要更多遍扫描 3-D 数组。然而，在这种情况下，块计算定序的基本原则是一样的。

“ROLAP 和 MOLAP 数据方计算，哪个更快？”借助于适当的稀疏数组压缩技术和仔细的方体计算定序，实验表明 MOLAP 数据方计算比 ROLAP（基于记录的关系）计算快得多。与 ROLAP 不同，MOLAP 的数组结构不需要节省空间存放查找关键字。此外，MOLAP 使用直接数组寻址，这比 ROLAP 基于关键字的寻址快。事实上，对于 ROLAP 数据方计算，不直接由表计算，而把表转换成数组，由数组进行数据方计算，然后将结果转换成表甚至还要快些。然而，这仅对维数相对较少的数据方成立，因为要计算的方体个数随维数指数增长。为了避免维增长灾难，最近的研究提出仅计算**冰山方**（iceberg cube）。这种方仅存放这样的方分划，分划中每个单元的聚集值（如，count）大于某个最小支持度或出现阈值。

“如果我想添加一些新数据到预计算的方中，或由其中存放的数据淘汰一些，怎么办？”对此，已经提出了一些有效的渐增更新方法，允许向预计算的方添加或由预计算的方淘汰数据，而不必重新计算方。这种更新的具体方法作为习题留给读者。

2.4.2 索引 OLAP 数据

为提供有效的数据访问，大部分数据仓库系统支持索引结构和物化视图（使用方体）。选择方体物化的方法前一节已讨论。本小节，我们考察如何使用位图索引和连接索引对 OLAP 数据进行索引。

位图索引方法在 OLAP 产品中很流行，因为它允许在数据方中快速检索。位图索引是 *record_ID*（*RID*）表的一种替代表示。在给定属性的位图索引中，属性域中的每个值 *v*，有一个不同的位向量 *Bv*。如果给定的属性域包含 *n* 个值，则位图索引中每项需要 *n* 位（即，*n* 位向量）。如果数据表中给定行的属性值为 *v*，则在位图索引的对应行，表示该值的位为 1，该行的其它位均为 0。

例 2.13　在 AllElectronics 数据仓库中，假定维 *item* 在顶层有 4 个值(代表商品类型)：“*home entertainment*”, “*computer*”, “*phone*”和“*security*”。每个值（例如，“*computer*”）用的位图索引表的一个位向量表示。假定数据方存放在一个具有 100,000 行的关系表中。由于 *item* 的域有 4 个值，位图索引需要 4 个位向量（或表），每个 100,000 位。图 2.17 给出了一个包含维 *item* 和 *city* 的基本（数据）表和它映射到每维的位图索引。□

基本表　　*Item* 位图索引表　　*City* 位图索引表

RID	*item*	*city*
R1	h	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

注：H 代表“*home entertainment*”，C 代表“*computer*”，P 代表“*phone*”，S 代表“*secrity*”，V 代表“Vancouver”，T 代表“Toronto”。

图 2.17　使用位图索引索引 OLAP 数据

与散列和树索引相比，位图索引有优势。对于基数较小的域它特别有用，因为比较、连接、和聚集操作都变成了位算术运算，大大减少了运行时间。由于字符串可以用单个位表示，位图索引大大降低了空间和 I/O 开销。对于基数较高的域，使用压缩技术，这种方法可以接受。**连接索引**方法的流行源于关系数据库查询处理。传统的索引将给定列上的值映射到具有该值的行表上。与之相反，连接索引登记来自两个关系数据库的可连接行。例如，如果两个关系 $R(RID, A)$ 和 $S(B, SID)$ 在属性 A 和 B 上连接，则连接索引记录包含 (RID, SID) 对，其中 RID 和 SID 分别为来自 R 和 S 的记录标识符。因此，连接索引记录识别可连接的元组，而不费事地执行连接操作。对于维护来自可连接的关系的外部关键字[6]和与之匹配的主关键字的联系，连接索引特别有用。

数据仓库的星形模式模型使得连接索引特别吸引人，因为事实表和它对应维表的连接属性是事实表的外关键字和维表的主关键字。连接索引维护域的属性值（例如，在一个维表内）和事实表的对应行的联系。连接索引可以跨越多维，形成**复合连接索引**。我们可以使用连接索引识别感兴趣的子方。

例 2.14　在例 2.4 中，我们定义了 AllElectronics 的一个星形模式，形如“*sales_star*[*time, item, branch, location*]: *dollars_sold = sum(sales_in_dollars)*”。*sales* 事实表与 *location* 和 *item* 维表之间联系的连接索引联系如图 2.18 所示。例如，*location* 维表的值”*Main Street*”与事实表中的元组 T57, T238 和 T884 连接。类似地，*item* 维表的值”*Sony-TV*”与事实表的元组 T57 和 T459 连接。对应的连接索引表如图 2.19 所示。

假定在数据方中有 360 个时间值，100 个商品，50 个分店，30 个地点，100 万个销售元组。如果事实表中只记录了 30 种商品，其余的 70 中商品显然不参与连接。如果不使用连接索引，必须执行额外的 I/O，将事实表的连接部分和维表一起读入。□

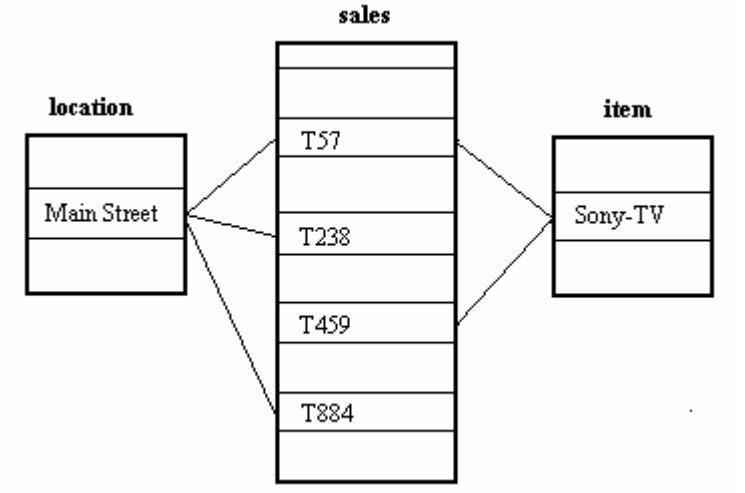


图 2.18　*sales* 事实表与 *location* 和 *item* 维表之间的链接

[6] 在一个关系模式中形成主关键字的属性集是另一个关系模式的外关键字。

location/sales
连接索引表

location	sales_key
...	...
Main Street	T57
Main Street	T238
Main Street	T884
...	

item/sales
连接索引表

item	sales_key
...	...
Sony-TV	T57
Sony-Tv	T459
...	...

location/item/sales
链接两个维的连接索引表

location	item	sales_key
...	...	...
Main Street	Sony-TV	T57
...	...	...

图 2.19　基于图 2.18 的 *sales* 事实表与 *location* 和 *item* 的维表之间的链接的连接索引表

为进一步加快查询处理，我们可以将连接索引与位图索引集成，形成**位图连接索引**。Microsoft SQL server 和 Sybase IQ 支持位图索引，Oracle 8 使用位图和连接索引。

2.4.3 OLAP 查询的有效处理

物化方体和构造 OLAP 索引结构，目的是加快数据方中的查询处理。给定物化的视图，查询处理应按如下步骤进行：

1. **确定哪些操作应当在可利用的方体上执行**：这涉及将查询中的选择、投影、上卷（分组）和下钻操作转换成对应的 SQL 和/或 OLAP 操作。例如，数据方上的切片和切块可能对应于物化方体上的选择和/或投影操作。
2. **确定相关操作应当使用哪些物化的方体**：这涉及找出可能用于回答查询的所有物化方体，使用方体之间的“统治”联系知识，剪去上集合，估计使用剩余物化方体的代价，并选择代价最低的方体。

例 2.15　假定我们为 AllElectronics 定义了一个数据方，形式为“*sales*[*time, item, location*]: sum(*sales_in_dollars*)”。所用的维层次，对于 *time* 是 *day* < *month* < *quarter* < *year*，对于 *item* 是 *item_name* < *brand* < *type*，而对于 *location* 是 *street* < *city* < *province_or_state* < *country*。

假定所处理的查询在{*brand, province_or_city*}上，选择常量为“*year* = 2000”。还假定有四个物化的方体可用，它们是

- 方体 1：{*item_name, city, year*}
- 方体 2：{*brand, country, year*}
- 方体 3：{*brand, province_or_state, year*}
- 方体 4：{*item_name, province_or_state*} ，其中 *year* = 2000

“以上四个方体，选择哪一个处理查询？”较细粒度的数据不能由较粗粒度的数据产生。这样，不能使用方体 2，因为 *country* 是比 *province_or_state* 更一般的概念。可以用方体 1，3 和 4 处理查询，因为（1）它们与查询具有相同的维集合，或是其超集；（2）查询中的选择可以蕴涵在方体的选择中；（3）与 *brand* 和 *province_or_state* 相比，这些方体中的 *item* 和 *location* 的抽象层在更细的层次。

“如果用于处理查询，如何比较每个方体的代价？”看来，使用方体 1 代价最高，因为 *item_name* 和 *city* 分别都在比查询中给出的 *brand* 和 *province_or_state* 更低的概念层。如果没有许多 *year* 值与 *item* 相关联，而对于每个 *brand* 值有许多 *item_name* 值，则方体 3 将比方体 4 小一些，因此应当选择方体 3 来处理查询。然而，如果方体 4 有有效的索引可用，方体 4 可能是较好的选择。因此，需要一些基于代价的估计，以确定应当使用哪个方体集处理查询。□

由于 MOLAP 服务器的存储模型是多维数组，前端的多维查询直接映射到提供直接寻址能力的服务器存储结构。数据方的简洁数组表示具有很好的索引性质，但当数据稀疏时，存储利用率很差。为了有效地存储和处理，应当采用稀疏矩阵和数据压缩技术（2.4.1 节）。

稠密和稀疏数组所用的存储结构可能不同，MOLAP 查询处理最好采用二级方法：对于稠密数组使用数组结构，对于稀疏数组使用稀疏矩阵结构。二维稠密数组可以用 B+树索引。

为处理 MOLAP 查询，首先确定一、二维稠密数组，然后对这些数组使用传统的索引结构建立索引。两级方法增加了存储的利用率，而不牺牲直接寻址能力。

2.4.4 元数据存储

“什么是元数据？”**元数据**是关于数据的数据。在数据仓库中，元数据是定义仓库对象的数据。对于给定数据仓库的数据名和定义，创建元数据。其它元数据包括对提取数据添加的时间标签、提取数据的源、被数据清理或集成处理添加的字段等。

元数据的存储应当包括：

- 数据仓库结构的描述，包括仓库模式、视图、维、层次结构、导出数据的定义，以及数据集市的位置和内容。
- 操作元数据，包括数据血统（移植数据的历史和用于它的转换序列），数据流通（主动的、档案的、或净化的），和管理信息（仓库使用统计、错误报告、审计跟踪）。
- 汇总用的算法，包括度量和维定义算法，数据所处粒度、分割、主题领域、聚集、汇总、预定义的查询与报告。
- 由操作环境到数据仓库的映射，包括源数据库和它们的内容、网关描述、数据分割、数据提取、清理、转换规则和缺省、数据刷新和剪裁规则、安全（用户授权 和存取控制）。
- 关于系统性能的数据，除刷新、更新定时和调度的规则与更新周期外，还包括索引和改善数据存取和提取性能的方法。
- 商务元数据，包括商务术语和定义、数据拥有者信息和收费策略。

数据仓库包含不同级别的综合，元数据是其中一种类型。其它类型包括当前的细节数据（几乎总是在磁盘上），老的细节数据（通常在三级存储器上），稍加综合的数据，和高度综合的数据（可以，也可以不物理地入仓）。

与数据仓库中的其它数据相比，元数据扮演很不相同的角色，并且由于种种原因，也是重要的角色。例如，元数据用作目录，帮助决策支持系统分析者对数据仓库的内容定位；当数据由操作环境到数据仓库环境转换时，作为数据映射指南；对于汇总的算法，它也是指南。汇总算法将当前细节数据汇总成稍加综合的数据，或将稍加综合的数据汇总成高度综合的数据。元数据应当持久存放（即，存放在磁盘上）。

2.4.5 数据仓库后端工具和实用程序

数据仓库系统使用后端工具和实用程序来加载和刷新它的数据。这些工具和机制包含以下功能：

- **数据提取**：通常，由多个、异种、外部数据源收集数据。
- **数据清理**：检测数据中的错误，可能时订正它们。
- **数据变换**：将数据由遗产或宿主格式转换成数据仓库格式。
- **装入**：排序、综合、加固、计算视图、检查整体性，并建立索引和划分。
- **刷新**：传播由数据源到数据仓库的更新。

除清理、装入、刷新和元数据定义工具外，数据仓库系统通常提供一组数据仓库维护工具。

数据清理和数据变换是提高数据质量，从而提高数据挖掘结果质量的重要步骤。将在第 3 章介绍。由于我们主要的兴趣在于数据仓库技术与数据挖掘之间的联系，我们将不深入讨论这些工具的细节，建议有兴趣的读者查阅数据仓库技术的书籍。

2.5 数据方技术的进一步发展

本节，你将学习数据方技术的进一步发展。2.5.1 小节介绍数据方发现驱动的探查；这里，数据异常将自动被检测出来，并以可视化的方式标识给用户。2.5.2 小节介绍多特征方，用于涉及多粒度上多个依赖聚集的复杂数据挖掘查询。其它进展在 2.5.3 小节介绍。

2.5.1 数据方发现驱动的探查

正如我们在本章看到的，可以将数据汇总并存放在 OLAP 系统的数据方中。用户或分析者可以通过使用一些诸如下钻、上卷、切片、切块等 OLAP 操作，检索方中感兴趣的模式。尽管这些工具可以用于帮助用户探查数据，但这一过程不是自动的。用户根据他的直观和假定，试图去识别数据中的例外和异常。这种**假定驱动的探查**有很多缺点。搜索空间可能非常大，使得人工检查地数据成为令人畏惧的任务。高层次的聚集可能指不出低层次的异常，这就容易忽略有趣的模式。即使查看方的一个子集（如一个切片）用户也常常面对大量需要考察的数据值。如果使用假定驱动的探查，堆积如山的数据值使得用户容易错过数据中的例外。

发现驱动的探查是一种替代的方法。这里，预计算的度量指出数据例外，在所有的聚集级指导用户的数据分析过程。以下我们称这种度量为例外指示符。直观地，**例外**是一个数据方单元值，基于某种统计模型，它显著地不同于预期值。该模型考虑单元所属的所有维上度量值的变化。例如，如果商品销售数据分析揭示，与其它所有月份相比，12 月份的销售增长了，这对时间维看来是一个例外。然而，如果考虑商品维，这不是一个例外，因为在 12 月份，其它商品的销售也有类似的增长。该模型考虑隐藏在数据方的所有分组聚集中的例外。可视方（如，使用背景色）反映每个单元的例外程度（基于预先计算的例外指示符）。如 2.4.1 小节讨论的，对于方结构，业已提出了一些有效的算法。例外指示符的计算可以与方构造重迭，使得数据方的总体结构更有效，更利于发现驱动的探查。

有三种度量用作例外指示符，帮助标识数据异常。这些度量指出单元中的量相对于期望值的奇异程度。对于所有的聚集层，计算这些度量，并将它们关联到每一个单元。它们是：

1. SelfExp: 指示相对于同一聚集层的其它单元的奇异程度。
2. InExp: 指示该单元之下某处的奇异程度，如果我们由它下钻的话。
3. PathExp: 指示由该单元的每条下钻路径的奇异程度。

这些度量的用法在下面的例子中解释。

例 2.16 假定我们想分析 AllElectronics 的月销售，按百分比与上月比较。所涉及的维是 *item*, *time* 和 *region*。开始，你研究每个月在所有商品在所有地区的聚集数据，如图 2.20 所示。

sum of sales	month											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Total		1%	-1%	0%	1%	3%	-1%	-9%	-1%	2%	-4%	3%

图 2.20 销售随时间变化

为观察例外指示符，你在屏幕上单击标记高亮度的例外按钮。这将 SelfExp 和 InExp 的值转换成可视提示，显示于每个单元。每个单元的背景色基于它的 SelfExp 值。此外，一个方框画在单元的周围，方框的粗细和颜色是其 InExp 值的函数。粗的框指示高 InExp 值。在两种情况下，颜色越深，例外的程度越高。例如，七、八、九月销售的黑粗框告诉用户通过下钻，探查这些单元的低层聚集。

下钻可以沿着 *item* 或 *region* 维进行。“哪一条路经更例外？”为找出它，你选择一个感兴趣的单元，并触发一个路经例外模块。该模块根据单元的 PathExp 值，为每个维上色。该值反映路经的奇异程度。假定沿着 *item* 的路经包含更多例外。

Avg. sales	month											
item	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Sony b/w printer		9%	-8%	2%	-5%	14%	-4%	0%	41%	-13%	-15%	-11%

Sony color printer		0%	0%	3%	2%	4%	-10%	-13%	0%	4%	-6%	4%
HP b/w printer		-2%	1%	2%	3%	8%	0%	-12%	-9%	3%	-3%	6%
HP color printer		0%	0%	-2%	1%	0%	-1%	-7%	-2%	1%	-4%	1%
IBM desktop computer		1%	-2%	-1%	-1%	3%	3%	-10%	4%	1%	-4%	-1%
IBM laptop computer		0%	0%	-1%	3%	4%	2%	-10%	-2%	0%	-9%	3%
Toshiba desktop computer		-2%	-5%	1%	1%	-1%	1%	5%	-3%	-5%	-1%	-1%
Toshiba laptop computer		1%	0%	3%	0%	-2%	-2%	-5%	3%	2%	-1%	0%
Logitech mouse		3%	-2%	-1%	0%	4%	6%	-11%	2%	1%	-4%	0%
Ergo-way mouse		0%	0%	2%	3%	1%	-2%	-2%	-5%	0%	-5%	8%

图 2.21　商品-时间组合的销售变化

沿着 *item* 下钻导致图 2.21 所示方切片，给出每种商品各时间段的销售。此时，提供了许多不同的销售值，供你分析。通过单击高亮度例外按钮，显示可视提示，将注意力引向例外。考虑“Sony b/w printer”九月份 41%的销售差。该单元具有深色背景，指示一个高 SelfExp 值，意味该单元是一个例外。现在考虑“Sony b/w printer”的十一月份-15%的销售差和十二月份-11%的销售差。十二月份-11%值被标记为一个例外，而-15%没有，尽管-15%比-11%的偏差更大。这是因为例外指示符考虑一个单元所在的所有维。注意，十二月份大部分其它商品的销售具有一个大的正值，而十一月份不是。这样，通过考虑单元在方中的位置，“Sony b/w printer”十二月份的销售差是一个例外，而该商品十一月份的销售差不是。

InExp 值可以用来指示在当前层不可见的、较低层上的例外。考虑七月和九月“IBM desktop computer”所在的单元，两个周围都有黑粗框，指明它们具有高 InExp 值。你可能决定沿 *region* 下钻，进一步探查“IBM desktop computer”的销售。按地区的销售差如图 2.22 所示，其中高亮度例外选项被调用。所显示的可视化提示使得我们立即注意到“IBM desktop computer”销售在南部地区的例外。那里，七月和九月份的销售分别下降 39%和 34%。在我们观察图 2.21 按商品-时间分组、按地区聚集的数据时，这些细节上的例外远非显而易见的。因此，对于搜索数据方的较低层次上的例外，InExp 值是有用的。由于图 2.22 中没有其它具有高 InExp 值的单元，你可能上卷，回到图 2.21 的数据，并选择另一个单元，由它下钻。按照这种方法，可以使用异常指示符，指导数据中有趣异常的发现。□

Avg. sales	Month											
Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
North		-1%	-3%	-1%	0%	3%	4%	-7%	1%	0%	-3%	-3%
South		-1%	1%	-9%	6%	-1%	-39%	9%	-34%	4%	1%	7%
East		-1%	-2%	2%	-3%	1%	18%	-2%	11%	-3%	-2%	-1%
West		4%	0%	-1%	-3%	5%	19%	-18%	8%	5%	-8%	1%

图 2.22　每个地区“IBM desktop computer”销售变化

“如何计算异常值？”SelfExp, InExp 和 PathExp 度量是基于表分析的统计方法。它考虑给定单元值涉及的所有分组（聚集）。一个单元值是否例外要根据它与它的期望值相差多少判定；这里，期望值按照下面介绍的统计模型确定。单元的值和它的期望值之间的差称为**余量**。直观地，余量越大，单元的值越例外。为比较余量值，需要按照与余量相关的期望标准差对值定标。这样，一个单元被视为例外，如果它的定标余量值超过一个预定的阈值。SelfExp, InExp 和 PathExp 度量就是基于这种定标余量。

一个给定单元的期望值是该给定单元高层分组的函数。例如，给定一个具有三个维 A、B 和 C 的方，在 A 的第 i 个位置、B 的第 j 个位置、C 的第 k 个位置的单元的期望值是 γ，γ_i^A，γ_j^B，γ_k^C，

γ_{ij}^{AB}，γ_{ik}^{AC}和γ_{jk}^{BC}的函数。其中，这些γ是所用统计模型的系数。系数反映了在较多细节层上值的差异，是基于观察高层聚集的一般印象。这样，一个单元的例外性建立在它下面的值的例外程度之上。因此，当看到例外时，用户自然通过下钻进一步探查例外。

“对于发现驱动的探查，如何有效地构造数据方？”该计算由三遍组成。第一遍涉及定义数据方的诸如 sum 与 count 等聚集值的计算。在这些聚集值上，将发现例外。有一些计算数据方的有效技术，如 2.4.1 小节讨论的多路数组聚集技术。第二遍包括模型符合。此步要确定上面提到的系数，并用于计算标准余量。这一遍可以与第一遍重迭，因为所涉及的计算是类似的。第三遍基于标准余量，计算 SelfExp, InExp 和 PathExp 的值。这一遍计算也与第一遍类似。因此，对于发现驱动的探查，数据方的计算可以有效地进行。

2.5.2 多粒度上的复杂聚集：多特征方

数据方利于回答数据挖掘查询，因为它们允许在多粒度层聚集数据。本小节，你将学习多特征方。多特征方计算复杂查询，这些查询涉及多粒度上多个依赖的聚集。在实践中，这些数据方非常有用，许多复杂的数据挖掘查询都可以用多特征方回答。与标准数据方上回答简单查询的方计算相比，并不明显增加计算费用。

本小节的所有例子都取自于 AllElectronics 的买卖数据。其中，*item* 是一个交易日(*year, month, day*)在一个销售地区(*region*)交易的商品；给定商品的货架寿命是商品存放在货架（*shelf*)上的月数；商品价格和销售额分别存放在 *price* 和 *sales*。为研究多特征方，我们首先看一个简单数据方的例子。

例 2.17 查询 1：简单数据方查询。找出 2000 年销售总和，按 *item, region* 和 *month* 划分，对每维求子和。

为回答查询 1，构造一个数据方，它在以下 8 个不同的粒度层上聚集销售和：{(*item, region, month*), (*item, region*), (*item, month*), (*month, region*), (*item*), (*region*), (*month*), ( )}；其中，( )代表 all。有多种技术可以有效地计算这种数据方（2.4.1 小节）。□

查询 1 使用一种我们本章研究过的数据方。我们称这种数据方为简单数据方，因为它不涉及任何依赖聚集。

“依赖聚集的含义是什么？”我们通过研究下面的例子，回答这一问题。

例 2.18 查询 2：一个复杂查询。按{*item, region, month*}的所有子集分组，对每组找出 2000 年最高价格，并在具有最高价格的元组中找出总销售额。

使用标准的 SQL，这种查询说明可能很长、很复杂，并且难以优化和维护。查询 2 可以用扩充的 SQL 精确地表示如下：

```
select          item,region,month,MAX(price),SUM(R.sales)
from            Purchases
where           year=2000
cube by         item,region,month: R
such that       R.price=MAX(price)
```

首先选择代表 2000 年交易的元组。**cube by** 子句对属性 *item, region* 和 *month* 的所有可能的组合计算聚集（或分组），它是 **group by** 子句的 *n* 维泛化。在 **cube by** 子句中说明的属性是**分组属性**。在所有分组属性上具有相同值的元组形成一个分组。设分组为 $g_1, g_2, \ldots, g_r$。对每个元组分组 g_i，计算该分组元组中的最高价格 $\max_{g_i}$。变量 *R* 是分组变量，遍取分组中价格等于 $\max_{g_i}$ 的所有元组（如子句 **such that** 所指明）。*R* 遍取 g_i 中的元组，计算销售和，并与 g_i 的分组属性值一起返回。结果方是一个**多特征方**，它支持复杂数据挖掘查询；对于它，多依赖的聚集在不同粒度计算。例如，查询 2 返回的销售和是依赖于每个分组的最高价格的集合。□

让我们看另一个例子。

例 2.19 查询 3：一个更复杂的查询。按{*item, region, month*}的所有子集分组，对每组找出 2000 年最高价格。在最高价格的元组中，找出最小和最大的商品货架寿命。还要在所有最高价格的元组中，找出具有最小货架寿命的元组的总销售额部分。

图 2.23 所示的多特征方图帮助解释查询中的聚集依赖。对于每个分组变量，图中有一个结点，另外添加一个结点 R0。由结点 R0 开始，先计算 2000 年最高价格的元组（结点 R1）。该图指出分

组变量 R2 和 R3“依赖于”R1，因为由 R1 到 R2 和 R3 分别有一条有向边。在多维特征方图中，由分组变量 R_i到 R_j的有向边意指 R_j总是遍取 R_i遍取的元组的一个子集。使用扩充的 SQL 表示时，我们用“R_j **in** R_i”作为这种情况的缩记。例如，R2 上最小货架寿命的元组遍取 R1 上最高价格的元组；即，R2 **in** R1。类似地，R3 上最大货架寿命的元组遍取 R1 上最高价格的元组；即，R3 **in** R1。

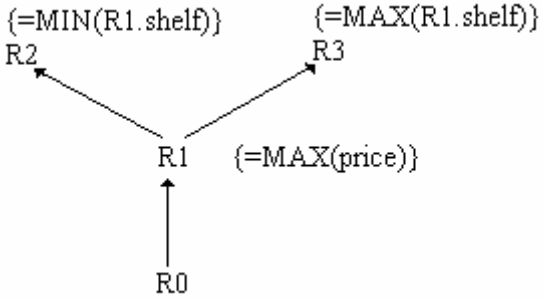


图 2.23　查询 3 的多特征方图

由该图，我们用扩充的 SQL 表示查询 3 如下：

```
select      item,region,month,MAX(price),MIN(R1.shelf),MAX(R1.shelf),
              SUM(R1.sales),SUM(R2.sales),SUM(R3.sales)
from        Purchases
where       year=2000
cube by     item,region,month:R1,R2,R3
such that R1.price=MAX(price) and
              R2 in R1 and R2.shelf=MIN(R1.shelf) and
              R3 in R1 and R3.shelf=MAX(R1.shelf)
```

□

“如何有效地计算多特征方？”多特征方的计算依赖于方中所用聚集函数的类型。在 2.2.3 小节，我们看到聚集函数分为分布的(如 count(), sum(), min()和 max())、代数的(如 avg(), min_N()和 max_N()) 或整体的（如 median(), mode()和 rank()）。多特征方可以同样分类。

多特征方的类型决定它所用的计算方法。有许多有效地计算数据方的方法。这些算法的基本策略是利用定义方多粒度的格结构，高层次粒度的聚集由低层次粒度的聚集来计算。这种方法适合于分布式多特征方。直观地，查询 2 是分布的多特征方，因为我们可以仅使用较低层粒度的方输出渐增地产生较高层粒度的方输出。较高层次粒度的 MAX(*price*)计算可以用较低层次粒度组的所有 MAX(*price*) 的最大值来计算。类似地，可以通过对低层分组的 SUM(*sales*)求和来计算高层分组的 SUM(*sales*)。有些有效方构造算法采用优化技术，基于估计数据方内分组回答的大小进行优化。由于多特征方内每一组的输出量是常数，相同的技术可以用于估计中间结果的大小。因此，简单数据方计算的有效算法可以用于计算复杂查询的分布式多特征方，而不增加 I/O 复杂性。如果多特征方的聚集函数比简单的 SUM()等复杂，可能 CPU 开销略有增加，但可忽略。代数的多特征方必须先转换成分布式多特征方，以便使用这些算法。整体的多特征方的计算，有时费用明显地比分布式高，尽管 CPU 开销一般是可接受的。这样，多特征方用于回答复杂查询，与简单数据方查询相比，费用只增加了一点点。

2.5.3 其它进展

“有无快速回答查询的策略？”快速回答查询的策略集中于为用户提供中间反馈。例如，在**联机聚集**中，数据挖掘系统可以显示“迄今为止所知道的”，而不是等待查询完全处理完。这种对数据挖掘查询的近似回答随着计算的进行周期性刷新。置信间隔与每个估计相关联，为用户提供关于回答可靠性的附加反馈。这促进与系统交互——用户可以洞察他是否沿着“正确的”方向探查，而不必等到查询结束。尽管联机聚集并不改进回答查询的总时间，但由于增加了与系统交互，整个数据挖掘过程应当快一些。

另一种方法是使用**最高 N 查询**。假定你感兴趣的是数以百万计的商品中销售最好的商品。与其等待得到所有商品的列表，按销售额递减序排序，你可能只希望看到最高的 N 项。使用统计，可以

优化查询处理，返回最高的 N 项，而不是整个排序的表。这导致较快的响应时间，有助于用户交互性和减少资源浪费。

2.6 由数据仓库到数据挖掘

“数据仓库和 OLAP 如何与数据挖掘联系？”本节，我们研究用于信息处理、分析处理和数据挖掘的数据挖掘技术。我们还将介绍联机分析挖掘（OLAM）。OLAM 将 OLAP 与数据挖掘集成在一起。

2.6.1 数据仓库的使用

数据仓库和数据集市已在广泛的应用领域使用。几乎每个行业的商务管理人员都使用收集、集成、预处理和存储在数据仓库与数据集市中的数据，进行数据分析和决策。在许多公司，数据仓库用作企业管理的计划-执行-评估“闭环”反馈系统的一部分。数据仓库广泛用在银行、金融服务、消费物品和零售分配部门，以及诸如基于需求的产品生产。

通常，数据仓库使用时间越长，它进化得越好。该进化进行多遍。开始，数据仓库主要用于产生报告和回答预先定义的查询。渐渐地，它用于分析汇总的和细节的数据，结果以报告和图表形式提供。稍后，数据仓库用于决策，进行多维分析和复杂的切片和切块操作。最后，数据仓库可能用于知识发现，并使用数据挖掘工具进行决策。在这种意义下，数据仓库工具可以分为存取与检索工具，数据库报表工具，数据分析工具和数据挖掘工具。

商业用户需要一种手段，知道数据仓库里有什么（通过元数据），如何访问数据仓库的内容，如何使用数据分析工具分析这些内容和如何提供分析结果。

有三种数据仓库应用：信息处理、分析处理和数据挖掘。

- **信息处理**支持查询和基本的统计分析，并使用交叉表、表、图表或图进行报告。数据仓库信息处理的当前趋势是构造低代价的基于网络的存取工具，然后与网络浏览器集成在一起。
- **分析处理**支持基本的 OLAP 操作，包括切片与切块、下钻、上卷和转轴。一般地，它在汇总的和细节的历史数据上操作。与信息处理相比，联机分析处理的主要优势是它支持数据仓库的多维数据分析。
- **数据挖掘**支持知识发现，包括找出隐藏的模式和关联，构造分析模型，进行分类和预测，并用可视化工具提供挖掘结果。

“数据挖掘与信息处理和联机数据分析的关系是什么？”信息处理基于查询，可以发现有用的信息。然而，这种查询的回答反映直接存放在数据库中的信息，或通过聚集函数可计算的信息；它们不反映复杂的模式，或隐藏在数据库中的规律。因此，信息处理不是数据挖掘。

联机分析处理向数据挖掘走近了一步，因为它可以由用户选定的数据仓库子集，在多粒度上导出汇总的信息。这种描述等价于第 1 章的类/概念描述。由于数据挖掘系统也能挖掘更一般的类/概念描述，这就有一个有趣的问题：“OLAP 进行数据挖掘吗？OLAP 系统实际就是数据挖掘系统吗？”

OLAP 和数据挖掘的功能可以视为不交的：OLAP 是数据汇总/聚集工具，它帮助简化数据分析；而数据挖掘自动地发现隐藏在大量数据中的隐含模式和有趣知识。OLAP 工具的目标是简化和支持交互数据分析；而数据挖掘的目标是尽可能自动处理，尽管允许用户指导这一过程。在这种意义下，数据挖掘比传统的联机分析处理前进了一步。

另一种更广泛的观点可能被接受：数据挖掘包含数据描述和数据建模。由于 OLAP 系统可以提供数据仓库中数据的一般描述，OLAP 的功能基本上是用户指挥的汇总和比较（通过上、下钻，旋转，切片，切块和其它操作）。这些尽管有限，但都是数据挖掘功能。同样根据这种观点，数据挖掘的涵盖面要比简单的 OLAP 操作宽得多，因为它不仅执行数据汇总和比较，而且执行关联、分类、预测 、聚类、时间序列分析和其它数据分析任务。

数据挖掘不限于分析数据仓库中的数据。它可以分析现存的、比数据仓库提供的汇总数据粒度更细的数据。它也可以分析事务的、文本的、空间的和多媒体数据，这些数据很难用现有的多维数据库技术建模。在这种意义下，数据挖掘涵盖的数据挖掘功能和处理的数据复杂性要比 OLAP 大得多。

由于数据挖掘涉及的分析比 OLAP 更自动化、更深入，数据挖掘应有更广的应用范围。数据挖掘可以帮助经理找到更合适的客户，也能获得对商务的洞察，帮助提高市场份额和增加利润。此外，数据挖掘能够帮助经理了解顾客的群体特点，并据此制定价格策略；不是根据直观，而是根据顾客的购买模式导出的实际商品组来修正商品的排放；在降低推销商品开销的同时，提高总体推销的纯效益。

2.6.2 由联机分析处理到联机分析挖掘

在数据挖掘领域，一些研究成果已被用于各种平台下的数据挖掘。包括事务数据库、关系数据库、时间序列数据库、展开的文件、数据仓库等。

在数据挖掘的许多不同范例和结构中，联机分析挖掘（**OLAM**，也称 **OLAP 挖掘**）将联机分析处理与数据挖掘以及在多维数据库中发现知识集成在一起，由于以下原因而特别重要：

- **数据仓库中数据的高质量**：大部分数据挖掘工具需要在集成的、一致的和清理过的数据上运行，这需要昂贵的数据清理、数据变换和数据集成作为预处理步骤。经由这些预处理而构造的数据仓库不仅用作 OLAP，而且也用作数据挖掘的高质量的、有价值的数据源。注意，数据挖掘也可以作为数据清理和集成的有价值的工具。
- **环绕数据仓库的有价值的信息处理基础**：全面的数据处理和数据分析基础已经，或将要围绕数据仓库而系统地建立，这包括存取，集成，加固，多个异种数据库的转换，ODBC/OLE DB 连接，网络访问和服务机制，报表和 OLAP 分析工具。谨慎的做法是尽量利用可用的基础，而不是一切从头做起。
- **基于 OLAP 的探测式数据分析**：有效的数据挖掘需要探测式数据分析。用户常常想穿越数据库，选择相关数据，在不同的粒度上分析它们，以不同的形式提供知识/结果。联机分析数据挖掘提供在不同的数据子集和不同的抽象层上进行数据挖掘的机制，在数据方和一些挖掘的中间结果数据上进行上卷、下钻、旋转、过滤、切块、切片。这些与数据/知识可视化工具一起，将大大增强探测式数据挖掘的能力和灵活性。
- **数据挖掘功能的联机选择**：用户常常不知道他想挖掘什么类型的知识。通过将 OLAP 与多种数据挖掘功能集成在一起，联机分析挖掘为用户选择所期望的数据挖掘功能，动态地改变数据挖掘任务提供了灵活性。

联机分析挖掘的结构

OLAM 服务器用与 OLAP 服务器进行联机分析处理类似的方式，在数据方上进行联机分析挖掘。一个集成的 OLAM 和 OLAP 结构如图 2.24 所示。其中，OLAM 和 OLAP 都通过图形用户界面 API 接受用户查询（命令），并通过数据方 API 与数据方一道进行数据分析。元数据目录用于指导对数据方的访问。通过 MDDB API 访问和/或集成多个数据库，或者过支持 OLE DB 或 ODBC 连接的数据库 API 过滤数据仓库，可以构造数据方。由于 OLAM 服务器可以执行如概念描述、关联、分类、预测、聚类、时间序列分析等多种数据挖掘任务，它通常由多个集成的数据挖掘模块组成，并且比 OLAP 服务器复杂得多。

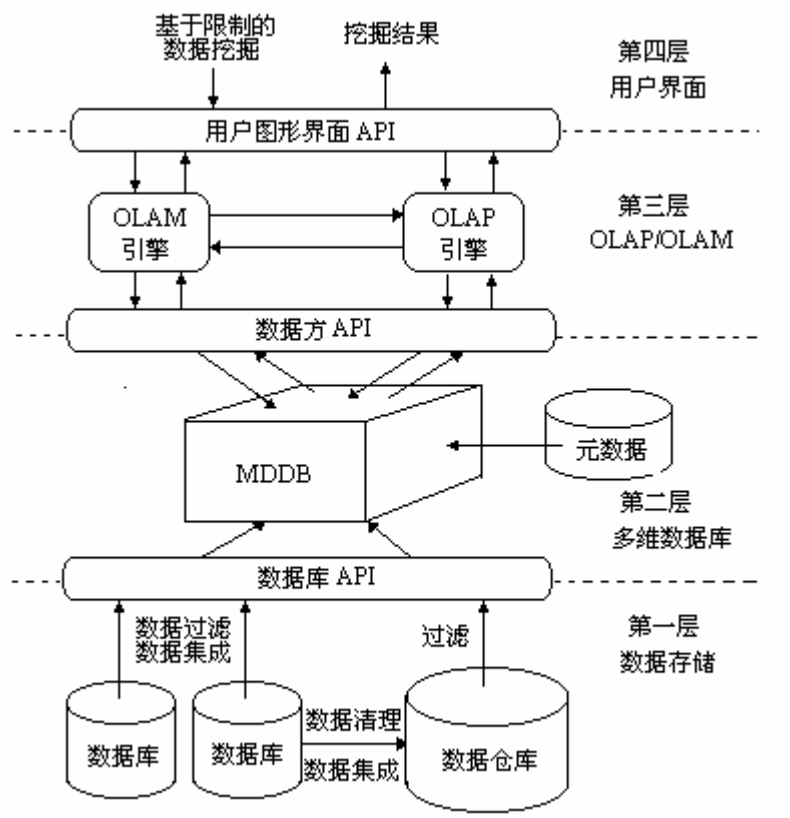


图 2.24　一个集成的 OLAM 和 OLAP 结构

本书的下面章节研究数据挖掘技术。正如我们已经看到的，本章介绍的数据仓库和 OLAP 技术对于数据挖掘的研究是基本的。这是因为数据仓库为用户提供了清洁的、有组织的、汇总的数据，大大地方便了数据挖掘。例如，数据仓库中不是存放每个销售事务的细节，而是存放每个部门每类商品的汇总数据，或对较高层次（如，国家）的汇总数据。OLAP 提供数据仓库中汇总数据的多个、动态的视图的能力，为成功的数据挖掘奠定了坚实的基础。

此外，我们也相信数据挖掘应当是以人为中心的过程。用户将经常与系统交互，进行探测式数据挖掘，而不是要求数据挖掘系统自动地产生模式和知识。OLAP 为交互式数据分析树立了一个好榜样，并为探测式数据挖掘做了必要的准备。例如，考虑关联模式的发现。应当允许用户沿着任意维上卷，而不是在原始的数据层，在事务间挖掘关联。例如，用户可能希望在 *item* 维上卷，由观察特定电视机的数据，到观察诸如索尼、松下等类电视机的数据。在搜索有趣的关联时，用户也可以由事务导航到顾客或顾客类型。这种 OLAP 风格的数据挖掘是 OLAM 的特点。在下一章研究数据挖掘原理时，我们特别强调 OLAP 挖掘。即，强调数据挖掘与 OLAP 技术的集成。

2.7 总结

- **数据仓库**是面向主题的、集成的、时变的和非易失的有组织的数据集合，支持管理决策制定。有一些因素区别数据仓库与操作数据库。由于两种系统提供相当不同的功能，需要不同类型的数据，有必要将数据仓库与操作数据库分开维护。
- 通常，**多维数据模型**用于数据仓库和数据集市的设计。这种模型采用星形模式、雪花模式或事实星座模式。多维数据模型的核心是**数据方**。数据方由大量事实（或度量）和许多维组成。维是一个组织想要记录的实体或透视，是自然分层的。
- **概念分层**将属性或维的值组织成渐进的抽象层。概念分层对于多抽象层上的挖掘是有用的。

- **联机分析处理**（**OLAP**）可以在使用多维数据模型的数据仓库或数据集市上进行。典型的 OLAP 操作包括上卷、下钻（钻过、钻透）、切片和切块、转轴（旋转），以及求等级、计算平均值和增长率等统计操作。使用数据方结构，OLAP 操作可以有效地实现。
- 数据仓库通常采用**三层结构**。底层是数据仓库服务器，通常是关系数据库系统。中间层是 OLAP 服务器。上层是客户，包括查询和报表工具。
- OLAP 服务器可以是**关系 OLAP**（**ROLAP**），**多维 OLAP**（**MOLAP**），或**混合 OLAP**（**HOLAP**）。ROLAP 服务器使用扩充的关系 DBMS，将多维数据上的 OLAP 操作映射成标准的关系操作。MOLAP 服务器直接将多维数据视图映射到数组结构。HOLAP 是 ROLAP 和 MOLAP 的结合。例如，它可以对历史数据使用 ROLAP，而将频繁访问的数据放在一个分离的 MOLAP 存储中。
- 数据方由**方体的格**组成，每个方体对应于给定多维数据的一个不同级别的汇总。**部分物化**是指有选择地物化格中方体的一个子集。**完全物化**是指物化格中所有的方体。如果方体使用 MOLAP 实现，则可以使用**多路数组聚集**。该技术将一些聚集计算重迭，使得整个物化计算更有效。
- 使用索引技术，OLAP 查询处理可以更有效地进行。在**位图索引**中，每个属性有它自己的位图索引表。位图索引将连接、聚集和比较归结成位算术运算。**连接索引**登记来自两个或多个关系的可连接行，降低 OLAP 连接操作的代价。**位图连接索引**结合位图和连接方法，可以进一步加快 OLAP 查询处理。
- 数据仓库**元数据**是定义仓库对象的数据。元数据库提供关于仓库结构、数据历史、汇总所使用的算法、由源数据到数据仓库的映射、系统性能和商务术语及含义等细节。
- 数据仓库包含加载和刷新数据仓库的**后端工具**和**实用程序**。这些包括数据的清理、数据变换、装入、刷新和仓库管理。
- 数据方的**发现驱动探查**使用预先计算的度量和可视方，指示所有聚集层中的数据例外，指导用户的分析进程。**多特征方**计算涉及多粒度上的多依赖的复杂查询。通过使用标准数据方计算的有效算法，发现驱动的探查和多特征方的计算可以有效地进行。
- 数据仓库用于信息处理（查询和报表）、分析处理（允许用户通过 OLAP 操作在汇总数据和细节数据之间导航）和数据挖掘（支持知识发现）。基于 OLAP 的数据挖掘称为 **OLAP 挖掘**或**联机分析挖掘**（**OLAM**），它强调 OLAP 挖掘的交互式和探测式特点。

习题

2.1 试述对于多个异种信息源的集成，为什么许多公司宁愿使用更新驱动的方法（构造使用数据仓库），而不愿使用查询驱动的方法（使用包装程序和集成程序）。描述一些情况，其中查询驱动方法比更新驱动方法更受欢迎。

2.2 简略比较以下概念,，你可以用例子解释你的观点。

(a) 雪花模式、事实星座、星形网查询模型。

(b) 数据清理、数据变换、刷新。

(c) 发现驱动数据方、多特征方、虚拟仓库。

2.3 假定 数据仓库包含三个维：*time, doctor* 和 *patient*；两个度量：*count* 和 *charge*；其中，*charge* 是医生对一位病人的一次来访的收费。

(a) 列举三种流行的数据仓库建模模式。

(b) 使用(a)列举的模式之一，画出上面数据仓库的模式图。

(c) 由基本方体[*day, doctor, patient*]开始，为列出 2000 年每位医生的收费总数，应当执行哪些 OLAP 操作？

(d) 为得到同样的结果，写一个 SQL 查询。假定数据存放在关系数据库中，其模式如下：

fee(day,month,year,doctor,hospital,patient,count,charge)

2.4 假定Big_University的数据仓库包含如下4个维：*student, course, semester*和 *instructor*；2 个度量：*count* 和 *avg_grade*。在最低的概念层（例如，对于给定的学生、课程、学期和教师的组合），度量 *avg_grade* 存放学生的实际成绩。在较高的概念层，*avg_grade* 存放给定组合的平均成绩。

(a) 为数据仓库画出雪花模式图。

(b) 由基本方体[*student, course, semester, instructor*]开始，为列出 Big_University 每个学生的 CS 课程的平均成绩，应当使用哪些 OLAP 操作（如，由学期上卷到学年）。

(c) 如果每维有 5 层（包括 all），如 *student* ≤ *major* ≤ *status* ≤ *university* ≤ all，该数据方包含多少方体（包括基本方体和顶点方体）？

2.5 假定数据仓库包含 4 个维：*date, spectator, location* 和 *game*；2 个度量：*count* 和 *charge*。其中，*charge* 是观众在给定的日期观看节目的付费。观众可以是学生、成年人或老人，每类观众有不同的收费标准。

(a) 画出该数据仓库的星形模式图。

(b) 由基本方体[*date, spectator, location, game*]开始，为列出 2000 年学生观众在 GM-Place 的总代价，应当执行哪些 OLAP 操作？

(c) 对于数据仓库，位图索引是有用的。以该数据方为例，简略讨论使用位图索引结构的优点和问题。

2.6 为地区气象局设计一个数据仓库。气象局大约有 1,000 观察点，散布在该地区的陆地、海洋，收集基本气象数据，包括每小时的气压、温度、降雨量。所有的数据都送到中心站，那里已收集了这种数据长达十年。你的设计应当有利于有效的查询和联机分析处理，有利于有效地导出多维空间的一般天气模式。

2.7 在数据方中计算度量：

(a) 根据计算数据方所用的聚集函数，列出度量的三种分类。

(b) 对于具有三个维，*time, location* 和 *product* 的数据方，函数 variance 属于哪一类？如果方被分割成一些块，描述如何计算它。

提示：计算 variance 函数的公式是：$\frac{1}{n}\sum_{i=1}^{n}(x_i)^2 - \bar{x}^2$，其中，$\bar{x}$ 是这些 x_i 的平均值。

(c) 假定函数是“最高的 10 个销售额”。讨论如何在数据方里有效地计算该度量。

2.8 假定需要在数据方中记录三种度量： min.、average 和 median。给定的数据方允许渐增地删除（即，每次一小部分），为每种度量设计有效的计算和存储方法。

2.9 数据仓库实现的流行方法是构造一个称为数据方的多维数据库。不幸的是，这常常产生大的、稀疏的多维矩阵。

(a) 给出一个例子，解释这种大的、稀疏的数据方。

(b) 设计一种实现方法，可以很好地克服这种稀疏矩阵问题。注意，你需要详细解释你的数据结构，讨论空间需求量，以及如何由你的结构中提取数据。

(c) 修改你在(b)的设计，处理渐增的数据更新。给出你的新设计的理由。

2.10 在数据仓库技术中，多维视图可以用多维数据库技术（MOLAP），或关系数据库技术（ROLAP），或混合数据库技术（HOLAP）实现。

(a) 简要描述每种实现技术。

(b) 对每种技术，解释如下函数如何实现：

i. 数据仓库的产生（包括聚集）。

ii. 上卷。

iii. 下钻。

iv. 渐增更新。

你喜欢哪种实现技术？为什么？

2.11 假定数据仓库包含 20 个维，每个维有 5 级粒度。

(a) 用户感兴趣的主要是 4 个特定的维，每维有 3 个上卷、下钻频繁访问的级。你如何设计数据方结构，有效地对此予以支持？

(b) 时常，用户想由一、两个特定的维钻透数据方，到原始数据。你如何支持这一特征？

2.12 假定基本方体有三个维(*A,B,C*)，其单元数如下：|*A*|=1,000,000, |*B*|=100, |*C*|=1,000。假定分块将每维分成 10 部分。

(a) 假定每维只有一层，画出完整的方的格。

(b) 如果每个方单元存放一个 4 字节的度量，若方是稠密的，所计算的方有多大？

(c) 指出空间需求量最小的块计算次序，并计算 2-维平面计算所需要的内存空间。

3.13 考虑下面的多特征方查询：按{*item, region, month*}的所有子集分组，对每组找出2000年的最小货架寿命，并对价格低于、其最小货架寿命在之间的元组找出总销售额部分。

(a) 画出该查询的多特征方图。

(b) 用扩充的SQL表示该查询。

(c) 这是一个分布式多特征方吗？为什么？

2.14 三种主要的数据仓库应用——信息处理、分析处理和数据挖掘——的主要区别是什么？讨论OLAP挖掘（OLAM）的动机。

文献注释

有大量关于建立数据仓库和OLAP技术的引论性教材，包括Inmon[Inm96], Kimball[Kim96], Berson和Smith[BS97], 以及Thomsen[Tho97]。Chaudhuri和Dayal[CD97]给出了建立数据仓库和OLAP技术的综述。

决策支持系统的历史可以追溯到20世纪60年代。然而，为多维数据分析构造大型数据仓库的提议归功于Codd[CCS93]，他创建术语OLAP表示联机分析处理。OLAP委员会成立于1995年。Widom[Wid95]列举了数据仓库的一些研究问题。Kimball[Kim96]总结了SQL在支持商业界常见的比较方面的不足。关于OLAP系统与统计数据库的比较综述，见Shoshani[Sho97]。

DMQL数据挖掘查询语言由Han等提出[HFW+96a]。数据挖掘查询语言的进一步讨论在第4章。其它基于SQL的数据挖掘语言在Imielinski, Virmani和Abdulghani[IVA96]、Meo, Psaila和Ceri[IPC96]、Baralis和Psaila[BP97]中提出。

Gray等[GCB+97]提出将数据方作为关系聚集操作符，泛化分组、交叉表、子和。Harinarayan, Rajaraman和Ullman[HRU96]提出数据方计算的有选择物化的贪心算法。Agarwal等[AAD+96]为ROLAP服务器多维聚集的有效计算提出了一些方法。2.4.1小节介绍的MOLAP中计算数据方的基于块的多路数组聚集算法由Zhao, Deshpande和Naughton[ZDN97]提出。数据方快速计算的其它方法可以在Beyer和Ramakrishnan[BR99]、Ross和Srivastava[RS97]中找到。Sarawagi和Stonebraker[SS94]开发了一种大型多维数组有效组织的基于块的计算技术。冰山查询在Fang, Shivakumar, Garcia-Molina等[FSGM+98]以及Beyer和Ramakrishnan[BR99]中介绍。

使用连接索引来加快关系查询处理由Valduriez[Val89]提出。O'Neil和Graefe[OG95]提出位图连接索引方法，以加快基于OLAP的查询处理。位映射和其它非常规索引技术的性能讨论在O'Neil和Quass[OQ97]中给出。

关于为有效的OLAP查询处理，物化方体选择的工作，见Chaudhuri和Dayal[CD97]、Harinarayan, Rajaraman和Ullman[HRU96]、Sristava等[SDJL96]。方体大小估计的方法可以在Deshpande等[DNR+97]、Ross和Srivastava[RS97]、Beyer和Ramakrishnan[BR99]中找到。Agrawal, Gupta和Sarawagi[AGS97]提出了多维数据库建模的操作。

最近，有一些用于数据挖掘的发现驱动的数据方实现方面的研究，包括由Sarawagi,Agrawal和Megiddo[SAM98]提出的OLAP数据方的发现驱动的探查，以及Ross, Srivastava和Chatziantonion[RSC98]的多特征方构造。通过联机聚集快速回答查询的方法在Hellerstein, Haas和Wang[HHW97]、Hellerstein等[HAC+99]]中介绍。估计最高N个查询的技术由Carey和Kossman[CK98]、Donjerkovic和Ramakrishnan[DR99]提出。OLAM技术的讨论见Han[Han98]。

第三章 数据预处理

当今现实世界中的数据库极易受噪音数据、遗漏数据和不一致性数据的侵扰，因为数据库太大，常常多达数千兆，甚至更多。“如何预处理数据，提高数据质量，从而提高挖掘结果的质量？”你可能会问。“怎样预处理数据，使得挖掘过程更加有效、更加容易？”

有大量数据预处理技术。数据清理可以去掉数据中的噪音，纠正不一致。数据集成将数据由多个源合并成一致的数据存储，如数据仓库或数据方。数据变换（如规范化）也可以使用。例如，规范化可以改进涉及距离度量的挖掘算法的精度和有效性。数据归约可以通过聚集、删除冗余特征或聚类等方法来压缩数据。这些数据处理技术在数据挖掘之前使用，可以大大提高数据挖掘模式的质量，降低实际挖掘所需要的时间。

本章，你将学习数据预处理的方法。这些方法包括：数据清理、数据集成和转换、数据归约。本章还讨论数据离散化和概念分层，它们是数据归约的一种替换形式。概念分层可以进一步用于多抽象层挖掘。你将学习如何由给定的数据自动地产生概念分层。

3.1 为什么要预处理数据？

想象你是 AllElectronics 的经理，负责分析涉及你部门的公司数据。你立即着手进行这项工作。你仔细地研究和审查公司的数据库或数据仓库，找出应当包含在你的分析中的属性或维，如 *item*, *price* 和 *units_sold*。啊！你注意到，许多元组在一些属性上没有值。对于你的分析，你希望知道每种销售商品是否通过广告降价销售，但你又发现这些信息根本未记录。此外，你的数据库系统用户已经报告一些错误、不寻常的值和某些事务记录中的不一致性。换言之，你希望使用数据挖掘技术分析的数据是**不完整的**（有些感兴趣的属性缺少属性值，或仅包含聚集数据），**含噪音的**（包含错误，或存在偏离期望的局外者），并且是**不一致的**（例如，用于商品分类的部门编码存在差异）。欢迎来到现实世界！

存在不完整的、含噪音的和不一致的数据是大型的、现实世界数据库或数据仓库的共同特点。不完整数据的出现可能有多种原因。有些感兴趣的属性，如销售事务数据中顾客的信息，并非总是可用的。其它数据没有包含在内，可能只是因为输入时认为是不重要的。相关数据没有记录是由于理解错误，或者因为设备故障。此外，记录历史或修改的数据可能被忽略。与其它数据不一致的数据可以删除。遗漏的数据，特别是某些属性上缺少值的元组可能需要推导出来。

数据含噪音（具有不正确的属性值）可能有多种原因。收集数据的设备可能出故障；人的或计算机的错误可能在数据输入时出现；数据传输中的错误也可能出现。这些可能是由于技术的限制，如用于数据传输同步的缓冲区大小的限制。不正确的数据也可能是由命名或所用的数据代码不一致而导致的。重复元组也需要数据清理。

数据清理例程通过填写遗漏的值，平滑噪音数据，识别、删除局外者，并解决不一致来“清理”数据。脏数据造成挖掘过程陷入困惑，导致不可靠的输出。尽管大部分挖掘例程都有一些过程，处理不完整或噪音数据，但它们并非总是强壮的。相反，它们更致力于避免数据过分适合所建的模型。这样，一个有用的预处理步骤是使用某些清理例程清理你的数据。3.2 节讨论清理数据的方法。

回到你在 AllElectronics 的任务，假定你想在你的分析中包含来自多个数据源的数据。这涉及集成多个数据库、数据方或文件，即**数据集成**。代表同一概念的属性在不同的数据库中可能具有不同的名字，这又导致不一致性和冗余。例如，关于顾客标识符的属性在一种数据存储中为 *customer_id*，而在另一种为 *cust_id*。命名的不一致还可能出现在属性值中。例如，同名的人可能在一个数据库中登记为 Bill，在第二个数据库中登记为 William，而在第三个数据库中登记为“B”。此外，你可能

会觉察到，有些属性可能是由其它属性导出的（例如，年收入）。含大量冗余数据可能降低知识发现过程的性能或使之陷入困惑。显然，除数据清理之外，必须采取步骤，避免数据集成时的冗余。通常，在为数据仓库准备数据时，数据清理和集成将作为预处理步骤进行。还可以再次进行数据清理，检测和移去可能由集成导致的冗余。

回到你的数据，如果你决定要使用诸如神经网络、最临近分类或聚类[7]这样的基于距离的挖掘算法进行你的分析。如果要分析的数据已规格化，即按比例映射到一个特定的区间[0.0,1.0]，这种方法能得到较好的结果。例如，你的顾客数据包含年龄和年薪属性。年薪属性的取值范围可能比年龄更大。这样，如果属性未规格化，在年薪上距离度量所取的权重一般要超过在年龄度量上所取的权重。此外，对于你的分析，得到每个地区的销售额这样的聚集信息可能有用的。这种信息不在你的数据仓库的任何预计算的数据方中。你很快意识到，**数据变换**操作，如规格化和聚集，是导向挖掘过程成功的预处理过程。数据集成和数据变换将在 3.3 节讨论。

随着你进一步考虑数据，你想知道“我所选择用于数据分析的数据集太大了——它肯定降低挖掘过程的速度。有没有办法使我能够‘压缩’我的数据集，而又不损害数据挖掘的结果？”**数据归约**得到数据集的压缩表示，它小得多，但能够产生同样的（或几乎同样的）分析结果。有许多数据归约策略，包括数据聚集（例如，建立数据方）、维归约（例如，通过相关分析，去掉不相关的属性）、数据压缩（例如，使用诸如最短编码或小波等编码方案）和数字归约（例如，使用聚类或参数模型等较短的表示“替换”数据）。泛化也可以“归约”数据。泛化用较高层的概念替换较低层的概念；例如，用地区或省/州替换城市。概念分层将概念组织在不同的抽象层。数据归约是 3.4 节的主题。由于概念分层对于多抽象层上的数据挖掘是非常有用的，我们另用一节来讨论这种重要数据结构的产生。3.5 节讨论概念分层的产生，通过数据离散化进行数据归约。

图 3.1 总结了这里讨论的数据预处理步骤。注意，上面的分类不是互斥的。例如，冗余数据的删除既是数据清理，也是数据归约。

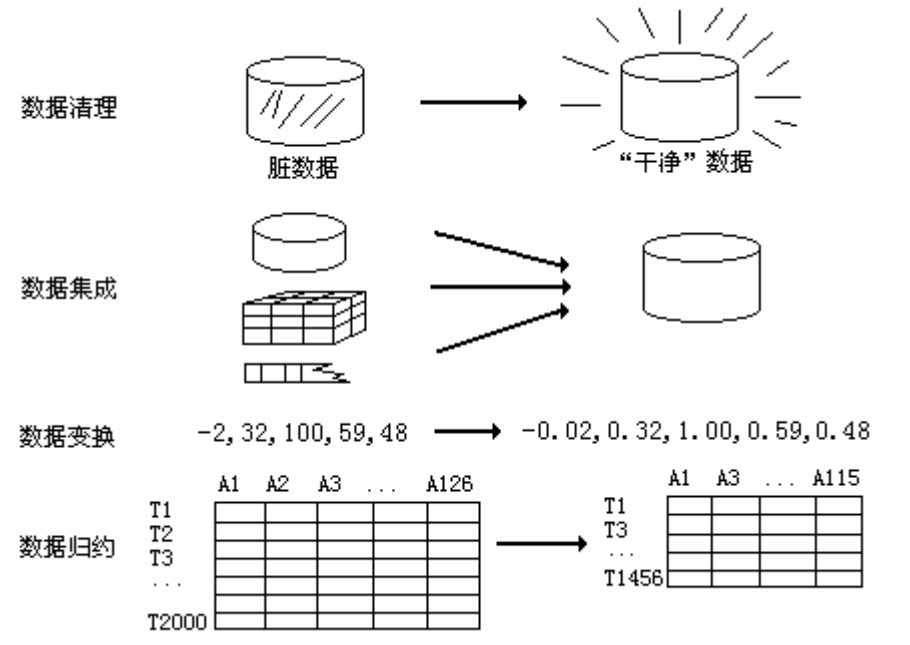


图 3.1　　数据预处理的形式

概言之，现实世界的数据一般是脏的、不完整的和不一致的。数据预处理技术可以改进数据的质量，从而有助于提高其后的挖掘过程的精度和性能。由于高质量的决策必然依赖于高质量的数据，因此数据预处理是知识发现过程的重要步骤。检测数据异常、尽早地调整数据，并归约待分析的数据，将在决策制定时得到高回报。

[7] 神经网络和最临近分类在第 7 章介绍，而聚类在第 8 章讨论。

3.2 数据清理

现实世界的数据一般是脏的、不完整的和不一致的。数据清理例程试图填充遗漏的值，识别局外者、消除噪音，并纠正数据中的不一致。本节，我们将研究数据清理的基本方法。

3.2.1 遗漏值

想象你要分析 AllElectronics 的销售和顾客数据。你注意到许多元组的一些属性，如顾客的收入，没有记录值。你怎样才能为该属性填上遗漏的值？让我们看看下面的方法，

1. **忽略元组**：当类标号缺少时通常这样做（假定挖掘任务涉及分类或描述）。除非元组有多个属性缺少值，否则该方法不是很有效。当每个属性缺少值的百分比很高时，它的性能非常差。
2. **人工填写遗漏值**：一般地说，该方法很费时，并且当数据集很大，缺少很多值时，该方法可能行不通。
3. **使用一个全局常量填充遗漏值**：将遗漏的属性值用同一个常数（如“*Unknown*”或 $-\infty$）替换。如果遗漏值都用“*Unknown*”替换，挖掘程序可能误以为它们形成了一个有趣的概念，因为它们都具有相同的值——“*Unknown*”。因此，尽管该方法简单，我们并不推荐它。
4. **使用属性的平均值填充遗漏值**：例如，假定 AllElectronics 顾客的平均收入为$28,000，则使用该值替换 *income* 中的遗漏值。
5. **使用与给定元组属同一类的所有样本的平均值**：例如，如果将顾客按 *credit_risk* 分类，则用具有相同信用度的顾客的平均收入替换 *income* 中的遗漏值。
6. **使用最可能的值填充遗漏值**：可以用回归、使用贝叶斯形式化方法或判定树归纳等基于推导的工具确定。例如，利用你的数据集中其他顾客的属性，你可以构造一棵判定树，来预测 *income* 的遗漏值。判定树将在第 7 章详细讨论。

方法 3 到 6 使数据倾斜，填入的值可能不正确。然而，方法 6 是最常用的方法。与其它方法相比，它使用现存数据的最多信息来推测遗漏值。在估计 *income* 的遗漏值时，通过考虑其它属性的值，有更大的机会保持 *income* 和其它属性之间的联系。

3.2.2 噪音数据

“什么是噪音？”**噪音**是测量变量的随机错误或偏差。给定一个数值属性，例如 *price*，我们怎样才能平滑数据，去掉噪音？让我们看看下面的数据平滑技术。

1. **分箱**：分箱方法通过考察“邻居”（即，周围的值）来平滑存储数据的值。存储的值被分布到一些“桶”或箱中。由于分箱方法导致值相邻，因此它进行局部平滑。图 3.2 图示了一些分箱技术。在该例中，*price* 数据首先被划分并存入等深的箱中（深度 3）。对于**按平均值平滑**，箱中每一个值被箱中的平均值替换。例如，箱 1 中的值 4，8 和 15 的平均值是 9；这样，该箱中的每一个值被替换为 9。类似地，可以使用**按中值平滑**。此时，箱中的每一个值被箱中的中值替换。对于**按边界平滑**，箱中的最大和最小值同样被视为边界。箱中的每一个值被最近的边界值替换。一般来说，宽度越大，平滑效果越大。箱也可以是等宽的，每个箱值的区间范围是个常量。分箱也可以作为一种离散化技术使用，将在 3.5 节和第 6 章进一步讨论。

   price 的排序后数据（元）：4, 8, 15, 21, 21, 24, 25, 28, 34

   划分为（等深的）箱：

   箱 1：4, 8, 15

   箱 2：21, 21, 24

箱 3：25, 28, 34

用平均值平滑：

箱 1：9, 9, 9

箱 2：22, 22, 22

箱 3：29, 29, 29

用边界平滑：

箱 1：4, 4, 15

箱 2：21, 21, 24

箱 3：25, 25, 34

图 3.2　数据平滑的分箱方法

2. **聚类：**局外者可以被聚类检测。聚类将类似的值组织成群或“聚类”。直观地，落在聚类集合之外的值被视为局外者（图 3.3）。第 9 章将研究聚类。

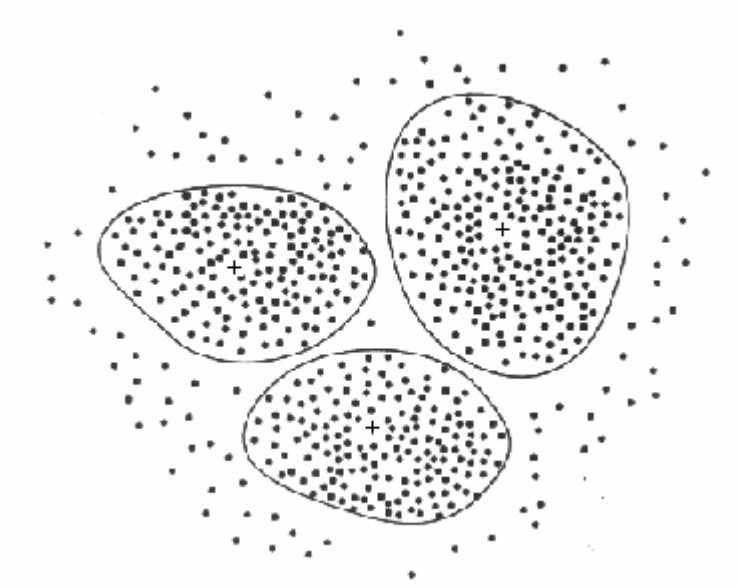


图 3.3　局外者可以被聚类检测

3. **计算机和人工检查结合：**可以通过计算机和人工检查结合的办法来识别局外者。例如，在一种应用中，使用信息理论度量，帮助识别手写体字符数据库中的局外者。度量值反映被判断的字符与已知的符号相比的“差异”程度。局外者模式可能是提供信息的（例如，识别有用的数据例外，如字符“0”或“7”的不同版本）或者是“垃圾”（例如，错误的字符）。其差异程度大于某个阈值的模式输出到一个表中。人可以审查表中的模式，识别真正的垃圾。这比人工地搜索整个数据库快得多。在其后的数据挖掘应用时，垃圾模式将由数据库中清除掉。

4. **回归：**可以通过让数据适合一个函数（如回归函数）来平滑数据。线性回归涉及找出适合两个变量的“最佳”直线，使得一个变量能够预测另一个。多线性回归是线性回归的扩展，它涉及多于两个变量，数据要适合一个多维面。使用回归，找出适合数据的数学方程式，能够帮助消除噪音。回归将在 3.4.4 小节以及第 7 章讨论。

许多数据平滑的方法也是涉及离散化的数据归约方法。例如，上面介绍的分箱技术减少了每个属性的不同值的数量。对于基于逻辑的数据挖掘方法（如判定树归纳），这充当了一种形式的数据归约。概念分层是一种数据离散化形式，也可以用于数据平滑。例如，*price* 的概念分层可以把 *price* 的值映射到 *inexpensive*、*moderately_priced* 和 *expensive*，从而减少了挖掘过程所处理的值的数量。

数据离散化将在 3.5 节讨论。有些分类方法，如神经网络，有内置的数据平滑机制。分类是第 7 章的课题。

3.2.3 不一致数据

对于有些事务，所记录的数据可能存在不一致。有些数据不一致可以使用其它材料人工地加以更正。例如，数据输入是的错误可以使用纸上的记录加以更正。这可以与用来帮助纠正编码不一致的例程一块使用。知识工程工具也可以用来检测违反限制的数据。例如，知道属性间的函数依赖，可以查找违反函数依赖的值。

由于数据集成，也可能产生不一致：一个给定的属性在不同的数据库中可能具有不同的名字。也可能存在冗余。数据集成和冗余数据删除在 3.3.1 小节讨论.

3.3 数据集成和变换

数据挖掘经常需要数据集成——由多个数据存储合并数据。数据还可能需要转换成适于挖掘的形式。本节介绍数据集成和数据变换。

3.3.1 数据集成

数据分析任务多半涉及数据集成。数据集成将多个数据源中的数据结合成、存放在一个一致的数据存储，如数据仓库中。这些源可能包括多个数据库、数据方或一般文件。

在数据集成时，有许多问题需要考虑。模式集成可能是有技巧的。来自多个信息源的现实世界的实体如何才能“匹配”？这涉及**实体识别**问题。例如，数据分析者或计算机如何才能确信一个数据库中的 *customer_id* 和另一个数据库中的 *cust_number* 指的是同一实体？通常，数据库和数据仓库有元数据——关于数据的数据。这种元数据可以帮助避免模式集成中的错误。

冗余是另一个重要问题。一个属性是冗余的，如果它能由另一个表“导出”；如年薪。属性或维命名的不一致也可能导致数据集中的冗余。

有些冗余可以被**相关分析**检测到。例如，给定两个属性，根据可用的数据，这种分析可以度量一个属性能在多大程度上蕴涵另一个。属性 *A* 和 *B* 之间的相关性可用下式度量：

$$r_{A,B} = \frac{\sum (A - \overline{A})(B - \overline{B})}{(n-1)\sigma_A \sigma_B} \qquad (3.1)$$

其中，*n* 是元组个数，$\overline{A}$ 和 $\overline{B}$ 分别是 *A* 和 *B* 的平均值，σ_A 和 σ_B 分别是 A 和 B 的标准差[8]。如果(3.1)式的值大于 0，则 *A* 和 *B* 是正相关的，意味 *A* 的值随 *B* 的值增加而增加。该值越大，一个属性蕴涵另一个的可能性越大。因此，一个很大的值表明 *A*（或 *B*）可以作为冗余而被去掉。如果结果值等于 0，则 *A* 和 *B* 是独立的，它们之间不相关。如果结果值小于 0，则 *A* 和 *B* 是负相关的，一个值随另一个减少而增加。这表明每一个属性都阻止另一个出现。(3.1)式可以用来检测上面的 *customer_id* 和 *cust_number* 的相关性。相关分析在 6.5.2 小节进一步讨论。

---

[8] A 的平均值是

$$\overline{A} = \frac{\sum A}{n}$$

A 的标准差是

$$\sigma_A = \sqrt{\frac{\sum (A - \overline{A})^2}{n-1}}$$

除了检测属性间的冗余外，“重复”也应当在元组级进行检测。重复是指对于同一数据，存在两个或多个相同的元组。

数据集成的第三个重要问题是数据值冲突的检测与处理。例如，对于现实世界的同一实体，来自不同数据源的属性值可能不同。这可能是因为表示、比例或编码不同。例如，重量属性可能在一个系统中以公制单位存放，而在另一个系统中以英制单位存放。不同旅馆的价格不仅可能涉及不同的货币，而且可能涉及不同的服务（如免费早餐）和税。数据这种语义上的异种性，是数据集成的巨大挑战。

仔细将多个数据源中的数据集成起来，能够减少或避免结果数据集中数据的冗余和不一致性。这有助于提高其后挖掘的精度和速度。

3.3.2 数据变换

数据变换将数据转换成适合于挖掘的形式。数据变换可能涉及如下内容:

- **平滑**：去掉数据中的噪音。这种技术包括分箱、聚类和回归。
- **聚集**：对数据进行汇总和聚集。例如，可以聚集日销售数据，计算月和年销售额。通常，这一步用来为多粒度数据分析构造数据方。
- **数据泛化**：使用概念分层，用高层次概念替换低层次“原始”数据。例如，分类的属性，如 *street*，可以泛化为较高层的概念，如 *city* 或 *country*。类似地，数值属性，如 *age*，可以映射到较高层概念，如 *young, middle-age* 和 *senior*。
- **规范化**：将属性数据按比例缩放，使之落入一个小的特定区间，如-1.0 到 1.0 或 0.0 到 1.0。
- **属性构造**（或特征构造）：可以构造新的属性并添加到属性集中，以帮助挖掘过程。

平滑是一种数据清理形式，已在 3.2.2 小节讨论。聚集和泛化也是一种数据归约形式，并分别将在 3.4 和 3.5 小节讨论. 本节，我们讨论规范化和属性构造。

通过将属性数据按比例缩放，使之落入一个小的特定区间，如 0.0 到 1.0，对属性规范化。对于距离度量分类算法，如涉及神经网络或诸如最临近分类和聚类的分类算法，规范化特别有用。如果使用神经网络后向传播算法进行分类挖掘（第 7 章），对于训练样本属性输入值规范化将有助于加快学习阶段的速度。对于基于距离的方法，规范化可以帮助防止具有较大初始值域的属性（例如，*income*）与具有较小初始值域的属性（例如，二进位属性）相比，权重过大。有许多数据规范化的方法，我们将学习三种：最小-最大规范化、z-score 规范化和按小数定标规范化。

最小-最大规范化对原始数据进行线性变换。假定 $\min_A$ 和 $\max_A$ 分别为属性 A 的最小和最大值。最小-最大规范化通过计算

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A \qquad (3.2)$$

将 *A* 的值 *v* 映射到区间[$new\_\min_A, new\_\max_A$]中的 *v’*。

最小-最大规范化保持原始数据值之间的关系。如果今后的输入落在 *A* 的原数据区之外，该方法将面临“越界”错误。

例 3.1 假定属性 *income* 的最小与最大值分别为\$12,000 和\$98, 000。我们想映射 *income* 到区间[0.0, 0.1]。根据最小-最大规范化，*income* 值\$73,600 将变换为: $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1-0) = 0.716$ 。□

在 **z- score 规范化**（或零-均值规范化）中，属性 *A* 的值基于 *A* 的平均值和标准差规范化。*A* 的值 *v* 被规范化为 *v’*，由下式计算：

$$v' = \frac{v - \overline{A}}{\sigma_A} \tag{3.3}$$

其中，$\overline{A}$ 和 σ_A 分别为属性 *A* 的平均值和标准差。当属性 *A* 的最大和最小值未知，或局外者左右了最大-最小规范化时，该方法是有用的。

例 3.2 假定属性 *income* 的平均值和标准差分别为\$54,000 和\$16,000。使用 z-score 规范化，值\$73,600 被转换为 $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$。□

小数定标规范化通过移动属性 *A* 的小数点位置进行规范化。小数点的移动位数依赖于 *A* 的最大绝对值。*A* 的值 v 被规范化为 *v'*，由下式计算：

$$v' = \frac{v}{10^j} \tag{3,4}$$

其中，*j* 是使得 *Max*(|*v'*|) < 1 的最小整数。

例 3.3 假定 *A* 的值由-986 到 917。*A* 的最大绝对值为 986。为使用小数定标规范化，我们用 1,000（即，j=3）除每个值。这样，-986 被规范化为-0.986。□

注意，规范化将原来的数据改变很多，特别是上述的后两种方法。有必要保留规范化参数（如平均值和标准差，如果使用 z-score 规范化），以便将来的数据可以用一致的方式规范化。

属性构造是由给定的属性构造和添加新的属性，以帮助提高精度和对高维数据结构的理解。例如，我们可能根据属性 *height* 和 *width* 添加属性 *area*。属性结构可以帮助平缓使用判定树算法分类的分裂问题。那里，沿着导出判定树[9]的一条路径重复地测试一个属性。属性构造操作符的例子包括二进位属性的 **and** 和名字属性的 **product**。通过组合属性，属性构造可以发现关于数据属性间联系的丢失信息，这对知识发现是有用的。

3.4 数据归约

假定你由 AllElectronics 数据仓库选择了数据，用于分析。数据集将非常大！在海量数据上进行复杂的数据分析和挖掘将需要很长时间，使得这种分析不现实或不可行。

数据归约技术可以用来得到数据集的归约表示，它小得多，但仍接近地保持原数据的完整性。这样，在归约后的数据集上挖掘将更有效，并产生相同（或几乎相同）的分析结果。

数据归约的策略如下：

1. **数据方聚集：**聚集操作用于数据方中的数据。
2. **维归约：**可以检测并删除不相关、弱相关或冗余的属性或维。
3. **数据压缩：**使用编码机制压缩数据集。
4. **数值压缩：**用替代的、较小的数据表示替换或估计数据，如参数模型（只需要存放模型参数，而不是实际数据）或非参数方法，如聚类、选样和使用直方图。

[9] 判定树将在第 3 章详细介绍。

5. **离散化和概念分层产生：**属性的原始值用区间值或较高层的概念替换。概念分层允许挖掘多个抽象层上的数据，是数据挖掘的一种强有力的工具。我们将概念分层的自动产生推迟到 3.5 节，那里整整一节讨论该课题。

策略 1 至 4 在本节的剩余部分讨论。用于数据压缩的时间不应当超过或“抵消”在归约后数据上挖掘节省的时间。

3.4.1 数据方聚集

想象你已经为你的分析收集了数据。这些数据由 AllElectronics 1997 到 1999 年每季度的销售数据组成。然而，你感兴趣的是年销售（每年的总和），而不是每季度的总和。可以对这种数据再聚集，使得结果数据汇总每年的总销售，而不是每季度的总销售。该聚集如图 3.4 所示。结果数据量小得多，并不丢失分析任务所需的信息。

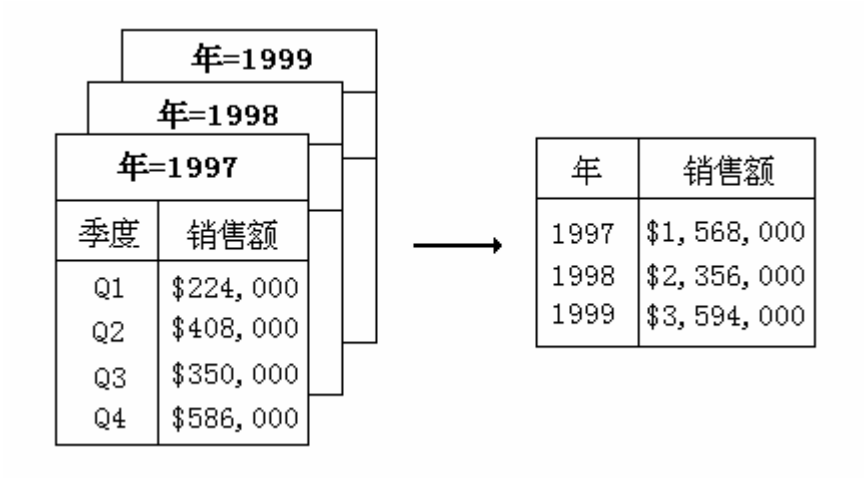


图 3.4　AllElectronics 1997 年到 1999 年的销售数据。左部销售数据按季度显示，右部数据聚集以提供年销售额

数据方已在第 2 章讨论。为完整起见，我们在这简略回顾一下。数据方存放多维聚集信息。例如，图 3.5 所示数据方用于 AllElectronics 所有分部每类商品年销售多维数据分析。每个单元存放一个聚集值，对应于多维空间的一个数据点。每个属性可能存在概念分层，允许在多个抽象层进行数据分析。例如，*branch* 的分层允许分部按它们的地址聚集成地区。数据方提供对预计算的汇总数据进行快速访问，因此它适合联机数据分析和数据挖掘。

创建在最低层的数据方称为**基本方体**。最高层抽象的数据方称为**顶点方体**。对于图 3.5 的销售数据，顶点方体将给出一个汇总值——所有商品类型、所有分部三年的总销售额。对不同层创建的数据方称为方体，因此“数据方”可以看作方体的格。每个较高层的抽象将进一步减少结果数据。

基本方体应当对应于感兴趣的实体，如 *sales* 或 *customer*。换言之，最低层对于分析应当是有用的。由于数据方提供了对预计算的汇总数据的快速访问，在响应关于聚集信息的查询时应当使用它们。当响应 OLAP 查询或数据挖掘查询时，应当使用与给定任务相关的最小方体。该问题也已在第 2 章讨论过。

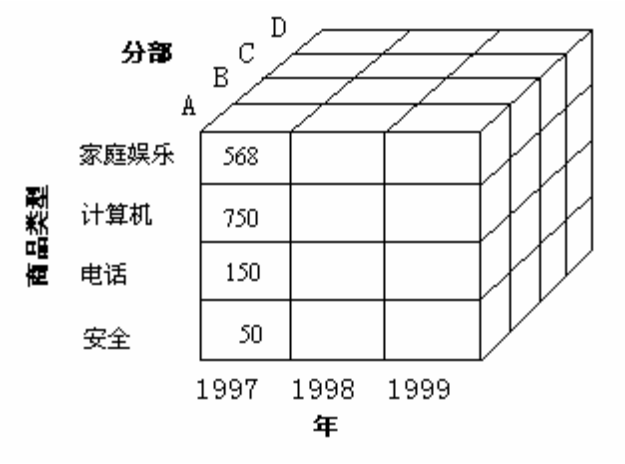


图 3.5　AllElectronics 销售数据方

3.4.2 维归约

用于数据分析的数据可能包含数以百计的属性，其中大部分属性与挖掘任务不相关，是冗余的。例如，如果分析任务是按顾客听到广告后，是否愿意在 AllElectronics 买流行的新款 CD 将顾客分类，与属性 *age*, *music_taste* 不同，诸如顾客的电话号码等属性多半是不相关的。尽管领域专家可以挑选出有用的属性，但这可能是一项困难而费时的任务，特别是当数据的行为不清楚的时候更是如此。遗漏相关属性或留下不相关属性是有害的，会导致所用的挖掘算法无所适从。这可能导致发现的模式质量很差。此外，不相关或冗余的属性增加了数据量，可能会减慢挖掘进程。

维归约通过删除不相关的属性（或维）减少数据量。通常使用属性子集选择方法。**属性子集选择**的目标是找出最小属性集，使得数据类的概率分布尽可能地接近使用所有属性的原分布。在压缩的属性集上挖掘还有其它的优点。它减少了出现在发现模式上的属性的数目，使得模式更易于理解。

“如何找出原属性的一个‘好的’子集？”*d* 个属性有 2^d 个可能的子集。穷举搜索找出属性的最佳子集可能是不现实的，特别是当 *d* 和数据类的数目增加时。因此，对于属性子集选择，通常使用压缩搜索空间的启发式算法。通常，这些算法是贪心算法，在搜索属性空间时，总是做看上去是最佳的选择。它们的策略是做局部最优选择，期望由此导致全局最优解。在实践中，这种贪心方法是有效的，并可以逼近最优解。

“最好的”（或“最差的”）属性使用统计测试来选择。这种测试假定属性是相互独立的。也可以使用一些其它属性估计度量，如使用信息增益度量建立分类判定树[10]。

属性子集选择的基本启发式方法包括以下技术，其中一些图示在图 3.6 中。

1. **逐步向前选择：**该过程由空属性集开始，选择原属性集中最好的属性，并将它添加到该集合中。在其后的每一次迭代，将原属性集剩下的属性中的最好的属性添加到该集合中。
2. **逐步向后删除：**该过程由整个属性集开始。在每一步，删除掉尚在属性集中的最坏属性。
3. **向前选择和向后删除的结合：**向前选择和向后删除方法可以结合在一起，每一步选择一个最好的属性，并在剩余属性中删除一个最坏的属性。

方法 1 到 3 的结束条件可以有多种。过程可以使用一个阈值来确定是否停止属性选择过程。

- **判定树归纳：**判定树算法，如 ID3 和 C4.5 最初是用于分类的。判定树归纳构造一个类似于流程图的结构，其每个内部（非树叶）结点表示一个属性上的测试，每个分枝对应于测试的一个输出；每个外部（树叶）结点表示一个判定类。在每个结点，算法选择“最好”的属性，将数据划分成类。

[10] 信息增益度量在 5.3.2 和 7.3.1 中详细介绍介绍。在 3.5.1 小节中结合属性离散归约要介绍。

当判定树**归纳**用于属性子集选择时，树由给定的数据构造。不出现在树中的所有属性假定是不相关的。出现在树中的属性形成归约后的属性子集。这种属性选择方法将在第 5 章讨论概念描述时更详细地讨论。

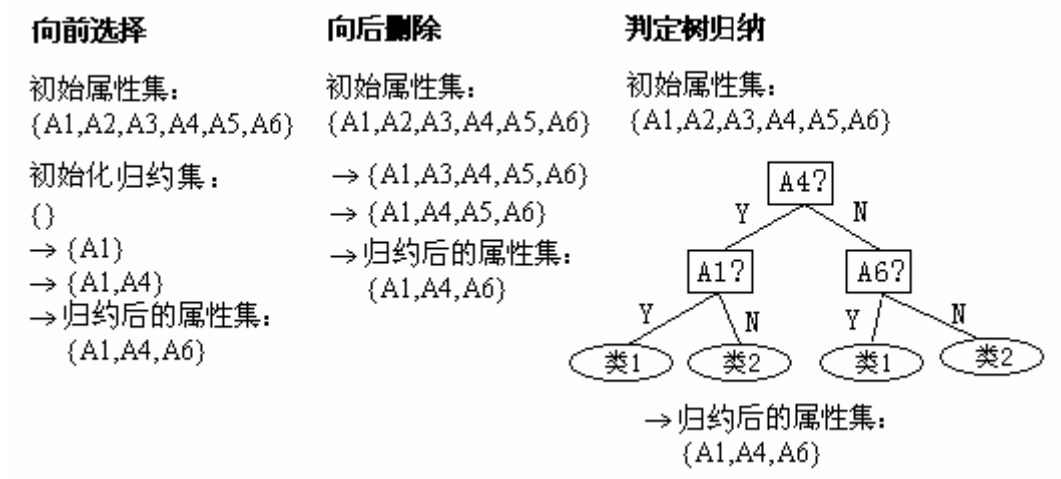


图 3.6: 属性子集选择的贪心(启发式)方法

3.4.3 数据压缩

在数据压缩时，应用数据编码或变换，以便得到原数据的归约或“压缩”表示。如果原数据可以由压缩数据重新构造而不丢失任何信息，则所使用的数据压缩技术是**无损的**。如果我们只能重新构造原数据的近似表示，则该数据压缩技术是**有损的**。有一些很好的串压缩算法。尽管它们是无损的，但它们只允许有限的数据操作。本小节我们介绍另外两种流行、有效的有损数据压缩方法：小波变换和主要成分分析。

小波变换

离散小波变换（DWT）是一种线性信号处理技术，当用于数据向量 *D* 时，将它转换成不同的数值向量**小波系数** *D'*。两个向量具有相同的长度。

“嗯”，你可能会奇怪。“如果小波变换后的数据与原数据的长度相等，这种技术如何用于数据压缩？”关键在于小波变换后的数据可以裁减。仅存放一小部分最强的小波系数，就能保留近似的压缩数据。例如，保留大于用户设定的某个阈值的小波系数，其它系数置为 0。这样，结果数据表示非常稀疏，使得如果在小波空间进行的话，利用数据稀疏特点的操作计算得非常快。该技术也能用于消除噪音，而不会平滑掉数据的主要特性，使得它们也能有效地用于数据清理。给定一组系数，使用所用的 DWT 的逆，可以构造原数据的近似。

DWT 与离散富里叶变换（DFT）有密切关系。DFT 是一种涉及正弦和余弦的信号处理技术。然而，一般地说，DWT 是一种较好的有损压缩。即，对于给定的数据向量，如果 DWT 和 DFT 保留相同数目的系数，DWT 将提供原数据更精确的近似。因此，对于等价的近似，DWT 比 DFT 需要的空间小。不象 DFT，小波空间局部性相当好，有助于保留局部细节。

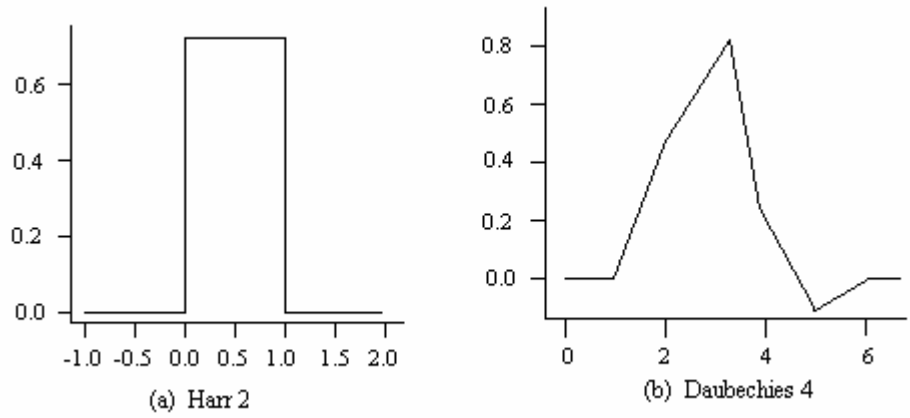


图 3.7 小波族的例子。

只有一种 DFT，但有若干族 DWT。图 3.7 给出一些小波族。流行的小波变换包括 Haar_2, Daubechies_4 和 Daubechies_6 变换。应用离散小波变换的一般过程使用一种分层金字塔算法，它在每次迭代将数据减半，导致很快的计算速度。该方法如下：

1. 输入数据向量的长度 *L* 必须是 2 的整数幂。必要时，通过在数据向量后添加 0，这一条件可以满足。
2. 每个变换涉及应用两个函数。第一个使用某种数据平滑，如求和或加权平均。第二个进行加权差分，产生数据的细节特征。
3. 两个函数作用于输入数据对，产生两个长度为 *L*/2 的数据集。一般地，它们分别代表输入数据的平滑后或低频的版本和它的高频内容。
4. 两个函数递归地作用于前面循环得到的数据集，直到结果数据集的长度为 2。
5. 由以上迭代得到的数据集中选择值，指定其为数据变换的小波系数。

等价地，可以将矩阵乘法用于输入数据，以得到小波系数。所用的矩阵依赖于给定的 DWT。矩阵必须是标准正交的。即，它们的列是单位向量并相互正交，使得矩阵的逆是它的转置。尽管受篇幅限制，这里我们不再讨论，但这种性质允许由平滑和平滑-差数据集重构数据。通过将矩阵分解成几个稀疏矩阵，对于长度为 *n* 的输入向量，“快速 DWT”算法的复杂度为 *O*(*n*)。

小波变换可以用于多维数据，如数据方。可以按以下方法做：首先将变换用于第一个维，然后第二个，如此下去。计算复杂性对于方中单元的个数是线性的。对于稀疏或倾斜数据、具有有序属性的数据，小波变换给出很好的结果。据报道，小波变换的有损压缩比当前的商业标准 JPEG 压缩好。小波变换有许多实际应用，包括手写体图象压缩、计算机视觉、时间序列数据分析和数据清理。

主要成分分析

这里，作为一种数据压缩方法，我们直观地介绍主要成分分析。详细的讨论已超出本书范围。

假定待压缩的数据由 *N* 个元组或数据向量组成，取自 *k*-维。**主要成分分析**（**PCA**，又称 Karhunen-Loeve 或 K-L 方法）搜索 *c* 个最能代表数据的 *k*-维正交向量；这里 $c \le k$。这样，原来的数据投影到一个较小的空间，导致数据压缩。PCA 可以作为一种维归约形式使用。然而，不象属性子集选择通过保留原属性集的一个子集来减少属性集的大小，PCA 通过创建一个替换的、较小的变量集“组合”属性的本质。原数据可以投影到该较小的集合中。

基本过程如下：

1. 对输入数据规范化，使得每个属性都落入相同的区间。此步确保具有较大定义域的属性不会主宰具有较小定义域的属性。

2. PCA 计算 *c* 个规范正交向量，作为规范化输入数据的基。这些是单位向量，每一个都垂直于另一个。这些向量被称为主要成分。输入数据是主要成分的线性组合。

3. 对主要成分按“意义”或强度降序排列。主要成分基本上充当数据的一组新坐标轴，提供重要的方差信息。即，对轴进行排序，使得第一个轴显示的数据方差最大，第二个显示的方差次之，如此下去。例如，图 3.8 展示对于原来映射到轴 *X1* 和 *X2* 的给定数据集的两个主要成分 *Y1* 和 *Y2*。这一信息帮助识别数据中的分组或模式。

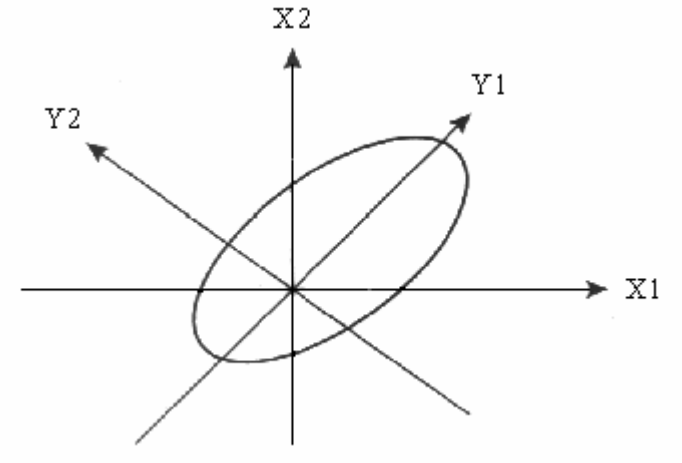


图 3.8 主要成分分析。*Y1* 和 *Y2* 是给定数据的前两个主要成分

4. 既然主要成分根据“意义”降序排列，就可以通过去掉较弱的成分（即，方差较小的那些）来压缩数据。使用最强的主要成分，应当可能重构原数据的很好的近似值。

PCA 计算花费低，可以用于有序和无序的属性，并且可以处理稀疏和倾斜数据。多于 2 维的数据可以通过将问题归约为 2 维来处理。例如，对于具有维 *item_type*, *branch* 和 *year* 的 3-D 数据方，你必须首先将它归约为 2-D 方体，如具有维 *item_type* 和 *branch* × *year* 的方体。与数据压缩的小波变换相比，PCA 能较好地处理稀疏数据，而小波变换更适合高维数据。

3.4.4 数值归约

“我们能通过选择替代的、‘较小的’数据表示形式来减少数据量吗？”数值归约技术可以用于这一目的。这些技术可以是有参的，也可以是无参的。对于有参方法，使用一个模型来评估数据，使得只需要存放参数，而不是实际数据。（局外者也可能被存放。）对数线性模型是一个例子，它估计离散的多维概率分布。存放数据归约表示的非参数的方法包括直方图、聚类和选样。

让我们来看看上面提到的每种数值归约技术。

回归和对数线性模型

回归和对数线性模型可以用来近似给定数据。在**线性回归**中，对数据建模，使之适合一条直线。例如，可以用以下公式，将随机变量 *Y*（称作响应变量）表示为另一随机变量 *X*（称为预测变量）的线性函数

$$Y = \alpha + \beta X \tag{3.6}$$

这里，假定 *Y* 的方差是常量。系数α和β（称为回归系数）分别为直线的 *Y* 轴截取和斜率。系数可以用最小平方法求得，使得分离数据的实际直线与该直线间的误差最小。**多元回归**是线性回归的扩充，响应变量是多维特征向量的线性函数。

对数线性模型近似离散的多维概率分布。基于较小的方体形成数据方的格，该方法可以用于估计具有离散属性集的基本方体中每个单元的概率。这允许由较低秩的数据方构造较高秩的数据方。

这样，对数线性对于数据压缩是有用的（因为较小秩的方体总共占用的空间小于基本方体占用的空间），对数据平滑也是有用的（因为与用基本方体进行估计相比，用较小秩的方体对单元进行估计选样变化小一些）。

回归和对数线性模型都可以用于稀疏数据，尽管它们的应用可能是受限的。虽然两种方法都可以用于倾斜数据，回归可望更好。当用于高维数据时，回归可能是计算密集的，而对数线性模型表现出很好的可规模性，可以扩展到 10 维左右。回归和对数线性模型将在 7.8 节进一步讨论。

直方图

直方图使用分箱近似数据分布，是一种流行的数据归约形式。属性 *A* 的直方图将 *A* 的数据分布划分为不相交的子集，或桶。桶安放在水平轴上，而桶的高度（和面积）是该桶所代表的值的平均频率。如果每个桶只代表单个属性值/频率对，则该桶称为单桶。通常，桶表示给定属性的一个连续区间。

例 3.4 下面的数据是 AllElectronics 通常销售的商品的单价表（按$取整）。已对数据进行了排序：1, 1, 5, 5, 5,5, 5, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30

图 3.9 使用单桶显示了这些数据的直方图。为进一步压缩数据，通常让一个桶代表给定属性的一个连续值域。在图 3.10 中每个桶代表 *price* 的一个不同的$10 区间。□

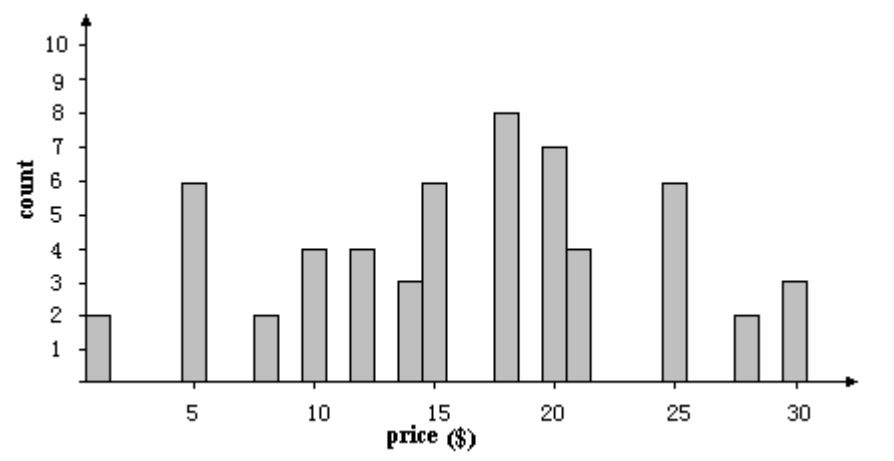


图 3.9　使用单桶的 *price* 直方图——每个桶代表一个 *price* 值/频率对

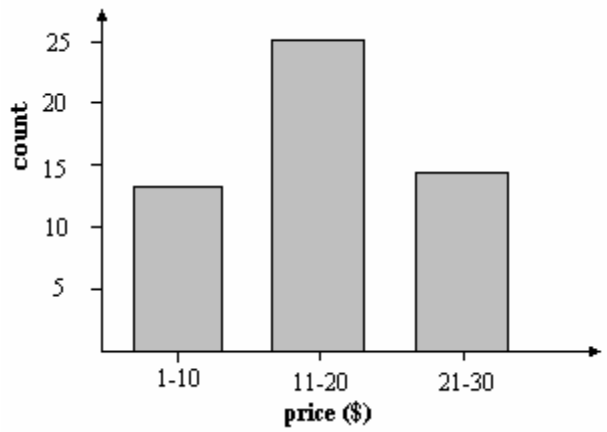


图 3.10　*price* 的直方图，值被聚集使得每个桶都有$10 宽

“如何确定桶和属性值的划分？”有一些划分规则，包括下面的一些：

- **等宽**：在等宽的直方图中，每个桶的宽度区间是一个常数（如图 3.10 中每个桶的宽度为$10）。

- **等深**（或等高）：在等深的直方图中，桶这样创建，使得每个桶的频率粗略地为常数（即，每个桶大致包含相同个数的临近样本）。
- **V-最优：**给定桶个数，如果我们考虑所有可能的直方图，V-最优直方图是具有最小偏差的直方图。直方图的偏差是每个桶代表的原数据的加权和，其中权等于桶中值的个数。
- **MaxDiff:** 在 MaxDiff 直方图中，我们考虑每对相邻值之间的差。桶的边界是具有β-1 个最大差的对；这里，β由用户指定。

V-最优和 MaxDiff 直方图看来是最精确和最实用的。对于近似稀疏和稠密数据，以及高倾斜和一致的数据，直方图是高度有效的。上面介绍的单属性直方图可以推广到多属性。多维直方图可以表现属性间的依赖。业已发现，这种直方图对于多达 5 个属性能够有效地近似数据。对于更高维，多维直方图的有效性尚需进一步研究。对于存放具有高频率的例外者，单桶是有用的。直方图将在 5.5 节进一步讨论。

聚类

聚类技术将数据元组视为对象。它将对象划分为群或聚类，使得在一个聚类中的对象“类似”，但与其它聚类中的对象“不类似”。通常，类似性基于距离，用对象在空间中的“接近”程度定义。聚类的“质量”可以用“直径”表示；而直径是一个聚类中两个任意对象的最大距离。质心距离是聚类质量的另一种度量，它定义为由聚类质心（表示“平均对象”，或聚类空间中的平均点）到每个聚类对象的平均距离。图 3.11 展示关于顾客在一个城市中位置的顾客数据 2-D 图，每个聚类的质心用“+”显示，三个数据聚类已标出。

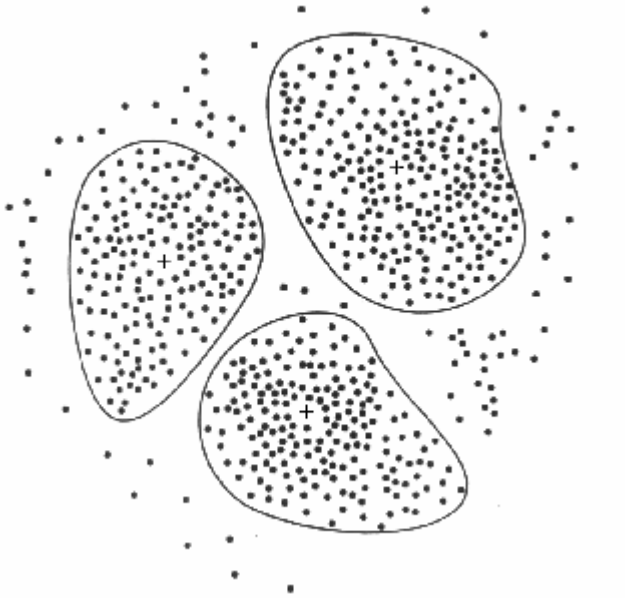


图 3.11　顾客数据的 2-D 图，展示关于顾客在一个城市中的位置；有三个聚类，每个聚类的质心用”+”标记

在数据归约时，用数据的聚类表示替换实际数据。该技术的有效性依赖于数据的性质。如果数据能够组织成不同的聚类，该技术有效得多。

在数据库系统中，**多维索引树**主要用于提供对数据的快速访问。它也能用于分层数据的归约，提供数据的多维聚类。这可以用于提供查询的近似回答。对于给定的数据集合，索引树动态地划分多维空间，其树根结点代表整个空间。通常，这种树是平衡的，由内部结点和树叶结点组成。每个父结点包含一些关键字和指向子女结点的指针，子女结点一起代表父结点代表的空间。每个树叶结点包含指向它所代表的数据元组（或实际元组）的指针。

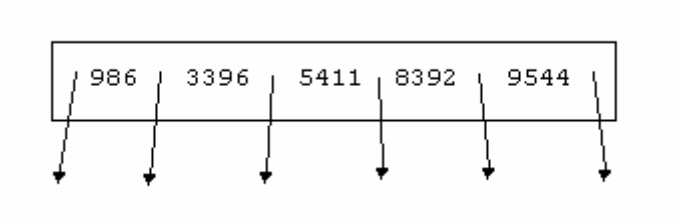


图 3.12　给定数据集的 B+树的根

这样，索引树可以在不同的清晰度或抽象层存放聚集和细节数据。它为数据集合的聚类提供了分层结构；其中，每个聚类有一个标号，存放包含在聚类中的数据。如果我们把父结点的每个子女看作一个桶，则索引树可以看作一个分层的直方图。例如，考虑图 3.10 所示 B+树的根，它具有指向数据键 986, 3396, 5411, 8392 和 9544 的指针。假定树包含 10,000 个元组，其键值由 1 到 9999。则树中的数据可以用 6 个桶的等深直方图近似，其键值分别从 1 到 985，986 到 3395，3396 到 5410，5411 到 8391，8392 到 9543，9544 到 9999。每个桶大约包含 10,000/6 个数据项。类似地，每个桶被分成较小的桶，允许在更细的层次聚集数据。作为数据清晰度的一种形式使用多维索引树依赖于每一维属性值的次序。多维索引树包括 R-树、四叉树和它们的变形。他们都非常适合处理稀疏数据和倾斜数据。

有许多定义聚类和聚类质量的度量。聚类方法将在第 8 章进一步讨论。

选样

选样可以作为一种数据归约技术使用，因为它允许用数据的较小随机样本（子集）表示大的数据集。假定大的数据集 D 包含 N 个元组。我们看看对 D 的可能选样。

- **简单选择 *n* 个样本，不回放(SRSWOR)**：由 D 的 N 个元组中抽取 n 个样本（$n < N$）；其中， D 中任何元组被抽取的概率均为 $1/N$。即，所有元组是等可能的。
- **简单选择 *n* 个样本，回放(SRSWR)**：该方法类似于 SRSWOR，不同在于当一个元组被抽取后，记录它，然后放回去。这样，一个元组被抽取后，它又被放回 D，以便它可以再次被抽取。
- **聚类选样**：如果 D 中的元组被分组放入 M 个互不相交的“聚类”，则可以得到聚类的 m 个简单随机选样；这里，$m < M$。例如，数据库中元组通常一次取一页，这样每页就可以视为一个聚类。例如，可以将 SRSWOR 用于页，得到元组的聚类样本，由此得到数据的归约表示。
- **分层选样**：如果 D 被划分成互不相交的部分，称作“层”，则通过对每一层的简单随机选样就可以得到 D 的分层选样。特别是当数据倾斜时，这可以帮助确保样本的代表性。例如，可以得到关于顾客数据的一个分层选样，其中分层对顾客的每个年龄组创建。这样，具有最少顾客数目的年龄组肯定能够表示。

这些选样如图 3.13 所示。它们代表最常用的数据归约选样形式。

采用选样进行数据归约的优点是，得到样本的花费正比例于样本的大小 n，而不是数据的大小 N。因此，选样的复杂性子线性(sublinear)于数据的大小。其它数据归约技术至少需要完全扫描 D。对于固定的样本大小，选样的复杂性仅随数据的维数 d 线性地增加；而其它技术，如使用直方图，复杂性随 d 指数增长。

用于数据归约时，选样最常用来回答聚集查询。在指定的误差范围内，可以确定（使用中心极限定理）估计一个给定的函数所需的样本大小。样本的大小 n 相对于 N 可能非常小。对于归约数据的渐进提炼，选样是一种自然选择。这样的集合可以通过简单地增加样本大小而进一步提炼。

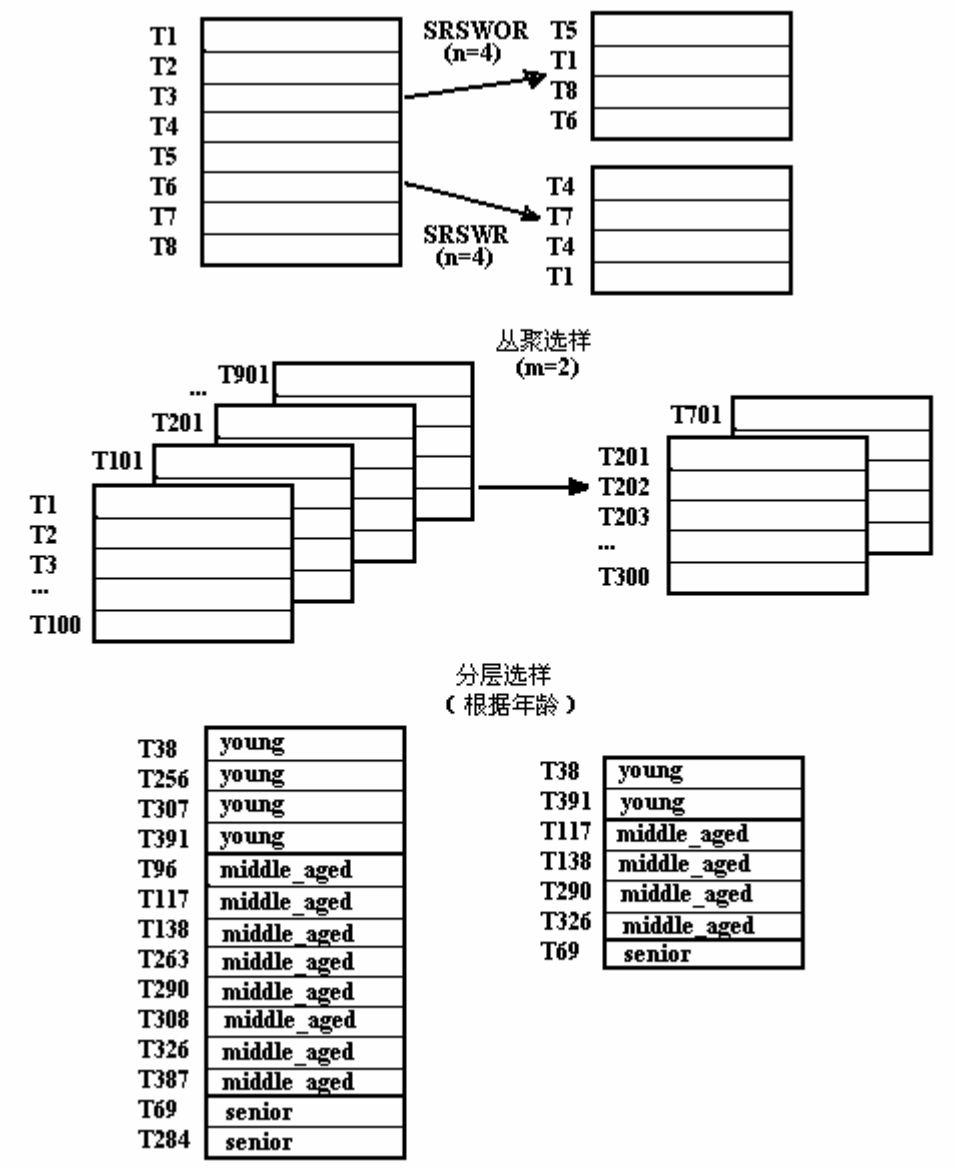


图 3.13　选样可以用于数据归约

3.5　离散化和概念分层产生

通过将属性域划分为区间，离散化技术可以用来减少给定连续属性值的个数。区间的标号可以替代实际的数据值。如果使用基于判定树的分类挖掘方法，减少属性值的数量特别有好处。通常，这种方法是递归的，大量的时间花在每一步的数据排序上。因此，待排序的不同值越少，这种方法就应当越快。许多离散化技术都可以使用，以便提供属性值的分层或多维划分——概念分层。概念分层在第 2 章引入，对于多个抽象层上的挖掘是非常有用的。

对于给定的数值属性，概念分层定义了该属性的一个离散化。通过收集并用较高层的概念（对于年龄属性，如 *young, middle-age* 和 *senior*）替换较低层的概念（如，年龄的数值值），概念分层可以用来归约数据。通过这种泛化，尽管细节丢失了，但泛化后的数据更有意义、更容易解释，并且所需的空间比原数据少。在归约的数据上进行挖掘，与在大的、未泛化的数据上挖掘相比，所需

的 I/O 操作更少，并且更有效。属性 *price* 的概念分层例子在图 3.14 给出。对于同一个属性可以定义多个概念分层，以适合不同用户的需要。

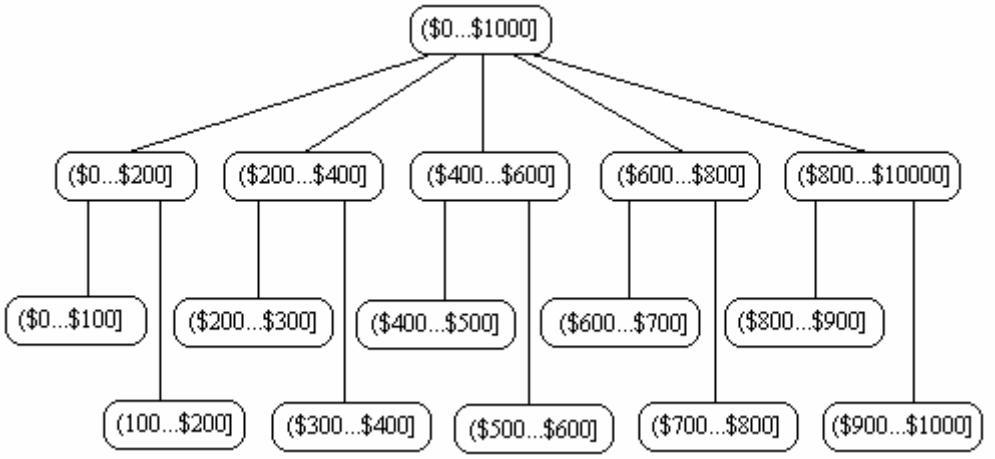


图 3.14　属性 *price* 的一个概念分层

对于用户或领域专家，人工地定义概念分层可能是一项令人乏味、耗时的任务。幸而，许多分层蕴涵在数据库模式中，并且可以在模式定义级定义。概念分层常常自动地产生，或根据数据分布的统计分析动态地加以提炼。

让我们来看看数值和分类数据的概念分层的产生。

3.5.1 数值数据的离散化和概念分层产生

对于数值属性，说明概念分层是困难的和令人乏味的，这是由于数据的可能取值范围发散和数据值的更新频繁。这种人工地说明还可能非常随意。

数值属性的概念分层可以根据数据分布分析自动地构造。我们考察五种数值概念分层产生方法：分箱、直方图分析、聚类分析、基于熵的离散化和通过“自然划分”的数据分段。

分箱

3.2.2 小节讨论了数据平滑的分箱方法。这些方法也是离散化形式。例如，通过将数据分布到箱中，并用箱中的平均值或中值替换箱中的每个值，可以将属性值离散化。就象用箱的平均值或箱的中值平滑一样。这些技术可以递归地作用于结果划分，产生概念分层。

直方图分析

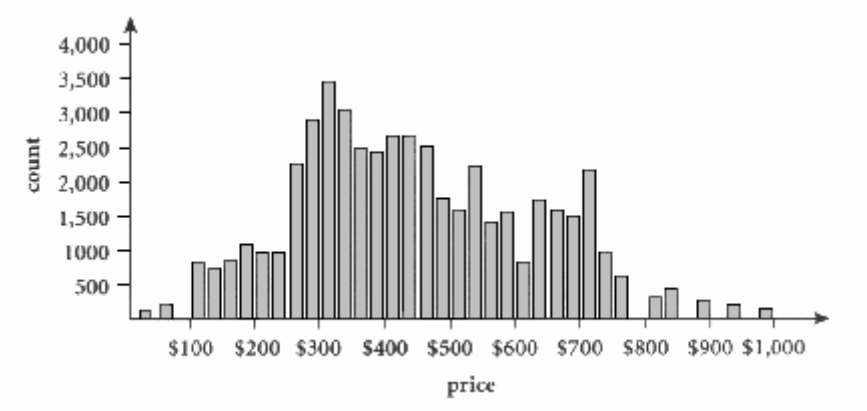


图 3.15　显示 *price* 属性的值分布的直方图

3.4.4 小节讨论的直方图也可以用于离散化。图 3.15 给出了一个直方图，显示某给定数据集 *price* 属性的数据分布。例如，频率最高的价格大约在$300-$325。可以使用划分规则定义值的范围。例如，在等宽的直方图中，将值划分成相等的部分或区间（如，($0,$100], ($100,$200],..., ($900,$1,000])。在等深的直方图中，值被划分使得每一部分包括相同个数的样本。直方图分析算法递归地用于每一部分，自动地产生多级概念分层，直到到达一个预先设定的概念层数，过程终止。也可以对每一层使用最小区间长度来控制递归过程。最小区间长度设定每层每部分的最小宽度，或每层每部分中值的最少数目。

聚类分析

聚类算法可以用来将数据划分成聚类或群。每一个聚类形成概念分层的一个结点，而所有的结点在同一概念层。每一个聚类可以进一步分成若干子聚类，形成较低的概念层。聚类也可以聚集在一起，以形成分层结构中较高的概念层。数据挖掘的聚类方法将在第 8 章讨论。

基于熵的离散化

一种基于信息的度量称作熵，可以用来递归地划分数值属性 *A* 的值，产生分层的离散化。这种离散化形成属性的数值概念分层。给定一个数据元组的集合 *S*，基于熵对 *A* 离散化的方法如下：

1. *A* 的每个值可以认为是一个潜在的区间边界或阈值 *T*。例如，*A* 的值 *v* 可以将样本 *S* 划分成分别满足条件 $A < v$ 和 $A \geq v$ 的两个子集，这样就创建了一个二元离散化。
2. 给定 *S*，所选择的阈值是这样的值，它使其后划分得到的信息增益最大。信息增益是

$$I(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2) \tag{3.6}$$

其中，S_1 和 S_2 分别对应于 *S* 中满足条件 $A < T$ 和 $A \geq T$ 的样本。对于给定的集合，它的熵函数 *Ent* 根据集合中样本的类分布来计算。例如，给定 *m* 个类，S_1 的熵是

$$Ent(S_1) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{3.7}$$

其中，p_i 是类 *i* 在 S_1 中的概率，等于 S_1 中类 *i* 的样本数除以 S_1 中的样本总数。$Ent(S_2)$的值可以类似地计算。

3. 确定阈值的过程递归地用于所得到的每个分划，直到满足某个终止条件，如

$$Ent(S) - I(S,T) > \delta \tag{3.8}$$

基于熵的离散化可以压缩数据量。与迄今为止提到的其它方法不同，基于熵的离散化使用类信息。这使得它更有可能将区间边界定义在准确位置，有助于提高分类的准确性。这里介绍的信息增益和熵也用于判定树归纳。这些度量的将在 5.3.2 和 7.3.1 小节更详细地讨论。

通过自然划分分段

尽管分箱、直方图、聚类和基于熵的离散化对于数值分层的产生是有用的，但是许多用户希望看到数值区域被划分为相对一致的、易于阅读、看上去直观或“自然”的区间。例如，更希望将年薪划分成象($50,000, $60,000]的区间，而不是象由某种复杂的聚类技术得到的($51263.98, $60872.34]那样。

3-4-5 规则可以用于将数值数据划分成相对一致、“自然的”区间。一般地，该规则根据最重要的数字上的值区域，递归地、逐层地将给定的数据区域划分为 3、4 或 5 个等长的区间。该规则如下：

- 如果一个区间在最重要的数字上包含 3、6、7 或 9 个不同的值，则将该区间划分成 3 个区间（对于 3、6 和 9，划分成 3 个等宽的区间；而对于 7，按 2-3-2 分组，划分成 3 个区间）；
- 如果它在最重要的数字上包含 2、4 或 8 个不同的值，则将区间划分成 4 个等宽的区间；

■ 如果它在最重要的数字上包含 1、5 或 10 个不同的值，则将区间划分成 5 个等宽的区间。

该规则可以递归地用于每个区间，为给定的数值属性创建概念分层。由于在数据集中可能有特别大的正值和负值，最高层分段简单地按最小和最大值可能导致扭曲的结果。例如，在资产数据集中，少数人的资产可能比其他人高几个数量级。按照最高资产值分段可能导致高度倾斜的分层。这样，顶层分段可以根据代表给定数据大多数的数据区间（例如，第 5 个百分位数到第 95 个百分位数）进行。越出顶层分段的特别高和特别低的值将用类似的方法形成单独的区间。

下面是一个自动构造数值分层的例子，解释 3-4-5 规则的使用。

例 3.5 假定 AllElectronics 所有分部 1999 年的利润覆盖了一个很宽的区间，由-$351,976.00 到$4,700,896.50。用户希望自动地产生利润的概念分层。为了改进可读性，我们使用记号(l...r]表示区间(l,r]。例如，(-$1,000,000...$0]表示由-$1,000,000（开的）到$0（闭的）的区间。

假定数据的 5%-片到 90%-片在-$159,876 和$1,838,761 之间。使用 3-4-5 规则的结果如图 3.16 所示。

1. 根据以上信息，最小和最大值分别为 *MIN* = -$351,976.00 和 *MAX* = $4,700,896.50。对于分段的顶层或第一层，要考虑的最低（第 5 个百分位数）和最高（第 95 个百分位数）值是：*LOW* = -$159,876，*HIGH* = $1,838,761。
2. 给定 *LOW* 和 *HIGH*，最重要的数字在一百万美元数字位（即，*msd*=1,000,000）。*LOW* 向下对一百万美元数字位取整，得到 *LOW*’= -$1,000,000；*HIGH* 向上对一百万美元数字位取整，得到 *HIGH*’ = +$2,000,000。

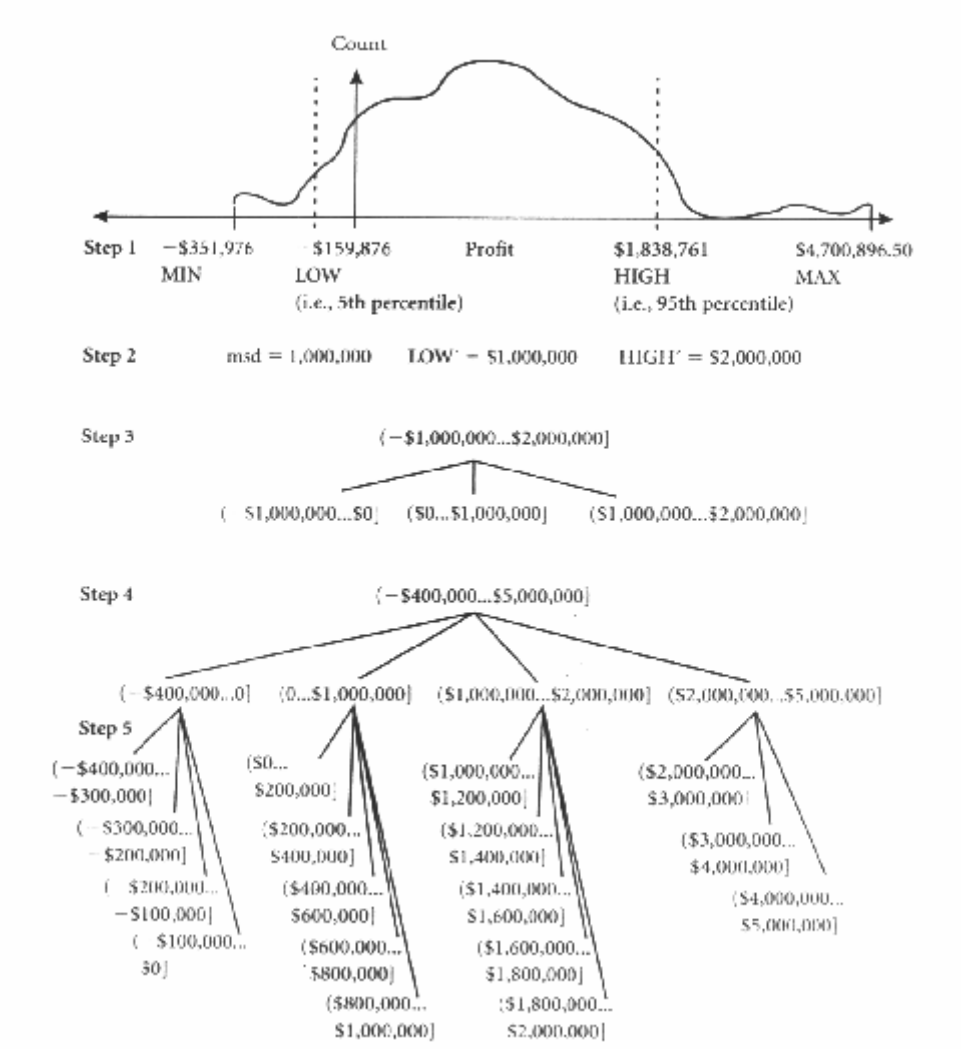


图 3.16　根据 3-4-5 规则，*profit* 概念分层的自动产生

3. 由于该区间在最重要的数字上跨越了三个值，即，(2,000,000 – (1,000,000)) /1,000,000 = 3，根据 3-4-5 规则，该区间被划分成三个等宽的区间：(-$1,000,000...$0], ($0...$1,000,000] 和 ($1,000,000...$2,000,000]。这代表分层结构的最顶层。

4. 现在，我们考察 *MIN* 和 *MAX*，看它们“适合”在第一层分划的什么地方。由于第一个区间 (-$1,000,000...$0]覆盖了 *MIN* 值（即，*LOW' < MIN*），我们可以调整该区间的左边界，使区间更小一点。*MIN* 的最重要数字在十万数字位。*MIN* 向下对十万数字位取整，得到 *MIN' =* -$400,000。因此，第一个区间被重新定义为(-$400,000...$0]。

   由于最后一个区间($1,000,000...$2,000,000]不包含 *MAX* 值，即 *MAX > HIGH'*，我们需要创建一个新的区间来覆盖它。对 *MAX* 向上对最重要数字位取整，新的区间为($2,000,000 ... $5,000,000]。因此，分层结构的最顶层包含 4 个区间：(-$400,000...$0]，($0...$1,000,000]，($1,000,000...$2,000,000]和($2,000,000...$5,000,000]。

5. 递归地，每一个区间可以根据 3-4-5 规则进一步划分，形成分层结构的下一个较低层：

   - 第一个区间(-$400,000...$0]划分成 4 个子区间：(-$400,000...-$300,000],(-$300,000...-$200,000], (-$200,000...-$100,000]和(-$100,000...$0]。
   - 第二个区间($0...$1,000,000]划分成 5 个子区间：($0...$200,000], ($200,000...$400,000], ($400,000...$600,000], ($600,000...$800,000]和($800,000...$1,000,000]。
   - 第三个区间($1,000,000...$2,000,000]划分成 5 个子区间：($1,000,000...$1,200,000], ($1,200,000...$1,400,000], ($1,400,000...$1,600,000], ($1,600,000...$1,800,000]和($1,800,000... $2,000,000]。
   - 最后一个区间($2,000,000...$5,000,000]划分成 3 个子区间：($2,000,000...$3,000,000], ($3,000,000...$4,000,000]和($4,000,000...$5,000,000]。

   类似地，如果必要的话，3-4-5 规则可以在较低的层上继续迭代。□

3.5.2　分类数据的概念分层产生

分类数据是离散数据。一个分类属性具有有限个（但可能很多）不同值，值之间无序。例子有地理位置、工作分类和商品类型。有一些典型的方法产生分类数据的概念分层。

由用户或专家在模式级显式地说明属性的部分序：通常，分类属性或维的概念分层涉及一组属性。用户或专家在模式级通过说明属性的部分序或全序，可以很容易地定义概念分层。例如，关系数据库或数据仓库的维 *location* 可能包含如下一组属性：*street, city, province_or_state* 和 *country*。可以在模式级说明一个全序，如 *street < city < province_or_state < country*，来定义分层结构。

通过显式数据分组说明分层结构的一部分：这基本上是人工地定义概念分层结构的一部分。在一个大型数据库中，通过显式的值枚举定义整个概念分层是不现实的。然而，对于一小部分中间层数据，显式指出分组是现实的。例如，在模式级说明了 *province* 和 *country* 形成一个分层后，可能想人工地添加某些中间层。如显式地定义“{Albert, Sakatchewan, Manitoba} ⊂ *prairies_Canada*”和“{British Columbia, prairies_Canada} ⊂ *Western_Canada*”。

说明属性集，但不说明它们的偏序：用户可以说明一个属性集，形成概念分层，但并不显式说明它们的偏序。然后，系统试图自动地产生属性的序，构造有意义的概念分层。

你可能会问：“没有数据语义的知识，如何找出一个任意的分类属性集的分层序？”考虑下面的事实：由于一个较高层的概念通常包含若干从属的较低层概念，定义在高概念层的属性与定义在较低概念层的属性相比，通常包含较少数目的不同值。根据这一事实，可以根据给定属性集中每个属性不同值的个数，自动地产生概念分层。具有最多不同值的属性放在分层结构的最低层。一个属性的不同值个数越少，它在所产生的概念分层结构中所处的层越高。在许多情况下，这种启发式规则都很顶用。在考察了所产生的分层之后，如果必要，局部层次交换或调整可以由用户或专家来做。

让我们看一个例子。

例 3.6 假定用户对于 AllElectronics 的维 *location* 选定了属性集：*street, country, province_or_state* 和 *city*，但没有指出属性之间的层次序。

location 的概念分层可以按如下步骤自动地产生。首先，根据每个属性的不同值个数，将属性按降序排列。其结果如下（每个属性的不同值数目在括号中）：*country*(15), *province_or_state*(365), *city*(3567), *street*(674,339)。其次，按照排好的次序，自顶向下产生分层，第一个属性在最顶层，最后一个属性在最底层。结果分层如图 3.17 所示。最后，用户可以考察所产生的分层，如果必要的话，修改它，以反映期望属性应满足的联系。在这个例子中，显然不需要修改产生的分层。□

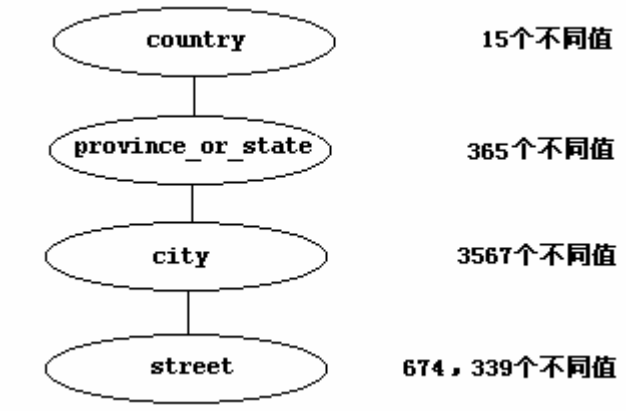


图 3.17　一个基于不同值个数的模式概念分层的自动产生

注意，不能把启发式规则推向极端，因为显然有些情况并不遵循该规则。例如，在一个数据库中，时间维可能包含 20 个不同的年，12 个不同的月，每星期 7 个不同的天。然而，这并不意味时间分层应当是“year < month < days_of_the_week”， days_of_the_week 在分层结构的最顶层。

只说明部分属性集：在定义分层时，有时用户可能不小心，或者对于分层结构中应当包含什么只有很模糊的想法。结果，用户可能在分层结构说明中只包含了相关属性的一小部分。例如，用户可能没有包含 *location* 所有分层的相关属性，而只说明了 *street* 和 *city*。为了处理这种部分说明的分层结构，重要的是在数据库模式中嵌入数据语义，使得语义密切相关的属性能够捆在一起。用这种办法，一个属性的说明可能触发整个语义密切相关的属性被“拖进”，形成一个完整的分层结构。然而，必要时，用户应当可以忽略这一特性。

例 3.7　关于 *location* 概念，假定数据库系统已将五个属性 *number, street, city, province_or_state* 和 *country* 捆绑在一起。如果用户在定义 *location* 的分层结构时只说明了属性 *city*，系统可以自动地拖进以上五个语义相关的属性，形成一个分层结构。用户可以去掉分层结构中的任何属性，如 *number* 和 *street*，让 *city* 作为该分层结构的最低概念层。□

3.6 总结

- **数据预处理**对于建立数据仓库和数据挖掘都是一个重要的问题，因为现实世界中的数据多半是不完整的、有噪音的和不一致的。数据预处理包括数据清理、数据集成、数据变换和数据归约。
- **数据清理**例程可以用于填充遗漏的值，平滑数据，找出局外者并纠正数据的不一致性。
- **数据集成**将来自不同数据源的数据整合成一致的数据存储。元数据、相关分析、数据冲突检测和语义异种性的解决都有助于数据集成。

- **数据变换**例程将数据变换成适于挖掘的形式。例如，属性数据可以规范化，使得它们可以落入小区间，如 0.0 到 1.0。
- **数据归约**技术，如数据方聚集、维归约、数据压缩、数值归约和离散化都可以用来得到数据的归约表示，而使得信息内容的损失最小。
- 数值数据的**概念分层自动产生**可能涉及诸如分箱、直方图分析、聚类分析、基于熵的离散化和根据自然划分分段。对于分类数据，概念分层可以根据定义分层的属性的不同值个数自动产生。
- 尽管已经提出了一些数据预处理的方法，数据预处理仍然是一个实际研究领域。

习题

3.1 数据的质量可以用精确性、完整性和一致性来评估。提出两种数据质量的其它尺度。

3.2 在现实世界的数据中，元组在某些属性上缺少值是常有的。描述处理该问题的各种方法。

3.3 假定用于分析的数据包含属性 *age*。数据元组中 *age* 的值如下（按递增序）：13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70

(a) 使用按箱平均值平滑对以上数据进行平滑，箱的深度为 3。解释你的步骤。评论对于给定的数据，该技术的效果。

(b) 你怎样确定数据中的局外者？

(c) 对于数据平滑，还有哪些其它方法？

3.4 讨论数据集成需要考虑的问题。

3.5 使用习题 3.3 给出的 *age* 数据，回答以下问题：

(a) 使用 *min-max* 规范化，将 *age* 值 35 转换到[0.0,1.0]区间。

(b) 使用 z-score 规范化转换 *age* 值 35，其中，*age* 的标准偏差为 12.94 年。

(c) 使用小数定标规范化转换 *age* 值 35。

(d) 指出对于给定的数据，你愿意使用哪种方法。陈述你的理由。

3.6 使用流程图解释如下属性子集选择过程

(a) 逐步向前选择

(b) 逐步向后删除

(c) 逐步向前选择和逐步向后删除的结合

3.7 使用习题 3.3 给出的 *age* 数据

(a) 画一个宽度为 10 的等宽的直方图。

(b) 为如下每种选样技术勾画例子：SRSWOR，SRSWR，聚类选样，分层选样。使用长度为 5 的样本和层“*young*”, “*middle_aged*”和“*senior*”。

3.8 对如下问题，使用伪代码或你喜欢用的程序设计语言，给出算法：

(a) 对于分类数据，基于给定模式中属性的不同值的个数，自动产生概念分层。

(b) 对于数值数据，基于等宽划分规则，自动产生概念分层。

(c) 对于数值数据，基于等深划分规则，自动产生概念分层。

文献注释

数据预处理在许多教科书中都有讨论，包括 Kennedy, Lee, Van Roy 等[KLV+98], Weiss 和 Indurkhya[WI98], 以及 Pyle[Pyl99]。预处理技术的更专门的文献在下面给出。

关于数据质量的讨论见 Redman[Red92]，Wang, Storey 和 Firth[WSF95], Wand 和 Wang [WW96]，以及 Ballou 和 Tayi[BT99]。缺少属性值的处理在 Friedman[Fri77]，Beriman, Friedman, Olshen 和 Stone[BFOS84]和 Quinlan[Qui89]中。在手写字符数据库中检测局外者或“垃圾”模式的方法由 Guyon, Matic 和 Vapnik 给出[GMV96]。分箱和数据规范化在一些教材中都有处理，包括[KLV+98, WI98, Pyl99]。包括属性构造的系统有 Langley, Simon, Bradshaw 和 Zytkow 的 BACON[LSBZ87]，Schlimmer 的 Stagger[Schl87]，Pagallo 的 FRINGE[Pag89]，以及 Bloedorn 和 Michalski 的 AQ17-DCI[BM98]。属性构造也在 Liu 和 Motoda[LM98a，LM98b]中介绍。

数据归约的一个很好的综述可以在 Barbará 等[BDF+97]中找到。数据方和它的预计算算法见[SS94, AAD+96, HRU96, RS97, ZDN97]。属性子集选择（或特征子集选择）在一些教材中都有介绍，如 Neter, Kutner, Nachtsheim 和 Wasserman[NKNW96]，Dash 和 Liu[DL97]，以及 Liu 和 Motoda[LM98a,LM98b]。结合向前选择和向后删除的方法由 Siedlecki 和 Skansky 提出[SS88]。一个属性选择的包装方法在 Kohavi 和 John 中[KJ97]。非主导属性子集选择在 Dash, Liu 和 Yao[DLY97]中介绍。关于数据压缩的小波介绍见 Press, Teukolosky, Vetterling 和 Flannery[PTVF96]。小波的一般说明可以在 Hubbard[Hub96]中找到。小波软件包的列表见 Bruce, Donoho 和 Gao[BDG96]。Daubechies 变换在 Daubechies [Dau92]中介绍。Press 等的书[PTVF96]中也包含了关于主要成分分析的单值分解的介绍。PCA 的例程包含在大部分统计软件包中，如 SAS(*http://www.sas.com/SASHome.html*)。

对回归和对数线性模型的介绍在若干教科书中可以找到，如[Jam85, Dob90, JW92, Dev95, NKNW96]。对数线性模型（在计算机科学界也称乘法模型）见 Pearl[Pea88]。直方图的一般介绍见 Barbará 等[BDF+97]和 Devore 和 Peck[DP97]。单属性直方图到多属性直方图的扩充见 Muralikrishna 和 DeWitt[MD88]，Poosala 和 Ioannidis[PI97]。关于聚类算法的引文在本书的第 8 章给出，该章讨论这一课题。多维索引结构的综述在 Caede 和 Günther[GG98]中。对于数据聚集使用多维索引树在 Aoki[Aok98]中讨论。索引树包括 R-树（Guttman[Gut84]），四叉树（Finkel 和 bentley[FB74]）和它们的变种。选样和数据挖掘的讨论见 Kivnen 和 Mannila[KM94] ，John 和 Langley[JL96]。

Kerber 的 ChiMerge[Ker92]，Liu 和 Setiono 的 Chi2[LS95]是数值属性的自动离散化，二者都使用了 χ^2 统计。Fayyad 和 Irani[FI93]使用最小描述长度原则确定数值离散化的区间数。在 Catlett[Cat91]，D-2 系统递归地二分数值特征。具有算法 C4.5 基于熵的离散化在 Quinlan[Qui93]中介绍。概念分层和由分类数据自动地产生它们在 Han 和 Fu[HF94]中描述。

第四章　数据挖掘原语、语言和系统结构

关于数据挖掘，一个流行的错误观点是：期望数据挖掘系统能够自动地挖掘出埋藏在给定的大型数据库中的所有有价值的知识，而不需要人的干预或指导。尽管有一个自动数据挖掘系统看上去是吸引人的，但在实践中，它将不可能涵盖大部分模式集。所产生的全部模式的大小很容易超过给定的数据库。让数据挖掘系统“放纵”地去发现模式，而不提供用户希望探查数据库的哪些部分，什么样的模式用户感兴趣，就是放纵数据挖掘“怪物”。所发现的大部分模式与用户的分析任务无关。此外，尽管有些模式与分析任务有关，但是它们可能太难理解，或缺乏有效性、新颖性或实用性——使得它们不令人感兴趣。这样，产生、存放或提供由给定的数据库可能发现的所有模式既不现实，又不是所期望的。

一个更现实的做法是：希望用户能够通过使用一组数据挖掘原语与数据挖掘系统通讯，以支持有效的和有成果的知识发现。这组原语包括说明数据库的部分或用户感兴趣的数据集（包括感兴趣的数据库属性或数据仓库维），要挖掘的知识类型，用于指导挖掘过程的背景知识，模式评估兴趣度量和如何显示所发现的知识。这些原语允许用户在知识发现时与数据挖掘系统通讯，从不同的角度和深度审查发现结果，并指导挖掘过程。

可以设计数据挖掘查询语言集成这些原语，允许用户自由地与数据挖掘系统交互。数据挖掘查询语言也为建立友好的图形用户界面提供了基础。此外，为了实现数据挖掘系统，一个精心设计的系统结构是非常重要的。这将有助于数据挖掘系统与其它信息系统通讯，有利于它与整个信息处理环境的集成。

本章，你将详细学习数据挖掘原语，研究根据这些原则设计数据挖掘查询语言。此外，你还将学习数据挖掘系统的系统结构。

4.1　数据挖掘原语：什么定义数据挖掘任务？

每个用户脑袋里都有一个**数据挖掘任务**，即，他想要进行的数据分析形式。一个数据挖掘任务可以用数据挖掘查询的形式说明，它是数据挖掘系统的输入。数据挖掘查询用以下原语定义，如图4.1所示。

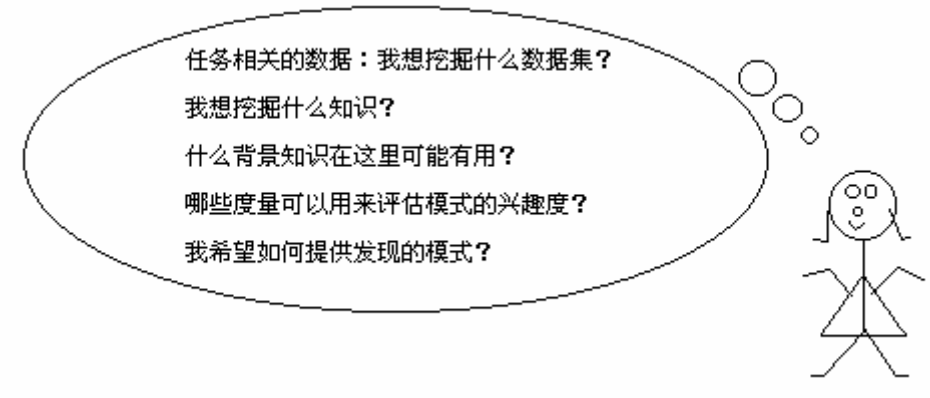


图4.1　定义数据挖掘任务或查询

- **任务相关的数据**：这是要考察的数据库部分。例如，假定你是 AllElectronics 的经理，负责美国和加拿大的销售。特殊地，你想研究加拿大顾客的购买趋势。你可能说明只提取加拿大顾客的购买数据，以及相关顾客的简要信息，而不是挖掘整个数据库。你还可以说明挖掘过程中

需要考虑的感兴趣的属性。这些属性称为**相关属性**[11]。例如，如果你只关心顾客购买的商品与其年收入和年龄之间的可能联系，则关系 *item* 的属性 *name*，关系 *customer* 的属性 *income* 和 *age* 可能被说明为挖掘任务相关的属性。

- **要挖掘什么类型的知识**：这是说明要执行的数据挖掘函数，如特征、区别、关联、分类、聚类或演变分析。例如，如果研究加拿大顾客的购买习惯，你可能选择挖掘顾客和他们喜爱买的商品之间的关联规则。
- **背景知识**：用户可以说明背景知识，或关于挖掘领域的知识。对于指导知识发现过程和评估发现的模式，这些知识是非常有用的。有多种类型的背景知识。本章，我们将注意力集中在一种称作概念分层的流行的背景知识上。概念分层是有用的，它允许在多个抽象层上挖掘数据。 其它例子包括用户对数据联系的确信。这些根据模式的非预期程度（这里，非预期的模式被认为是感兴趣的）或预期程度（这里，验证了某种用户假定的模式是有趣的）评估发现的模式。

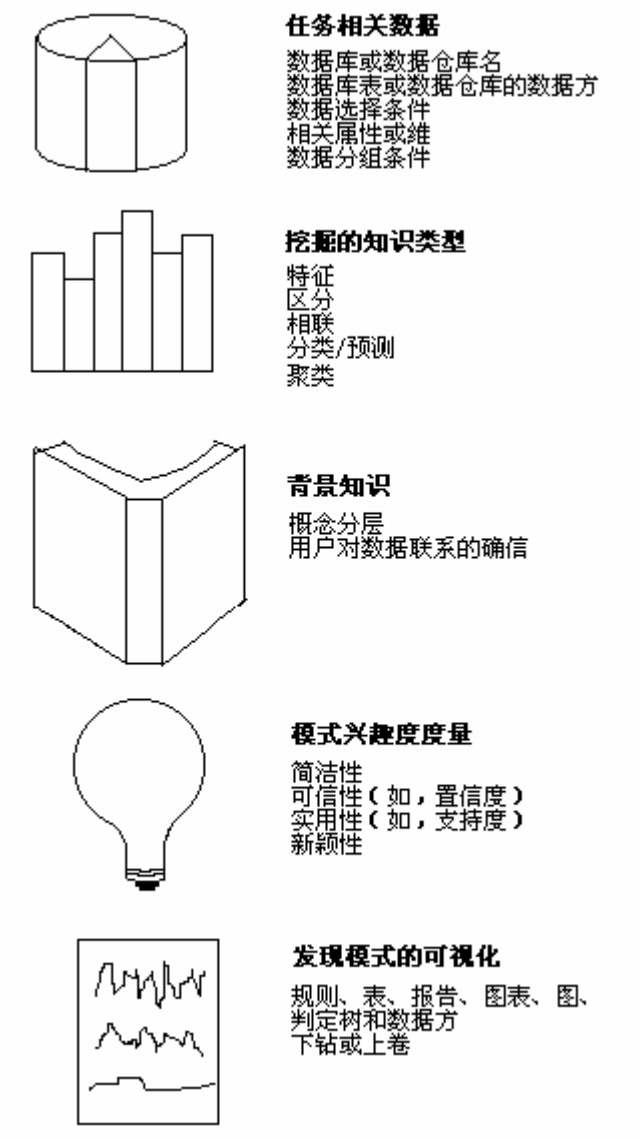


图 4.2　说明数据挖掘任务的原语

- **兴趣度度量**：这些功能用于将不感兴趣的模式从知识中分开。它们可以用于指导挖掘过程，或在挖掘之后，评估发现的模式。不同类型的知识需要不同的兴趣度度量。例如，对于关联规则，

[11] 如果挖掘在多维数据方上进行，用户可以指定**相关维**。

兴趣度度量包括**支持度**（出现规则模式的任务相关元组所占的百分比）和**置信度**（规则的蕴涵强度估计）。其支持度和置信度小于用户指定的阈值的规则被认为是不感兴趣的。

- **发现模式的提供和可视化**：这涉及发现模式的显示形式。用户可以选择不同的知识表现形式，如规则、表、图、判定树和数据方。

下面，我们仔细考察这些原语。这些原语的说明总结在图 4.2 中。

4.1.1 任务相关的数据

第一个原语是说明待挖掘的数据。通常，用户感兴趣的只是数据库的一个子集。不加区分地挖掘整个数据库是不现实的，特别是由于所产生的模式可能随数据库的大小指数地增长，使得挖掘过程效率很低。此外，所发现的许多模式与用户的兴趣无关。

在关系数据库中，任务相关的数据集可以通过涉及如选择、投影、连接和聚集等操作的关系查询来收集。这种数据提取可以认为是数据挖掘任务的一个“子任务”。数据收集过程产生一个新的数据关系，称作**初始数据关系**。初始数据关系可以根据查询中指定的条件排序或分组。在用于数据挖掘分析之前，数据可能被清理或转换（例如，在某些属性上聚集）。初始关系可以对应于，也可以不对应于数据库中的物理关系。由于虚拟关系在数据库领域称为视图，这种用于数据挖掘的任务相关的数据集称作**可挖掘的视图**。

例 4.1 如果数据挖掘任务是研究在 AllElectronics 经常购买的商品和加拿大顾客之间的关联，则任务相关的数据可以由以下信息指定：

- 所用的数据库或数据仓库的名字（如，*AllElectronics_db*），
- 包含相关数据的表或数据方的名字（如，*item, customer, purchass* 和 *item_sold*），
- 选择相关数据的条件（如，提取关于当年在加拿大进行购买的数据），
- 相关的属性或维（如，来自 *item* 表的 *name* 和 *price*，来自 *customer* 表的 *income* 和 *age*）。

此外，用户可能说明提取的数据按某些属性分组，如“**group by** date”。给出这些信息，可以用一个 SQL 查询提取任务相关的数据。□

在数据仓库中，数据通常存放在称为数据方的多维数据库中。数据方可以使用多维数组结构、关系结构或二者的结合来实现，我们在第 2 章已讨论。任务相关的数据集可以通过基于条件的过滤，数据方的切片（对于给定的属性值提取数据）或切块（提取若干片的交）来指定。

注意，在数据挖掘查询中，数据选择条件可以在比数据库或数据仓库中的数据更高的概念层上。例如，用户可以使用概念 *type =“home entertainment”*在 AllElectronics 的商品上指定选择，尽管数据库中的商品可能不是按类型存放，而是按较低层的概念，如“TV”、“CD 播放机”或“VCR”存放。在商品上的概念分层将“*home entertainment*”说明为较高层概念，由较低层概念{“TV”,“CD 播放机”,“VCR”}组成，可以用于收集任务相关的数据。

对于用户，说明相关属性或维可能是一个困难的任务。对于可能进行的探查，什么属性是感兴趣的，用户可能只有一个粗略的想法。此外，在说明待挖掘的数据时，用户可能会忽略与之有很强语义联系的数据。例如，某些商品的销售可能与诸如圣诞节或鬼节，或特定的人群等特定的事件密切相关，但这些因素可能没有包含在一般的数据分析请求中。对于这种情况，有些机制可以帮助给出任务相关数据的更精确说明。此外，搜索具有强语义联系属性的技术也可以用来加强用户说明的初始数据集。

4.1.2 要挖掘的知识的类型

说明挖掘什么类型的知识是非常重要的，因为这决定使用什么数据挖掘功能。知识类型包括概念描述（特征和区别）、关联、分类、预测、聚类和演变分析。

对于给定的数据挖掘任务，除说明要挖掘的知识类型外，用户可能想进一步说明和提供所有发现模式必须匹配的模式模板。这些模板，或**元模式**（又称**元规则**或**元查询**）可以用于指导发现过程。这些元模式的使用在以下例子中解释。

例 4.2 一个研究 AllElectronics 的顾客购买习惯的用户可能选择挖掘如下形式的关联规则

$$P(X:customer,W)\wedge Q(X,Y)\Rightarrow buys(X,Z)$$

其中，X是关系 *customer* 的关键字；P和 Q是**谓词变量**，它们可以被例示为作为任务相关数据的一部分说明的相关属性或维；而 W，Y和 Z是**对象变量**，它们可以在关于顾客 X的谓词上取值。

关联规则的搜索限于匹配给定的元规则的那些，如

$$age(X,"30...39")\wedge income(X,"40K...49K")\Rightarrow buys(X,"VCR") \quad [2.2\%,60\%] \quad (4.1)$$

和

$$occupation(X,"student")\wedge age(X,"20...29")\Rightarrow buys(X,"computer") \quad [1.4\%,70\%] \quad (4.2)$$

前一个规则是说 30 多岁的顾客，其年收入在 40K 和 49K 之间，多半（置信度 60%）会买 VCR，这种情况占事务总数的 2.2%。后一个规则是说 20 多岁的学生多半（置信度 70%）会买计算机，这种情况占事务总数的 1.4%。□

4.1.3 背景知识：概念分层

背景知识是关于挖掘领域的知识，它们在发现过程中是非常有用的。本小节，我们将我们的注意力放在一种简单但功能很强，称作概念分层的背景知识上。概念分层允许在多个抽象层上发现知识。

正如第 2 章介绍的，**概念分层**定义了一组由低层概念集到高层概念集的映射。一个关于 *location* 维的概念分层如图 4.3 所示，将较低层的概念（即，城市）映射到较高层更一般的概念（即，国家）。

注意，概念分层结构以组织成树的**结点**集表示，其中每个结点本身代表一个概念。一个特殊的结点 all 作为树根，它表示给定维的最一般的值。如果不显式给出，它是蕴涵的。该概念分层结构由 4 层组成。为方便计，概念分层结构中的层自顶向下编号，结点 all 为 0 层。在我们的例子中，层 1 表示概念 *country*，而层 2 和 3 分别表示概念 *province_or_state* 和 *city*。概念分层的树叶对应于维的原始数据值（**原始层数据**）。这些是给定属性或维的最细节的值或概念。尽管概念分层结构通常用树形分类的形式表示，但它们也可以形成一般的格或偏序。

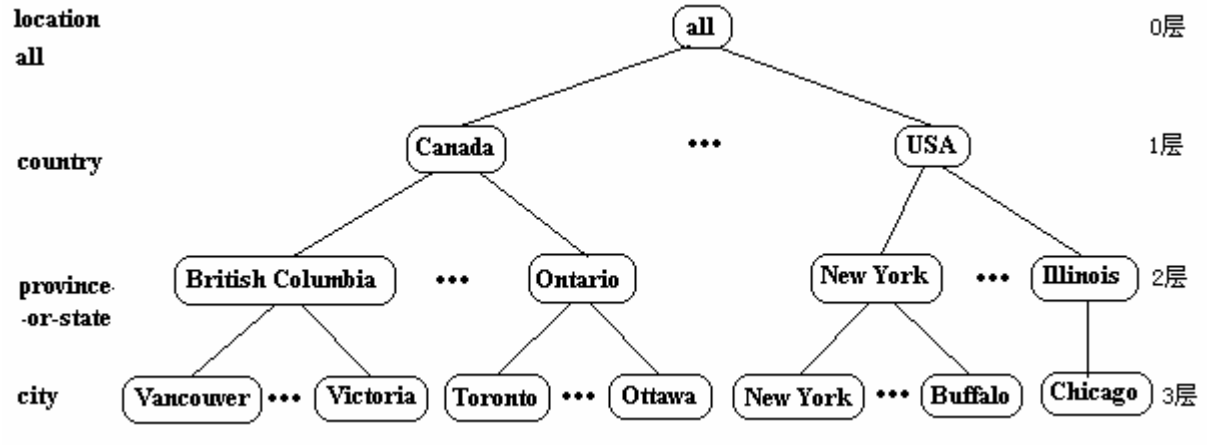


图 4.3 维 *location* 的一个概念分层

概念分层是一种有用的背景知识形式，它使得原始数据可以在较高的、一般化的抽象层上进行处理。数据的泛化或**上卷**可以通过用较高层概念（如 *location* 的国家，*age* 的诸如“20...39”，“40...59”和“60+” 这样的区间）替换较低层的概念（如 *location* 的城市，*age* 的数值值）来实现。这使得用户可以在更有意义、更明显的抽象层观察数据，使得发现的模式更易于理解。泛化的另一个优点是压缩数据。与在大的、未压缩的数据上挖掘相比，在压缩的数据集上挖掘需要较少的 I/O 操作，并将更有效。

如果结果数据过于一般化，概念分层也允许特化或**下钻**，概念值用较低层的概念替代。使用上卷和下钻，用户可以用不同的视图来观察数据，洞察隐藏的数据联系。

概念分层结构可以由系统用户、领域专家或知识工程师提供。通常，这些映射是面向特定数据或应用的。正如我们在下面将看到的，许多概念分层结构蕴涵在数据库模式中。此外，概念分层结构通常可以自动地发现，或根据数据分布的统计分析动态地提炼。概念分层结构的自动产生已在第3章详细讨论。

对于给定的属性或维，根据不同用户的观点，可能有多个概念分层结构。例如，假定AllElectronics的地区销售经理想要研究不同地方顾客的购买习惯，对于这样的挖掘任务，图4.3关于*location*的概念分层结构应当是有用的。然而，市场部经理可能更希望*location*按语言组织，以利于商业广告的分发。

概念分层有4种类型。第2章介绍了最常用的类型——模式分层和集合分组分层，这里我们将简略回顾。此外，我们还研究操作导出的分层和基于规则的分层。

模式分层：模式分层（或更严格地，模式定义的分层）是数据库模式属性间的全序或偏序。模式分层可以形式地表示属性间的语义联系。通常，一个模式分层指定了数据仓库的一个维。

例4.3 给定关系模式*address*，包含属性*street, city, province_or_state*和*country*，我们可以用如下全序定义*location*模式分层结构：

$$street < city < province\_or\_state < country$$

这意味*street*的概念层低于*city*，*city*低于*province_or_state*，而*province_or_state*低于*country*。模式分层提供元数据（即，关于数据的数据）信息。使用全序或偏序的说明比列出所有街道、省或州、国家的等价定义要简明得多。□

集合分组分层：集合分组分层将给定属性或维的值组织成常量组或区间值。组之间可以定义全序或偏序。当两种类型的分层结构结合时，集合分组分层可以用于精炼或丰富模式定义的分层。通常，集合分组分层用于定义对象联系的小集合。

例4.4 属性*age*的集合分组分层结构可以用范围来指定，如下所示

$$\{young, middle\_aged, senior\} \subset \mathbf{all}(age)$$
$$\{20...39\} \subset young$$
$$\{40...59\} \subset middle\_aged$$
$$\{60...89\} \subset senior$$

注意，正如3.5节解释的，类似的说明也可以自动产生。□

操作导出的分层：操作导出的分层是根据用户、专家或数据挖掘系统说明的操作分层。操作可能包括信息编码串的解码，由复杂数据对象提取信息和数据聚类。

例4.5 一个*e-mail*地址或*WWW*的*URL*可能包含涉及部门、学校（或公司）和国家的层次信息。可以使用解码操作来提取信息，形成概念分层。

例如，*e-mail*地址“*dmbook@cs.sfu.ca*”给出部分序“*login-name < department < university < country*”，形成了*e-mail*地址的一个概念分层。类似地，可以对*URL*地址“*http://www.cs.sfu.ca/research/DB/DBMiner*”解码，提供一个形成*URL*的概念分层的基。□

基于规则的分层：基于规则的分层是指整个概念分层或它的一部分由一组规则定义，并且根据当前数据库数据和规则定义动态地计算。

例4.6 下面的规则可以用于将AllElectronics的商品分类为*low_profit_margin*, *medium_profit_margin*和*high_profit_margin*。其中，商品*X*的价格差定义为*X*的销售价格和实际价格的差。价格差小于$50的商品定义为*low_profit_margin*商品，获利$50和$250之间的商品定义为*medium_profit_margin*商品，而获利多于$250的商品定义为*high_profit_margin*商品。

$$low\_profit\_margin(X) \Leftarrow price(X, P1) \wedge cost(X, P2) \wedge ((P1-P2) < \$50)$$
$$medium\_profit\_margin(X) \Leftarrow price(X, P1) \wedge cost(X, P2) \wedge ((P1-P2) > \$50) \wedge ((P1-P2) \leq \$250)$$

$high\_profit\_margin(X) \Leftarrow price(X,P1) \wedge cost(X,P2) \wedge ((P1-P2) > \$250)$ □

使用概念分层进行数据挖掘在本书的剩余章节介绍。

4.1.4 兴趣度度量

尽管任务相关的数据和要挖掘的知识类型(例如，特征、关联等)的说明可以大幅度减少产生规则的数量，数据挖掘过程仍然可能产生大量模式。通常，这些模式中只有一小部分是特定用户感兴趣的。这样，用户需要进一步限制挖掘过程产生的不感兴趣的模式数量。这可以通过设定兴趣度度量来实现。兴趣度度量评估模式的简洁性、确定性、实用性和新颖性。

本小节，我们研究模式兴趣度的客观度量。这种客观度量基于模式的结构和统计。一般地，每一种度量都有一个可以由用户控制的阈值。不满足阈值的规则被认为是不感兴趣的，因而不作为知识向用户提供。

简洁性：模式兴趣度的一个重要因素是对于人的理解，模式的总体简洁性。模式简洁性的客观度量可以看作模式结构的函数，用模式的二进位位数，或属性数，或模式中出现的操作符数来定义。例如，一个规则的结构越复杂，它就越难解释，从而就可能对它没多少兴趣。

例如，**规则长度**是一种简洁性的度量。对于用合取范式（即，合取谓词的集合）表达的规则，规则的长度简单地定义为规则中合取符的个数。关联、区别或分类规则的长度超过用户定义的阈值时，被认为是不感兴趣的。对于以判定树表达的模式，简洁性可以是树叶或树结点的个数的函数。

确定性：每个发现的模式都应当有一个表示其有效性或“值得信赖性”的确定性度量。对于形如“$A \Rightarrow B$”的关联规则，其确定性度量是**置信度**。给定一个任务相关的数据元组集合（或事务数据库事务的集合），“$A \Rightarrow B$”的置信度定义为：

$$confidence(A \Rightarrow B) = \frac{包含A和B的元组数}{包含A的元组数} \quad (4.3)$$

例 4.7 假定任务相关数据由 AllElectronics 的计算机部的事务数组成。一个置信度为 85%的关联规则

$$buys(X,"computer") \Rightarrow buys(X,"software") \quad (4.4)$$

意味买计算机的顾客 85%也买软件。□

置信度为 100%或 1 意味在数据分析时，该规则总是正确的。这种规则称为**准确的**。

对于分类规则，置信度称为**可靠性**或**准确性**。分类规则提出了一个模型，将目标类（如，*bigSpenders*）的对象或元组与对比类（如，*budgetSpenders*）的对象相区别。低可靠性表明不正确的分类，对比类的许多对象也在目标类中。规则的可靠性也称为**规则的强度**，**规则的质量**，**确定性因子**和**区分权**。

实用性：一个模式的潜在的有用性是定义其兴趣度的一个重要因素。它可以用一个实用性函数（如**支持度**）来评估。关联模式的支持度是模式为真的任务相关的元组（或事务）所占的百分比。对于形如“$A \Rightarrow B$”的关联规则，支持度定义为

$$support(A \Rightarrow B) = \frac{包含A和B的元组数}{元组总数} \quad (4.5)$$

例 4.8 假定任务相关数据由 AllElectronics 的计算机部的事务数组成。一个支持度为 30%的关联规则(4.4)意味计算机部的所有顾客的 30%同时购买了计算机和软件。□

同时满足用户定义的最小置信度阈值和最小支持度阈值的关联规则称为**强关联规则**，并认为是有趣的。具有较低支持度的规则多半是提供噪音，少见或例外的情况。

支持度定义的分子通常称作规则**计数**。我们常常显示该值而不是支持度。支持度容易由它导出。

特征和区分描述基本上是泛化元组。其代表的元组数少于整个任务相关元组数的 Y%的泛化元组都被视为噪音。因此，这样的元组不向用户提供。Y 值称为**噪音阈值**。

新颖性：新颖的模式是那些提供信息或提高给定模式集性能的模式。例如，一个数据例外可以认为是新颖的，它不同于根据统计模型和用户的信念所期望的模式。检测新颖性的另一策略是删除冗余模式。如果发现的规则被已在知识库中或导出的规则集中的另一规则所蕴涵，则两个规则都要重新审查，以便去掉潜在的冗余。

使用概念分层挖掘可能导致大量冗余规则。例如，假定下列关联规则使用图 4.3 关于 *location* 的概念分层，由 AllElectronics 的数据库中挖掘出：

$$location(X,"Canada") \Rightarrow buys(X,"SONY\_TV") \quad [8\%, 70\%] \quad (4.6)$$

$$location(X,"Montreal") \Rightarrow buys(X,"SONY\_TV") \quad [2\%, 71\%] \quad (4.7)$$

假定规则(4.6)具有 8%的支持度和 70%的置信度。可以预料规则(4.7)也大约有 70%的置信度，因为代表的 Montreal 所有数据对象也是 Canada 的数据对象。规则(4.6)比规则(4.7)更一般，因此我们预料前一个规则比后一个规则更常出现。结果，两个规则不应当具有相同的支持度。假定的销售大约有四分之一来自 Montreal。我们预料涉及 Montreal 的规则的支持度是涉及 Canada 的规则的支持度的四分之一。换一句话说，我们预料规则(4.7)的支持度为 $8\% \times \frac{1}{4} = 2\%$ 。如果规则(4.7)的实际置信度和支持度是可预料的，则它应当是冗余的，因为它不提供附加的信息，并且一般性不如规则(4.6)。这些思想在第 6 章关于关联规则挖掘时进一步讨论。

数据挖掘系统应当允许用户自由地、交互地说明、测试和修改兴趣度度量和它们对应的阈值。除了以上我们研究的基本度量之外，还有许多其它客观度量。除了客观的统计度量之外，主观度量同样存在。主观度量考虑用户对数据间联系的信赖。兴趣度度量更详尽的讨论将在贯穿本书。

规则

age(X,"young") and income(X,"high") => class(X,"A")
age(X,"young") and income(X,"low") => class(X,"B")
age(X,"old") => class(X,"C")

表

age	income	class	count
young	high	A	1,402
young	low	B	1,038
old	high	C	786
old	low	C	1,374

交叉表

	income		class		
age	high	low	A	B	C
young	1,402	1,038	1,402	1,038	0
old	786	1,374	0	0	2,160
count	2,188	2,412	1,402	1,038	2,160

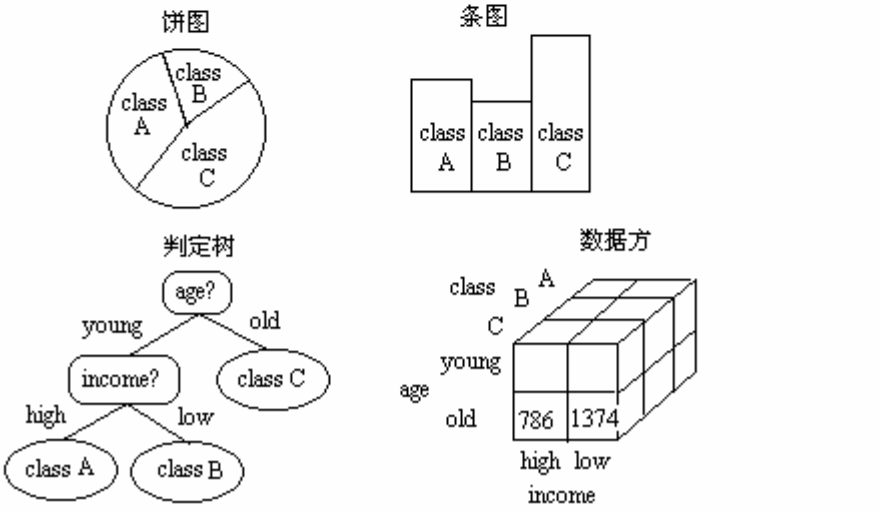


图 4.4　发现的模式的表示和可视化的各种形式

4.1.5　发现模式的提供和可视化

“如何‘观看’发现的模式？”数据挖掘要成为有效的，数据挖掘系统就应当能够以多种形式显示所发现的模式，如规则、表、交叉表、饼图或条图、判定树、数据方或其它可视化表示（图 4.4）。允许发现的模式以多种形式表示可以帮助不同背景的用户识别有趣的模式，并与系统交互或指导进一步的发现。用户应当能够指定用于显示发现模式的表示形式。

概念分层的使用在帮助用户观察发现的模式中起重要作用。使用概念分层挖掘允许发现的模式在高层概念表示，这可能比用原始数据概念表达的规则（如，函数或多值依赖规则，或整体性限制）更容易理解。此外，数据挖掘系统应当能够利用概念分层实现下钻、上卷操作，使得用户可以在多个抽象级审视发现的模式。转轴（旋转）、切片和切块操作也能帮助用户从不同视角观察泛化数据和知识。这些操作已在第 2 章详细讨论。一个数据挖掘系统应当对任意维，以及每个维的特定值提供这些交互操作。

对于特定的知识类，某些表示形式可能比其它的更合适。例如，对于特征描述，泛化规则和对应的交叉图或饼图/条图是好的表示形式；而对于分类，判定树是通常的选择。诸如 4.1.4 小节介绍的兴趣度度量可以显示在每个发现的模式上，帮助用户识别提供有用知识的那些模式。

4.2 一种数据挖掘查询语言

“为什么有一个数据挖掘查询语言很重要？”回忆一下，数据挖掘系统的期望特点是能够支持特别的和交互的数据挖掘，以利于灵活和有效的知识发现。可以设计数据挖掘查询语言，来支持这种特点。

通过观察关系数据库系统的历史，也可以明白设计一个好的数据挖掘查询语言的重要性。关系数据库系统已经主宰数据库市场几十年。关系查询语言的标准化始于关系数据库开发的早期阶段，广泛认为它对关系数据库的成功起了重要作用。尽管每个商品化的关系数据库系统都有自己图形用户界面，但是每个界面下面的核都是标准关系查询语言。关系查询语言的标准化为关系数据库系统的发展和进化提供了基础。它促进了信息交换和技术转换，推动了关系数据库技术的商品化和广泛接受。数据库系统最近的标准化活动，如涉及 SQL-3 的工作，进一步说明具有一种标准的数据库语言对于数据库系统的成功开发和商品化的重要性。因此，具有一个好的数据挖掘查询语言将有助于数据挖掘系统平台开发标准化。

设计一个易理解的数据挖掘语言是一个挑战，因为数据挖掘任务涉及面宽。由数据特征到挖掘关联规则、数据分类和进化分析，每种任务都有不同的需求。有效的数据挖掘语言的设计需要深入理解各种数据挖掘任务的能力、限制和潜在机制。

如何设计数据挖掘查询语言？在本章的前面，我们考虑了用数据挖掘查询形式的定义数据挖掘任务的原语。这些原语说明：

- 待挖掘的相关数据集
- 要挖掘的数据类型
- 用于发现过程的背景知识
- 模式评估的兴趣度度量和阈值
- 可视化发现模式的期望表示

基于这些原语，我们设计一种数据挖掘查询语言 DMQL。DMQL 是 Data Mining Query Language（数据挖掘查询语言）的缩写。DMQL 允许在多个抽象层上，由关系数据库和数据仓库进行多种类型知识的特殊挖掘[12]。

该语言采用类似于 SQL 的语法，因此它易于和关系查询语言 SQL 集成在一起。DMQL 的语法采用扩充的 BNF 文法定义，其中“[ ]”表示 0 次或 1 次出现，“{ }”表示 0 次或多次出现，黑体字表示关键词。

由 4.2.1 到 4.2.5 小节介绍每种数据挖掘原语的 DMQL 语法。在 4.2.6 小节，我们用该语法给出一个数据挖掘查询的例子。语言的顶层概述如图 4.5 所示。

[12] 定义数据仓库和数据集市的 DMOL 语法已在第 2 章给出。

```
<DMQL>                    ::= <DMQL_Statement>;{<DMQL_Statement>}
<DMQL_Statement>  ::= <Data_Mining_Statement>
                    | <Concept_Hierarchy_Definition_Statement>
                    | <Visualization_and_Presentation>
<Data_Mining_Statement> ::=
        use database <database_name> | use data warehouse
ta_warehouse_name>
        {use hierachy <hierachy_name> for <attribute_or_dimension>}
        <Mine_Knowledge_Specification>
        in relevance to < attribute_or_dimension_list>
        from <relation(s)/cube>
        [where <condition>]
        [order by <order_list>]
        [group by <grouping_list>]
        having <condition>]
        {with [<intetest_measure_name>] threshold = < threshold_value>
        [for <attribute(s)>]}
<Mine_Knowledge_Specification> ::=<Mine_Char> | <Mine_Discr> | <Mine_ Assoc>
                            | <Mine_ Class >
<Mine_Char>   ::=  mine characteristics [as <pattern_name>]
              analyze <measure(s)>
<Mine_Discr>  ::=  mine comparison [as <pattern_name>]
              for <target_class> where <target_condition>
              {versus <contract_class_i> where <contract_cndition_i>}
              analyze <measure(s)>
<Mine_Assoc>  ::=  mine associations [as <pattern_name>]
              [matching <metapattern>]
<Mine_Class>   ::=  mine classfication [as <pattern_name>]
              analyze <classfying_attribute_or_dimension>
<Concept_Hierarchy_Definition_Statement>  ::=
              define hierarchy < hierarchy_name>
              [for <attribute_or_dimension>]
              on <relation_or_cube_or_ hierarchy>
              as < hierarchy_description>
              [where <condition>]
<Visualization_and_Presentation>  ::=
              display as  <result_form> | {<Multilevel_Manipulation>}
<Multilevel_Manipulation> ::=  roll up on <attribute_or_dimension>
              |       drill down on <attribute_or_dimension>
              |       add <attribute_or_dimension>
              |       drop <attribute_or_dimension>
```

图 4.5：数据挖掘查询语言 DMQL 顶层语法

4.2.1 任务相关数据说明的语法

定义数据挖掘任务的第一步是说明任务相关的数据。这涉及说明包含相关数据的数据库和表或数据仓库，选择相关数据的条件，相关属性或维和关于所提取的数据的排序和分组的指令。DMQL 提供了一些子句来说明这些信息，如下所述

- **use database** <database_name>或 **use data warehouse** <data_warehouse_name>：**use** 子句将数据挖掘任务指向说明的数据库或数据仓库。
- **from** <relation(s)/cub(s)> [**where** <condition>]：**from** 和 **where** 子句分别指定所涉及的表或数据方和定义提取数据的条件。
- **in relevance to** < attribute_or_dimension _list>：该子句列出要探查的属性和维。
- **order by** <order_list>：**order by** 子句说明任务相关的数据排序的次序。
- **group by** <grouping_list>：**group by** 子句说明数据分组标准。
- **having** <condition>：**having** 子句说明相关数据分组条件。

这些子句形成一个 SQL 查询，收集任务相关的数据。

例 4.9 本例展示如何用 DMQL 说明例 4.1 描述的任务相关的数据。例 4.1 是挖掘由加拿大顾客在 AllElectronics 经常购买的商品之间的关联规则，涉及顾客的 *income* 和 *age*。此外，用户指出他想将数据按日期分组。这些数据由关系数据库提取。

```
use database AllElectronics_db
in relevance to I.name,I.price,C.income,C.age
from customer C, item I, purchases P, items_sold S
where I.item_ID = S.itme_ID and S.trans_ID = P_trans_ID and P.cust_ID = C.cust_ID
      and C.country = “Canada”
group by P.date
```

□

4.2.2 说明挖掘知识类型的语法

<Mine_Knowledge_Specification>语句用于说明挖掘知识的类型。换一句话说，它指定用于执行的挖掘函数。下面定义的语法用于特征、区分、关联和分类。

特征

```
<Mine_Knowledge_Specification>::=
    mine characteristics [as <pattern_name>]
    analyze <measure(s)>
```

这说明挖掘特征描述。当用于特征时，**analyze** 子句指定聚集度量，如 count, sum 或 count%（百分比计数，即指定的特征在相关数据元组中的百分比）。这些度量将对每个找到的数据特征进行计算。

例 4.10 下面说明挖掘的知识类型是反映顾客购买习惯的特征描述。对于每一个特征，显示满足特征的任务相关元组的百分比。

```
mine characteristics as customerPurchasing
analyze                                                    count%
```

□

区分

```
<Mine_Knowledge_Specification>::=
    mine comparison [as <pattern_name>]
    for <target_class> where <target_condition>
    {versus <contract_class_i> where <contract_cndition_i>}
    analyze <measure(s)>
```

这说明挖掘区分描述。区分将给定的目标类的对象与一个或多个对比类的对象进行比较。因此，这类知识也称为比较。与特征一样，**analyze** 子句指定聚集度量，如 count, sum 或 count%，将对每个描述进行计算和显示。

例 4.11 用户可以定义顾客类，然后挖掘每类的区分。例如，用户可以定义 *bigSpenders* 为购买商品平均价格$100 或更多的顾客，而 *budgetSpenders* 为购买商品平均价格不足$100 的顾客。挖掘每类顾客的区分描述可以用如下 DMQL 说明，其中，*I* 表示 *item* 关系。满足每个区分的任务相关的元组将被显示。

```
mine comparision as purchaseGroups
for bigSpenders where avg(I.price)≥$100
versus budgetSpenders where avg(I.price)<$100
analyze count
```

□

关联

```
<Mine_Knowledge_Specification>::=
    mine associations [as <pattern_name>]
    [matching <metapattern>]
```

这说明关联模式的挖掘。在说明关联挖掘时，用户可以选用 **matching** 子句，提供模板（又称元模式或元规则）。元模式可以用来将发现集中于与给定元模式匹配的模式，从而强化了对挖掘任务的句法限制。除提供句法限制外，元模式提供了用户有兴趣探查的数据束或假定。具有元模式的挖掘，或**元模式指导的挖掘**，允许特定挖掘的更多灵活性。尽管元模式可以用于其它形式知识的挖掘，但是它们对关联规则的挖掘最有用，因为潜在的关联规则数目太大。

例 4.12 例 4.2 的元模式可以指定用来下面描述顾客购买习惯的关联规则挖掘.

```
mine association as buyingHabits
matching P(X:customer,W) ∧ Q(X,Y) ⇒ buys(X,Z)
```

其中，*X* 是关系 *customer* 的关键字；*P* 和 *Q* 是谓词变量，它们可以被例示作为任务相关数据说明的相关属性或维；*W*、*Y* 和 *Z* 是对象变量，它们可以分别在对应的顾客 *X* 的谓词上取值。□

分类

```
<Mine_Knowledge_Specification>::=
    mine classfication [as <pattern_name>]
    analyze <classfying_attribute_or_dimension>
```

这说明挖掘数据分类模式。**analyze** 子句说明根据<classfying_attribute_or_dimension>的值进行分类。对于分类属性或维，每个值通常代表一个类（对于属性 *credit_rating*，如“*low-risk*”，“*medium-risk*”，“*high-risk*”等）。对于数值属性或维，每个类可以用一个值区间定义（对于 *age*，如“20-39”，“40-59”，“60-89”）。分类提供了一个简明的框架，它最好地描述了每个类并将它们与其它类相区别。

例 4.13 为挖掘顾客的信用等级模式，可以使用以下 DMQL 说明。这里，信用等级由属性 *credit_rating* 确定。

```
mine classfication as classfyCustomerCreditRating
analyze                                                    credit_rating
```

□

除上述类型外，数据挖掘查询语言也应当允许说明挖掘其它类型的知识。包括数据聚类、演变规则或序列模式和偏差的挖掘。

4.2.3 概念分层说明的语法

概念分层允许在多个抽象层上挖掘知识。为适应用户看待数据的不同角度，每个属性或维可能有多个概念分层。例如，有的用户想按省或州组织分店地址，而另一些用户可能想根据所用的语言组织。在这种情况下，用户可以使用如下语句指出他要用哪个概念分层

use hierachy <hierachy> **for** <attribute_or_dimension>

否则，将使用属性或维的系统设定的概念分层。

“我们如何用 DMQL 定义概念分层？”在 4.1.3 小节，我们研究了四种类型概念分层。让我们看一些定义概念分层的例子。

例 4.13 前面，我们用全序 *street* < *city* < *province_or_state* < *country* 为 *address* 定义了一个模式分层。这可以用数据挖掘查询语言定义为

define hierarchy location_hierarchy **on** address **as** [street,city,province_or_state,country]

属性列出的次序是重要的。事实上，它定义了一个全序，指出 *street* 比 *city* 低一个概念层，而 *city* 比 *province_or_state* 低一个概念层，如此下去。□

例 4.15 一个**集合分组分层的定义。**对于例 4.4 中 *age* 的集合分组分层可以用值区间定义如下。其中，为方便计，最一般的概念 **all** 放在分层结构的根上（即，在第 0 层）。

```
define hierarchy age_hierarchy for age on customer as
    level1: {young,middle_aged,sinior} < level0: all
    level2: {20...39} < level1: young
    level2: {40...59} < level1: middle_aged
    level2: {60...89} < level1: senior
```

记号“...”表示给定区间内所有可能的值。例如，“{20...39}”代表包括端点在内 20 到 39 之间的所有整数。区间也可以用实数作为端点。对应的概念分层如图 4.6 所示。□

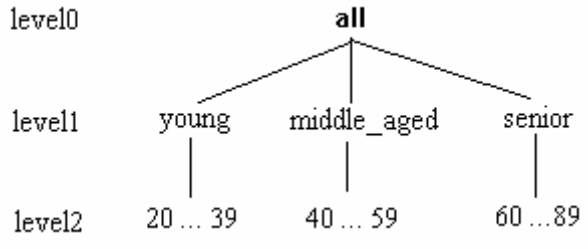


图 4.6 *age* 的一个概念分层

附加的例子留作习题。

4.2.4 兴趣度度量说明的语法

用户可以通过说明模式的兴趣度和它们的对应阈值，控制数据挖掘系统返回的不感兴趣的模式数量。兴趣度度量包括 4.1.4 小节介绍的置信度、支持度、噪音和新颖性度量。用户可以用如下语句说明兴趣度度量和它的阈值：

with [<interest_measure_name>] **threshold** = < threshold_value>

例 4.16 在挖掘关联规则时，用户可能用如下语句限制找到的规则分别满足最小支持度和最小置信度阈值 5%和 70%

```
with support threshold = 5%
with confidence threshold = 70%
```
□

可以交互地设置和修改兴趣度度量和阈值。

4.2.5 模式提供和可视化说明的语法

“用户如何说明显示发现模式的表示和可视化形式？”我们的数据挖掘语言需要一种语法，使得用户可以说明用一种或多种形式显示发现的模式。这些形式包括规则、表、交叉表、饼图、判定树、方、曲线等。为此，我们定义 **display** 语句：

display as <result_form>

其中，<result_form>可以是以上列举的知识表示和可视化形式。

交互挖掘应当允许由不同的概念层或不同的角度来观察发现的模式。这可以用第 2 章介绍的上卷和下钻等操作来实现。通过沿属性或维的概念分层向上攀升（用较高层次的概念值替换较低层次的概念值），模式可以被上卷或在更一般的层次上观察。泛化也可以用丢弃属性或维来进行。例如，假定模式中包含了属性 *city*。给定分层结构 *city* < *province_or_state* < *country* < *continent*，则由模式丢弃 *city* 将泛化数据到下一个较高层 *province_or_state*。通过沿属性或维的概念分层下降，模式可以被下钻或在较细的层次上观察。通过添加属性或维到模式，也可以使它特化。添加的属性必须是任务相关说明的 **relevance to** 子句中列出的属性。用户可以用以下 DMQL 语法交替地在不同抽象级观察模式：

<Multilevel_manipulation> ::= **roll up on** <attribute_or_dimension>
| **drill down on** <attribute_or_dimension>
| **add** <attribute_or_dimension>
| **drop** <attribute_or_dimension>

例 4.17 假定描述是基于维 *location*, *age* 和 *income* 的挖掘。可以 “**roll up on** location”，或“**drop** age”，泛化发现的模式。□

4.2.6 汇集 —— 一个 DMQL 查询的例子

在上面的讨论中，我们用五种数据挖掘原语给出了说明数据挖掘查询的语法。对于一个给定的查询，这些原语定义任务相关的数据，要挖掘的知识类型，概念分层和兴趣度度量，以及模式可视化表示形式。这里，我们把这些成分汇集在一起，看一个 DMQL 查询的完整说明。

例 4.18 挖掘特征描述。假定你作为 AllElectronics 的市场经理，希望知道购买商品的价格不低于$100 的顾客的购买习惯特征，涉及顾客的年龄、所购商品类型和商品的产地。对于所发现的每个特征，你希望知道具有这种特征的顾客所占的百分比。特殊地，你只关心在加拿大的购买，并且用 American Express(“AmEx”)信用卡付款。你希望以表的形式观察发现结果。该数据挖掘查询用 DMQL 表达如下：

```
use database AllElectronics_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age,I.type,I.place_made
from customer C,item I,purchases P,items_sold S,works_at W,branch B
where I.item_ID=S.item_ID and S.trans_ID=P.trans_ID and P.cust_ID=C.cust_ID
      and P.methold_paid="AmEx" and P.empl_ID=W.empl_ID
      and W.branch_ID=B.branch_ID and B.address="Canada" and I.price>100
with noise threshold=5%
display as table
```

该数据挖掘查询被分析，形成由 AllElectronics 数据库提取任务相关数据的 SQL 查询。使用图 4.3 的概念分层结构 *location_hierarchy* 产生分店位置高层次概念，如“Canada”。然后，运行用于产生泛化数据的挖掘特征规则的算法。挖掘特征规则的算法在第 5 章介绍。挖掘的特征规则由属性 *age, type* 和 *place_made* 导出，以表或广义关系（表 4.1）的形式显示。满足每个广义元组的任务相关的元组数作为 count%的结果以百分比表示。如果不指定可视化形式，则使用省缺的形式。5%的噪音阈值意味总计数低于 5%的发现元组都被忽略而不显示。□

表 4.1：表形式的特征描述，或泛化关系

age	type	place_made	count%
30...39	家庭安全系统	美国	19
40...49	家庭安全系统	美国	15
20...29	CD 播放机	日本	26
30...39	CD 播放机	美国	13
40...49	大屏幕 TV	日本	8
...	...	...	...
			100%

类似地，可以给出区分、关联、分类和预测的数据挖掘查询的完整的 DMQL 说明。查询的例子将在下面几章给出，那里分别研究这些知识类型的挖掘。

4.2.7 其它数据挖掘语言和数据挖掘原语的标准化

“在本书中，你已介绍了 DMQL。还有其它数据挖掘语言吗？”在本书中，DMQL 用于介绍数据挖掘原语和概念。除此以外，还有其它一些设计数据挖掘语言的研究成果和对挖掘原语和语言标准化的产业成果。这里，我们介绍一些例子。关于语言和标准的详细信息可以在本章文献注释开列的文献中找到。

MSQL 是一种数据挖掘查询语言，由 Imielinski 和 Vermani 提出[IV99]。该语言使用类 SQL 语法和包括排序和分组在内的 SQL 原语。由于在数据挖掘时可能产生大量规则，MSQL 提供了象 GetRules 和 SelectRules 这样的原语，用于规则泛化和规则选择。由于它对数据和规则的处理是一致的，因此，通过进行有选择的、基于查询的数据泛化，或通过操作或查询产生的规则集，可以实现优化。

数据挖掘语言设计方面的其它研究成果包括 MINE RULE 操作，由 Meo, Psaila 和 Ceri 提出[MPC96]。它遵循类 SQL 语法，并作为规则产生查询用于挖掘关联规则。另一种提议由 Tsur, Ullman, Abitboul, Clifton 等提出[TUA+98]，使用 Datalog 语法表示查询群。这有利于挖掘和测试规则。

“数据挖掘方面有标准化成果吗？”最近，Microsoft 提出了一种数据挖掘语言，称作**数据挖掘（DM）OLE DB**。这是朝着数据挖掘语言原语标准化前进的引人注目的努力。有了标准的数据挖掘语言将帮助加强数据挖掘产业，有利于数据挖掘平台和数据挖掘系统的开发，有利于数据挖掘结果的共享。

DM OLE DB 与 OLE DB，OLAP OLE DB 一起组成 Microsoft 迈向数据库、数据仓库和数据挖掘标准化的三个重要步骤。DM OLE DB 涵盖了若干重要的数据挖掘模块的创建和使用，包括关联、预测建模（分类和预测）和聚类。由于在本书出版时 DM OLE DB 还未完成，我们只好选择 DMQL 作为本书的数据挖掘语言。一个关于 DM OLE DB 的简要介绍在附录 A 中。

CRISP-DM（Cross-Industry Standard Process for Data Mining）（*http://www.crisp-dm.org/*）是另一项关于数据挖掘标准化的成果。它抛开对技术的关注，强调所有层次的用户在使用数据挖掘技术解决商务问题的需要。CRISP-DM 是一个国际项目，它集中了一些数据仓库公司、数据挖掘公司和用户公司一道工作，旨在提供有效的数据挖掘平台和处理结构。该项目要定义和确认在不同的产业部门广泛可用的数据挖掘处理。它强调（1）由商务问题到数据挖掘问题的映射，（2）捕获和理解数据，（3）识别和解决数据中的问题，（4）应用数据挖掘技术，（5）在商务环境下解释数据挖掘结果，（6）使用和管理数据挖掘结果，（7）收集和交换专家意见，确保今后的项目能由经验受益。该项目提供了一个实现数据挖掘的处理结构和数据挖掘项目可能出现的潜在问题指南。

随着数据挖掘系统的进一步开发和数据挖掘语言的标准化，可以预料，本书再版时使用的数据挖掘语言将朝着类似于 DM OLE DB 的原语进化，或最终被更完整的标准数据挖掘语言所取代，不管这种语言那时叫什么。

4.3 基于数据挖掘查询语言设计图形用户界面

数据挖掘查询语言提供了必要的原语，允许用户与数据挖掘系统通讯。然而，没有经验的用户可能发现数据挖掘查询语言很难使用，并且语法太难记。用户可能更愿意与图形用户界面（GUI）通讯。对于关系数据库技术，SQL 充当关系系统标准的“核心”语言，通过它很容易设计 GUI。类似地，数据挖掘查询语言可以充当数据挖掘系统实现的“核心语言”，为有效的数据挖掘系统 GUI 的开发提供基础。

数据挖掘 GUI 可能包含以下成分：

- **数据收集和数据查询编辑**：该部分允许用户说明任务相关的数据集，编写数据挖掘查询。它类似于关系查询说明所用的 GUI。
- **发现模式的表示**：该成分允许以各种形式显示发现的模式，包括表、图、图表、曲线或其它可视化技术。
- **分层结构说明和操纵**：该成分允许说明概念分层，或者由用户手动地，或者自动地（基于手头数据的分析）说明。此外，该成分还应当允许用户修改概念分层，或根据给定的数据分布，自动地对概念分层进行调整。
- **数据挖掘原语的操作**：该成分允许动态地调整数据挖掘阈值，选择、显示和修改概念分层。它还可能允许修改先前的数据挖掘查询或条件。
- **交互的多层挖掘**：该成分应当允许在发现的模式上进行上卷、下钻操作。
- **其它各种信息**：这部分可能包含联机帮助手册、索引查找、调试和其它交互图形机制。

图形用户界面的设计还应当考虑数据挖掘系统的不同类型用户。一般地，数据挖掘系统的用户可以分成两类：商务分析者和商务执行者。商务分析者希望在选择数据的不同部分、操纵维和层、设定挖掘参数和调整挖掘过程方面具有灵活性和方便性。另一方面，商务执行者需要清楚地提供和解释数据挖掘结果、灵活地观察和比较不同的数据挖掘结果、容易地将数据挖掘结果集成到报告书写中。一个良好设计的数据挖掘系统应当为这两类用户提供友好的用户界面。

“数据挖掘查询语言可以进化，形成设计数据挖掘 GUI 的标准吗？”如果这种进化是可能的，标准将有利于数据挖掘软件的开发和系统通讯。然而，一些 GUI 原语，如指向曲线或图形中的一个特定点，很难使用象 DMQL 这样的基于文本数据挖掘查询语言来说明。标准化的基于 GUI 的语言可能演变，并取代类 SQL 的数据挖掘语言。只有时间能够告诉我们。

4.4 数据挖掘系统的结构

随着数据挖掘的流行和扩散，可以预料：在未来的几年中，将会设计和开发各种数据挖掘系统。尽管丰富和强大的数据挖掘功能形成了数据挖掘系统的核心，象大部分软件系统一样，数据挖掘系统的结构和设计是至关重要的。一个好的系统结构将有利于系统更好地利用软件环境，有利于有效、及时地完成数据挖掘任务，有利于与其它信息系统协调和交换信息，有利于系统适应用户的种种需求，并随时间进化。

“数据挖掘系统的期望结构是什么？”数据库和信息产业界研究和开发历经数十年，数据库系统和数据仓库系统已经成为主流信息系统。海量数据和信息已经存储和/或集成在这些系统中。此外，全面的信息处理和数据分析基础已经或将持续不断地、系统地围绕数据库系统和数据仓库构造。

这包括多个异种数据库的访问、集成、统一和转换，ODBC/OLE DB 连接，Web 访问和服务机制，报告和 OLAP 分析工具。

在这种情况下，数据挖掘系统设计的一个重要问题是：我们是否应当将数据挖掘（DM）系统与数据库（DB）系统和/或数据仓库（DW）系统耦合或集成？如果应当，怎样正确地进行？为回答这些问题，我们需要考察耦合或集成 DM 系统和 DB/DW 系统的可能途径。基于不同的结构设计，用以下耦合模式可以将 DM 系统与 DB/DW 系统集成：不耦合、松散耦合、半紧密耦合和紧密耦合。让我们逐一考察它们。

不耦合：不耦合意味 DM 系统不利用 DB 或 DW 系统的任何功能。它可能由特定的源（如，文件系统）提取数据，使用某些数据挖掘算法处理数据，然后再将挖掘结果存放到另一个文件中。

这种系统尽管简单，但有不少缺点。首先，DB 系统在存储、组织、访问和处理数据方面提供了很大的灵活性和有效性。不使用 DB/DW 系统，DM 系统可能要花大量的时间查找、收集、清理和转换数据。在 DB 和/或 DW 系统，数据多半被很好地组织、索引、清理、集成或统一，使得找出任务相关的、高质量的数据成为一件容易的任务。其次，在 DB 或 DW 系统中，有许多被测试的、可规模化的算法和数据结构。使用这种系统开发有效的、可规模化的实现是切实可行的。此外，大部分数据已经或将要存放在 DB/DW 系统中。不与这些系统耦合，DM 系统就需要使用其它工具提取数据，使得很难将这种系统集成到信息处理环境。这样，不耦合是一种很糟糕的设计。

松散耦合：松散耦合意味 DM 系统将使用 DB 或 DW 的某些机制，从这些系统管理的数据存储提取数据，进行数据挖掘，然后将挖掘的结果或者存放到文件中，或者存放到数据库或数据仓库的指定位置。

松散耦合比不耦合好，因为它可以使用查询处理、索引和其它机制，提取存放在数据库或数据仓库中数据的任意部分。这带来了这些系统提供的灵活性、有效性等优点。然而，许多松散耦合的系统是基于内存的。由于挖掘本身不使用 DB 或 DW 提供的数据结构和查询优化方法，对于大的数据集，松散耦合系统很难获得可规模性和良好的性能。

半紧密耦合：半紧密耦合意味除了将 DM 系统连接到一个 DB/DW 系统之外，一些基本数据挖掘原语（通过分析频繁遇到的数据挖掘功能确定）可以在 DB/DW 系统中实现。这些原语可能包括排序、索引、聚集、直方图分析、多路连接和一些基本的统计度量（如，求和、计数、最大、最小、标准差等）的预计算。此外，一些频繁使用的中间结果也可以预计算，并存放在 DB/DW 系统中。由于这些中间挖掘结果或者预计算，或者可以有效地计算，这种设计将提高 DM 系统的性能。

紧密耦合：紧密耦合意味 DM 系统平滑地集成到 DB/DW 系统中。数据挖掘子系统被视为信息系统的一个成分。数据挖掘查询和功能根据 DB 或 DW 系统的挖掘查询分析、数据结构、索引模式和查询处理方法优化。随着技术进步，DM、DB 和 DW 将进化和集成在一起，成为一个具有多种功能的信息系统。这将提供一个一致的信息处理环境。

这种方法是高度期望的，因为它有利于数据挖掘功能的有效实现，有利于提高系统性能，有利于实现集成的信息处理环境。

有了这些分析，可以看出数据挖掘系统应当与一个 DB/DW 系统耦合。松散耦合尽管不太有效，也比不耦合好，因为它可以使用 DB/DW 的数据和系统机制。紧密耦合是高度期望的，但其实现并非易事，在此领域需要更多的研究。半紧密耦合是松散和紧密耦合之间的折衷。重要的是识别常用的数据挖掘原语，提供这些原语在 DB/DW 系统中的有效实现。

4.5 总结

- 我们研究了在**数据挖掘查询**形式下，说明数据挖掘任务的五种原语。这些原语说明任务相关的数据（即，要挖掘的数据集）、要挖掘的知识的类型（即，特征、区分、关联、分类和预测）、背景知识（通常，以概念分层形式表示）、兴趣度度量、发现模式的知识表示和可视化形式。

- 在定义**任务相关数据**时，用户说明包含被挖掘数据的数据库和表（或数据仓库和数据方）、选择数据和分组的条件、挖掘时要考虑的属性（或维）。
- **概念分层**提供了有用的背景知识，有利于使用简明的高层术语表达发现模式，并有助于多个抽象层的知识挖掘。
- **模式兴趣度**的度量评估发现模式的简洁性、确定性、实用性和新颖性。这些度量能够帮助减少返回用户的不感兴趣的模式数量。
- 用户应当能够说明显示发现模式的期望形式，如规则、表、图表、判定树、方、图或报告。上卷和下钻操作也应当能够用于多个抽象层的模式观察。
- 可以设计**数据挖掘查询语言**，支持特定的和交互的数据挖掘。数据挖掘语言（如 DMQL）应当为说明数据挖掘原语，产生和操纵概念分层提供命令。这样的查询语言是基于 SQL 的，并可能最终形成标准，成为数据挖掘图形用户界面的基础。
- **数据挖掘系统结构**包括将数据挖掘系统与数据库/数据仓库系统偶合的各种考虑。有多种可能的设计：不耦合、松散耦合、半紧密耦合和紧密偶合。一个良好设计的数据挖掘系统应当提供与数据库和/或数据仓库的紧密或半紧密耦合。

习题

4.1 列出和描述说明数据挖掘任务的五种原语。

4.2 说明为什么概念分层在数据挖掘中是有用的.

4.3 概念分层的四种类型是：模式分层、集合分组分层、操作导出的分层和基于规则的分层。
(a) 简略定义每种类型的分层。
(b) 对于每种类型的分层，给出一个不在本章中的例子。

4.4 假定 Big-University 的大学课程数据库包含下列属性:每个学生的 *name, address, status*(本科生或研究生)，*major* 和 *GPA*（累计平均等级）。
(a) 对属性 *status, major, GPA* 和 *address* 提出一个概念分层。
(b) 对于你上面提出的每个概念分层，你提出的概念分层结构的类型是什么？
(c) 使用 DMQL 语法定义每个概念分层。
(d) 写一个 DMQL 查询，找出 GPA 为优秀的学生的特征。
(e) 写一个 DMQL 查询，比较科学和艺术专业的学生。
(f) 写一个 DMQL 查询，找出任课教师、学生成绩和你选择的其它属性之间的关联规则。使用元规则说明你想要找的关联规则形式。为列出的关联规则指定最小置信度和支持度阈值。
(g) 写一个 DMQL 查询，根据学生的 GPA 和任课教师预测 “计算机科学 101”课程的学生成绩。

4.5 考虑下面的由 Big-University 的学生数据库挖掘的关联规则(4.8):

$$major(X,"science") \Rightarrow status(X,"undergrad") \quad (4.8)$$

假定学校的学生人数（即，任务相关的元组数）为 5000，其中 56%的在校本科生的专业是科学，64%的学生注册本科学位课程，70%的学生主修科学。
(a) 计算规则(4.8)的支持度和置信度。
(b) 考虑下面的规则(4.9):

$$major(X,"biology") \Rightarrow status(X,"undergrad") \quad [17\%,80\%] \quad (4.9)$$

假定主攻科学的学生 30%专业为 *biology*。与规则(4.8)对比，你认为规则(4.9)新颖吗？解释你的结论。

4.6 <Mine_Knowledge_Specification>语句可以用于挖掘特征、区分、关联、分类和预测规则。为聚类的挖掘提出一个语法定义。

4.7 下面的练习涉及定义概念分层的 DMQL 语法。

(a) 典型地，对于模式 *date*(*day, month, quarter, year*)，数据挖掘系统有一个预定义的概念分层。使用 DMQL 提供该概念分层的定义。

(b) 概念分层定义可能设计多个关系。例如，*item_hierachy* 可以涉及两个关系 *item* 和 *supplier*，由如下模式定义：

item(*item_ID, branch, type, place_made, supplier*)

supplier(*name, type, headquarter_location, owner, size, assets, revenue*)

为 *item_hierachy* 提出 DMQL 定义。（提示：你可以使用 **where** 结构和"."记号，如在 SQL 中那样。）

(c) 对集合分组分层使用 DMQL 语法，通过添加概念层 *continent*，精炼例 4.14 中 *location* 的模式分层。

(d) 作为例 4.15 中集合分组分层的一种替代，用户可能希望根据数据聚类例程定义一个操作导出的分层。为 *age* 的这种分层提出 DMQL 语法。例如，根据 5 个聚类。

(e) 概念分层可以根据一个规则集合定义。给出 DMQL 语法，根据利润，为例 4.6 中 AllElectronics 的 *items* 定义基于规则的分层。

4.8 讨论建立标准化的数据挖掘查询语言的重要性。涉及这一任务的一些潜在好处和挑战是什么？列举一些该领域的最近提议。

4.9 描述如下将数据挖掘系统与数据库或数据仓库系统集成的结构之间的差别：不耦合、松散耦合、半紧密耦合和紧密耦合。陈述你认为哪种结构最流行，为什么？

文献注释

大量的兴趣度客观度量已在一些文献中提出。简洁性度量在 Michalski[Mic83]中给出。本章介绍的关联规则的置信度和支持度度量由 Agrawal,Imielinski 和 Swami[AIS93a]提出。我们介绍的识别冗余多层关联规则由 Srikant 和 Agrawal [SA95,SA96]提出。其它客观兴趣度度量在[HM91, PS91ab, SG92, AIS93a, MM95, CHY96, Ka96]中。兴趣度的主观度量考虑用户对于数据联系的确信，在[PSM94, MPSM96, ST96, LHC97]中讨论。

DMQL 数据挖掘语言由 Han, Hu 和 Wang 等提出[HFW+96a]，用于 DBMiner 数据挖掘系统。

发现插板（Discovery Board）由 Imielinski,Virmani 和 Abdulghani[IVA96]作为应用开发界面原型提出，它涉及数据挖掘查询说明和规则提取的基于 SQL 的操作。其相关的数据挖掘查询语言 MSQL 由 Imielinski 和 Vermani[IV99]提出。MINE RULE，一种挖掘单维关联规则的类 SQL 操作由 Meo,Psaila 和 Ceri[MPC96]提出，并被 Baralis 和 Psaila[BP97]扩展。一种关联规则产生语言使用 *Datalog* 语法[RGOO]，并基于查询群的概念由 Tsur, Ullman, Abitboul, Clifton 等[TUA+98]提出。具有元规则的挖掘在 Klemettinen, Mannila, Ronkainen 等[KMR+94]，Fu 和 Han[FH95]，Shen, Ong, Mitbander 和 Zaniolo[SOMZ96]和 Kamber, Han 和 Chiang[KHC97]中描述。其它想法涉及在挖掘中使用模板和谓词限制，已被[AK93, DT93, LHC97, ST96, SVA97, NLHP98]讨论。

关系数据库系统成功的一个重要因素[SH98]是关系数据库语言 SQL 的标准化。最近的标准化活动，如关于 SQL-3 等的工作进一步表明具有一个标准的数据库语言对于数据库系统的成功实现和商品化的重要性。通过提出数据挖掘（DM）OLE DB[MicOO]，Microsoft 公司已经做出数据挖掘标准化的努力。DM OLE DB 的数据挖掘原语的简介可以在本书的附录 A 中找到。CRISP-DM（Cross-Industry Standard Process for Data Mining）(*http://www.crisp-dm.org/*)是涉及数据挖掘标准化的又一成果。然而，它抛开对技术的关注，强调所有层次的用户在使用数据挖掘技术解决商务问题的需要。

有许多为数据挖掘开发的图形用户界面和可视化工具，可以在各种数据挖掘产品中找到。一些数据挖掘的书籍，如 Westphal 和 Blaxton 的 *Data Mining Solution*[WB98]，给出了一些很好的例

子和可视快照。可视化技术的综述参见 Keim[Kei97]的 *Visual Techniques for Exploring Databases*。

数据挖掘系统的结构已被许多研究者在会议的小组讨论和大会上讨论。数据挖掘语言的最近设计（如[Mic00]），联机分析挖掘（如[Han98]）和数据挖掘查询优化的研究（如[NLHP98, STA98, LNHP99]）可以看作迈向数据挖掘系统与数据库系统和数据仓库系统紧密集成的步骤。Sarawagi, Thomas 和 Agrawal 提出的一些数据挖掘原语[STA98]，如 KWayJoin, GatherPrune 等，也可以用作关系或对象-关系系统中的建筑构件，用于这些数据库系统中数据挖掘的有效实现。

第五章 概念描述：特征与比较

从数据分析的角度，数据挖掘可以分为两类：描述式数据挖掘和预测式数据挖掘。描述式数据挖掘以简洁概要的方式描述数据，并提供数据的有趣的一般性质。预测式数据挖掘分析数据，建立一个或一组模型，并试图预测新数据集的行为。

数据库通常存放大量的细节数据。然而，用户通常希望以简洁的描述形式观察汇总的数据集。这种数据描述可以提供一类数据的概貌，或将它与对比类相区别。此外，用户希望方便、灵活地以不同的粒度和从不同的角度描述数据集。这种描述性数据挖掘称为概念描述，它是数据挖掘的一个重要部分。

本章，你将学习概念描述如何有效地进行。

5.1 什么是概念描述？

描述性数据挖掘的最简单类型是概念描述。概念通常指数据的汇集，如 *frequent_buyers, graduate_students* 等。作为一种数据挖掘任务，概念描述不是数据的简单枚举。**概念描述**产生数据的特征和比较描述。当被描述的概念涉及对象类时，有时也称概念描述为**类描述**。**特征**提供给定数据汇集的简洁汇总，而概念或类的**比较**（也称为**区分**）提供两个或多个数据汇集的比较描述。由于概念描述涉及特征和比较，我们将逐一研究这些任务的实现技术。

概念描述与数据泛化密切相关。给定存放在数据库中的大量数据，能够以简洁的形式在更一般的（而不是在较低的）抽象层描述数据是很有用的。允许数据集在多个抽象层泛化，便于用户考察数据的一般行为。例如，给定 AllElectronics 数据库，销售经理可能不想考察每个顾客的事务，而愿意观察泛化到高层的数据。如，根据地区按顾客的分组汇总，观察每组顾客的购买频率和顾客的收入。这种多维、多层数据泛化类似于数据仓库中的多维数据分析。在这种意义下，概念描述类似于第 2 章讨论的数据仓库的联机分析处理（OLAP）。

“大型数据库的概念描述和数据仓库的联机分析处理有何不同？”二者之间的主要差别如下：

复杂的数据类型和聚集：数据仓库和 OLAP 工具基于多维数据模型，将数据看作数据方形式，由维（或属性）和度量（聚集函数）组成。然而，对于这些系统的大部分商品化版本，维和度量的数据类型都是很有限的。许多当前的 OLAP 系统限制维必须是非数值数据[13]。类似地，在当前的 OLAP 系统中，度量（如 count(), sum(), avg()）也仅用于数值数据。相反，对于概念形成，数据库属性可以是各种各样的数据类型，包括数值的、非数值的、空间的、文本的或图象的。此外，数据库中属性的聚集也可能包括复杂的数据类型，如非数值数据的集合，空间区域的合并，图象的合成，文本的集成，和对象指针分组等。这样，由于可能的维和度量类型的限制，OLAP 只表现为一种简单的数据分析模型。需要时，数据库中的概念描述可以处理具有复杂数据类型的属性和它们的聚集。

用户控制与自动处理：数据仓库中的联机分析处理纯是用户控制的过程。维的选择和诸如下钻、上卷、切块和切片等 OLAP 操作的使用都由用户指挥和控制。尽管在大部分 OLAP 系统中，用户控制的界面是相当友好的，但用户确实需要对每个维的作用有透彻的理解。此外，为了找到一个满意的描述，用户需要使用一长串 OLAP 操作。相反，数据挖掘系统中的概念描述努力成为更自动化的过程，帮助用户确定哪些维（或属性）应当包含在分析中，给定的数据应当泛化到什么程度，以便产生有趣的数据汇总。

正如第 2 章所讨论的，最近，数据仓库和 OLAP 技术正在朝着处理更复杂的数据类型和嵌入更多的知识发现机制方向进化。随着技术的进一步发展，预期更多的描述性数据挖掘功能将集成到未来的 OLAP 系统中。

本章，你将学习概念描述的方法，包括多层泛化、汇总、特征和比较。这些方法形成实现数据挖掘的两个主要功能模块的基础：多层特征和比较。此外，你还将考察以多种形式表示概念描述的

[13] 注意，在第 3 章中，我们介绍了概念分层可以由数值数据自动产生，形成数值维。然而，这一特点是数据挖掘的最近研究成果，在大多数商品化系统中还未使用。

技术，包括表、图表、图和规则。

5.2 数据泛化和基于汇总的特征

数据库中的数据和对象通常包含原始概念层的细节信息。例如，*sales* 数据库中的 *item* 关系可能包含描述商品的低层信息，如 *item_ID, name, brand, category, supplier, place_made* 和 *price*。能够对大的数据集合进行汇总并在高层概念提供结果是有用的。例如，圣诞节期间销售的大量商品的汇总提供这些数据的一般描述，对于销售和市场经理都是非常有帮助的。这要求一个重要的数据挖掘功能：数据泛化。

数据泛化是一个过程，它将大的、任务相关的数据集从较低的概念层抽象到较高的概念层。大的数据集有效的、灵活的泛化方法可以分为两类：（1）数据方（或 OLAP）方法，和（2）面向属性归纳方法。数据方方法已在第 2 章介绍。本节，我们介绍面向属性的归纳方法。

5.2.1 面向属性归纳

对于数据泛化和基于汇总的特征，面向属性的归纳于 1989 年首次提出，比数据方方法的提出早几年。数据方方法可以认为是基于数据仓库的、面向预计算的、物化视图的方法。它在 OLAP 或数据挖掘查询提交处理之前，脱机计算聚集。另一方面，面向属性的归纳，至少在它被提出时，是面向关系数据库查询、基于泛化的、联机的数据分析处理技术。然而，根据联机聚集和脱机预计算区分两种方法并非是固有的。数据方中有些聚集也可以联机计算，而多维空间的脱机预计算也可以加快面向属性的归纳速度。

让我们先介绍面向属性的归纳方法。然后，我们将讨论该方法的细节，它的变形和扩充。

面向属性归纳的基本思想是：首先使用关系数据库查询收集任务相关的数据；然后，通过考察任务相关数据中每个属性的不同值的个数，进行泛化。泛化或者通过属性删除，或者通过属性泛化进行。聚集通过合并相等的泛化元组，并收集它们对应的计数值进行。这压缩了泛化后的数据集合。结果泛化关系可以映射到不同形式，如图表或规则，提供用户。

下面一系列例子解释面向属性归纳的处理过程。

例 5.1 用 DMQL 说明特征数据挖掘查询。假定用户想描述 Big-University 数据库中研究生的一般特征。给定的属性有 *name,gender,major,birth_place,residence,phone#*（电话号码）和 *gpa*（平均等级分）。该特征的数据挖掘查询可以用数据挖掘查询语言 DMQL 表示如下：

```
use Big_University_DB
mine characteristics as "Science_Students"
in relevance to name,gender,major,birth_place,birth_date,residence,phone#,gpa
from student
where status in "graduate"
```

我们将看看这个典型的数据挖掘查询例子如何使用面向属性的归纳挖掘特征描述。□

“面向属性归纳的第一步做什么？”首先，在面向属性归纳之前进行**数据聚焦**。这一步对应于第 4 章介绍的说明任务相关数据（或用于分析的数据），根据数据挖掘查询提供的信息进行数据收集。由于数据挖掘查询通常只涉及数据库的一部分，选择相关的数据集不仅使得挖掘更有效，而且与在整个数据库挖掘相比，能够产生更有意义的规则。

对于用户来说，指定相关的数据集（即，挖掘的属性，如 DMQL 的 **in relevance to** 子句所指出的）可能是困难的。有时，用户只能选择少量他感到可能重要的属性，而遗漏在描述中可能起作用的其它属性。例如，假定 *birth_place* 由属性 *city,province_or_state* 和 *country* 定义。这些属性，用户只想到说明 *city*。为了能在 *birth_place* 维上泛化，定义该维的其它属性也应当包括进来。换一句话说，系统自动地包括 *province_or_state* 和 *country* 作为相关属性，使得 *city* 可以在归纳过程中泛化到较高的概念层。

另一个极端是，用户可能引进太多属性，如用“**in relevance to ***”指定所有可能的属性。在这种情况下，被 **from** 子句说明的关系的所有属性将包含在分析中。许多属性对于有趣的描述是没有用的。5.3 节介绍一种方法，通过过滤掉统计不相关或弱相关属性来处理这种情况。

“子句‘**where** status **in** “graduate”’ 是什么意思？”该 **where** 子句意味在属性 *status* 上存在概念分层。这种概念分层将 *status* 的原始层的值，如*”M.Sc”,”M.A”,M.B.A”,”Ph.D”,”B.Sc”,”B.A”*组织成较高层次的概念，如”*graduate*”和”*undergraduate*”。这种概念分层在传统的关系查询语言中没有，而在数据挖掘语言中是普遍的。

例 5.2　转化数据挖掘查询为关系查询。例 5.1 的数据挖掘查询被转换成如下关系查询，收集任务相关的数据集。

```
use Big_university_DB
select name,gender,major,birth_place,birth_date,residence,phone#,gpa
from student
where status in {”M.Sc”,”M.A”,M.B.A”,”Ph.D”}
```

转换后的查询在关系数据库 *Big_university_DB* 上执行，返回表 5.1 所示数据。该表称作（任务相关）**初始工作关系**，是要进行归纳的数据。注意，事实上每个元组是属性-值对的合取。因此，我们可以认为关系的元组是合取规则，而关系上的归纳是这些规则的一般化。□

表 5.1：初始工作关系：任务相关数据集合

name	gender	major	birth_place	birth_date	residence	phone#	gpa
Jim Woodman	M	CS	Vancouver,BC,Canada	8-12-76	3511,Main St., Richmand	687-4598	3.67
Scott Lachance	M	CS	Montreal,Que,Canada	28-7-75	345, IstAve.,Vancouver	253-9106	3.70
Laura Lee	F	physics	Seattle,WA,USA	25-8-70	125,Austin Ave.,Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…

“对于面向属性归纳，现在数据已经准备好，如何进行面向属性归纳？”面向属性归纳的基本操作是数据泛化，它可以用两种方法之一在初始关系上进行：属性删除，属性泛化。

属性删除基于如下规则：如果初始工作关系的某个属性有大量不同的值，但是（1）在此属性上没有泛化操作符（例如，对该属性没有定义概念分层），或者（2）它的较高层概念用其它属性表示，则该属性应当从工作关系中删除。

该规则的理由何在？一个属性-值对表示泛化元组或规则的一个合取。删除一个合取就删除了一个限制，从而泛化了规则。如果是情况 1，属性具有大量的不同值，但对它没有泛化操作符，应当将属性删除，因为它不能被泛化，保留它就意味着保留与产生的简洁规则相矛盾的大量不同值。另一方面，考虑情况 2，属性的高层次概念用其它属性表示。例如，假定该属性是 *street*，它的高层次概念被属性(*city, province_or_state, country*)表示。删除 *street* 等价于使用泛化操作。该规则对应于机器学习中示例学习的删除条件。

属性泛化基于如下规则：如果初始工作关系的某个属性有大量不同的值，并且该属性上存在泛化操作符，则应当选择该泛化操作符，并将它用于该属性。该规则基于如下理由：使用泛化操作符泛化工作关系中元组的属性值或规则，将使得规则涵盖更多的原数据元组，从而泛化了它所表示的概念。这对应于泛化规则，在示例学习中称为沿泛化树攀升或沿概念树攀升。

属性删除和属性泛化两个规则都表明，如果某属性有大量的不同值，应当进行进一步泛化。这就提出了一个问题：多大才算“属性具有大量不同值”？

这取决于属性或应用，有的用户愿意让有些属性留在较低的抽象层，而另一些用户愿意将它们泛化到较高的抽象层。控制将属性泛化到多高的抽象层通常是相当主观的。该过程的控制称为**属性泛化控制**。如果属性泛化得“太高”，可能导致过分泛化，产生的规则可能没有多少信息。另一方面，如果属性不泛化到“足够高的层次”，可能泛化不足，得到的规则可能也不含多少信息。这样，面向属性的泛化应当把握好尺度。

有一些方法控制泛化过程。我们介绍两种常用的方法。

第一个技术称作**属性泛化阈值控制**，或者对所有的属性设置一个泛化阈值，或者对每个属性设置一个阈值。如果属性的不同值个数大于属性泛化阈值，则应当进行进一步的属性删除或属性泛化。

典型地，数据挖掘系统有一个省缺的属性阈值（取值范围一般为 2 到 8），并且也允许用户或专家修改该阈值。如果用户感到对于一个特定的属性，泛化达到的层次太高，他可以加大阈值；这对应于沿着属性下钻。为进一步泛化关系，用户也可以减小属性阈值；这对应于沿属性上卷。

第二种技术称作**泛化关系阈值控制**，为泛化关系设置一个阈值。如果泛化关系中不同元组的个数超过该阈值，则应当进行进一步泛化；否则，不再进一步泛化。这样的阈值也可以在数据挖掘系统中预先设定（通常取值范围为 10 到 30），或者由用户或专家设置，并且允许调整。例如，如果用户感到泛化的关系太小，他可以加大阈值；这意味下钻。否则，为进一步泛化关系，他可以减小阈值；这意味上卷。

这两种技术可以顺序使用：首先使用属性泛化阈值控制技术泛化每个属性，然后使用关系阈值控制进一步压缩产生的关系。无论使用哪种泛化控制技术，都应当允许用户调整泛化阈值，以便得到有趣的概念描述

在许多面向数据库的归纳过程中，用户感兴趣的是在不同的抽象层得到数据的量化或统计信息。这样，在归纳过程中收集计数和其它聚集值是非常重要的。概念上讲，这件事可以用如下办法做。一个特殊的度量或数值属性为聚集函数 count，它与每个数据库元组相关联。对于初始工作关系的每个元组，它的值被初始化为 1。通过删除属性和属性泛化，在初始关系中的元组可能被泛化，导致相等的元组分组。在这种情况下，形成一组的所有相等元组应当合并成一个元组。泛化的元组的新计数设置成初始关系中被新的泛化元组代表的元组的计数和。例如，假定根据面向属性归纳，初始关系中 52 个数据元组被泛化成同一个元组 *T*。即，这 52 个元组的泛化产生元组 *T* 的 52 个相等的实例。这 52 个相等的元组合并，形成 *T* 的一个实例，其计数设置成 52。其它流行的聚集函数包括 sum 和 avg。对于一个给定的泛化的元组，sum 包含产生该泛化元组的初始关系的给定数值属性值的和。假定元组 *T* 包含 sum（*units_sold*）作为聚集函数，元组 *T* 的 sum 值应当设置为 53 个元组的 *units_sold* 总和。聚集函数 avg 根据公式 avg = sum/count 计算。

例 5.3　面向属性归纳。这里，我们看看面向属性归纳如何在例 5.2 得到的初始工作关系表 5.1 上进行归纳。对于关系的每个属性，泛化过程如下：

1. *name*：由于 *name* 存在大量不同值，并且其上没有泛化操作符，该属性被删除。
2. *gender*：由于 *gender* 只有两个不同值，该属性保留，并且不对其进行泛化。
3. *major*：假定已定义了一个概念分层，允许将属性 *major* 泛化到值{*arts&science, engineering, business*}。还假定该属性的泛化阈值设置为 5，并且初始关系中 *major* 有 20 不同值。根据属性泛化和属性泛化控制，沿概念分层向上攀升，*major* 被泛化。
4. *birth_place*：该属性有大量不同值，因此应当泛化它。假定存在 *birth_place* 的概念分层，定义为 *city* < *province_or_state* < *country*。如果初始工作关系中 *country* 的不同值个数大于属性泛化阈值，则 *birth_place* 应当删除，因为尽管存在泛化操作符，泛化阈值也不会满足。如果假定 *country* 的不同值个数小于泛化阈值，则 *birth_place* 应当泛化到 *birth_country*。
5. *birth_date*：假定存在概念分层，可以将 *birth_date* 泛化到 *age*，而 *age* 到 *age_range*，并且 *age_range* 的不同值（区间）数小于对应的属性泛化阈值，则应当对 *birth_date* 进行泛化。
6. *residence*：假定 *residence* 被属性 *number, street,residence_city, residence_province_or_ state* 和 *residence_country* 定义。*number* 和 *street* 的不同值多半很多，因为这些概念的层次相当低。因此，*number* 和 *street* 应当删除，将 *residence* 泛化到 *residence_city*，其包含较少的不同值。
7. *phone#*：从名字可以看出，该属性包含太多不同值，因此应当在泛化中删除。
8. *gpa*：假定存在 *gpa* 的概念分层，将等级分分成数值区间，如{3.75-4.0, 3.5-3.75,...}，它又被用描述值{*excellent, very good*,...}分组。这样，该属性可以被泛化。

泛化过程将产生相等元组的组。例如，表 5.1 的前两个元组被泛化成相同的元组（即，表 5.2 的第一个元组）。这些相同的元组被合并成一个，同时累计它们的计数值。这一过程导致表 5.2 所示的泛化关系。

表 5.2：通过对表 5.1 的数据进行面向属性归纳得到的泛化关系

gender	major	birth_country	age_range	residence_city	gpa	count
M	Science	Canada	20..25	Richmond	very_good	16
F	Science	Foreign	25..30	Burnaby	excellent	22

..	ience	.30	...	llent	2
	..	...		...	.
	.				..

根据 OLAP 的术语，我们可以把 count 看作度量，而其它属性看作维。注意，聚集函数，如 sum，可以用于如 *salary, sales* 等数值属性。这些属性称为度量属性。□

在下面的小节，提供泛化的实现技术和方法。

5.2.2 面向属性归纳的有效实现

“面向属性的归纳如何实际实现？”前一小节介绍了面向属性的归纳。一般过程总结在图 5.1 中。算法的有效性分析如下：

- 算法的第 1 步基本上是关系查询，将任务相关的数据收集到**工作关系** W 中。其有效性依赖于所用的查询处理方法。有大量成功实现的商品化数据库系统，该步可望具有很好的性能。
- 第 2 步收集初始关系上的统计。这最多需要对该关系扫描一次。对每个属性计算最低期望层和确定映射对（v, v'）依赖于每个属性的不同值数量，它比初始关系的元组数 n 小。

算法：面向属性归纳。根据用户的数据挖掘请求，在关系数据库上挖掘泛化特征。

输入：(i) 关系数据库 *DB*；(ii) 数据挖掘查询 *DMQuery*；(iii) 属性表 *a_list*（包含属性 a_i 等）；(iv)概念分层或属性 a_i 上的泛化操作符的集合 Gen(a_i)；(v) 每个属性 a_i 的泛化阈值 *a_gen_thresh*(a_i)。

输出：主泛化关系 *P*。

方法：方法如下。

1. $W \leftarrow$ get_task_relevant_data(*DMQuery, DB*)； // 工作关系 W 存放任务相关的数据。
2. prepare_for_generalization(W) ； // 该步实现如下。
   (a) 扫描 W，收集每个属性 a_i 的不同值。（注意：如果 W 很大，可以通过考察 W 的样本来做。）
   (b) 对于每个属性 a_i，根据给定的或省缺的属性阈值，确定 a_i 是否应当删除；如果不删除，则计算它的最小期望层次 L_i，并确定映射对（v, v'），其中，v 是 W 中 a_i 的不同值，而 v' 是其对应的在层 L_i 的泛化值。
3. $P \leftarrow$ generalization(W) ；
   通过用其在映射中对应的 v' 替换 W 中每个值 v，累计计数并计算所有聚集值，导出主泛化关系 *P*。
   这一步可以用两种方法有效地实现：
   (a) 对于每个泛化元组，通过二分检索将它插入主关系 P 中。如果元组已在 P 中，则简单地增加它的计数值并相应地处理其它聚集值；否则，将它插入 *P*。
   (b) 在大部分情况下，由于主关系不同值的个数很少，可以将主关系编码，作为 m-维数组，其中 m 是 P 中的属性数，而每个维包含对应的泛化属性值。如果有的话，数组的每个元素存放对应的计数和其它聚集值（如果有的话）。泛化元组的插入通过对应的数组元素上的度量聚集进行。

图 5.1 面向属性归纳的基本算法

- 第 3 步导出**主关系** *P*。这通过将泛化元组插入到 P 中完成。W 中有 n 个元组，P 中有 p 个元组。对于 W 中的每个元组 t，根据导出的映射对替换它的属性值，产生泛化元组 t'。如果采用方法(a)，每个 t' 需要 $O(\log p)$ 时间找到计数增值或插入位置。这样，所有泛化元组总的时间复杂性为 $O(n \times \log p)$。如果采用方法(b)，每个 t' 需要 $O(1)$ 时间找到计数增值元组。这样，所有泛化元组的时间复杂性为 $O(n)$。

许多数据分析任务需要考察很多维或属性。例如，交互式数据挖掘系统可能动态地引入和测试属性，而不仅仅是挖掘查询中说明的那些属性。高级的描述数据挖掘任务，如解析特征（5.3 节讨论），需要大量属性的属性相关分析。此外，不太知道真正的相关数据集的用户可能简单地在挖掘查询中指定“in relevance to *”。在这些情况下，聚集值的预计算将加快大量维或属性的分析。因此，数据方实现是以上介绍的数据库实现的一种吸引人的替换。

面向属性归纳的数据方实现可以采用两种方法进行。

对给定的数据挖掘查询临时构造数据方：该方法根据任务相关的数据集，动态地构造数据方。如果任务相关的数据集太特殊，不能与任何预定义的数据方匹配，或者任务相关的数据集不太大时，该方法是所期望的。由于这种数据方仅当查询提交之后才计算，构造这种数据方的主要动机是便于有效地下钻。有了这种数据方，下钻到主关系层之下，只需要简单地从方中提取数据，或由存放在方中的某中间层稍做泛化，而不是从基本层数据进行泛化。然而，由于面向属性的数据泛化涉及查询相关的数据方的计算，与简单地计算主关系相比，这可能涉及更多的处理，从而增加了响应时间。二者之间的一个折衷是计算方结构的“次主”关系，其泛化关系的每个维层次比主关系的层次稍深一点。这将便于以合理的存储和处理开销下钻到这些层，尽管超过这些层的进一步下钻仍然需要从原始层数据泛化。注意，这种下钻多半是局部的，而不是散布在整个数据方上。

使用预定义的数据方：另一种方法是：在数据挖掘查询提交系统之前构造数据方，并对其后的数据挖掘使用预定义的数据方。如果任务相关的数据的粒度与预定义的数据方一致，并且任务相关的数据量相当大，该方法是所期望的。由于这种数据方是预计算的，它便于属性相关的分析、面向属性归纳、切片和切块、上卷和下钻。必须付出的代价是不容忽视的数据方计算开销和存储开销。计算/存储开销和访问速度之间的一个折衷是，选择性地预计算所有可能物化的方体的一个子集，如第 2 章所述。

5.2.3 导出泛化的表示

“面向属性归纳产生一个或一组泛化描述。如何直观地表示这些描述？”可以用多种不同的形式将描述提供给用户。由面向属性归纳方法产生的泛化描述通常以**泛化关系**形式显式。

例 5.4 假定在 AllElectronics 的 *sales* 关系上进行面向属性归纳，产生 1999 年销售的泛化描述表 5.3。该描述以泛化关系的形式给出。例 5.3 的表 5.2 是泛化关系的另一个例子。□

表 5.3: 1997 年销售的泛化关系

location	item	sales($1,000,000)	count(1,000)
亚洲	TV	15	300
欧洲	TV	12	250
北美	TV	28	450
亚洲	计算机	120	1000
欧洲	计算机	150	1200
北美	计算机	200	1800

描述也可以用交叉表的形式显示。在二维交叉表中，每行显示一个属性的值，每列显示另一个属性的值。在 n 维(n > 2)交叉表中，列可以显示多个属性的值，子和显示属性-值组。这种表示类似于电子数据表。容易直接将数据方结构映射到交叉表。

例 5.5 表 5.3 的泛化关系可以转换成 3-D 交叉表，如表 5.4 所示。□

表 5.4: 1997 年销售的交叉表

location\item	TV		计算机		两项商品	
	sales	count	sales	count	sales	count
亚洲	15	300	120	1000	135	300

	欧洲		1			150		1		1		1
		2		250			200		62		450	
	北美		2			200		1		2		2
		8		450			800		28		250	
	所有地		5		1	470		4		5		5
区		5		000			000		25		000	

泛化数据也可以用图的形式表示，如条形图、饼图和曲线。数据分析中使用图表示是很流行的。这种图和曲线可以表示 2-D 和 3-D 数据。

例 5.6 表 5.4 的交叉表销售数据可以转换成图 5.2 的条形图表示和图 5.3 的饼图表示。□

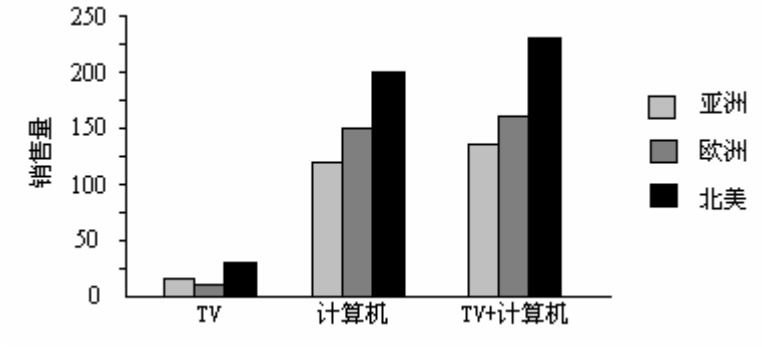


图 5.2 1999 年销售的条形图表示

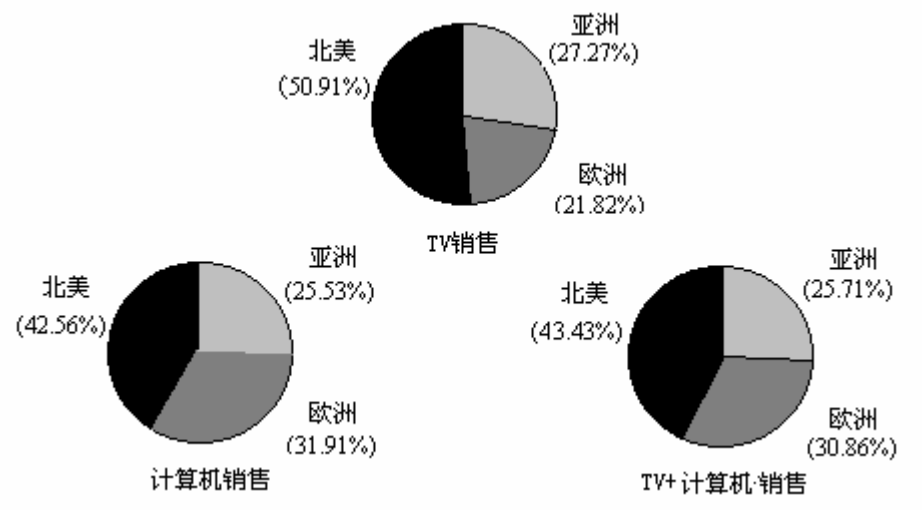


图 5.3 1999 年销售的饼图表示

最后，3-D 泛化关系或交叉表可以用 3-D 数据方表示。这种 3 维数据方视图是一种吸引人的数据方浏览工具。

例 5.7 考虑图 5.4 所示关于维 *item*, *location* 和 *cost* 的数据方。单元的 *size*(用小方体显示)代表对应单元的计数，而单元的亮度可以用于表示单元的另一个度量，如 sum(*sales*)。旋转、上卷、下钻、切片和切块操作可以点击鼠标，在数据方浏览器上进行。□

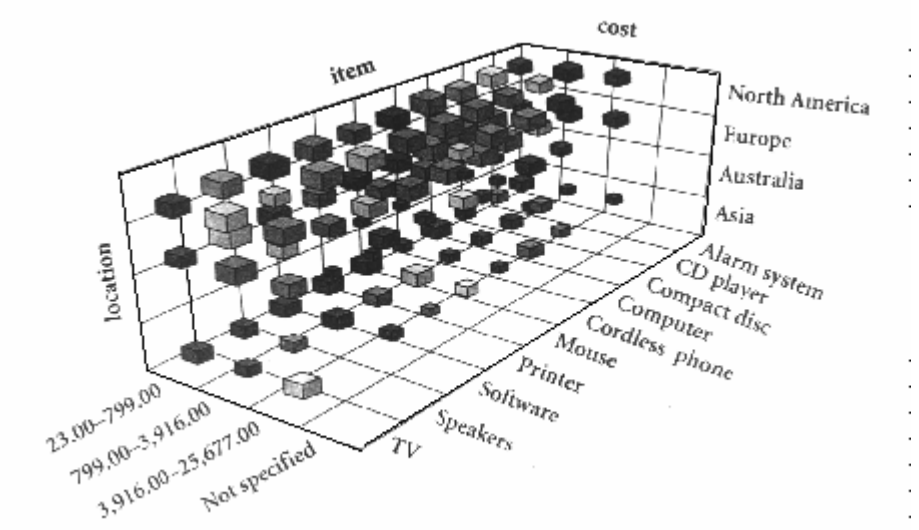


图 5.4　1999 年销售的 3-D 方视图表示

泛化关系也可以用逻辑规则的形式表示。典型地，每个泛化元组代表一个规则析取。由于大型数据库中的数据通常分布在一个发散的区间，单个泛化元组不太可能代表初始工作关系 100%的元组。这样，每个规则应当带上量化信息，如满足规则左部，也满足规则右部的元组所占的百分比。带有量化信息的逻辑规则称为**量化规则**。

为定义量化特征规则，我们引入 ***t*-权**作为兴趣度度量，描述规则中每个析取或对应泛化关系的每个元组的典型性。该度量定义如下：设待特化的（或被规则描述的）对象类称为目标类，q_a 是一个描述目标类的泛化元组。q_a 的 *t*-权是来自初始工作关系被 q_a 涵盖的目标类元组的百分比。形式地，我们有

$$t\_weight = count(q_a) \Big/ \sum_{i=1}^{n} count(q_i) \tag{5.1}$$

其中，n 是泛化关系中目标类元组的个数，$q_1, \ldots, q_n$ 泛化关系中目标类元组，q_a 在 $q_1, \ldots, q_n$ 中。显然，*t*-权的取值区间为[0.0,1.0]或[0%,100%]。

一个**量化特征规则**可以表示为（1）逻辑形式，涵盖目标类的每个析取带有一个对应的 *t*-权；或者（2）关系表或交叉表形式，表中目标类元组的 *count* 值换成对应的 *t*-权值。

量化特征规则的每个析取代表一个条件。一般地，这些条件的析取形成目标类的必要条件，因为该条件是根据目标类的所有情况导出的。即，目标类的所有元组必须满足这一条件。然而，规则可能不是目标类的充分条件，因为满足同一条件的元组可能属于其它类。因此，规则应当表示成如下形式：

$$\forall X,\ target\_class(X) \Rightarrow condition_1(X)\ [t{:}w_1] \vee \ldots \vee condition_m(X)\ [t{:}w_m]. \tag{5.2}$$

该规则指出，如果 X 在 *target_class* 中，则 X 满足 $condition_i$ 的可能性是 w_i，其中 w_i 是 $condition_i$ 或析取 i 的 *t*-权值，而 i 在 $\{1,\ldots,m\}$ 中。

例 5.8　表 5.4 所示交叉表可以转换成逻辑规则形式。设目标类是计算机商品的集合。对应的规则的逻辑形式为：

$$\begin{aligned}\forall X, item(X) = \text{“}computer\text{”} \Rightarrow\\ (location(X) = \text{“}Asia\text{”})\ [t{:}25.00\%] \vee (location(X) = \text{”}Europe\text{”})\ [t{:}30.00\%] \vee\\ (location(X) = \text{“}North\_America\text{)}\ [t{:}45.00\%]\end{aligned} \tag{5.3}$$

注意，第一个 *t*-权值 25%是对应于“(*computer,Asia*)”处的值 1000 被对应于“(*computer,all_regions*)”处的值 4000 除的结果。（即，4000 表示计算机销售总数量。）其它两个 *t*-权用类似方法得到。其它目标类的量化特征规则可以用类似的方法计算。□

“一般地，数据挖掘系统如何使用 *t*-权和兴趣度度量，仅显示客观评估是有趣的那样一些概念描述？”为此目的，可以设定一个阈值。例如，如果一个泛化元组的 *t*-权低于该阈值，则可以认为

该元组代表数据库的不重要部分，并因此可以作为无兴趣的而被忽略。忽略这种不重要的元组并不意味要将它们从中间结果（即，主泛化关系，或数据方，这取决于实现）删除，因为对于其后用户通过其它维或抽象层上的交互式上卷、下钻，进行进一步数据探查，它们可能是有用的。这种阈值可以视为**重要性阈值**或**支持度阈值**。后一术语在关联规则挖掘中很流行。

5.3 解析特征：属性相关性分析

“如果我不能确定哪个属性应当包含在类特征或类比较中，怎么办？我可能指定了太多属性，这可能降低系统性能。”属性相关分析度量可以帮助识别不相关或弱相关属性，可以将它们排除在概念描述过程之外。这一预处理步骤与类特征或类比较结合分别称作解析特征和解析比较。本节介绍属性相关分析的基本方法，以及它与面向属性归纳的集成。

5.3.1 为什么进行属性相关性分析?

对于数据仓库和 OLAP 工具中的多维数据分析，类特征的第一个局限是处理复杂对象。这在 5.2 节已讨论。第二个局限是缺乏自动泛化过程：用户必须显式告诉系统，哪些维应当包含在类分析中，每个维应当泛化到多高的层次。事实上，在任何维上泛化和特化的每一步都必须由用户指定。

通常，用户告诉数据挖掘系统每个维应当泛化到多高层次并不困难。例如，用户可以设置泛化阈值，或使用诸如“**generalize dimension** location **to** country **level**”的命令，说明给定维应当达到的泛化层次。即使没有用户的显式说明，数据挖掘系统也可以设置一个省缺的阈值 2 到 8，使得每个维都可以泛化到只包含 2 到 8 个不同值的层次。如果用户对当前的泛化层次不满意，他可以指定需要上卷或下钻的维。

然而，对于用户来说，确定那些维应当包含在类特征分析中则不是一件平凡的事。数据关系通常包含 50 到 100 个属性，对于有效的数据挖掘，应当选择哪些属性或维，用户所知甚少。用户可能在分析中包含的属性太少，造成挖掘的描述结果不完全。另一方面，用户也可能包含太多分析属性（例如，使用“**in relevance to** *”，它包含指定关系中的所有属性）。

应当引进一些方法进行属性（或维）相关性分析，以过滤掉统计不相关或弱相关的属性，而保留对手头挖掘任务最相关的属性。包含属性/维相关性分析的类特征称为**解析特征**。包含这种分析的类比较称为**解析比较**。

直观地，对于给定的类，一个属性或维被认为是高度相关的，如果该属性或维的值可能用于区分该类与其它类。例如，汽车的颜色多半不能区分贵的和便宜的汽车，但型号、制造商、款式和汽缸数可能是更相关的属性。此外，即使在同一个维内，对于区分一个类与其它类，不同层的概念也可能有很不相同的能力。例如，在 *birth_data* 维，*birth_day* 和 *birth_month* 看上去与雇员的 *salary* 不相关。然而，*birth_decade*（即，年龄区间）可能与雇员的工资是高度相关的。这意味维相关性分析应当在多个抽象层进行，并且只有那些最相关的维层次应当包含在分析中。

上面，我们指出，属性/维的相关性要根据属性/维区分一个类与其它类的能力来评估。在挖掘类比较（或区分）时，目标类和对比类要明显地在挖掘查询中给出。如我们将在下面看到的，相关分析应当进行这些类的比较。然而，在挖掘类特征时，只有一个特征化的类。即，没有说明对比类。这样，什么对比类应当用于相关分析并非明显的。在这种情况下，除特征化的数据集外，数据库中可比较的数据集都作为对比类。例如，为特征化研究生，对比类为不是研究生的学生的集合。

5.3.2 属性相关分析方法

关于属性相关分析，在机器学习、统计、模糊和粗糙集理论等方面都有许多研究。属性相关分析的基本思想是计算某种度量，用于量化属性与给定类或概念的相关性。这种度量包括信息增益、Gini 索引、不确定性和相关系数。

这里，我们介绍一种方法，它将信息增益分析技术（诸如在学习决策树 ID3 和 C4.5 算法中提供的[14]）和基于多维数据分析的方法集成在一起。该方法删除信息量较少的属性，收集信息量较多的属性，用于概念描述分析。

“信息增益计算如何工作？”设 S 是训练样本的集合，其中每个样本的类标号是已知的。事实上，每个样本是一个元组，一个属性用于确定训练样本的类。例如，属性 *status* 可以用于定义每个样本的类标号或者是“*graduate*”，或者是“*undergraduate*”。假定有 m 个类。设 S 包含 s_i 个 C_i 类样本，i = 1, ..., m。一个任意样本属于类 C_i 的可能性是 s_i / s，其中 s 是集合 S 中对象的总数。对一个给定的样本分类所需的**期望信息**是

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s} \tag{5.4}$$

具有值{$a_1, a_2, ..., a_v$}的属性 A 可以用来将 S 划分为子集{ $S_1, S_2, ..., S_v$ }，其中，S_j 包含 S 中 A 值为 a_j 的那些样本。设 S_j 包含类 C_i 的 s_{ij} 个对象。根据 A 的这种划分的期望信息称作 A 的**熵**。它是加权平均：

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + ... + s_{mj}}{s} I(s_{1j} + ... + s_{mj}) \tag{5.5}$$

A 上该划分的信息增益定义为

$$Gain(A) = I(s_1, s_2, ..., s_m) - E(A) \tag{5.6}$$

在这种相关分析方法中，我们可以计算定义 S 中样本的每个属性的信息增益。具有最高信息增益的属性是给定集合中具有最高区分度的属性。通过计算信息增益，我们可以得到属性的秩评定。这种秩评定可用于相关分析，选择用于概念描述的属性。

概念描述的属性相关分析执行步骤如下：

1. **数据收集**：通过查询处理，收集目标类和对比类的数据。对于类比较，目标类和对比类都由用户在数据挖掘查询中提供。对于类特征，目标类是要特征化的类，而对比类是不在目标类中的可比较数据。
2. **使用保守的 AOI 进行预相关分析**：这一步识别属性和维的集合，选择的相关性度量用于它们。由于维的不同层次对于给定的类具有很不相同的相关性，原则上，定义维概念层的每个属性都应当包含在相关分析中。通过删除或泛化具有大量不同值的属性（如，*name* 和 *phone#*），面向属性的归纳（AOI）可以用来进行一些预相关分析。对于概念描述，具有大量的不同值的属性多半没有意义。保守一点，这里进行的 AOI 使用的属性分析阈值要合理的大，使得更多的（但非所有的）属性在进一步相关分析（下面的步骤 3）中被考虑。这样使用 AOI 得到的关系称作挖掘任务的**候选关系**。
3. **使用选定的相关分析度量删除不相关和弱相关属性**：使用选定的相关分析度量，评估候选关系中的每个属性。此步所用的相关性度量可以建立在数据挖掘系统中，或由用户提供。例如，可以使用上面介绍的信息增益度量。根据计算的属性与数据挖掘任务的相关性，对属性排序（即，确定秩）。然后删除与类描述任务不相关或弱相关的属性。可以设置一个阈值来定义“弱相关”。其结果为**初始目标类工作关系**和**初始对比类工作关系**。
4. **使用 AOI 产生概念描述**：使用一组不太保守的属性泛化阈值进行 AOI。如果类描述任务是类特征，这里只包含初始目标类工作关系。如果类描述任务是类比较， 初始目标类工作关系和初始对比类工作关系都要包含在分析中。

该过程的复杂性类似于图 5.1 中的算法，因为归纳过程进行了两次，一次是预相关分析（步骤 2），另一次是在初始工作关系上归纳（步骤 4）。以选定度量进行属性相关性分析（步骤 3）所用的统计可以在步骤 2 的数据库扫描时确定。

5.3.3 解析特征：一个例子

[14] 判定树是一个类似于流程图的树结构，其中，每个结点表示属性上的一个测试，每个分枝代表一个测试的输出，而树叶代表类或类分布。判定树对于分类是有用的，并且容易转换成逻辑规则。判定树归纳的讨论在第 7 章。

如果挖掘的类描述涉及许多属性，应当运行解析特征。该过程在进行特化之前，首先删除不相关或弱相关的属性。让我们考察一个解析挖掘过程的例子。

例 5.9 假定我们想使用解析特征挖掘 Big-University 的研究生的一般特征描述。给定的属性是 *name, gender, major, birth_place, birth_date, phone#*和 *gpa*。

“解析特征如何执行？”第 1 步，收集目标类数据，它由研究生的集合组成。还需要对比类的数据，以便进行相关分析。对比类取本科生的集合。

第 2 步，用保守的属性泛化阈值进行面向属性的归纳，通过属性删除和属性泛化进行预相关分析。类似于例 5.3，属性 *name* 和 *phone#*被删除，因为它们的不同值个数超过了它们对应的属性分析阈值。与例 5.3 相同，使用概念分层将 *birth_place* 泛化到 *birth_country*，*birth_date* 泛化到 *age_range*。属性 *major* 和 *gpa* 也使用例 5.3 的概念分层泛化到较高的抽象概念层。因此，候选关系中剩下的属性是 *gender, major, birth_country, age_range* 和 *gpa*。结果关系在表 5.5 和 5.6 中。

第 3 步，使用选定的相关分析度量(如，信息增益)，评估候选关系中的属性。设 C_1对应于研究生类，C_2对应于本科生类。在研究生类有 120 个样本，本科生有 130 个样本。为计算每个属性的信息增益，我们先用(5.4)式计算对给定的样本分类所需要的期望信息。即

$$I(s_1,s_2)=I(120,130)=-\frac{120}{250}\log_2\frac{120}{250}-\frac{130}{250}\log_2\frac{130}{250}=0.9988$$

表 5.5 由解析特征得到的候选关系：目标类（研究生）

gender	major	birth_country	age_range	gpa	count
M	Science	Canada	21...25	very_good	16
F		Foreign			
M	Science	Foreign	26...30	excellent	22
F		Foreign			
M	Engineering	Canada	26...30	excellent	18
F		Canada			
	Science		26...30	excellent	25
	Science		21...25	excellent	21
	Engineering		21...25	excellent	18

表 5.6 由解析特征得到的候选关系：对比类（本科生）

gender	major	birth_country	age_range	gpa	count
M	Science	Foreign	<= 20	very_good	16
F		Canada	<= 20		8
M	Business	Canada	<= 20	fair	22
F		Canada	21...25	fair	0
M	Business	Foreign	21...25	fair	22
F		Canada	21...25	very_good	24
	Science		21...25	good	22
	Science		<=20	excellent	24
	Engineering				22
	Engineering				24

下一步，我们需要计算每个属性的熵。让我们试属性 *major*。我们需要观察对于属性 *major* 的每个值，研究生和本科生的分布。对每个分布，计算期望信息。

对于 *major* = "*Science*"

s_{11} = 84　　s_{21} = 42　　$I(s_{11}, s_{21})$ = 0.9183

对于 *major* = "*Engineering*"

s_{12} = 36　　s_{22} = 46　　$I(s_{12}, s_{22})$ = 0.9892

对于 *major* = "*Business*"

s_{13} = 0　　s_{23} = 42　　$I(s_{13}, s_{23})$ = 0

使用(5.5)式，如果样本根据 *major* 划分，则对给定的样本进行分类所需的期望信息是：

$$E(major) = \frac{126}{250} I(s_{11}, s_{21}) + \frac{82}{250} I(s_{12}, s_{22}) + \frac{42}{250} I(s_{13}, s_{23}) = 0.7873$$

因此，由这样的划分的信息增益是：

$$Gain(major) = I(s_1, s_2) - E(major) = 0.2115$$

类似地，我们可以对剩下的属性计算信息增益。对于每个属性，它们的信息增益按递增序分别是：*gender:* 0.0003，*birth_country:* 0.0407，*major:* 0.2115，*gpa:* 0.4490 和 *age_range:* 0.5971。假定我们用于识别弱相关性的属性相关阈值为 0.1。属性 *gender* 和 *birth_country* 的信息增益小于该阈值，因此被认为是弱相关的。这样，它们被删除。对比类也被删除，产生初始目标类工作关系。

第 4 步，按照图 5.1 算法，将面向属性归纳用于初始目标类工作关系。

5.4 挖掘类比较：区分不同的类

在许多应用中，人们可能对单个类（或概念）的描述或特征化不感兴趣，而希望挖掘一种描述，它将一个类（或概念）与其它可比较的类（或概念）相区分。类区分或比较（此后称为**类比较**）挖掘将目标类与对比类相区分的描述。注意，目标类和对比类必须是可比较的，意指它们具有相似的维或属性。例如，三个类 *person, address* 和 *item* 是不可比较的。然而，过去三年的销售是可比较的，计算机系的学生和物理系的学生同样是可比较的。

在前几节，我们关于类特征的讨论处理单个类中的多层数据的汇总和特征。所开发的技术应当能够扩充，处理多个可比较类的类比较。例如，可以修改类特征的属性泛化处理，使得泛化在所有比较类上同步地进行。这使得所有类的属性可以泛化到同一抽象层。例如，假定给定 1998 和 1999 年 AllElectronics 的销售数据，并希望比较这两个类。例如，考虑具有抽象层 *city, province_or_state* 和 *country* 的维 *location*。每个数据类都应当泛化到相同的 *location* 层。即，它们要同步地都泛化到 *city* 层，或 *province_or_state* 层，或 *country* 层。理想地，这种比较比用 1998 年 Vancouver 的销售和 1999 年 USA 的销售相比较（即，每个销售数据集泛化到不同的层次）更有用。然而，用户应当有选择，在他愿意时，用他自己的选择替代这种自动的同步比较。

5.4.1 类比较方法和实现

"如何进行类比较？"一般地，该过程如下：

1. **数据收集**： 通过查询处理收集数据库中相关数据集，并将它划分成一个目标类和一个或多个对比类。
2. **维相关分析**：如果有多个维并且希望解析类比较，则应当在这些类上进行 5.3 节介绍的维相关分析，并且在后面的分析中仅包含强相关的维。
3. **同步泛化**：泛化在目标类上进行，泛化到用户或专家指定的维阈值控制的层，产生**主目标类关系/方体**。对比类概念泛化到与主目标类关系/方体相同的层次，形成**主对比类关系/方体**。

4. **导出比较的提供**：结果类比较描述可以用表、图或规则的形式可视化。这种表示通常包括“对比”度量（如 *count%*），反映目标类和对比类的比较。如果需要，用户可以在目标类和对比类上使用下钻、上卷和其它 OLAP 操作，调整比较描述。

上面的讨论给出了在数据库中挖掘解析类比较算法的一般轮廓。与挖掘解析类特征的算法相比，上面的算法涉及目标类与对比类的同步泛化，使得这些类可以在同一抽象层同时进行比较。

“使用数据方技术，类比较挖掘可以有效地实现吗？”回答是肯定的——过程类似与 5.3.2 小节讨论的挖掘数据特征的实现。可以用一个标志指示一个元组是否代表目标类或对比类，其中标志可以看作数据方的一个维。由于目标类和对比类的其它所有维共享数据方的相同部分，同步泛化和特化可以通过数据方的上卷和下钻自动地实现。

下面的例子挖掘描述 Big-University 的研究生和本科生的类比较。

例 5.10　挖掘类比较。假定我们想比较 Big-University 的研究生和本科生的一般性质，给定了属性 *name, gender, major, birth_place, birth_date, residence, phone#*和 *gpa*。

该数据挖掘任务可以用 DMQL 表达如下：

```
use Big_University_DB
mine comparison as “grad_vs_undergrad_students”
in relevance to name, gender, major, birth_place, birth_date, residence, phone#, gpa
for “graduate_students”
where status in “graduate”
versus “undergraduate”
where status in “undergraduate”
analyze count%
from student
```

让我们看看这个典型的挖掘比较描述的数据挖掘查询如何处理。

首先，将查询转换成两个关系查询，收集两个任务相关的集合：一个是目标类工作关系，另一个是对比类工作关系，如表 5.7 和表 5.8 所示。这可以看作是构造数据方，其中状态{graduate, undergraduate}作为一个维，其它属性形成剩下的维。

其次，在两个数据类上进行维相关分析。分析后，不相关或弱相关的维，如 *name, gender, birth_place, residence* 和 *phone#*，从结果类删除。只有那些强相关的属性包含在其后的分析中。

再次，进行同步泛化：泛化在目标类上进行，泛化到用户或专家指定的维阈值控制的层，产生主目标类关系/方体。对比类概念泛化到与主目标类关系/方体相同的层次，形成主对比类关系/方体，如表 5.9 和表 5.10 所示。与本科生相比，研究生一般趋向于年龄稍大，GPA 较高。

表 5.7：初始工作关系（研究生）

name	gender	major	birth_place	birth_date	residence	phone#	pa
Jim Woodman	M	CS	Vancouver,BC,Canada	8-12-76	3511,Main St., Richmand	687-4598	3.67
Scott Lachance	M	CS	Montreal,Que,Canada	28-7-75	345, IstAve.,Vancouver	253-9106	3.70
Laura Lee	F	physics	Seattle,WA,USA	25-8-70	125,Austin Ave.,Burnaby	420-5232	3.83
…	…	…	…	…	…	…	…

表 5.8：初始工作关系（本科生）

name	gender	major	birth_place	birth_date	residence	phone#	gpa
Bob Schumann	M	Chemistry	Calgary,Alt,Canada	10-1-78	2642 Halifax St.Burnaby	294-4291	2.96
Amy Eau	F	Biology	Golden,BC,Canada	30-3-76	463Sunset Cres,Vancouver	681-5417	3.52
…	…	…	…	…	…	…	…

表 5.9　目标类的主泛化关系（研究生）

major	age_range	gpa	count%

Science	21...25	good	5.53%
Science	26...30	good	5.02%
Science	> 30	very_good	5.86%
...	...	...	...
Business	> 30	excellent	4.68%

表 5.10　对比类主泛化关系（本科生）

major	age_range	gpa	count%
Science		fair	5.53%
Science	16...20	good	4.53%
...	16...20	...	...
Science	...	good	2.32%
...	26...30	...	...
Business	...	excellent	0.68%
	> 30		

最后，结果类比较描述以表、图和/或规则的形式提供。这种直观表示包括对比度量（如 count%），比较目标类和对比类。例如， 5.02%的研究生出生在加拿大，年龄在 26 到 30 之间，GPA 为“good”，而只有 2.32%的本科生具有这种性质。如果必要，用户可以在目标类和对比类上进行上、下钻和其它 OLAP 操作，调整最终描述的抽象级。□

5.4.2 类比较描述的表示

“类比较描述如何可视化？”如同类特征，类比较也可以以多种形式向用户提供，包括泛化关系、交叉表、条形图、饼图、曲线和规则。除逻辑规则外，类比较与类特征所用的形式相同。本小节，我们讨论以判别规则的形式显示类比较。

与特征描述类似，比较描述中的目标类和对比类的区分特性也可以用量化区分规则量化地描述。量化区分规则对描述中每个泛化元组附上一个统计兴趣度度量 *d*-权。

设 q_a是一个泛化元组，而 C_j是目标类。其中，q_a覆盖目标类的某些元组。注意，q_a也可能覆盖对比类的某些元组，因为我们处理的是比较描述。q_a的 ***d*-权**是初始目标类工作关系中被 q_a覆盖的元组数与初始目标类和对比类工作关系中被 q_a覆盖的元组数的比。形式地，q_a关于 C_j的 *d*-权定义为

$$d\_weight = count(q_a \in C_j) / \sum_{i=1}^{m} count(q_a \in C_i) \tag{5.7}$$

其中，m是目标类和对比类的个数，C_j在$\{C_1, ..., C_m\}$中，而 count($q_a \in C_i$) 是类 C_i中被 q_a覆盖的元组数。*d*-权的取值范围为[0.0,1.0]（或[0%,100%]）。

高 *d*-权的目标类表明被泛化元组代表的概念主要从目标类导出，而低 *d*-权表明概念主要从对比类导出。可以设定一个阈值，根据 *d*-权或如 5.2.3 小节介绍的其它度量控制有趣规则的显示。

例 5.11　在例 5.10 中，假定对泛化元组计数，由表 5.9 和表 5.10 得到泛化元组 *major* = "*Science*"，*age_range* = "21...25"，　*gpa* = "*good*"的计数分布，如表 5.11 所示。

表 5.11：泛化元组研究生和本科生的计数分布

status	major	age_range	gpa	count
graduate	Science	21...25	good	90
undergraduate	Science	21...	good	210

	e	25	ood 10

给定的泛化元组的 *d*-权关于目标类是 90/（90+210）= 30%，关于对比类是 210/（90+210）= 70%。即，如果一个学生专业是 *Science*，年龄在 21 和 26 之间，*gpa* 为“*good*”，则基于给定的数据，他是研究生的概率为 30%，是本科生的概率为 70%。类似地，也可以导出表 5.9 和表 5.10 其它泛化元组的 *d*-权。□

关于给定比较描述的目标类的**量化区分规则**记作

$$\forall X,\ target\_class(X) \Leftarrow condition(X) \qquad [d:\ d\_weight] \tag{5.8}$$

其中，条件由描述的泛化元组形成。该规则不同于类特征得到的规则，那里的蕴涵箭头是从左向右。

例 5.12 根据例 5.11 中的泛化元组和计数分布，目标类 *graduate_student* 的量化区分规则可以表示如下：

$$\forall X, status(X) = "graduate\_student" \Leftarrow major(X) = "Science" \wedge age\_range(X) = "21...25" \wedge gpa(X) = "good" \quad [d:30\%] \tag{5.9}$$

□

注意，对于一个对象在目标类中，区分规则给出了充分条件，但不是必要条件。例如，规则(5.9)表明如果 *X* 满足条件，则 *X* 是研究生的概率为 30%。然而，给定 *X* 是研究生，这并不表明 *X* 满足条件的可能性。这是因为，尽管满足条件的元组在目标类中，其它不满足条件的元组也可能在目标类中，因为规则可能并不涵盖数据库中目标类的所有实例。因此，条件是充分的，但不是必要的。

5.4.3 类描述： 提供特征和比较

“既然类特征和类比较是形成类描述的两个方面，我们能在同一个表或同一个规则中提供二者吗？”事实上，只要我们清楚地理解 *t*-权和 *d*-权度量，并且能够正确地解释它们，就没有其它困难在同一个表里表示它们。让我们考察一个在同一个交叉表表示类特征和类区分的例子。

例 5.13 表 5.12 是一个交叉表，显示 AllElectronics 1999 年销售的 TV 和计算机的总数（单位：千台）。

表 5.12：TV 和计算机 1999 年销售总量的交叉表（单位：千台）

location/item	TV	*computer*	*both_item*
Europe	80	240	320
North_America	120	560	680
both_region	200	800	1000

设 *Europe* 是目标类，*North_America* 是对比类。两个类之间的销售分布的 *t*-权和 *d*-权提供在表 5.13 中。根据该表，对于一个给定的类（例如，目标类 *Europe*），一个泛化元组或对象（例如，*item*=“TV”)的 *t*-权表明该元组是给定类元组的可能性有多大(例如,欧洲的销售 TV 占多大比例？）。元组的 *d*-权表明给定（目标或对比）类的元组与其对手相比，有多大区别（例如，欧洲的 TV 销售与北美相比，情况如何？）。

表 5.13：与表 5.12 相同的交叉表，但同时显示每个类相关的 *t*-权和 *d*-权

location/item	TV	*computer*	*both_item*

	count	*t*-权	*d*-权	*count*	*t*-权	*d*-权	*count*	*t*-权	*d*-权
Europe	80	25%	40%	40	2%	0%	20	75%	3%
North_America	20	1.65%	60%	60	5.35%	0%	80	82%	8%
both_region	200	20%	100%	100	80%	100%	1000	1%	1%

例如，"(*Europe*,TV)"的 *t*-权是 25%，因为欧洲的 TV 销售量(80,000)只占欧洲两种商品销售量(320,000)的 25%。"(*Europe*,TV)"的 *d*-权是 40%，因为欧洲的 TV 销售量(80,000)占目标类和对比度类，即欧洲和北美 TV 销售量(200,000)的 40%。□

注意，交叉表 5.13 中的计数度量遵守交叉表的一般性质：对于计数，每行和每列计数值的和分别等于对应的两种商品和两个地区中的总和。然而，*t*-权和 *d*-权度量不遵守这一性质。正如我们在例 5.13 中解释的，这些度量的语义都不同于计数。

"量化特征规则和量化区分规则可以一起用一个规则的形式表示吗？"回答是可以——同一个类的量化特征规则和量化区分规则可以结合在一起，形成该类的量化描述规则，它显示与对应的特征和区分规则相关联的 *t*-权和 *d*-权。为看清如何做，让我们快速地回顾一下量化特征规则和量化判定规则如何表示。

正如在 5.2.3 小节讨论的，量化特征规则提供了给定目标类的必要条件，因为对于每个可能在目标类出现的性质，它提供了一种概率度量。这种规则的形式为

$$\forall X, target\_class(X) \Rightarrow condition_1(X)[t:w_1] \vee \ldots \vee condition_m(X)[t:w_m] \quad (5.10)$$

其中，每个条件表示目标类的一个性质。该规则指出，如果 *X* 在 *target_class* 中，则 *X* 满足 $condition_i$ 的概率是 *t*-权 w_i 的值。这里，*i* 在{1,...,*m*}中。

正如前面在 5.4.1 小节所讨论的，一个量化区分规则提供了目标类的一个充分条件，因为它提供的是一个出现在目标类和出现在对比类的比例的量化度量。这种规则的形式为

$$\forall X, target\_class(X) \Leftarrow condition_1(X)[d:w_1] \vee \ldots \vee condition_m(X)[d:w_m]$$

该规则指出，如果 *X* 满足 $condition_i$，则 *X* 在 *target_class* 中的可能性（d-权值）为 w_i。这里，*i* 在{1,...,*m*}中。

一个给定类的量化特征规则和量化区分规则可按如下方法结合，形成一个**量化描述规则**：（1）对于每个条件，显示相关联的 *t*-权和 *d*-权，并且（2）在给定的类和条件之间使用双向箭头。即，量化描述规则形如

$$\forall X, target\_class(X) \Leftrightarrow condition_1(X)[t:w_1, d:w_1'] \vee \ldots \vee condition_m(X)[t:w_m, d:w_m'] \quad (5.11)$$

该规则指出，对于从 1 到 *m*，*X* 在 *target_class* 中，则 *X* 满足 $condition_i$ 的可能性是 w_i；而 X 满足 $condition_i$，则 *X* 在 *target_class* 中的可能性是 w_i'。

例 5.14 将例 5.13 的交叉表表 5.13 转换成量化描述规则形式的类描述是直接了当的。例如，对于目标类 *Europe*,，其量化描述规则为

$$\forall X, location(X) = "Europe" \Leftrightarrow (item(X) = "TV")[t:25\%, d:40\%] \vee (item(X) = "computer")[t:75\%, d:30\%] \quad (5.12)$$

该规则表明，对于 1999 年 *AllElectronics* 的 TV 和计算机销售，如果一个这样的商品在欧洲售出，则该商品是 TV 的概率为 25%，而是计算机的概率为 75%。另一方面，如果我们比较欧洲和北美的销售，则 40%的 TV 在欧洲销售（由此，我们推出 60%的 TV 在北美销售）。此外，关于计算机销售，30%的销售在欧洲。□

5.5 在大型数据库中挖掘描述统计度量

在本章的前面，我们讨论了流行的度量（如 count, sum 和 average）下的类描述。关系数据库系统提供了五种内部聚集函数：count(), sum(), avg(), max()和 min()。这些函数在数据方中也可以有效地（以渐增和分布的方式）计算。在多维数据的描述挖掘中，包含这些函数作为基本度量并不成问题。

然而，对于许多数据挖掘任务，用户更希望了解关于数据的中心趋势和发散特征。中心趋势的度量包括 *mean, median, mode* 和 *midrange*，而数据发散度量包括 *quartiles, outliers, variance* 和其它统计度量。这些描述性统计对于理解数据的分布很有帮助。这些度量在统计界已广泛研究。然而，从数据挖掘的角度，我们需要考察在大型多维数据库中如何有效地计算它们。

5.5.1 度量中心趋势

数据集最常用、最有效的 “中心”数值度量是（算术）平均值。设 $x_1, x_2, \ldots, x_n$ 是值或观测的集合。该值集的**平均值**是

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.13}$$

这对应于关系数据库系统提供的内部聚集函数 *average*（SQL 中, avg()）。在大部分数据方中，*sum* 和 *count* 在预计算时保存。这样，使用公式 *average = sum/count* 导出 *average* 是直接了当的。

有时，集合中每个值 x_i 与一个权 w_i 相关联，$i = 1, \ldots, n$。权反映对应的值的意义、重要性或出现频率。在这种情况下，我们可以计算

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{5.14}$$

这称为**加权算术平均**或**加权平均**。

在第 2 章,度量被定义为代数的,如果它能由分布聚集度量计算。由于 avg()可以被 sum()/count()计算，其中，sum()和 count()都是分布聚集度量，能够以分布方式计算，因而 avg()是代数度量。可以验证加权平均也是代数度量。

尽管平均值是我们用于描述数据集合最有用的单个量，但它不是唯一的，并非总是最好的度量数据集合中心的方法。对于倾斜数据，数据中心较好的度量是中位数 *M*。假定数据集合的值是数值有序的。如果值的个数 *n* 是奇数，则**中位数**是有序集合的中间值；否则（即，如果 *n* 是偶数），中位数是中间两个数的平均值。

根据第 2 章度量的分类，中位数既不是分布度量，也不是代数度量——它是一个整体度量。即，它不能用以下方法计算：将值的集合任意地划分成小的子集，独立地计算它们的中位数，然后合并每个子集的中位数。相反，count(),sum(),max()和 min()可以用这种方式计算（是分布度量），因而比中位数容易计算。

尽管在大型数据库中不容易计算准确的中位数值，但是可以有效地计算一个近似的中位数。例如，对于分组数据，由插值得到的中位数由下式给出

$$median = L_1 + (\frac{n/2 + (\sum f)_l}{f_{median}})c \tag{5.15}$$

其中，L_1 是包含中位数的较低的类边界（即，最小值），n 是数据中值的个数，$(\Sigma f)_l$ 是低于中位数类的所有类的频率和，f_{median} 是中位数类的频率，而 c 是中位数类的区间长度。

另一种中心趋势度量是模。数据集合的**模**是集合中出现频率最高的值。可能最高频率对应多个不同值，导致多个模。具有一个、两个、三个模的数据集合分别称为**单模态**、**双模态**和**三模态**。具有两个或更多模的数据集合是**多模态**。在另一种极端情况下，如果每个数据值仅出现一次，则它没有模。

对于适度倾斜（不对称的）的单模态频率曲线，我们有下面的经验关系

$$mean - mode = 3 \times (mean - median) \tag{5.16}$$

这意味如果平均值和中位数已知，适度倾斜的单模态频率曲线的模容易计算。

中列数，即数据集合的最大和最小值的平均值，可以用于度量数据集合的中心趋势。使用 SQL 的聚集函数 max()和 min()计算中列数是平凡的。

5.5.2 度量数据的发散

数值数据趋向于发散的程度称为数据的**发散度**或**方差**。数据发散度的最常用度量是五数概括(基于四分位数）、中间四分位数区间和标准偏差。盒图的绘制（展现局外者值）也用作一种有用的图形方法。

四分位数、局外者和盒图

数值序下的数据集合的第 k 个**百分位数**是具有如下性质的值 x：数据项的百分之 k 在 x 上或低于 x。在中位数 M（上一小节讨论过）上或低于 M的值对应于第 50 个百分位数。

除中位数外，最常用的百分位数是四分位数。第一个四分位数记作 Q_1，是第 25 个百分位数；第三个四分位数记作 Q_3，是第 75 个百分位数。四分位数与中位数一起给出中心、发散和分布形状的某种指示。第一个和第三个四分位数之间的距离是分布的一种简单度量，它给出被数据的中间一半所覆盖的范围。该距离称为**中间四分位数区间**（IQR），定义为

$$IQR = Q_3 - Q_1 \tag{5.17}$$

我们应当明白，对于描述倾斜分布，单个分布数值度量（如 IQR）不是非常有用的。倾斜分布两边的分布是不等的。因此，还提供两个四分位数 Q_1和 Q_3，以及中位数 M信息更丰富。一个识别**局外者**的常用规则是：挑出落在至少高于第三个四分位数或低于第一个四分位数 1.5×IQR 处的值。

因为 Q_1、M 和 Q_3不包含数据端点的信息，分布形状的更完整的概括可以通过同时也提供最高和最低数据值得到。这称作五数概括。分布的**五数概括**由中位数 M，四分位数 Q_1和 Q_3，最小和最大观测值组成，按以下次序写出 $Minimum, \ Q_1, \ M, \ Q_3, \ Maximum$。

分布的一种流行的直观表示是盒图。在盒图中：

- 典型地，盒的端点在四分位数上，使得盒的长度是中间四分位数区间 IQR。
- 中位数用盒内的线标记。
- 盒外的两条线（称作胡须）延伸到最小（$Minimum$）和最大（$Maximum$）观测值。

当处理数量适中的观测值时，值得个别绘出潜在的局外者。在盒图中做，仅当这些值超过四分位数不到 1.5×IQR 时，胡须扩展到最高和最低观测值。否则，胡须在在出现在四分位数的 1.5×IQR 之内的最极端的观测值处终止，剩下的情况个别绘出。盒图可以用来比较若干可比较的数据集。图 5.5 给出在给定的时间段，在 AllElectronics 的 4 个分店销售的商品单价的盒图。对于分店 1，我们看到销售商品的中位数是\$80，Q_1是\$60，Q_3是\$100。

根据与 5.5.1 小节中位数分析的类似理由，我们可以断定 Q_1和 Q_3是整体度量，IQR 也同样。对于大型数据集的挖掘，盒图的有效计算，甚至近似的盒图都是令人感兴趣的。

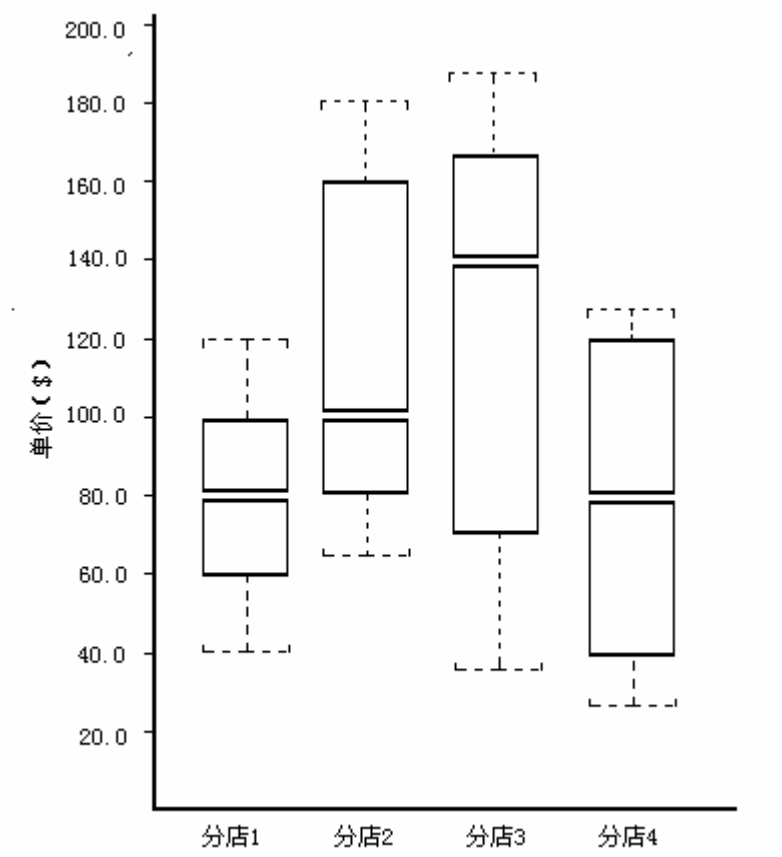


图 5.5：在给定的时间段，AllElectronics 的 4 个分店销售的商品单价的盒图

方差和标准差

n 个观测值 $x_1, x_2, \ldots, x_n$ 的**方差**是

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}[\sum_{i=1}^{n}x_i^2 - \frac{1}{n}(\sum_{i=1}^{n}x_i)^2] \tag{5.18}$$

标准差 s 是方差 s^2 的平方根。

标准差 s 作为发散的度量，其基本性质是：

- s 度量关于平均值的发散，仅当选择平均值作为中心度量时使用。
- 仅当不存在发散时，即当所有的观测值都相同时，$s = 0$。否则，$s > 0$。

注意，方差和标准差是代数度量，因为 n（是 SQL 的 count()），Σx_i（是 x_i 的 sum()），而 Σx_i^2（是 x_i^2 的 sum()）都可以以任何划分进行计算，然后合并形成(5.18)式。这样，两个度量的计算在大型数据库都是可规模化的。

表 5.14：在 *AllElectronics* 的一个分店销售的商品单价数据

单价($)	商品销售量
40	275
43	300
47	250
...	...
74	360
75	515
78	540
...	...
115	320
117	270
120	350

5.5.3 基本统计类描述的图形显示

除本章前面介绍的条形图、饼图和线图之外，还有一些常用的图用于显示数据汇总和分布，这包括直方图、分位数图、q-q 图、散布图和局部回归（loess）曲线。

直方图，或频率直方图是一种单变量图形方法。直方图由一组矩形组成，这些矩形反映类在给定数据中出现的计数或频率。每个矩形的基在水平轴上，中心是“类”标记，基的长度等于类的宽度。通常，类的宽度是一致的，类定义为分类属性的值，或离散化连续属性的等宽区间。在这种情况下，每个矩形的高等于它代表的类的计数或相对频率，并且也称直方图为**条形图**。连续属性类也可以用不等宽的区间定义。在这种情况下，对于给定的类，类的宽度等于区间的宽度，而矩形的高是类的密度（即，类的计数或相对频率除以类的宽度）。构造直方图的划分规则在第 3 章已讨论。

图 5.6 给出表 5.14 数据的直方图，其中，类定义成等宽的，代表增量$20，频率是商品的销售数量。直方图至少有一个世纪了，是一种广泛使用的单变量图形方法。然而，对于比较单变量观测组，它可能不如分位数图、q-q 图和盒图方法有效。

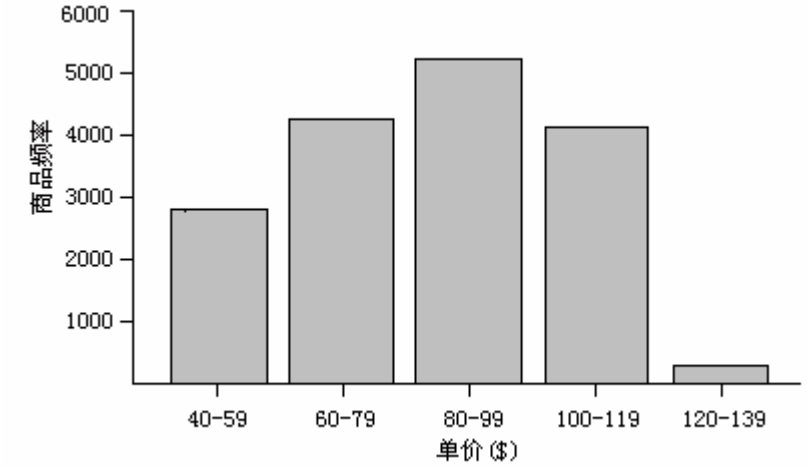


图 5.6：表 5.14 数据集的直方图

分位数图是一种观察数据分布的简单有效的方法。首先，它显示所有的数据（允许用户评估总的情况和不寻常的出现）。其次，它绘出了分位数信息。此步使用这种机制与百分位数计算稍微有点不同。设 $x_{(i)}$ (i = 1,...,n)是由小到大排序的数据，使得 $x_{(1)}$是最小的观测值，而 $x_{(n)}$是最大的。每个观测值 $x_{(i)}$与一个百分数 f_i配对，指出大约 100 f_i %的数据小于等于 $x_{(i)}$。我们说“大约”，因为可能没有一个精确的小数值 f_i，使得数据的 f_i%小于或等于 $x_{(i)}$。注意，0.25 分位数对应于 Q_1，0.50 分位数对应于中位数，而 0.75 分位数对应于 Q_3。设

$$f_i = \frac{i - 0.5}{n}$$

这些数由 1/2n（稍大于 0）到 1-1/2n（稍小于 1），以相同的步长 1/n 递增。在分位数图中，$x_{(i)}$对着 f_i画出。这使得我们可以基于分位数，比较不同的分布。例如，给定两个不同时间段销售数据的分位数图，我们一眼就可以比较它们的 Q_1，中位数，Q_3，以及其它 f_i值。图 5.7 给出表 5.14 单价数据的分位数图。

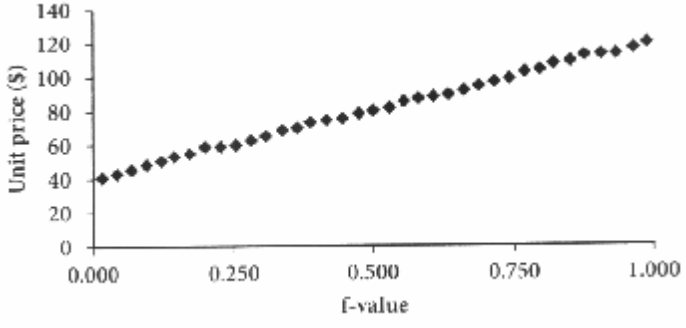


图 5.7　表 5.14 单价数据的分位数图

分位数-分位数图，或 q-q 图对着另一个的对应分位数，绘制一个单变量分布的分位数。它是一种强有力的直观表示工具，使得用户可以观察从一个分布到另一个是否有移位。

假定对于变量单价，我们有两个观测集，取自两个不同的分店。设 $x_{(1)}, \ldots, x_{(n)}$是取自第一个分店的数据，$y_{(1)}, \ldots, y_{(m)}$是取自第二个分店的数据，每组数据都已按递增序排序。如果 $m = n$（即，每个集合中的点数相等）则我们简单地对着 $x_{(i)}$画 $y_{(i)}$，其中，$y_{(i)}$和 $x_{(i)}$都是它们的数据集的第$(i - 0.5)/n$个分位数。如果 $m < n$（第二个分店的的观测值比第一个少），则可能只有 m 个点在 q-q 图中。这里，$y_{(i)}$是 y 数据的第$(i - 0.5)/m$个分位数，对着 x 数据的第$(i - 0.5)/m$个分位数画。典型地，该计算涉及插值。

图 5.8 给出在给定的时间段，AllElectronics 的两个不同分店销售的商品单价数据的分位数-分位数图。对于每个数据集，左下角的点对应相同的分位数 0.03。（为帮助比较，我们也画了一条直线，它代表对于给定的分位数，两个分店的单价相同的情况。此外，加黑的点分别对应 Q_1、中位数和 Q_3。）例如， 我们看到，在分位数 0.03，在分店 1 销售的商品单价比分店 2 稍低。换言之，在分店 1 销售的商品 3%低于或等于\$40，而在分店 2 销售的商品 3%低于或等于\$42。在最高分位数，我们看到分店 2 的商品单价稍微低于分店 1。一般地，我们注意到分店 1 的分布相对于分店 2 有一个移位，分店 1 销售的商品单价趋向于比分店 2 低。

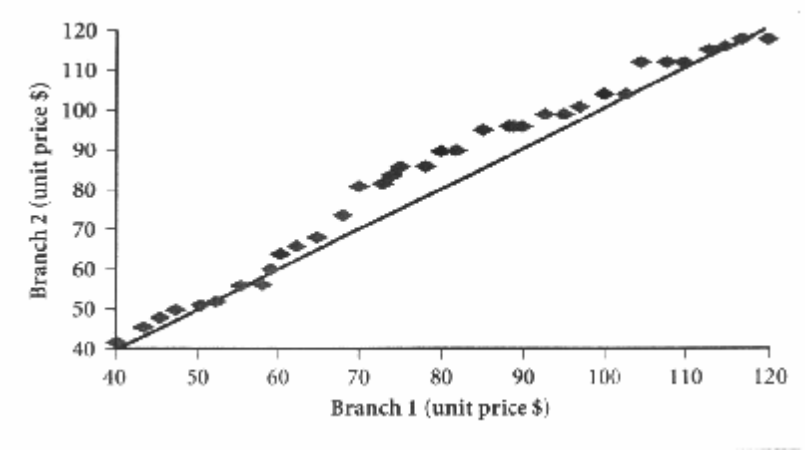


图 5.8　两个不同分店的单价数据的分位数-分位数图

散布图是确定两个量化变量之间看上去是否有联系、模式或趋势的最有效的图形方法之一。为构造散布图，每个值对视为一个代数坐标对，并作为一个点画在平面上。散步图是一种有用的探查方法，一眼就看出双变量数据在整个平面上如何分布，例如，点的聚类、例外者等。图 5.9 给出表 5.14 中数据的散布图。

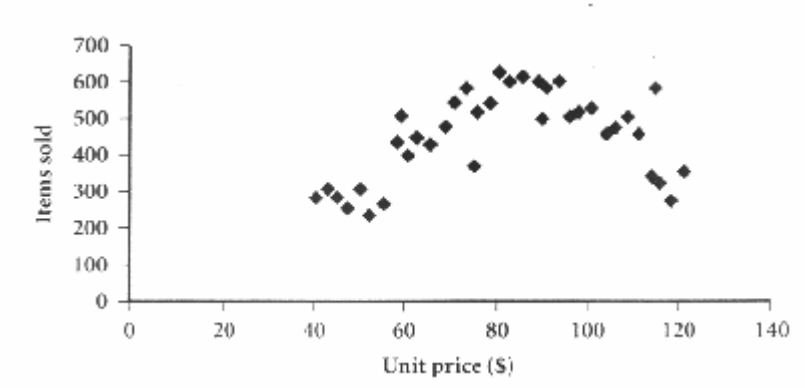


图 5.9　表 5.14 中数据集的散布图

loess 曲线是另一种重要的探查图形工具，它添加一条平滑曲线到散布图，以便更好地理解依赖模式。loess 一词是“局部回归”（local regression）的缩写。图 5.10 给出表 5.14 中数据的 loess 曲线。

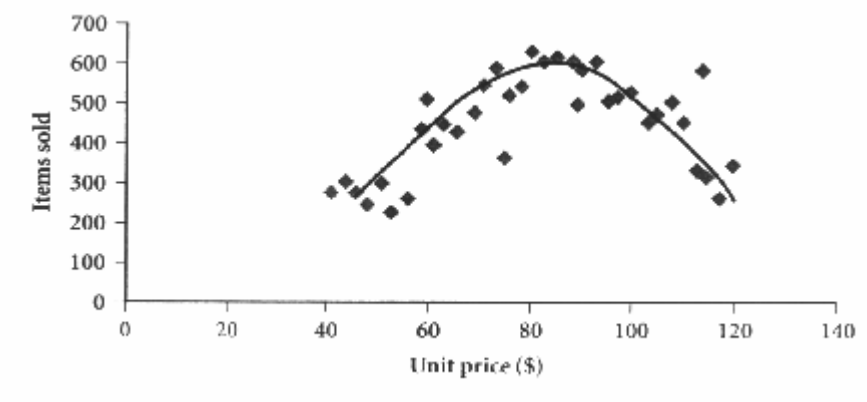


图 5.10　表 5.14 中数据集的 loess 曲线

为了拟合 loess 曲线，需要设置两个参数——平滑参数α，被回归拟合的多项式的阶λ。α可以是任意正数（典型值在 1/4 和 1 之间），而λ可以是 1 或 2。选择α的目的是产生一个拟合，它尽可能平滑，而不过份破坏数据中潜在的模式。曲线随α增大而变得平滑。然而，可能出现拟合不足，表明可能“丢失”数据模式。如果α太小，跟踪了潜在的模式，但可能过份适合数据，曲线中的局部“摆动”可能不被数据支持。如果数据的潜在模式具有“温和的”曲率，而没有局部极大和极小，则局部线性拟合就足够了（λ = 1）。然而，如果有局部极大和极小，则二次拟合（λ = 2）一般做得更好，它遵循数据模式并且保持局部平滑性。

5.6　讨论

我们已经为挖掘大型数据库中的概念或类描述提出了一些可规模化的方法。本节，我们讨论关于这些描述的相关问题。包括基于数据方和面向属性归纳与典型的机器学习方法的比较，概念描述的增量和并行挖掘的实现，概念描述的兴趣度度量。

5.6.1　概念描述：与典型的机器学习方法比较

本章，我们介绍了挖掘大型数据库中概念描述的面向数据库的方法。这些方法包括基于数据方的和面向属性归纳的概念描述数据泛化方法。自 80 年代以来，机器学习界已经提出了一些有影响的概念描述方法。典型的概念描述机器学习方法遵循示例学习的范例。一般地，这些算法在概念或标定类训练样本集上运行，导出或学习描述学习类的假定。

“示例学习方法与这里介绍的数据挖掘方法之间的主要区别是什么？”首先，机器学习和数据挖掘方法的哲学体系不同，它们关于概念描述问题的基本假定也不同。在机器学习开发的大部分示例学习算法中，分析样本划分成两个集合：正样本和负样本，分别代表目标类和对比类。学习过程随机地选取一个正样本，并用它形成描述该类对象的一个假定。然后，学习过程使用其余正样本在假定上进行泛化，并使用负样本进行特化。一般地，结果假定涵盖所有的正样本，但不涵盖负样本。

通常，数据库不显式存放否定数据。这样，特化时就没有显式说明的负样本可用。这就是为什么对于解析特征挖掘和比较挖掘要收集不在目标类（正样本）中的可比较数据，用作反面数据（5.3 和 5.4 节）。因此，大部分面向数据库方法倾向基于泛化。尽管它们多半提供下钻（特化）操作，但该操作的实现本质上是回溯泛化过程。

其次，概念描述的机器学习和面向数据库的方法之间的另一个主要不同是训练样本集的大小。对于传统的机器学习方法，数据样本集通常比使用面向数据库技术进行数据分析所用的数据样本集小。因此，对于机器学习方法，容易找到涵盖所有的正样本，而不涵盖任何负样本的描述。然而，考虑存放在现实世界数据库中的数据的发散性和数量巨大，这种数据分析多半不会导出涵盖所有正样本，而不涵盖任何负样本的规则或模式。我们可以期望找到的是一组特性或规则，它们涵盖正类中数据的大部分，最大限度地区分正样本和负样本。这也可以看作概率分布。

第三，关于所用的泛化方法，机器学习和面向数据库的方法还存在差别。两种方法确实都将属性删除和属性泛化作为它们的主要泛化技术使用。考虑训练样本集是元组的集合。机器学习方法逐个元组进行泛化，而面向数据库的方法逐个属性（或整个维）进行泛化。

在机器学习方法的逐个元组泛化策略中，一次考察一个训练样本，导出泛化概念。为形成与所有正样本一致，而与负样本都不一致的最特殊的假定（或概念描述），算法必须搜索空间中的每个结点，这些结点代表由每个训练样本泛化导出的可能概念。由于元组的不同属性可能泛化到不同的抽象层，给定训练样本的搜索结点数目将涉及大量的可能组合。

另一方面，在泛化的早期阶段，使用逐个属性策略的数据库方法对数据关系的所有元组在每个属性或维上一致地进行泛化。本质上，这种方法将注意力聚焦在单个属性，而不是属性组合上。这称作分解解释空间，这里，**解释空间**定义为与训练样本一致的假定子集。分解解释空间可以显著地提高计算性能。假定用于泛化的概念分层有 k 个，每个概念分层中有 p 个结点。k 个分解的解释空间大小为 $p \times k$。与此相比，对于相同的概念树，被机器学习方法搜索的未分解的解释空间为 p^k。

注意，在泛化的早期阶段，探查给定大量元组的不同属性-值条件的许多可能组合，这种算法不可能是多产的，因为这些组合在进一步泛化时最终将被合并。不同的可能组合只有在关系先被泛化成相对较小的关系时才需要探查，正如本章介绍的面向数据库的方法做的那样。

与其它机器学习算法相比，面向属性的方法另一个明显的优点是数据挖掘过程与面向集合的数据库操作的集成。与其它已有的不利用数据库机制的学习算法不同，面向属性归纳基本采用关系操作，如选择、连接、投影（提取任务相关数据和删除属性）元组置换（攀升概念树）和排序（在类中发现公共元组）。这些关系操作是面向集合的，并通常在大部分数据库系统中是优化的。这样，面向属性的方法不仅有效，而且易于输出到其它关系系统。这些解释同样适用于基于数据方的泛化算法。通过结合稀疏方技术、各种方计算方法、索引和存取技术，基于数据方的方法使用比传统的数据库查询处理技术更优化的技术。这样，在处理大型数据集时，面向数据库的算法获得优于机器学习技术的高性能是在预料之中的。

5.6.2 概念描述的增量和并行挖掘

给定数据库中的大量数据，更可取的是渐增地更新数据挖掘结果，而不是从每次数据库更新的结果挖掘。对于许多种类的大型数据库或数据仓库挖掘，增量数据挖掘是一个诱人的目标。幸而，扩充面向数据库的概念描述算法，进行增量数据挖掘是直接了当的。

“我们如何扩充面向属性的归纳，用于增量数据挖掘？”假定泛化关系 R 存放在数据库中。当一批新元组 ΔDB 插入数据库时，可以在 ΔDB 上进行面向属性的归纳，将属性泛化到与泛化关系 R 的对应属性相同的概念层。相关的聚集信息，如计数、求和等可以通过将泛化算法用于 ΔDB 来计算。然后，在 ΔDB 上导出的泛化关系 ΔR 容易合并到泛化关系 R，因为 R 和 ΔDB 具有相同的维，并且每个维在相同的抽象层上。并 $R \cup \Delta R$ 成为新的泛化关系 R'。如果需要，可以按照用户说明对 R' 进行微调，如维泛化或特化。增量删除可以用类似的方法进行。细节留作习题。

基于相同的思想，可以研究概念描述的选样方法、并行算法和分布式算法。例如，面向属性的归纳可以通过从大量任务相关数据中选取一个数据子集进行，或者先在任务相关数据集的分划上并行执行归纳，然后合并泛化结果。

5.7 总结

- 数据挖掘可以分成描述式数据挖掘和预测式数据挖掘。**概念描述**是描述式数据挖掘的最基本形式。它以简洁汇总的形式描述给定的任务相关数据集，提供数据的有趣的一般特性。
- 概念（或类）描述由**特征**和**比较**（**区分**）组成。前者汇总并描述称作**目标类**的数据集，而后者汇总并将一个称作**目标类**数据集与称作**对比类**的其它数据集相区别。
- 概念特征有两种一般方法：**基于数据方 OLAP 的方法**和**面向属性归纳**方法。二者都是基于属性或维泛化的方法。面向属性归纳可以用关系或数据方结构实现。
- **面向属性归纳方法**包含以下技术：数据聚焦、通过属性删除或属性泛化泛化数据、计数和聚集值累计、属性泛化控制和泛化数据可视化。

- 泛化数据可以用多种形式**可视化**，包括泛化关系、交叉表、条形图、饼图、数据方视图、曲线和规则。下钻和上卷可以交互地在泛化数据上进行。
- **解析特征/比较**进行属性或维相关分析，在归纳处理之前过滤掉不相关和弱相关属性。
- **概念比较**可以用类似于概念特征的方式，使用面向属性归纳或数据方方法进行。可以量化地比较和对比从目标类和对比类泛化的元组。
- 特征和比较描述（形成概念描述）可以在同一个泛化关系、交叉表或量化规则中直观表示，尽管他们以不同的兴趣度度量显示。这些度量包括 **t-权**（元组的典型性）和 **d-权**（元组的可区分性）。
- 从统计学角度，可以使用**统计度量**描述数据的中心趋势和发散。四分位数、变差和局外者也是有用信息，可以从数据库挖掘。盒图、分位数图、散布图和分位数-分位数图是描述式挖掘中有用的可视化工具。
- 与机器学习算法相比，面向数据库的概念描述导致在大型数据库和数据仓库中的有效性和可规模性。
- 对基本方法稍加扩充，概念描述挖掘可以**增量**地、**并行**地、或**分布**地进行。

习题

5.1 对于类特征.基于数据方的实现与诸如面向属性归纳的关系实现之间的主要不同是什么？讨论哪种方法最有效，在什么条件下最有效。

5.2 假定下面的表从面向属性归纳导出。

class	birth_place	count
Programmer	Canada	180
	orthers	0
DBA	Canada	120
	orthers	0

(a) 将该表转换成显示相关 *t*-权和 *d*-权的交叉表。

(b) 将类 *Programmer* 转换成（双向的）量化描述规则。例如，$\forall X$, $Programmer(X) \Leftrightarrow (birth\_place(X) = \text{“}Canada\text{”} \wedge ...)[t: x\%, d: y\%]...\vee[t: w\%, d: z\%]$。

5.3 讨论为什么需要解析特征和如何进行。比较两种归纳方法的结果：（1）包含相关分析和（2）不包含相关分析。

5.4 对于数据发散特征，另外给出三个常用统计度量（未在本章解释），并讨论如何在大型数据库中有效地计算它们。

5.5 假定分析数据包含属性 age。数据元组的 age 值（以递增序）是：13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70

(a) 该数据的平均值是多少？中值是多少?

(b) 该数据的模是多少？评论数据的模（即，双模、三模等）。

(c) 数据的中间区间是什么？

(d) 你能找出（粗略地）数据的第一个四分位数（Q1）和第三个四分位数（Q3）吗？

(e) 给出数据的五数概括。

(f) 画出数据的盒图。

(g) 分位数-分位数图与分位数图的不同之处是什么？

5.6 给定由数据库 *DB* 导出的泛化关系 *R*，假定元组的集合ΔDB需要从 *DB* 中删除。简要给出用于 *R* 的必要删除的更新过程。

5.7 简要给出挖掘解析类比较的基于数据方的增量算法。

5.8 简要给出数据方环境下数据发散统计度量挖掘的（1）并行，（2）分布式方法。

文献注释

在计算机出现之前，泛化与汇总已在统计界研究。统计描述数据挖掘方法的总结包括Cleveland[Cle93]和 Devore[Dev95]。基于泛化的归纳技术，如示例学习，在数据挖掘研究开始之前已在机器学习界提出和研究。归纳学习的理论和方法由Michalski[Mic83]提出。示例学习方法由Michalski[Mic83]提出。解释空间由Mitchell[Mit77, Mit82]提出。5.6.1小节介绍的分解解释空间方法由 Subramanian 和 Feigenbaum [SF86b]提供。机器学习的一般介绍可以在 Dietterich 和Michalski[DM83]，Michalski, Carbonell 和Mitchell[MCM86] 和Mitchell [Mit97]中找到。

基于数据方的泛化技术最初由E. E. Codd, S. B. Codd 和Salley[CCS93]提出，并在许多基于OLAP的数据仓库中实现。Gary, Chaudhuri, Bosworth, Layman 等[GCB+97]提出了数据方中计算聚集的方操作。最近，关于有效的数据方计算有许多研究，对泛化的有效计算作出了贡献。该课题的一个全面综述可以在Chaudhuri 和Dayal[CD97]中找到。

概念描述的面向数据库方法开发描述数据库和数据仓库中大量数据的可扩展的和有效的技术。本章介绍的面向属性的归纳方法首先由 Cai, Cercone 和 Han[CCH91]提出，并被 Han, Cai 和Cercone[HCC93],Han 和Fu[HF96],Carter 和Hamilton[CH98],Han, Nishio, Kawano 和Wang [HNKW98]扩充。

有许多方法评估属性相关性，每个都有自己的偏爱。信息增益度量偏向于具有许多值的属性。许多替代已经提出，包括增益率（Quinlan[Qui93]），它考虑每个属性值的可能性。其它相关性度量包括Gini 索引（Breiman, Frieman, Olshen 和Stone[BFOS84]），χ^2应急表统计和不确定系数（Johnson 和 Wichern[JW92]）。关于判定树归约的属性选择度量比较， 见 Buntine 和Niblett[BN92]。关于另外的方法，见Liu 和Motoda[LM98ab]，Dash 和Liu[DL97]，Almuallim 和Dietterich[AD91]。

关于使用盒图、分位数图、分位数-分位数图、散布图和loess 曲线的基于统计的数据可视化，见Cleveland[Cle93]和Devore[Dev95]。Knorr 和Ng[KN98]研究了定义和计算局外者的一致方法。

第六章 挖掘大型数据库中的关联规则

关联规则挖掘发现大量数据中项集之间有趣的关联或相关联系。随着大量数据不停地收集和存储，许多业界人士对于从他们的数据库中挖掘关联规则越来越感兴趣。从大量商务事务记录中发现有趣的关联关系，可以帮助许多商务决策的制定，如分类设计、交叉购物和贱卖分析。

关联规则挖掘的一个典型例子是**购物篮分析**。该过程通过发现顾客放入其购物篮中不同商品（图6.1）之间联系，分析顾客的购买习惯。通过了解哪些商品频繁地被顾客同时购买，这种关联的发现可以帮助零售商制定营销策略。例如，在同一次去超级市场，如果顾客购买牛奶，他也购买面包（和什么类型的面包）的可能性有多大？通过帮助零售商有选择地经销和安排货架，这种信息可以引导销售。例如，将牛奶和面包尽可能放近一些，可以进一步刺激一次去商店同时购买这些商品。

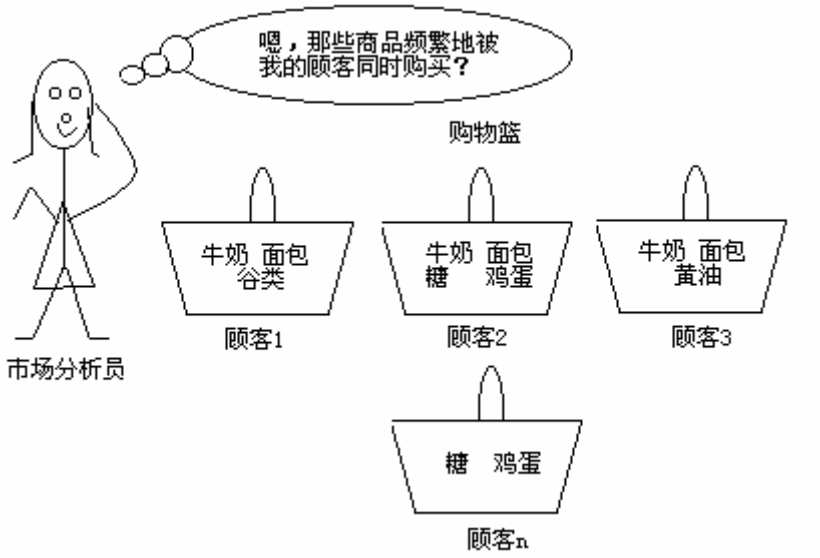


图 6.1　购物篮分析

数据是事务的或关系的，如何由大量的数据中发现关联规则？什么样的关联规则最有趣？我们如何帮助或指导挖掘过程发现有趣的关联规则？对于关联规则挖掘，什么样的语言结构对于定义关联挖掘查询是有用的？本章我们将深入研究这些问题。

6.1 关联规则挖掘

关联规则挖掘寻找给定数据集中项之间的有趣联系。本节简要介绍关联规则挖掘。6.1.1 小节给出一个购物篮分析的例子，这是关联规则挖掘的最初形式。挖掘关联规则的基本概念在 6.1.2 小节给出。6.1.3 小节给出一个路线图，指向可挖掘的各种不同类型关联规则。

6.1.1 购物篮分析：一个引发关联规则挖掘的例子

假定作为 AllElectronics 的分店经理，你想更加了解你的顾客的购物习惯。例如，你想知道“什么商品组或集合顾客多半会在一次购物时同时购买？”为回答你的问题，你可以在你的商店顾客事务零售数据上运行购物篮分析。分析结果可以用于市场规划、广告策划、分类设计。例如，购物篮分析可以帮助经理设计不同的商店布局。一种策略是：经常一块购买的商品可以放近一些，以便进一步刺激这些商品一起销售。例如，如果顾客购买计算机也倾向于同时购买财务软件，将硬件摆放离软件陈列近一点，可能有助于增加二者的销售。另一种策略是：将硬件和软件放在商店的两

端，可能诱发买这些商品的顾客一路挑选其它商品。例如，在决定购买一台很贵的计算机之后，去看软件陈列，购买财务软件，路上可能看到安全系统，可能会决定也买家庭安全系统。购物篮分析也可以帮助零售商规划什么商品降价出售。如果顾客趋向于同时购买计算机和打印机，打印机降价出售可能既促使购买打印机，又促使购买计算机。

如果我们想象全域是商店中可利用的商品的集合，则每种商品有一个布尔变量，表示该商品的有无。每个篮子则可用一个布尔向量表示。可以分析布尔向量，得到反映商品频繁关联或同时购买的购买模式。这些模式可以用**关联规则**的形式表示。例如，购买计算机也趋向于同时购买财务管理软件可以用以下关联规则表示：

$$computer \Rightarrow financial\_management\_software$$
$$[\text{support}=2\%,\text{confidence}=60\%] \qquad (6.1)$$

规则的**支持度**和**置信度**是两个规则兴趣度度量，已在前面 4.1.4 小节介绍。它们分别反映发现规则的有用性和确定性。关联规则(6.1)的支持度 2%意味分析事务的 2%同时购买计算机和财务管理软件。置信度 60%意味购买计算机的顾客 60%也购买财务管理软件。关联规则是有趣的，如果它满足**最小支持度阈值**和**最小置信度阈值**。这些阈值可以由用户或领域专家设定。

6.1.2 基本概念

设 $\boldsymbol{I} = \{ i_1, i_2, ..., i_m \}$是项的集合。设任务相关的数据 D 是数据库事务的集合，其中每个事务 T 是项的集合，使得 $T \subseteq \boldsymbol{I}$。每一个事务有一个标识符，称作 TID。设 A 是一个项集，事务 T 包含 A 当且仅当 $A \subseteq T$。**关联规则**是形如 $A \Rightarrow B$ 的蕴涵式，其中 $A \subset \boldsymbol{I}$，$B \subset \boldsymbol{I}$，并且 $A \cap B = \varnothing$。规则 $A \Rightarrow B$ 在事务集 D 中成立，具有支持度 s，其中 s 是 D 中事务包含 $A \cup B$（即，A 和 B 二者）的百分比。它是概率 $P(A \cup B)$。规则 $A \Rightarrow B$ 在事务集 D 中具有置信度 c，如果 D 中包含 A 的事务同时也包含 B 的百分比是 c。这是条件概率 $P(B|A)$。即

$$support\ (A \Rightarrow B) = P(A \cup B) \qquad (6.2)$$
$$confidence\ (A \Rightarrow B) = P(B|A) \qquad (6.3)$$

同时满足最小支持度阈值(*min_sup*)和最小置信度阈值(*min_conf*)的规则称作**强规则**。为方便计，我们用 0%和 100%之间的值，而不是用 0 到 1 之间的值表示支持度和置信度。

项的集合称为**项集**[15]。包含 k 个项的项集称为 ***k*-项集**。集合{*computer, financial_management_software*}是一个 2-项集。项集的**出现频率**是包含项集的事务数，简称为项集的**频率**、**支持计数**或**计数**。项集满足**最小支持度** *min_sup*，如果项集的出现频率大于或等于 *min_sup* 与 D 中事务总数的乘积。如果项集满足最小支持度，则称它为**频繁项集**[16]。频繁 k -项集的集合通常记作 L_k。[17]

“如何由大型数据库挖掘关联规则？”关联规则的挖掘是一个两步的过程：

1. **找出所有频繁项集**：根据定义，这些项集出现的频繁性至少和预定义的最小支持计数一样。
2. **由频繁项集产生强关联规则**：根据定义，这些规则必须满足最小支持度和最小置信度。

如果愿意，也可以使用附加的兴趣度度量。这两步中，第二步最容易。挖掘关联规则的总体性能由第一步决定。

6.1.3 关联规则挖掘：一个路线图

购物篮分析只是关联规则挖掘的一种形式。事实上，有许多种关联规则。根据下面的标准，关联规则有多种分类方法：

---

[15] 在数据挖掘研究界，“itemset”比“item set”更常用。

[16] 在早期的工作中，满足最小支持度的项集称为**大的**。然而，该术语有时易混淆，因为它具有项集中项的个数的内涵，而不是集合出现的频率。因此，我们使用当前术语**频繁**。

[17] 尽管**频繁**已取代**大的**，由于历史的原因，频繁 *k*-项集仍记作 L_k。

- **根据规则中所处理的值类型**：如果规则考虑的关联是项的在与不在，则它是**布尔关联规则**。例如，上面的规则(6.1)是由购物篮分析得到的布尔关联规则。

  如果规则描述的是量化的项或属性之间的关联，则它是**量化关联规则**。在这种规则中，项或属性的量化值划分为区间。下面的规则(6.4)是量化关联规则的一个例子，其中，X 是代表顾客的变量。

$$age(X,"30...39") \wedge income(X,"42K...48K") \Rightarrow buys(X,"high\_resolution\_TV") \quad (6.4)$$

  注意，量化属性 *age* 和 *income* 已离散化。

- **根据规则中涉及的数据维**：如果关联规则中的项或属性每个只涉及一个维，则它是**单维关联规则**。注意，(6.1)式可以写作

$$buys(X,"computer") \Rightarrow buys(X,"financial\_management\_software") \quad (6.5)$$

  规则(6.1)是单维关联规则，因为它只涉及一个维 *buys*[18]。如果规则涉及两个或多个维，如维 *buys, time_of_transaction* 和 *customer_category*，则它是**多维关联规则**。上面的规则(6.4)是一个多维关联规则，因为它涉及三个维 *age, income* 和 *buys*。

- **根据规则集所涉及的抽象层**：有些挖掘关联规则的方法可以在不同的抽象层发现规则。例如，假定挖掘的关联规则集包含下面规则：

$$age(X,"30...39") \Rightarrow buys(X,"laptop\ computer") \quad (6.6)$$

$$age(X,"30...39") \Rightarrow buys(X,"computer") \quad (6.7)$$

  在规则(6.6)和(6.7)中，购买的商品涉及不同的抽象层（即，"*computer*"在比"*laptop computer*"高的抽象层）。我们称所挖掘的规则集由**多层关联规则**组成。反之，如果在给定的规则集中，规则不涉及不同抽象层的项或属性，则该集合包含**单层关联规则**。

- **根据关联挖掘的各种扩充**：关联挖掘可以扩充到相关分析，那里，可以识别项是否相关。还可以扩充到挖掘最大模式（即，最大的频繁模式）和频繁闭项集。**最大模式**是频繁模式 p，使得 p 的任何真超模式[19]都不是频繁的。**频繁闭项集**是一个频繁的、闭的项集；其中，项集 c 是闭的，如果不存在 c 的真超集 c'，使得每个包含 c 的事务也包含 c'。使用最大模式和频繁闭项集可以显著地压缩挖掘所产生的频繁项集数。

  本章的其余部分，你将学习上述每种关联规则的挖掘方法。

6.2 由事务数据库挖掘单维布尔关联规则

本节，你将学习挖掘最简单形式的关联规则的方法。这种关联规则是单维、单层、布尔关联规则，如 6.1.1 小节所讨论的购物篮分析中的那些。我们以提供 **Apriori** 算法开始（6.2.1 小节）。Apriori 算法是一种找频繁项集的基本算法。由频繁项集产生强关联规则的过程在 6.2.2 小节给出。6.2.3 小节介绍 Apriori 算法的一些变形，用于提高效率和可规模性。6.2.4 小节提出一些挖掘关联规则方法，不象 Apriori，它们不涉及“候选”频繁项集的产生。6.2.5 小节介绍如何将 Apriori 的原则用于提高冰山查询的效率。冰山查询在购物篮分析中是常见的。

[18] 按照多维数据库使用的术语，我们把规则中的每个不同谓词称作维。

[19] q 是 p 的超模式，如果 p 是 q 的子模式；即，如果 q 包含 p。

6.2.1 Apriori 算法：使用候选项集找频繁项集

Apriori 算法是一种最有影响的挖掘布尔关联规则频繁项集的算法。算法的名字基于这样的事实：算法使用频繁项集性质的先验知识，正如我们将看到的。Apriori 使用一种称作逐层搜索的迭代方法，k-项集用于探索(k+1)-项集。首先，找出频繁 1-项集的集合。该集合记作 L_1。L_1 用于找频繁 2-项集的集合 L_2，而 L_2 用于找 L_3，如此下去，直到不能找到频繁 k-项集。找每个 L_k 需要一次数据库扫描。

为提高频繁项集逐层产生的效率，一种称作 Apriori 性质的重要性质用于压缩搜索空间。我们先介绍该性质，然后用一个例子解释它的使用。

Apriori 性质：频繁项集的所有非空子集都必须也是频繁的。Apriori 性质基于如下观察：根据定义，如果项集 I 不满足最小支持度阈值 s，则 I 不是频繁的，即 $P(I) < s$。如果项 A 添加到 I，则结果项集（即，$I \cup A$）不可能比 I 更频繁出现。因此，$I \cup A$ 也不是频繁的，即 $P(I \cup A) < s$。

该性质属于一种特殊的分类，称作**反单调**，意指如果一个集合不能通过测试，则它的所有超集也都不能通过相同的测试。称它为反单调的，因为在通不过测试的意义下，该性质是单调的。

“如何将 Apriori 性质用于算法？”为理解这一点，我们必须看看如何用 L_{k-1} 找 L_k。下面的两步过程由**连接**和**剪枝**组成。

1. **连接步**：为找 L_k，通过 L_{k-1} 与自己连接产生候选 k-项集的集合。该候选项集的集合记作 C_k。设 l_1 和 l_2 是 L_{k-1} 中的项集。记号 l_i[j]表示 l_i 的第 j 项（例如，l_1[k-2]表示 l_1 的倒数第 3 项）。为方便计，假定事务或项集中的项按字典次序排序。执行连接 L_{k-1} L_{k-1}；其中，L_{k-1} 的元素是可连接的，如果它们前(k-2)个项相同；即，L_{k-1} 的元素 l_1 和 l_2 是可连接的，如果(l_1 [1] = l_2 [1]) ∧ (l_1 [2] = l_2 [2]) ∧ ... ∧ (l_1 [k-2] = l_2 [k-2]) ∧ (l_1 [k-1] < l_2 [k-1])。条件(l_1 [k-1] < l_2 [k-1])是简单地保证不产生重复。连接 l_1 和 l_2 产生的结果项集是 l_1 [1] l_1 [2]... l_1 [k-1] l_2 [k-1]。
2. **剪枝步**：C_k 是 L_k 的超集；即，它的成员可以是，也可以不是频繁的，但所有的频繁 k-项集都包含在 C_k 中。扫描数据库，确定 C_k 中每个候选的计数，从而确定 L_k（即，根据定义，计数值不小于最小支持度计数的所有候选是频繁的，从而属于 L_k）。然而，C_k 可能很大，这样所涉及的计算量就很大。为压缩 C_k，可以用以下办法使用 Apriori 性质：任何非频繁的(k-1)-项集都不是可能是频繁 k-项集的子集。因此，如果一个候选 k-项集的(k-1)-子集不在 L_{k-1} 中，则该候选也不可能是频繁的，从而可以由 C_k 中删除。这种子集测试可以使用所有频繁项集的散列树快速完成。

例 6.1 让我们看一个 Apriori 的具体例子。该例基于图 6.2 的 AllElectronics 的事务数据库。数据库中有 9 个事务，即|D| = 9。Apriori 假定事务中的项按字典次序存放。我们使用图 6.3 解释 Apriori 算法发现 D 中的频繁项集。

AllElectronics 数据库

TID	List of item_ID’s
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

图 6.2 AllElectronics 某分店的事务数据

1. 在算法的第一次迭代，每个项都是候选 1-项集的集合 C_1 的成员。算法简单地扫描所有的事务，对每个项的出现次数计数。
2. 假定最小事务支持计数为 2（即，min_sup = 2/9 = 22%）。可以确定频繁 1-项集的集合 L_1。它由具有最小支持度的候选 1-项集组成。

3. 为发现频繁 2-项集的集合 L_2，算法使用 L_1 L_1 产生候选 2-项集的集合 C_2 [20]。C_2 由 $\binom{|L_1|}{2}$ 个 2-项集组成。
4. 下一步，扫描 D 中事务，计算 C_2 中每个候选项集的支持计数，如图 6.3 的第二行的中间表所示。
5. 确定频繁 2-项集的集合 L_2，它由具有最小支持度的 C_2 中的候选 2-项集组成。

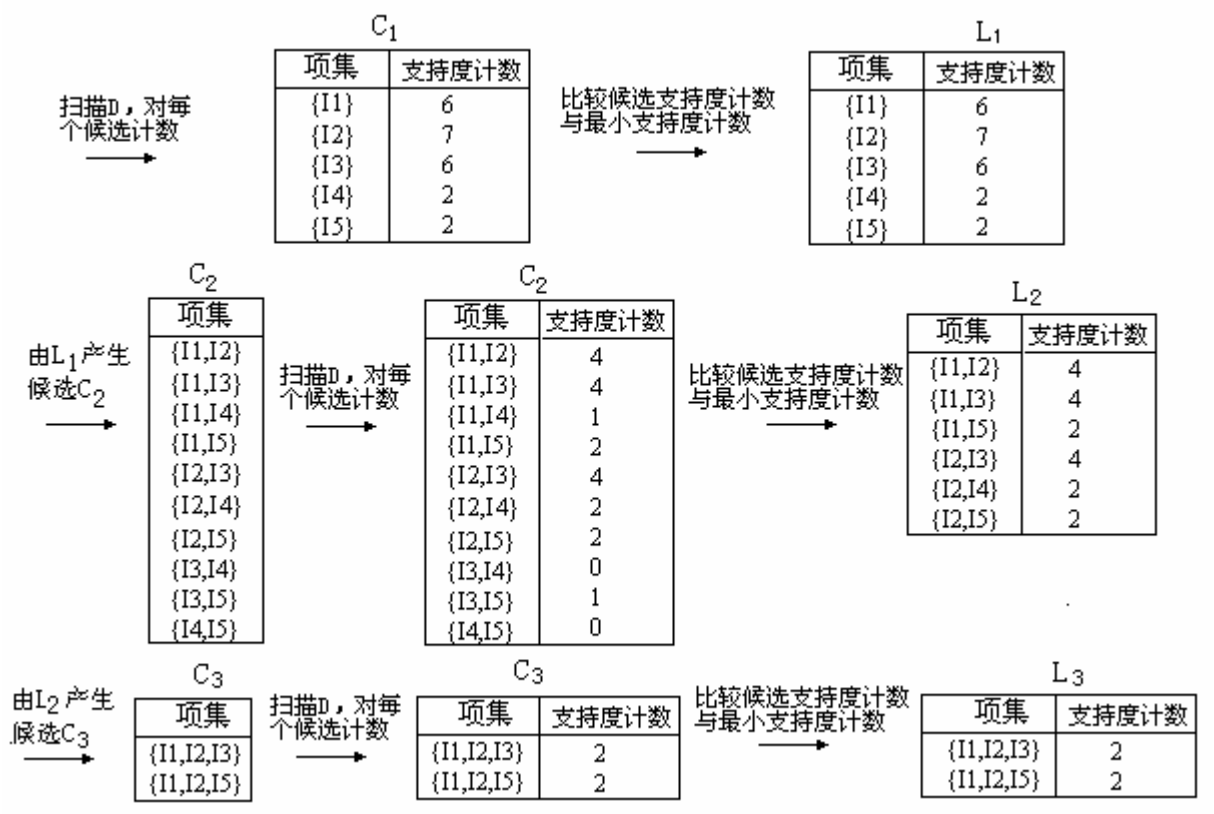


图 6.3　候选项集和频繁项集的产生，最小支持计数为 2

---

1. 连接：$C_3 = L_2$ L_2={{I1,I2},{I1,I3},{I1,I5},{I2,I3},{I2,I4},{I2,I5}} {{I1,I2},{I1,I3},{I1,I5},{I2,I3},{I2,I4},{I2,I5}} = {{I1,I2,I3},{I1,I2,I5},{I1,I3,I5},{I2,I3,I4},{I2,I3,I5},{I2,I4,I5}}
2. 使用 Apriori 性质剪枝：频繁项集的所有子集必须是频繁的。存在候选项集，其子集不是频繁的吗？
   - {I1,I2,I3}的 2-项子集是{I1,I2}，{I1,I3}和{I2,I3}。{I1,I2,I3}的所有 2-项子集都是 L_2 的元素。因此，保留{I1,I2,I3}在 C_3 中。
   - {I1,I2,I5}的 2-项子集是{I1,I2}，{I1,I5}和{I2,I5}。{I1,I2,I5}的所有 2-项子集都是 L_2 的元素。因此，保留{I1,I2,I5}在 C_3 中。
   - {I1,I3,I5}的 2-项子集是{I1,I3}，{I1,I5}和{I3,I5}。{I3,I5}不是 L_2 的元素，因而不是频繁的。这样，由 C_3 中删除{I1,I3,I5}。
   - {I2,I3,I4}的 2-项子集是{I2,I3}，{I2,I4}和{I3,I4}。{I3,I4}不是 L_2 的元素，因而不是频繁的。这样，由 C_3 中删除{I2,I3,I4}。
   - {I2,I3,I5}的 2-项子集是{I2,I3}，{I2,I5}和{I3,I5}。{I3,I5}不是 L_2 的元素，因而不是频繁的。这样，由 C_3 中删除{I2,I3,I5}。

---

[20] L_1 L_1 等价于 $L_1 \times L_1$，因为 L_k L_k 的定义要求两个连接的项集共享 k-1=0 个项。

- {I2,I4,I5}的 2-项子集是{I2,I4}，{I2,I5}和{I4,I5}。{I4,I5}不是 L_2 的元素，因而不是频繁的。这样，由 C_3 中删除{I2,I3,I5}。

3．这样，剪枝后 C_3 = {{I1,I2,I3},{I1,I2,I5}}。

图 6.4　使用 Apriori 性质，由 L_2 产生候选 3-项集 C_3

6. 候选 3-项集的集合 C_3 的产生详细地列在图 6.4 中。首先，令 $C_3 = L_2 \quad L_2$ = {{I1,I2,I3}, {I1,I2,I5}, {I1,I3,I5}, {I2,I3,I4}, {I2,I3,I5}, {I2,I4,I5}}。根据 Apriori 性质，频繁项集的所有子集必须是频繁的，我们可以确定后 4 个候选不可能是频繁的。因此，我们把它们由 C_3 删除，这样，在此后扫描 D 确定 L_3 时就不必再求它们的计数值。注意，Apriori 算法使用逐层搜索技术，给定 k-项集，我们只需要检查它们的(k-1)-子集是否频繁。

7. 扫描 D 中事务，以确定 L_3，它由具有最小支持度的 C_3 中的候选 3-项集组成（图 6.3）。

8. 算法使用 $L_3 \quad L_3$ 产生候选 4-项集的集合 C_4。尽管连接产生结果{{I1,I2,I3,I5}}，这个项集被剪去，因为它的子集{I1,I3,I5}不是频繁的。这样，$C_4 = \varnothing$，因此算法终止，找出了所有的频繁项集。□

图 6.5 给出 Apriori 算法和它的相关过程的伪代码。Apriori 的第 1 步找出频繁 1-项集的集合 L_1。在 2-10 步，L_{k-1} 用于产生候选 C_k，以找出 L_k。Apriori_gen 过程产生候选，然后使用 Apriori 性质删除那些具有非频繁子集的候选（步骤 3）。该过程在下面描述。一旦产生了所有的候选，就扫描数据库（步骤 4）。对于每个事务，使用 subset 函数找出事务中是候选的所有子集（步骤 5），并对每个这样的候选累加计数（步骤 6 和 7）。最后，所有满足最小支持度的候选形成频繁项集 L。然后，调用一个过程，由频繁项集产生关联规则。该过程在 6.2.2 小节介绍。

算法 6.2.1 (Apriori)　使用逐层迭代找出频繁项集

输入：事务数据库 D；最小支持度阈值。

输出：D 中的频繁项集 L。

方法：

```
1) L1 = find_frequent_1_itemsets(D);
2) for (k = 2; Lk-1 ≠ ∅; k++) {
3) Ck = aproiri_gen(Lk-1,min_sup);
4) for each transaction t∈D{                    //scan D for count
5) Ct = subset(Ck,t);                           //get subsets of t that are candidates
6) for each candidate c∈Ct
7) c.count++;
8) }
9) Lk={c∈Ck | c.count ≥ min_sup}
10)  }
11)  return L = ∪kLk;

procedure apriori_gen(Lk-1: frequent (k-1)-itemset; min_sup: support)
1) for each itemset l1∈Lk-1
2) for each itemset l2∈Lk-1
3) if (l1[1]=l2[1])∧...∧(l1[k-2]=l2[k-2])∧(l1[k-1]<l2[k-2]) then {
4) c = l1    l2;                                //join step: generate candidates
5) if has_infrequent_subset(c,Lk-1) then
6) delete c;                                    // prune step: remove unfrequent cadidate
7) else add c to Ck;
8) }
9) return Ck;
```

procedure has_infrequent_subset(c:candidate k-itemset; L_{k-1}:frequent (k-1)-itemset)
// use priori knowledge
1) **for each** (k-1)-subset s **of** c
2) **if** $c \notin L_{k-1}$ **then**
3) return TRUE;
4) **return** FALSE;

图 6.5　对于布尔关联规则发现频繁项集的 Apriori 算法

如上所述，Apriori_gen 做两个动作：**连接**和**剪枝**。在连接部分，L_{k-1} 与 L_{k-1} 连接产生可能的候选（步骤 1-4）。剪枝部分（步骤 5-7）使用 Apriori 性质删除具有非频繁子集的候选。非频繁子集的测试在过程 has_infrequent_subset 中。

6.2.2　由频繁项集产生关联规则

一旦由数据库 *D* 中的事务找出频繁项集，由它们产生强关联规则是直接了当的（强关联规则满足最小支持度和最小置信度）。对于置信度，可以用下式，其中条件概率用项集支持度计数表示。

$$confidence(A \Rightarrow B) = P(A \mid B) = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)} \tag{6.8}$$

其中，*support_count*($A \cup B$)是包含项集 $A \cup B$ 的事务数，*support_count*(*A*)是包含项集 *A* 的事务数。根据该式，关联规则可以产生如下：

- 对于每个频繁项集 *l*，产生 *l* 的所有非空子集。
- 对于 *l* 的每个非空子集 *s*，如果 $\frac{\text{support\_count}(l)}{\text{support\_count}(s)} \geq \text{min\_conf}$，则输出规则“$s \Rightarrow (l\text{-}s)$”。其中，*min_conf* 是最小置信度阈值。

由于规则由频繁项集产生，每个规则都自动满足最小支持度。频繁项集连同它们的支持度预先存放在 hash 表中，使得它们可以快速被访问。

例 6.2　让我们试一个例子，它基于图 6.2 中 AllElectronics 事务数据库。假定数据包含频繁项集 *l* = {I1, I2, I5}，可以由 *l* 产生哪些关联规则？*l* 的非空子集有{I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}和{I5}。结果关联规则如下，每个都列出置信度。

$I1 \wedge I2 \Rightarrow I5$,	*confidence* = 2/4 = 50%
$I1 \wedge I5 \Rightarrow I2$,	*confidence* = 2/2 = 100%
$I2 \wedge I5 \Rightarrow I1$,	*confidence* = 2/2 = 100%
$I1 \Rightarrow I2 \wedge I5$,	*confidence* = 2/6 = 33%
$I2 \Rightarrow I1 \wedge I5$,	*confidence* = 2/7 = 29%
$I5 \Rightarrow I1 \wedge I2$,	*confidence* = 2/2 = 100%

如果最小置信度阈值为 70%，则只有 2、3 和最后一个规则可以输出，因为只有这些是强的。□

6.2.3　提高 Apriori 的有效性

“怎样能够提高 Apriori 的有效性？”已经提出了许多 Apriori 算法的变形，旨在提高原算法的效率。其中一些变形列举如下。

使用散列函数
hash(x,y)=((order of x)*10+(order of y)) mod 7
创建散列表 H_2

H_2

桶地址	0	1	2	3	4	5	6
桶计数	2	2	4	2	2	4	4
桶内容	{I1,I4} {I3,I5}	{I1,I5} {I1,I5}	{I2,I3} {I2,I3} {I2,I3} {I2,I3}	{I2,I4} {I2,I4}	{I2,I5} {I2,I5}	{I1,I2} {I1,I2} {I1,I2} {I1,I2}	{I1,I3} {I1,I3} {I1,I3} {I1,I3}

图 6.6　候选 2-项集的散列表 H_2：该散列表在由 C_1 确定 L_1 时通过扫描图 5.2 的事务数据库产生。如果最小支持度为 3，在桶 0, 1，3 和 4 中的项集不可能是频繁的，因而它们不包含在 C_2 中

基于散列的技术（散列项集计数）：一种基于散列的技术可以用于压缩候选 *k*-项集 C_k (k >1)。例如，当扫描数据库中每个事务，由 C_1 中的候选 1-项集产生频繁 1-项集 L_1 时，我们可以对每个事务产生所有的 2-项集，将它们散列（即，映射）到散列表结构的不同桶中，并增加对应的桶计数（图 6.6）。在散列表中对应的桶计数低于支持度阈值的 2-项集不可能是频繁 2-项集，因而应当由候选项集中删除。这种基于散列的技术可以大大压缩要考察的 *k*-项集（特别是当 $k = 2$ 时）。

事务压缩（压缩进一步迭代扫描的事务数）：不包含任何 *k*-项集的事务不可能包含任何(k+1)-项集。这样，这种事务在其后的考虑时，可以加上标记或删除，因为为产生 *j*-项集($j > k$)，扫描数据库时不再需要它们。

划分（为找候选项集划分数据）：可以使用划分技术，它只需要两次数据库扫描，以挖掘频繁项集（图 6.7）。它包含两遍。在第 I 遍，算法将 *D* 中的事务划分成 *n* 个非重叠的部分。如果 *D* 中事务的最小支持度阈值为 *min_sup*，则每个部分的最小支持度计数为 *min_sup*×该部分中事务数。对每一部分，找出该部分内的频繁项集。这些称作**局部频繁项集**。该过程使用一种特殊的数据结构，对于每个项集，记录包含项集中项的事务的 TID。这使得对于 k = 1,2,..，找出所有的局部频繁 *k*-项集只需要扫描一次数据库。

局部频繁项集可能不是整个数据库 *D* 的频繁项集。*D* 的任何频繁项集必须作为局部频繁项集至少出现在一个部分中。这样，所有的局部频繁项集作为 *D* 的候选项集。所有部分的频繁项集的集合形成 *D* 的**全局候选项集**。在第 II 遍，第二次扫描 *D*，评估每个候选的实际支持度，以确定全局频繁项集。每一部分的大小和划分的数目这样确定，使得每一部分能够放入内存，这样每遍只需要读一次。

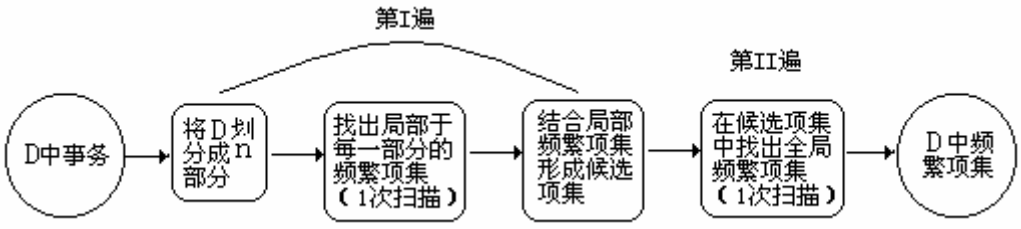


图 6.7　　通过划分挖掘

选样（在给定数据的一个子集挖掘）：选样方法的基本思想是：选取给定数据库 *D* 的随机样本 *S*，然后，在 *S* 而不是在 *D* 中搜索频繁项集。用这种方法，我们牺牲了一些精度换取了有效性。样本 *S* 的大小这样选取，使得可以在内存搜索 *S* 中频繁项集；这样，总共只需要扫描一次 *S* 中的事务。由于我们搜索 *S* 中而不是 *D* 中的频繁项集，我们可能丢失一些全局频繁项集。为减少这种可能性，我们使用比最小支持度低的支持度阈值来找出局部于 *S* 的频繁项集（记作 L^S）。然后，数据库的其余部分用于计算 L^S 中每个项集的实际频繁度。有一种机制可以用来确定是否所有的频繁项集都包含在 L^S 中。如果 L^S 实际包含了 *D* 中的所有频繁项集，只需要扫描一次 *D*。否则，可以做第二次扫描，以找出在第一次扫描时遗漏的频繁项集。当效率最为重要时，如计算密集的应用必须在不同的数据上运行时，选样方法特别合适。

动态项集计数（在扫描的不同点添加候选项集）：动态项集计数技术将数据库划分为标记开始点

的块。不象 Apriori 仅在每次完整的数据库扫描之前确定新的候选，在这种变形中，可以在任何开始点添加新的候选项集。该技术动态地评估已被计数的所有项集的支持度，如果一个项集的所有子集已被确定为频繁的，则添加它作为新的候选。结果算法需要的数据库扫描比 Apriori 少。

其它变形涉及多层和多维关联规则挖掘，在本章的其余部分讨论。涉及空间数据、时间序列数据和多媒体数据的关联挖掘在第 9 章讨论。

6.2.4 不产生候选挖掘频繁项集

正如我们已经看到的，在许多情况下，Apriori 的候选产生-检查方法大幅度压缩了候选项集的大小，并导致很好的性能。然而，它有两种开销可能并非微不足道的。

- 它可能需要产生大量候选项集。例如，如果有 10^4 个频繁 1-项集，则 Apriori 算法需要产生多达 10^7 个候选 2-项集，累计并检查它们的频繁性。此外，为发现长度为 100 的频繁模式，如 $\{a_1, \ldots, a_{100}\}$，它必须产生多达 $2^{100} \approx 10^{30}$ 个候选。
- 它可能需要重复地扫描数据库，通过模式匹配检查一个很大的候选集合。对于挖掘长模式尤其如此。

“可以设计一种方法，挖掘全部频繁项集，而不产生候选吗？”一种有趣的方法称作**频繁模式增长**，或简单地，**FP-增长**，它采取如下分治策略：将提供频繁项集的数据库压缩到一棵**频繁模式树**（或 **FP-树**），但仍保留项集关联信息；然后，将这种压缩后的数据库分成一组条件数据库（一种特殊类型的投影数据库），每个关联一个频繁项，并分别挖掘每个数据库。让我们看一个例子。

例 6.3 使用频繁模式增长方法，我们重新考察例 6.1 中图 6.2 事务数据库 D 的挖掘。

数据库的第一次扫描与 Apriori 相同，它导出频繁项（1-项集）的集合，并得到它们的支持度计数（频繁性）。设最小支持度计数为 2。频繁项的集合按支持度计数的递减序排序。结果集或表记作 L。这样，我们有 L = [I2:7, I1:6, I3:6, I4:2, I5:2]。

然后，FP-树构造如下：首先，创建树的根结点，用“null”标记。二次扫描数据库 D。每个事务中的项按 L 中的次序处理（即，根据递减支持度计数排序）并对每个事务创建一个分枝。例如，第一个事务“T100: I1, I2, I5”按 L 的次序包含三个项{ I2, I1, I5}，导致构造树的第一个分枝<(I2:1), (I1:1), (I5:1)>。该分枝具有三个结点，其中，I2 作为根的子女链接，I1 链接到 I2，I5 链接到 I1。第二个事务 T200 按 L 的次序包含项 I2 和 I4，它导致一个分枝，其中，I2 链接到根，I4 链接到 I2。然而，该分枝应当与 T100 已存在的路径共享**前缀**<I2>。这样，我们将结点 I2 的计数增加 1，并创建一个新结点(I4:1)，它作为(I2:2)的子女链接。一般地，当为一个事务考虑增加分枝时，沿共同前缀上的每个结点的计数增加 1，为随在前缀之后的项创建结点并链接。

为方便树遍历，创建一个项头表，使得每个项通过一个**结点链**指向它在树中的出现。扫描所有的事务之后得到的树展示在图 6.8 中，附上相关的结点链。这样，数据库频繁模式的挖掘问题就转换成挖掘 FP-树问题。

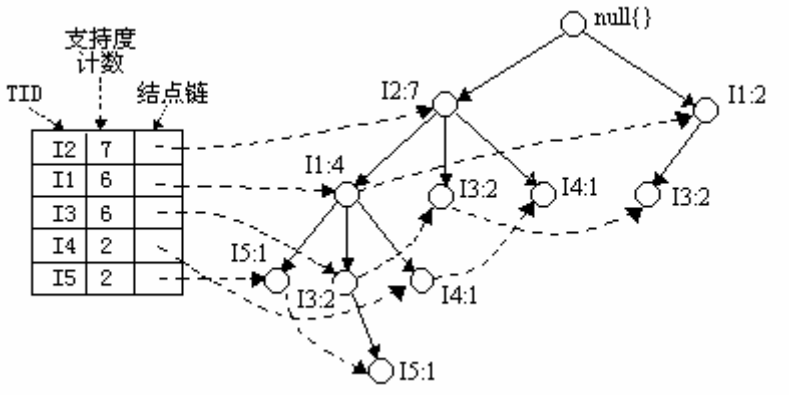


图 6.8　存放压缩的频繁模式信息的 FP-树

FP-树挖掘处理如下。由长度为 1 的频繁模式（初始**后缀**模式）开始，构造它的**条件模式基**（一个“子数据库”， 由 FP-树中与后缀模式一起出现的前缀路径集组成）。然后，构造它的（条件）FP-树，并递归地在该树上进行挖掘。模式增长通过后缀模式与由条件 FP-树产生的频繁模式连接实现。

FP-树的挖掘总结在表 6.1 中，细节如下。让我们首先考虑 I5，它是 *L* 中的最后一个项，而不是第一个。其原因随着我们解释 FP-树挖掘过程就会清楚。I5 出现在图 6.8　的 FP-树的两个分枝。（I5 的出现容易通过沿它的结点链找到。）这些路径由分枝<(I2 I1 I5:1)>和<(I2 I1 I3 I5:1)>形成。这样，考虑 I5 为后缀，它的两个对应前缀路径是<(I2 I1:1)>和<(I2 I1 I3:1)>，它们形成 I5 的条件模式基。它的条件 FP-树只包含单个路径<(I2:2 I1:2)>；不包含 I3，因为它的支持度计数为 1，小于最小支持度计数。该单个路径产生频繁模式的所有组合：I2 I5:2, I1 I5:2, I2 I1 I5:2。

表 6.1　通过创建条件（子）模式基挖掘 FP-树

item	条件模式基	条件 FP-树	产生的频繁模式
I5	{(I2 I1:1), (I2 I1 I3:1))}	<I2:2, I1:2>	I2 I5:2, I1 I5:2, I2 I1 I5:2
I4	{(I2 I1:1), (I2:1)}	<I2:2>	I2 I4:2
I3	{(I2 I1:2), (I2:2), (I1:2)}	<I2:4, I1:2>, <I1:2>	I2 I3:4, I1 I3:4, I2 I1 I3:2
I1	{(I2:4)}	<I2:4>	I2 I1:4

对于 I4，它的两个前缀形成条件模式基{(I2 I1:1), (I2:1)，产生一个单结点的条件 FP-树<I2:2>，并导出一个频繁模式 I2 I4:2。注意，尽管 I5 跟在第一个分枝中的 I4 之后，也没有必要在此分析中包含 I5，因为涉及 I5 的频繁模式在 I5 的考察时已经分析过。这就是我们为什么由后面，而不是由前面开始处理的原因。

与以上分析类似，I3 的条件模式基是{(I2 I1:2), (I2:2), (I1:2)}。它的条件 FP-树有两个分枝<I2:4, I1:2>和<I1:2>，如图 6.9 所示，它产生模式集：{I2 I3:4, I1 I3:2, I2 I1 I3:2}。最后，I1 的条件模式基是{(I2,4)}，它的 FP-树只包含一个结点<I2:4>，产生一个频繁模式 I2 I1:4。挖掘过程总结在图 6.10。□

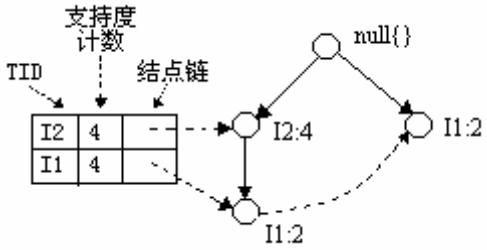


图 6.9　具有条件结点 I3 的条件 FP-树

算法：FP-增长。使用 FP-树，通过模式段增长，挖掘频繁模式。

输入：事务数据库 D；最小支持度阈值 *min_sup*。

输出：频繁模式的完全集。

方法：

1. 按以下步骤构造 FP-树：

(a) 扫描事务数据库 *D* 一次。收集频繁项的集合 *F* 和它们的支持度。对 *F* 按支持度降序排

序，结果为频繁项表 *L*。

(b) 创建 FP-树的根结点，以“null”标记它。对于 D 中每个事务 *Trans*，执行：

选择 *Trans* 中的频繁项，并按 *L* 中的次序排序。设排序后的频繁项表为[*p* | *P*]，其中，*p* 是第一个元素，而 *P* 是剩余元素的表。调用 **insert_tree**([*p* | *P*], *T*)。该过程执行情况如下。如果 *T* 有子女 *N* 使得 *N.item-name* = *p.item-name*，则 *N* 的计数增加 1；否则创建一个新结点 *N*，将其计数设置为 1，链接到它的父结点 *T*，并且通过结点链结构将其链接到具有相同 *item-name* 的结点。如果 *P* 非空，递归地调用 **insert_tree**(*P*, *N*)。

2．FP-树的挖掘通过调用 **FP_growth**(FP_tree, null)实现。该过程实现如下：

```
procedure  FP_growth(Tree, α)
(1) if  Tree 含单个路径 P  then
(2)     for  路径 P 中结点的每个组合（记作β）
(3) 产生模式β∪α，其支持度 support = β中结点的最小支持度；
(4) else for each  a_i 在 Tree 的头部  {
(5) 产生一个模式β = a_i ∪ α，其支持度 support = a_i.support；
(6) 构造β的条件模式基，然后构造β的条件 FP-树 Treeβ；
(7) if Treeβ ≠ ∅ then
(8) 调用 FP_growth (Treeβ, β)；}
```

图 6.10　不产生候选，发现频繁项集的 FP-增长算法

FP-增长方法将发现长频繁模式的问题转换成递归地发现一些短模式，然后与后缀连接。它使用最不频繁的项作后缀，提供了好的选择性。该方法大大降低了搜索开销。

当数据库很大时，构造基于内存的 FP-树是不现实的。一种有趣的替换是首先将数据库划分成投影数据库的集合，然后在每个投影数据库上构造 FP-树并挖掘它。该过程可以递归地用于投影数据库，如果它的 FP-树还不能放进内存。

对 FP-树方法的性能研究表明：对于挖掘长的和短的频繁模式，它都是有效的和可规模化的，并且大约比 Apriori 算法快一个数量级。它也比树-投影算法快。树-投影算法递归地将数据库投影为投影数据库树。

6.2.5　冰山查询

Apriori 算法可以用来提高回答冰山查询的效率。冰山查询在数据挖掘中经常用，特别是对购物篮分析。**冰山查询**在一个属性或属性集上计算一个聚集函数，以找出大于某个指定阈值的聚集值。给定关系 *R*，它具有属性 *a*_1, *a*_2, ... , *a_n* 和 *b*，一个聚集函数 *agg_f*，冰山查询形如

```
select    R.a_1, R.a_2, ... ,R.a_n, agg_f(R.b)
from      relation R
group by  R.a_1, R.a_2, ..., R.a_n
having    agg_f(R.b) >= threshold
```

给定大量输入数据元组，满足 **having** 子句中的阈值的输出元组数量相对很少。输出结果看作“冰山顶”，而“冰山”是输入数据集。

例 6.4　一个冰山查询：假定给定销售数据，你想产生这样的一个顾客-商品对的列表，这些顾客购买商品的数量达到 3 件或更多。这可以用下面的冰山查询表示

```
select    P.cust_ID, P.Item_ID,SUM(P.qty)
from      Purchases P
group by  P.cust_ID, Pitem_ID
having    SUM(P.qty)                    >=3
```

□

“如何回答例 6.4 的查询？”你可能会问。一个常用的策略是使用散列或排序，对所有顾客-商品分组，计算聚集函数 SUM 的值，然后删除被给定的顾客购买的商品数量少于 3 的那些。相对于处理的元组总数，满足该条件的元组多半很少，为改进性能留下了空间。我们可以使用 Apriori 性质的变形，裁减需要考虑的顾客-商品对。即，不是考查每个顾客购买的每种商品的数量，我们可以

- 产生 *cust_list*，一个总共购买 3 件或更多商品的顾客表。例如

```
select      P.cust_ID
from        Purchases P
group by    P.cust_ID
having      SUM(P.qty) >= 3
```

- 产生 *item_list*，被顾客购买的、数量为 3 或更多的商品表。例如

```
select      P. item_ID
from        Purchases P
group by    P.item_ID
having      SUM(P.qty) >= 3
```

由先验知识，我们可以删除许多被散列/排序方法产生的顾客-商品对：仅对 *cust_list* 中的顾客和在 *item_list* 中的商品产生候选顾客-商品对。对每个这样的对，维持一个计数。尽管该方法通过预先裁减许多对或分组提高了性能，所产生的顾客-商品对数量可能依然很大，不能放进内存。可以将散列和选样策略集成到该过程，帮助提高该查询回答技术的总体性能。

6.3 由事务数据库挖掘多层关联规则

本节，你将学习挖掘多层关联规则的方法。多层关联规则是这样一些规则，它们涉及多个抽象层中的项。本节还讨论检查冗余多层规则的方法。

6.3.1 多层关联规则

对于许多应用，由于多维数据空间数据的稀疏性，在低层或原始层的数据项之间很难找出强关联规则。在较高的概念层发现的强关联规则可能提供普遍意义的知识。然而，对一个用户代表普遍意义的知识，对另一个用户可能是新颖的。这样，数据挖掘系统应当提供一种能力，在多个抽象层挖掘关联规则，并容易在不同的抽象空间转换。

让我们考察下面的例子。

例 6.5 假定给定表 6.2 事务数据的任务相关数据集，它是 AllElectronics 分店的计算机部的销售数据，对每个事务 TID 给出了购买的商品。商品的概念分层在图 6.11 给出。概念分层定义了由低层概念到高层更一般的概念的映射序列。可以通过将数据内的低层概念用概念分层中的其高层概念（祖先）替换，对数据进行泛化[21]。图 6.11 的概念分层有 4 层，记作 0, 1, 2 和 3 层。为方便计，概念分层中的层自顶向下编号，根结点 **all**（最一般的抽象概念）为第 0 层。因此，第 1 层包括 *computer, software, printer* 和 *computer accessory*，第 2 层包括 *desktop computer, laptop computer, education software, financial management software, ...*，而第 3 层包括 *IBM desktop computer,..., Microsoft education software*，等等。第 3 层是该分层结构的最特定的抽象层。概念分层可以由熟悉数据的用户指定，也可以在数据中蕴涵存在。

表 6.2 任务相关数据 *D*

TID	购买的商品
T1	IBM desktop computer, Sony b/w printer
T2	Microsoft educationsoftware, Microsoft finacial management software
T3	Logitech mouse computer accessory, Ergo-way wrist pad computer accessory
T4	IBM desktop computer, Microsoft finacial management software

[21] 概念分层已在第 2、4 章详细介绍。为了使得本书的没章尽可能自包含，我们在此再次给出它的定义。泛化在第 5 章已介绍。

T5 ...	IBM desktop computer .....

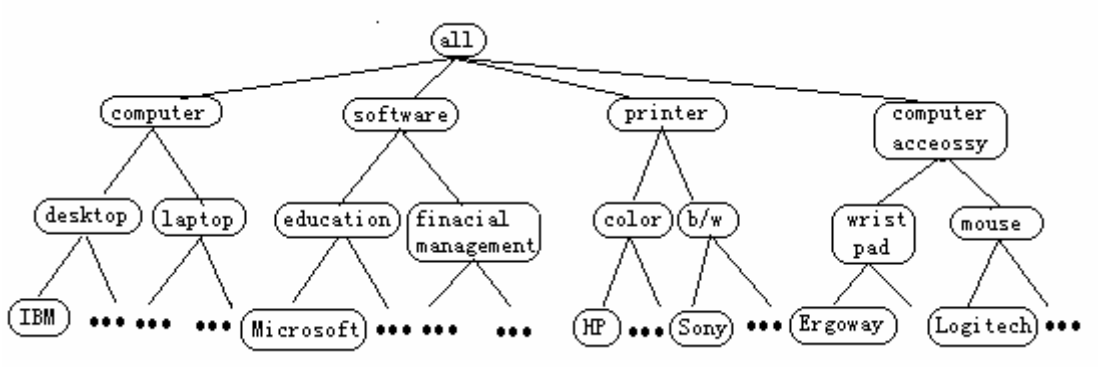


图 6.11　AllElectronics 计算机商品的概念分层

表 6.2 中的项在图 6.11 概念分层的最低层。在这种原始层很难找出有趣的购买模式。例如，如果“*IBM desktop computer*”和“*Sony b/w* (黑白) *printer*”每个都在很少一部分事务中出现，则可能很难找到涉及它们的强关联规则。很少人同时购买它们，使得“{ *IBM desktop computer, Sony b/w printer* }”不太可能满足最小支持度。然而，考虑将“*Sony b/w printer*”泛化到“*b/w printer*”。在“*IBM desktop computer*”和“*b/w printer*”之间比在“*IBM desktop computer*”和“*Sony b/w printer*”可望更容易发现强关联。类似地，许多人同时购买“*computer*”和“*printer*”，而不是同时购买特定的“*IBM desktop computer*”和“*Sony b/w printer*”。换句话说，包含更一般项的项集，如“{ *IBM desktop computer, b/w printer* }”和“{ *computer, printer* }”，比仅包含原始层数据的项集，如“{ *IBM desktop computer, Sony b/w printer* }”，更可能满足最小支持度。因此，在多个概念层的项之间找有趣的关联比仅在原始层数据之间更容易找。□

由具有概念分层的关联规则挖掘产生的规则称为**多层关联规则**，因为它们考虑多个概念层。

6.3.2　挖掘多层关联规则的方法

“我们如何使用概念分层有效地挖掘多层关联规则？”让我们看一些基于支持度-置信度框架的方法。一般地，可以采用自顶向下策略，由概念层 1 开始向下，到较低的更特定的概念层，在每个概念层累加计数计算频繁项集，直到不能再找到频繁项集。即，一旦找出概念层 1 的所有频繁项集，就开始在第 2 层找频繁项集，如此下去。对于每一层，可以使用发现频繁项集的任何算法，如 Apriori 或它的变形。这种方法有许多变形，介绍如下，并用图 6.12 到图 6.16 解释。图中矩形指出项或项集已被考察，而粗边框的矩形指出已考察的项或项集是频繁的。

- **对于所有层使用一致的支持度**（称作**一致支持度**）：在每一层挖掘时，使用相同的最小支持度阈值。例如，在图 6.12 中，整个使用最小支持度阈值 5%（例如，对于由“*computer*”到“*laptop computer*”）。“ *computer*”和“*laptop computer*”都是频繁的，但“*desktop computer*”不是。

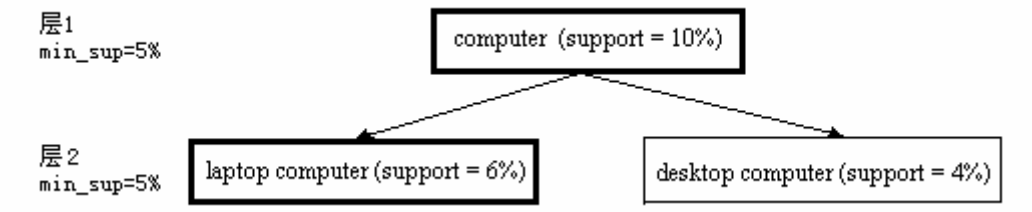


图 6.12　具有一致支持度的多层挖掘

使用一致的最小支持度阈值时，搜索过程是简单的，并且用户也只需要指定一个最小支持

度阈值。根据祖先是其后代的超集的知识，可以采用优化策略：搜索时避免考察这样的项集，它包含其祖先不具有最小支持度的项。

然而，一致支持度方法有一些困难。较低层次抽象的项不大可能象较高层次抽象的项出现得那么频繁。如果最小支持度阈值设置太高，可能丢掉出现在较低抽象层中有意义的关联规则。如果阈值设置太低，可能会产生出现在较高抽象层的无兴趣的关联规则。这导致了下面的方法。

- **在较低层使用递减的支持度**（称作**递减支持度**）：每个抽象层有它自己的最小支持度阈值。抽象层越低，对应的阈值越小。例如，在图 6.13，层 1 和层 2 的最小支持度阈值分别为 5%和 3%。用这种方法，“*computer*”、“*laptop computer*”和“*desktop computer*”都是频繁的。

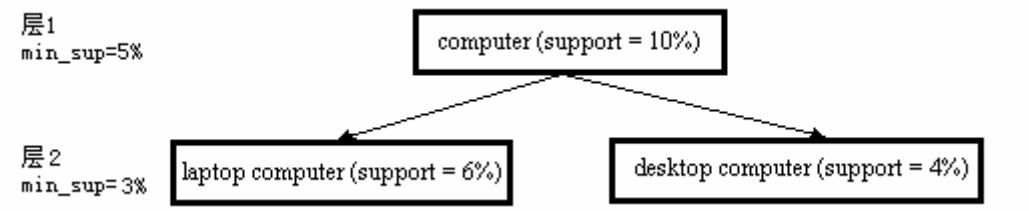


图 6.13　具有递减支持度的多层挖掘

对于具有递减支持度的多层关联规则挖掘，有许多可用的搜索策略，包括：

- **逐层独立**：这是完全的宽度搜索，没有频繁项集的背景知识用于剪枝。考察每一个结点，不管它的父结点是否是频繁的。
- **层交叉用单项过滤**：一个第 *i* 层的项被考察，当且仅当它在第(*i*-1)层的父结点是频繁的。换一句话说，我们由较一般的关联考察更特定的关联。如果一个结点是频繁的，它的子女将被考察；否则，它的子孙将由搜索中剪枝。例如，在图 6.14 中，“*computer*”的后代结点（即，“*laptop computer*”和“*desktop computer*”）将不被考察，因为“*computer*”不是频繁的。

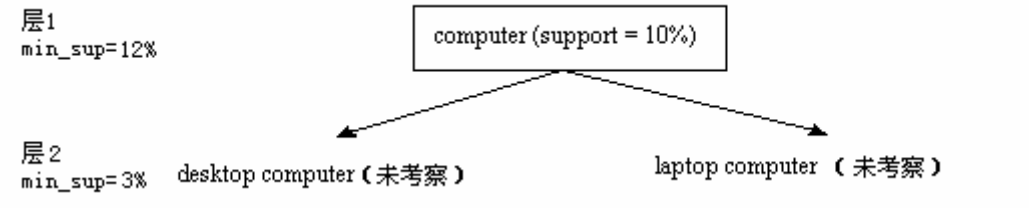


图 6.14　具有递减支持度的多层挖掘，使用层交叉用单项过滤

- **层交叉用 *k*-项集过滤**：一个第 *i* 层的 *k*-项集被考察，当且仅当它在第(*i*-1)层的对应父结点 *k*-项集是频繁的。例如，在图 6.15 中，2-项集“{*computer, printer*}”是频繁的，因而结点“{*laptop computer, b/w printer*}”、“ {*laptop computer, color printer*}”、“ {*desktop computer, b/w printer*}”和“{*desktop computer, color printer*}”被考察。

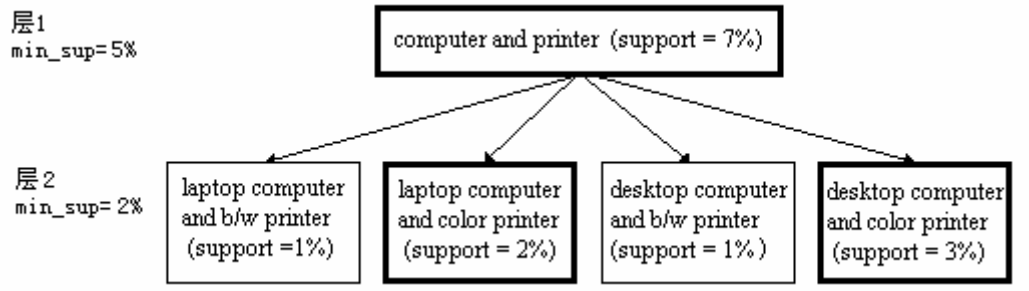


图 6.15　具有递减支持度的多层挖掘，使用层交叉用 *k*-项集过滤，其中 *k*=2

“如何比较这些方法？”逐层独立策略的条件很松，可能导致在低层考察大量非频繁的项，找出一些不太重要的关联。例如，如果“*computer furniture*”很少购买，考察特定的“*computer chair*”

是否与“*laptop computer*”关联没什么意思。然而，如果“*computer accessories*”经常出售，考察“*laptop*”与“*mouse*”之间是否存在关联购买模式可能是有意义的。

层交叉用 *k*-项集过滤策略允许系统仅考察频繁 *k*-项集的子女。这一限制可能太强，通常没有多少 *k*-项集组合后仍是频繁的（特别是当 $k > 2$ 时）。因此，有些有价值的模式可能被该方法过滤掉。

层交叉用单项过滤策略是上两个极端的折衷。然而，这种方法也可能丢失低层项之间的关联；根据递减的最小支持度，这些项是频繁的，但它们的祖先不满足最小支持度（由于每层的支持度阈值可能不同）。例如，如果根据第 *i* 层的最小支持度阈值，“*color monitor*”在第 *i* 层是频繁的，但是它在(*i*-1)层的父结点“*monitor*”，根据第(*i*-1)层的最小支持度阈值，不是频繁的，则频繁的关联“*desktop computer* ⇒ *color monitor*”将丢失。

层交叉单项过滤策略有一个修改版本，称作**受控的层交叉单项过滤策略**。可以设置一个称作**层传递阈值**的阈值，用于向较低层“传递”相对频繁的项（称作**子频繁项**）。换句话说，如果满足层传递阈值，则该方法允许考查不满足最小支持度阈值项的子女。每个概念层可以有它自己的层传递阈值。通常，对于一个给定层，层传递阈值设置为下一层的最小支持度阈值和给定层的最小支持度阈值之间的一个值。用户可以在较高层选择“下滑”或降低层传递阈值，使得子频繁项的后代在较低层被考察。降低层传递阈值到最低层的最小支持度阈值将使得项的所有后代被考察。例如，在图 6.16 中，设置层 1 的层传递阈值(*level_passage_sup*)为 8%，使得第 2 层的“*laptop computer*”和“*desktop computer*”被考察，并发现是频繁的，尽管它们的父结点“*computer*”不是频繁的。通过增加这种机制，用户对进一步控制多概念层上的挖掘过程有了更多的灵活性，同时减少无意义关联的考察和产生。

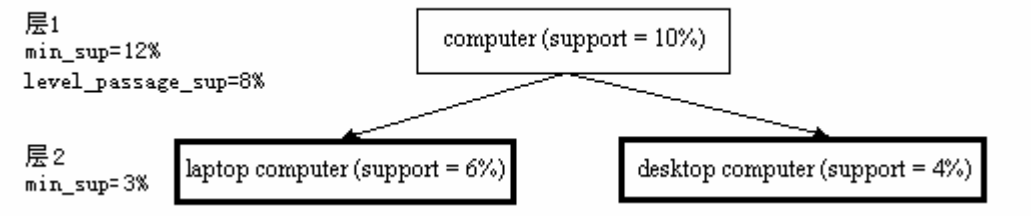


图 6.16　多层挖掘，受控的层交叉单项过滤

迄今为止，我们的讨论集中在发现这样的频繁项集，它的所有项都属于同一概念层。这可能导致形如“*computer* ⇒ *printer*”（其中，”*computer*”和”*printer*”都在概念层 1）和“*desktop computer* ⇒ *b/w printer*”（其中，”*desktop computer*”和”*b/w printer*”都在给定概念层的第 2 层）的规则。假定我们想要发现跨越概念层边界的规则，如“*computer* ⇒ *b/w printer*”，规则中的项不要求属于同一概念层。这些规则称作**交叉层关联规则**。

“如何挖掘交叉层关联规则？”如果挖掘由层 *i* 到层 *j* 的关联，其中层 *j* 比层 i 更特定（即，在较低的抽象层），则应当使用层 *j* 的最小支持度阈值，使得层 *j* 的项可以包含在分析中。

6.3.3　检查冗余的多层关联规则

概念分层在数据挖掘中是有用的，因为它们允许不同的抽象层的知识发现，如多层关联规则。然而，当挖掘多层关联规则时，由于项之间的“祖先”关系，有些发现的规则将是冗余的。例如，考虑下面的规则，其中，根据图 6.11 的概念分层，”*desktop computer*”是”*IBM desktop computer*”的祖先。

$$desktop\ computer \Rightarrow b/w\ printer, \quad [support=8\%,\ confidence=70\%] \tag{6.9}$$

$$IBM\ desktop\ computer \Rightarrow b/w\ printer, \quad [support=2\%,\ confidence=72\%] \tag{6.10}$$

“如果挖掘出规则 6.9 和 6.10，那么后一个规则是有用的吗？”你可能怀疑“它真的提供新颖的信息吗？”如果后一个具有较小一般性的规则不提供新的信息，应当删除它。让我们看看如何来确定。规则 R1 是规则 R2 的**祖先**，如果将 R2 中的项用它在概念分层中的祖先替换，能够得到 R1。

例如，规则(6.9)是规则(6.10)的祖先，因为”*desktop computer*”是”*IBM desktop computer*”的祖先。根据这个定义，一个规则被认为是冗余的，如果根据规则的祖先，它的支持度和置信度都接近于“期望”值。作为解释，假定规则(6.9)具有 70%置信度，8%支持度，并且大约四分之一的”*desktop computer*”销售是”*IBM desktop computer*”。可以期望规则(6.10)具有大约 70%的置信度（由于所有的”*IBM desktop computer*”样本也是” *desktop computer*”样本）和 2%（即，8%×1/4）的支持度。如果确实是这种情况，规则(6.10)不是有趣的，因为它不提供附加的信息，并且它的一般性不如规则(6.9)。

6.4 由数据库和数据仓库挖掘多维关联规则

本节，你将学习挖掘多维关联规则的方法。多维关联规则是涉及多个属性或谓词的规则（例如，关于顾客的 *buys* 和顾客的 *age* 的规则）。这些方法可以根据他们对量化属性的处理组织。

6.4.1 多维关联规则

到本章这里，我们研究了蕴涵单个谓词，即谓词 *buys* 的关联规则。例如，在挖掘 AllElectronics 数据库时，我们可能发现布尔关联规则“*IBM desktop computer* $\Rightarrow$ *Sony b/w printer*”，它也可以写成

$$buys(X,\text{“}IBM\ desktop\ computer\text{”}) \Rightarrow buys(X,\text{“}Sony\ \ b/w\ printer\text{”}) \tag{6.11}$$

其中，*X* 是变量，代表在 AllElectronics 购物的顾客。沿用多维数据库使用的术语，我们把每个不同的谓词称作维。这样，我们称规则(6.11)为**单维**或**维内关联规则**，因为它们包含单个不同谓词（即，*buys*）的多次出现（即，谓词在规则中出现多次）。正如我们在本章的前几节看到的，这种规则通常由事务数据挖掘。

然而，假定不是使用事务数据库，销售和相关数据存放在关系数据库或数据仓库中。根据定义，这种存储是多维的。例如，除记录购买的商品之外，关系数据库可能记录与商品有关的其它属性，如购买数量，或价格，或销售分店的地址。另外，关于购物顾客的信息，如顾客的年龄、职业、信誉度、收入和地址等也可能存储。将数据库的每个属性或数据仓库的每个维看作一个谓词，这样就能挖掘多维关联规则，如

$$age(X,\text{“}20...29\text{”}) \wedge occupation(X,\text{“}student\text{”}) \Rightarrow buys(X,\text{“}laptop\text{”}) \tag{6.12}$$

涉及两个或多个维或谓词的关联规则称为**多维关联规则**。规则(6.12)包含三个谓词（*age*, *occupation* 和 *buys*），每个在规则中仅出现一次。因此，我们称它具有**不重复谓词**。具有不重复谓词的关联规则称作**维间关联规则**。我们也对挖掘具有重复谓词的关联规则感兴趣。这种规则包含某些谓词的多次出现，称作**混合维关联规则**。这种规则的一个例子是规则(6.13)，其中谓词 *buys* 是重复的。

$$age(X,\text{“}20...29\text{”}) \wedge buys(X,\text{“}laptop\text{”}) \Rightarrow buys(X,\text{“}b/w\ printer\text{”}) \tag{6.13}$$

注意，数据库属性可能是分类的或量化的。**分类属性**具有有限个不同值，值之间无序（例如，*occupation, brand, color*）。分类属性也称**标称属性**，因为它们的值是“事物的名字”。**量化属性**是数值的，并在值之间具有一个蕴涵的序（例如，*age, income, price*）。挖掘多维关联规则的技术可以根据量化属性的处理分为三类。

第一种方法，使用预定义的概念分层对量化属性离散化。这种离散化在挖掘之前进行。例如，*income* 的概念分层可以用于以区间值，如“0...20K”、“21...30K”、“31...40K”等，替换属性的原来的数值值。这里，离散化是静态的、预确定的。离散化的数值属性具有区间值，可以象分类属性一样处理（每个区间看作一类）。我们称这种方法为**使用量化属性的静态离散化挖掘多维关联规则**。

第二种方法，根据数据的分布，将量化属性离散化到“箱”。这些箱可能在挖掘过程中进一步组

合。离散化的过程是动态的，以满足某种挖掘标准，如最大化所挖掘的规则的置信度。由于该策略将数值属性的值处理成量，而不是预定义的区间或分类，由这种方法挖掘的关联规则称为**量化关联规则**。

第三种方法，量化属性离散化，以紧扣区间数据的语义。这种动态离散化过程考虑数据点之间的距离。因此，这种量化关联规则称作**基于距离的关联规则**。

让我们逐个研究这些挖掘多维关联规则方法。为简明起见，我们将讨论限于维间关联规则。注意，不是搜索频繁项集（象单维关联规则挖掘那样），在多维关联规则挖掘中，我们搜索频繁谓词集。***k*-谓词集**是包含 *k* 个合取谓词的集合。例如，规则(6.12)中的谓词集{*age, occupation, buys*}是 3-谓词集。类似于项集使用的记号，我们用 L_k 表示频繁 *k*-谓词集的集合。

6.4.2 使用量化属性的静态离散化挖掘多维关联规则

在这种情况下，量化属性使用预定义的概念分层，在挖掘之前离散化，数值属性的值用区间替代，如果期望的话，分类属性可以泛化到较高的概念层。如果任务相关的结果数据存放在关系表中，则 Apriori 算法只需要稍加修改就可以找出所有的频繁谓词，而不是频繁项集（即，通过搜索所有的相关属性，而不是仅搜索一个属性，如 *buys*）。找出所有的频繁 *k*–谓词将需要 *k* 或 *k*+1 次表扫描。其它策略，如散列、划分和选样可以用来改进性能。

替换地，变换后的任务相关的数据可以存放在数据方中。数据方很适合挖掘多维关联规则，由于根据定义，它们是多维的。数据方和它们的计算已在第 2 章详细讨论。回顾一下，数据方由方体的格组成，方体是多维数据结构。这种结构可以存放任务相关的数据，以及聚集、分组信息。图 6.17 给出了一个方体的格，定义维 *age, income* 和 *buys* 的数据方。*n*-维方体的单元用于存放对应 *n*-谓词集的计数或支持度。基本方体按 *age, income* 和 *buys* 聚集了任务相关的数据；2-*D* 方体(*age, income*)按 *age* 和 *income* 聚集；0-*D*（顶点）方体包含任务相关数据中事务的总数，等等。

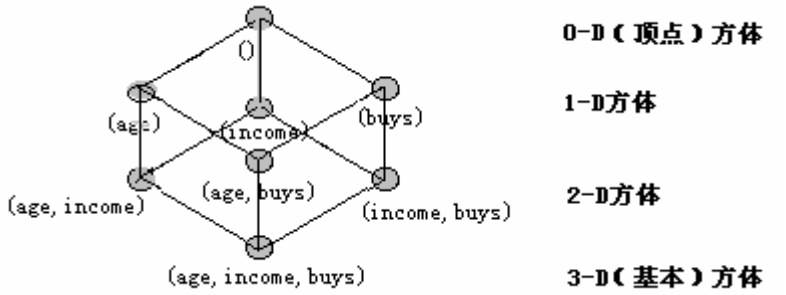


图 6.17　方体的格，形成 3-*D* 数据方。每个方体代表一个不同分组。基本方体包含三个谓词 *age, income* 和 *buys*

由于数据仓库和 OLAP 技术的日益增长的使用，包含用户感兴趣的维的数据方可能已经存在，并完全物化。“如果是这种情况，我们如何去找频繁谓词集？”可以使用一种类似于 Apriori 所用的策略，基于先验知识：频繁谓词集的每个子集也必须是频繁的。这个性质可以用于减少产生的谓词集候选数量。

在没有挖掘任务相关数据方存在时，必须创建。第 2 章介绍了数据方快速、有效计算的算法。这些算法可以修改，在数据方构造时搜索频繁项集。

6.4.3 挖掘量化关联规则

量化关联规则是多维关联规则，其中数值属性动态离散化，以满足某种挖掘标准，如最大化挖掘规则的置信度或紧凑性。在本小节，我们将特别关注如何挖掘左部有两个量化属性，右部有一个分类属性的量化关联规则，例如

$$A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$$

其中，A_{quan1} 和 A_{quan2} 是在量化属性的区间（其中，区间动态地确定）上测试，A_{cat} 测试任务相关数据的分类属性。这种规则称作 **2-维量化关联规则**，因为它们包含两个量化维。例如，假定我们关心

象 *age* 和 *income* 这样的量化属性对和这样的顾客喜欢什么类型的电视机之间的关联关系。这种 2-*D* 量化关联规则的一个例子是

$$age(X,\text{"}30...39\text{"}) \wedge income(X,\text{"}42K...48K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.15)$$

“如何找出这种规则？”让我们看看系统 **ARCS**（Association Rule Clustering System，关联规则聚类系统）使用的方法，其思想源于图形处理。本质上，该方法将量化属性对映射到满足给定分类属性条件的 2-*D* 栅格上。然后，搜索栅格点的聚类，由此产生关联规则。下面是 ARCS 涉及的步骤：

分箱。量化属性可能具有很宽的取值范围，定义它们的域。如果我们以 *age* 和 *income* 为轴，每个 *age* 的可能值在一个轴上赋予一个唯一的位置；类似地，每个 *income* 的可能值在另一个轴上赋予一个唯一的位置；想想 2-*D* 栅格会有多么大！为了使得栅格压缩到可管理的尺寸，我们将量化属性的范围划分为区间。这些区间是动态的，在挖掘期间它们可能进一步合并。这种划分过程称作**分箱**；即，区间被看作“箱”。三种常用的分箱策略是：

- **等宽分箱：**每个箱的区间长度相同，
- **等深分箱：**每个箱赋予大致相同个数的元组，和
- **基于同质的分箱：**箱的大小这样确定，使得每个箱中的元组一致分布。

在 ARCS 中，使用等宽分箱，每个量化属性的箱尺寸由用户输入。对于涉及两个量化属性的每种可能的箱组合，创建一个 2-*D* 数组。每个数组单元存放规则右部分类属性每个可能类的对应的计数分布。通过创建这种数据结构，任务相关的数据只需要扫描一次。基于相同的两个量化属性，同样的 2-*D* 数组可以用于产生分类属性的任何值的规则。分箱在第 3 章也进行了讨论。

找频繁谓词集。一旦包含每个分类计数分布的 2-*D* 数组设置好，就可以扫描它，以找出也满足最小置信度的频繁谓词集（满足最小支持度）。然后，使用类似于 6.2.2 小节介绍的规则产生算法，由这些谓词集产生关联规则。

关联规则聚类。将上一步得到的强关联规则映射到 2-D 栅格上。图 6.18 显示给定量化属性 *age* 和 *income*，预测规则右端条件 *buys(X,"high resolution TV")*的 2-D 量化关联规则。四个“×”对应于规则

$$age(X,34) \wedge income(X,\text{"}31K...40K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.15)$$
$$age(X,35) \wedge income(X,\text{"}31K...40K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.16)$$
$$age(X,34) \wedge income(X,\text{"}41K...50K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.17)$$
$$age(X,35) \wedge income(X,\text{"}41K...50K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.18)$$

“我们能找到一个更简单的规则替换上面四个规则吗？”注意，这些规则都相当“接近”，在栅格中形成聚类。的确，这些规则可以组合或“聚”在一起，形成下面的规则(6.19)，它更简单，将上面四个规则汇总在一起，并取代它们。

$$age(X,\text{"}34...35\text{"}) \wedge income(X,\text{"}31K...50K\text{"}) \Rightarrow buys(X,\text{"high resolution TV"}) \quad (6.20)$$

ARCS 使用聚类算法做这件事。该算法扫描栅格，搜索规则的矩形聚类。用这种方法，出现在规则聚类中的量化属性的箱可能进一步合并，从而对量化属性动态地离散化。

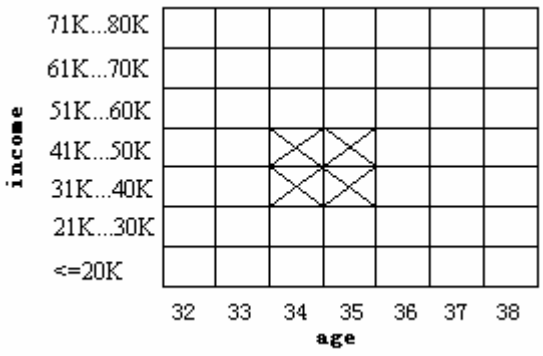


图 6.18　表示购买高分辨 TV 的顾客元组的 2-D 栅格

这里介绍的基于栅格的技术假定初始关联规则可以聚集到矩形区域。在进行聚集前，可以使用平滑技术，帮助消除数据中噪音和局外者。矩形聚类可能过分简化数据。已经提出了一些替换技术，基于其它形状的区域，看来能够更适合数据，但需要更大的计算量。

已经提出了一种非基于栅格的技术，发现更一般的关联规则，其中任意个数的量化属性和分类属性可以出现在规则的两端。在这种技术下，量化属性使用等深分箱动态划分，划分根据部分完全性度量组合，该度量量化由于划分而导致的信息丢失。关于这些 ARCS 替代方法的引文，参见文献注释。

6.4.4　挖掘基于距离的关联规则

前一小节我们介绍了量化关联规则，其量化属性开始用分箱的方法离散化，然后将结果区间组合。然而，这种方法可能不能紧扣区间数据的语义，因为它未考虑数据点之间或区间之间的相对距离。

例如，考虑表 6.3 中属性 *price* 的数据，根据等宽和等深分箱与基于距离的划分对比。基于距离的划分看来最直观，因为它将接近的值分在同一区间内（例如，[20, 22]）。相比之下，等深划分将很远的值放在一组（例如，[22, 50]）。等宽划分可能将很近的值分开，并创建没有数据的区间。显然，基于距离的划分既考虑稠密性或区间内的点数，又考虑一个区间内点的“接近性”，这帮助产生更有意义的离散化。每个量化属性的区间可以通过聚集该属性的值建立。

关联规则的一个缺点是它们不允许近似的属性值。考虑关联规则

$$item\_type(X,\text{“}electronic\text{”}) \wedge manufacture(X,\text{“}foreign\text{”}) \Rightarrow price(X,\$200) \quad (6.20)$$

在现实中，更可能的是国外的电子产品的价格大约\$200，而不是恰好\$200。使得关联规则可以表达这种接近性是有用的。注意，支持度和置信度度量不考虑属性值的接近性。这导致**基于距离的关联规则**挖掘。这种规则紧扣区间数据的语义，并允许数据值的近似。一个两遍算法可以用于挖掘基于距离的关联规则。第一遍使用聚类找出区间或聚类。第二遍搜索频繁地一起出现的聚类组得到基于距离的关联规则。

表 6.3　分箱方法，如等宽和等深，不能总是紧扣区间数据的语义

price($)	等宽 （宽度\$10）	等深 （深度 2）	基于距离
7	**[0,10]**	**[7,20]**	**[7,7]**
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

“如何在第一遍形成聚类？”这里，我们给出如何形成聚类的直观介绍。感兴趣的读者可以阅读第 8 章，以及本章文献注释中给出的关于基于距离的关联规则的引文。设 $S[X]$是 N 个元组 $t_1, t_2,..., t_N$ 投影到属性集 X 的集合。定义一个直径度量，评估元组的接近性。$S[X]$的直径是投影到 X 的元组两两距离的平均值。距离度量可以使用诸如欧几里德距离或曼哈坦距离[22]。$S[X]$的直径越小，其元组投影到 X 上时越接近。因此，直径度量评估聚类的稠密性。聚类 C_X 是定义在属性集 X 上的元组的集合；其中，元组满足**稠密度阈值**和**频繁度阈值**。频繁度阈值限定聚类中元组的最少个数。聚类方法，如第 8 章介绍的那些，可以修改，用于该挖掘过程的第一遍。

第二遍，将聚类组合，形成基于距离的关联规则。考虑一个简单的形如 $C_X \Rightarrow C_Y$ 的基于距离的关联规则。假定 X 是属性集{*age*}，而 Y 是属性集{*income*}。我们想确保 *age* 的聚类 C_X 和 *income* 的

[22] 两个元组 $t_1=(x_{11},x_{12},...,x_{1m})$和 $t_2=(x_{21},x_{22},...,x_{2m})$之间的欧几里德距离和曼哈坦距离分别是 $Euclidean\_d(t_1,t_2)=\sqrt{\sum_{i=1}^{m}(x_{1i}-x_{2i})^2}$ 和 $Manhattan\_d\ (t_1,t_2)=\sum_{i=1}^{m}|x_{1i}-x_{2i}|$。

聚类 C_Y 之间的蕴涵是强的。这意味当 *age* 聚类的元组投影到属性 *income* 上时，它们对应的值落在 *income* 聚类 C_Y 之内，或接近它。聚类 C_X 投影到属性集 Y 上记作 $C_X[Y]$。这样，$C_X[Y]$和 $C_Y[Y]$之间的距离必须很小。该距离度量 C_X 和 C_Y 之间的关联程度。$C_X[Y]$和 $C_Y[Y]$之间的距离越小，C_X 和 C_Y 之间的关联程度越强。关联程度度量可以使用标准统计度量定义，如平均内聚类距离，质心曼哈坦距离。其中，聚类的质心代表聚类的“平均”元组。

一般地，可以组合聚类，找出如下形式的基于距离的关联规则

$$C_{X_1} C_{X_2} ... C_{X_x} \Rightarrow C_{Y_1} C_{Y_2} ... C_{Y_y}$$

其中，X_i 和 Y_j 是两两不相交的属性集，并满足以下三个条件：（1）规则前件的每个聚类与后件的每个聚类是强关联的；（2）前件中的聚类一起出现；（3）后件中的聚类一起出现。关联程度取代了非基于距离的关联规则框架下的置信度，而稠密度阈值取代了支持度。

6.5 由关联挖掘到相关分析

“挖掘了关联规则之后，数据挖掘系统如何指出哪些规则是用户感兴趣的？”大部分关联规则挖掘算法使用支持度-置信度框架。尽管使用最小支持度和置信度阈值排除了一些无兴趣的规则的探查，仍然会产生一些对用户来说不感兴趣的规则。本节，我们首先看看即便是强关联规则为何也可能是无兴趣的并可能误导；然后，讨论基于统计独立性和相关分析的其它度量。

6.5.1 强关联规则不一定是有趣的：一个例子

“在数据挖掘中，所有的强关联规则（即，满足最小支持度和最小置信度阈值）都有兴趣，值得向用户提供吗？”并不一定。规则是否有兴趣可能用主观或客观的标准来衡量。最终，只有用户能够确定规则是否是有趣的，并且这种判断是主观的，因不同用户而异。然而，根据数据“支持”的统计，客观兴趣度度量可以用于清除无兴趣的规则，而不向用户提供。

“我们如何识别哪些强关联规则是真正有兴趣的？”让我们考查下面的例子。

例 6.6 假定我们对分析 AllElectronics 的事务感兴趣，涉及计算机游戏和录像。设事件 *game* 表示包含计算机游戏的事务，而 *video* 表示包含录像的事务。在所分析的 10,000 个事务中，数据显示 6000 个顾客事务包含计算机游戏，7500 个事务包含录像，而 4000 个事务包含计算机游戏和录像。假定发现关联规则的数据挖掘程序在该数据上运行，使用最小支持度 30%，最小置信度 60%。将发现下面的关联规则

$$buys(X, "computer\ games") \Rightarrow buys(X, "videos")$$

$$[\text{support} = 40\%, \text{confidence} = 66\%] \qquad (6.21)$$

规则(6.21)是强关联规则，因而向用户报告，因为其支持度为 $\frac{4,000}{10,000} = 40\%$，置信度为 $\frac{4,000}{6,000} = 66\%$，分别满足最小支持度和最小置信度阈值。然而，规则(6.21)是误导，因为购买录像的可能性是 75%，比 66%还大。事实上，计算机游戏和录像是负相关的，买一种实际上减少了买另一种的可能性。不完全理解这种现象，可能根据导出的规则作出不明智的决定。□

上面的例子也表明规则 $A \Rightarrow B$ 的置信度有一定的欺骗性，它只是给定 A，B 的条件概率的估计。它并不度量 A 和 B 之间蕴涵的实际强度。因此，寻求支持度-置信度框架的替代，对挖掘有趣的数据联系可能是有用的。

6.5.2 由关联分析到相关分析

使用支持度-置信度框架的关联规则挖掘对于许多应用是有用的。然而，支持度-置信度框架可

能误导，当 A 的出现事实上并不蕴涵 B 的出现时，识别出 $A \Rightarrow B$ 是有趣的。本小节，我们考虑一种替代框架，根据相关性挖掘数据项之间有趣的联系。

项集 A 的出现**独立**于项集 B 的出现，如果 $P(A \cup B) = P(A)P(B)$；否则，项集 A 和 B 是**依赖的**和**相关的**。这个定义容易推广到多于两个项集。A 和 B 的出现之间的相关性通过计算下式度量

$$\frac{P(A \cup B)}{P(A)P(B)} \tag{6.22}$$

如果(6.22)式的值小于 1，则 A 的出现和 B 的出现是负相关的。如果结果值大于 1，则 A 和 B 是正相关的，意味每一个的出现都蕴涵另一个的出现。如果结果值等于 1，则 A 和 B 是独立的，它们之间没有相关性。

让我们回头看例 6.6 计算机游戏和录像。

例 6.7 为了帮助过滤掉形如 $A \Rightarrow B$ 的误导的“强”关联，我们需要研究两个项集 A 和 B 怎样才是相关的。设 $\overline{game}$ 表示例 6.6 中不包含计算机游戏的事务，$\overline{video}$ 表示不包含录像的事务。事务可以汇总在**相依表**中。例 6.6 的数据的相依表如表 6.4 所示。由该表可以看出，购买计算机游戏的概率 $P(\{game\}) = 0.60$，购买录像的概率 $P(\{video\}) = 0.75$，而购买二者的概率 $P(\{game, video\}) = 0.40$。根据(6.22)式，$P(\{game, video\}) / (P(\{game\}) \times P(\{video\})) = 0.40 / (0.75 \times 0.60) = 0.89$。由于该值明显比 1 小，{*game*}和{*video*}之间存在负相关。分子是顾客购买二者的可能性，而分母是如果两个购买是完全独立的可能性。这种负相关不能被支持度-置信度框架识别。□

表 6.4 汇总关于购买计算机游戏和录像事务的 2 × 2 相依表

	game	$\overline{game}$	Σ_{row}
video	**4,000**	**3,500**	7,500
$\overline{video}$	2, 000	500	2, 500
Σ_{col}	6, 000	4, 000	10, 000

这激发了识别相关性规则或相关规则的挖掘。**相关规则**形如$\{i_1, i_2, \ldots, i_m\}$，其中，项$\{i_1, i_2, \ldots, i_m\}$的出现是相关的。给定由(6. 22)式确定的相关值，χ^2统计可以确定相关是否是统计意义上的相关。χ^2统计也可以确定负蕴涵。

相关性的一个优点是它是向上封闭的。这意味，如果项集 S 是相关的（即，S 中的项是相关的），则 S 的超集也是相关的。换句话说，添加项到相关集合中，不影响已存在的相关性。χ^2统计在每个有意义的层也是向上封闭的。

在搜索相关集，形成相关规则时，可以使用相关性和χ^2的向上封闭性。由空集开始，考察项集空间（或项集的格），一次添加一个项，寻找**最小相关项集**——相关的项集，其子集都不相关。这些项集形成格的**边界**。由于封闭性，边界以下的项集都不是相关的。由于最小相关项集的所有超集都是相关的，我们可以停止向上搜索。在项集空间进行这种一系列“行走”的算法称作**随机行走算法**。这种算法可以与支持度测试结合，以进行进一步的剪枝。随机行走算法容易使用数据方实现。将这里介绍的过程用于超大规模数据库是一个尚待解决的问题。另一个限制是，当相依表数据稀疏时，χ^2统计不够精确，并且对于大于 2 × 2 相依表可能误导。替代支持度-置信度框架评估关联规则兴趣度的提议在文献注释中给出。

6.6 基于限制的关联挖掘

对于给定的任务相关的数据集，数据挖掘过程可能发现数以千计的规则，其中许多用户并不感兴趣。在**基于限制的挖掘**中，挖掘在用户提供的各种限制的指导下进行。这些限制包括

- **知识类型限制**：指定要挖掘的知识类型，如关联规则。
- **数据限制**：指定任务相关的数据集。
- **维/层限制**：指定所用的维或概念分层结构的层。

- **兴趣度限制**：指定规则兴趣度阈值或统计度量，如支持度和置信度。
- **规则限制**：指定要挖掘的规则形式。这种限制可以用元规则（规则模板）表示，如可以出现在规则前件或后件中谓词的最大或最小个数，或属性、属性值和/或聚集之间的联系。

以上限制可以用高级数据挖掘查询语言说明，如用第 4 章介绍的语言。

上面的前 4 种限制已在本书或本章的前面讨论。本节，我们讨论使用规则限制对挖掘任务聚焦。这种基于限制的挖掘允许用户根据他们关注的目标，说明要挖掘的规则，因此使得数据挖掘过程更有功效。此外，可以使用复杂的挖掘查询优化程序，以便利用用户指定的限制，从而使得挖掘过程更有效率。基于限制的挖掘促进交互式探查挖掘与分析。在 6.6.1 小节，你将学习元规则制导的挖掘，那里用规则模板的形式说明了语法规则限制。6.6.2 小节讨论进一步的规则限制使用，指定集合/子集联系、变量的常量初始化和聚集函数。

6.6.1 关联规则的元规则制导挖掘

“元规则有什么作用？”元规则使得用户可以说明他们感兴趣的规则的语法形式。规则的形式可以作为限制，帮助提高挖掘过程的性能。元规则可以根据分析者的经验、期望或对数据的直觉，或者根据数据库模式自动产生。

例 6.8 假定作为 AllElectronics 的市场分析员，你已经访问了描述顾客的数据（如，顾客的年龄、地址和信誉度等），以及顾客事务的列表。你对找出顾客的特点和他购买的商品之间的关联关系感兴趣。然而，不是要找出反映这种联系的所有关联规则，你特别对什么样的一对顾客特点促进教育软件的销售感兴趣。可以使用一个元规则来说明你感兴趣的规则形式。这种元规则的一个例子是

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“}education\ software\text{”}) \quad (6.23)$$

其中，P_1和 P_2是**谓词变量**，在挖掘过程中被例示为给定数据库的属性；X是变量，代表顾客；Y和 W分别取赋给 P_1和 P_2的属性值。典型地，用户要说明一个例示 P_1和 P_2需考虑的属性列表；否则，将使用省缺的属性集。

一般地，元规则形成一个关于用户希望探查或证实的、他感兴趣联系的假定。然后，挖掘系统可以寻找与给定元规则匹配的规则。例如，规则(6.24)匹配或遵守元规则(6.23)。

$$age(X, \text{“}30...39\text{”}) \wedge income(X, \text{“}41...60K\text{”}) \Rightarrow buys(X, \text{“}education\ software\text{”}) \quad (6.24)$$

□

“元规则如何用于指导挖掘过程？”让我们进一步考察这个问题。假定我们希望挖掘维间关联规则，如上例所示。元规则是形如

$$P_1 \wedge P_2 \wedge ... \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge ... \wedge Q_r \quad (6.31)$$

的规则模板。其中，P_i (i = 1, 2 ,..., l) 和 Q_j (j = 1, 2, ...,r)是例示谓词或谓词变量。设元规则中谓词的个数为 $p = l + r$。为找出满足该模板的维间关联规则

- 我们需要找出所有的频繁 p-谓词集 L_p。
- 我们还必须有 L_p中的 l-谓词子集的支持度或计数，以计算由 L_p导出的规则的置信度。

这是挖掘多维关联规则的典型情况，我们在 6.4 节已介绍。正如在那里看到的，数据方很适合多维关联规则的挖掘，因为它们具有存放聚集维值的能力。由于数据仓库和 OLAP 技术的流行，适合给定挖掘任务的、完全物化的 n-D 数据方可能已经存在。这里，n 是被考虑的对谓词变量例示的属性数，加上在给定元规则中已经例示的谓词数，并且 $n \geq p$。通常，这种 n-D 数据方用方体的格表示，类似于图 6.17 中的那种。在这种情况下，我们只需要扫描 p-D 方体，将每个单元中的计数与最小支持度计数比较，以找出 L_p。由于 1-D 方体已经计算，并包含 L_p的 1-D 谓词子集的计数，然后调用规则产生过程，返回与元规则匹配的强规则。我们称这种方法为**缩减的 n-D 方体搜索**，因为它只考察 p-D 和 1-D 方体，而不是搜索整个 n-D 数据方。

如果用于元规则制导的挖掘任务的 n-D 数据方不存在，我们必须构造和搜索它。只需要计算 p-D 和 1-D 方体，而不是整个数据方。数据方的构造方法已在第 2 章讨论。

6.6.2 用附加的规则限制制导的挖掘

用户可以说明集合/子集联系，变量的常量初始化和聚集函数。这些可以与元规则制导的挖掘一起使用，或作为它的替代。本节，我们考察规则限制，看看怎样使用它们，使得挖掘过程更有效。让我们研究一个例子，其中规则限制用于挖掘混合维关联规则。

例 6.9 假定 AllElectronics 有一个销售多维数据库，包含以下相互关联的关系：

- *sales(customer_name, item_name, transaction_id)*
- *lives(customer_name, region, city)*
- *item(item_name, category, price)*
- *transaction(transaction_id, day, month, year)*

其中，*lives*, *item* 和 *transaction* 是三个维表，通过三个关键字 *customer_name*, *item_name* 和 *transaction_id* 分别链接到事实表 *sales*。

我们的关联挖掘查询是“找出这样的销售，对于 Vancouver 的 1999 年的顾客，什么样的便宜商品（价格和低于$100）能够促进同类贵商品（最低价为$500）的销售？”该查询可以用 DMQL 数据挖掘查询语言表达如下。为方便讨论，查询的每一行已经编号。

(1) **mine associations as**
(2) lives(C,_, “vancouver”) ∧ sales+(C,?{I},{S})⇒ sales+(C,?{J},{T})
(3) **from** sales
(4) **where** S.year = 1999 **and** T.year = 1999 **and** I.category = J.category
(5) **group by** C,I.category
(6) **having** sum(I.price) < 100 **and** min(J.price)≥500
(7) **with** support **threhold** = 1%
(8) **with** confidence **threhold** = 50%

在讨论规则限制之前，让我们仔细看看上面的查询。行 1 是知识类型限制，说明要发现关联模式。行 2 说明了元规则。这是下面混合维关联规则（多维关联规则，其中重复谓词是 *sales*）的元规则的缩写形式：

$$lives(C,\_,\text{"}Vacouver\text{"})$$
$$\wedge sales(C,?I_1,S_1)\wedge\ldots\wedge sales(C,?I_k,S_k)\wedge I=\{I_1,\ldots,I_k\}\wedge S=\{S_1,\ldots,S_k\}$$
$$\Rightarrow sales(C,?J_1,T_1)\wedge\ldots\wedge sales(C,?J_m,T_m)\wedge J=\{J_1,\ldots,J_m\}\wedge T=\{T_1,\ldots,T_m\}$$

这意味一个或多个 *sales* 记录以“$sales(C,?I_1,S_1) \wedge \ldots \wedge sales(C,?I_k,S_k)$”形式驻留在规则的前件（左部），问号“？”表示只有项的名字 $I_1,\ldots, I_k$需要打印。“I={ $I_1,\ldots, I_k$ }”意味出现在前件的所有的 *I* 取集合形式，由行 4 的类 SQL 的 **where**-子句得到。类似的记号用于后件（右端）。

该元规则可能允许类似于下面的关联规则产生

$$lives(C,\_,\text{"}Vancouver\text{"})\wedge sales(C,\text{"}Census\_CD\text{"},\_)\wedge sales(C,\text{"}MS/Office\text{"},\_)\Rightarrow sales(C,\text{"}MS/SQLServer\text{"},\_),\quad [1.5\%,68\%] \tag{6.26}$$

该规则意味，如果顾客住在 Vancouver，买了“*Census_CD*”和“*MS/Office*”，他多半会买“*MS/SQLServer*”（概率 68%），并且所有顾客的 1.5%买这三样。

数据限制在元规则的“*lives*(_,_,”Vancouver”)”部分指定（即，住在 Vancouver 的所有顾客），并在行 3 指出只有事实表 *sales* 需要显示引用。在多维数据库中，变量的引用被简化。例如，“S.year=1999”等价于 SQL 语句“**from** *sales S, transaction R* **where** *S.transaction_ID = R.transaction_ID* **and** *R.year* = 1999”。所有三个维(*lives, item* 和 *transaction*)都使用。层限制如下：对于 *lives*，我们只考虑 *customer_name*，因为只有 *city* = “Vancouver”在选择中使用；对于 *item*，我们考虑 *item_name* 和 *category*，因为它们在查询中使用；对于 *transaction*，我们只考虑 *transaction_ID*，因为 *day* 和 *month* 未被引用，而 *year* 只在选择中使用。

规则限制包含在 **where**（行 4）和 **having**（行 6）子句的大部分，如“*S.year* = 1999”、“*T.year* = 1999”、“ *I.category = J.category*”、“*sum(I.price)* < 100 ”和“*min(J.price)* ≥ 500”。最后，行 7 和 8 说明了两个兴趣度限制（即，阈值）：1%的最小支持度和 50%的最小置信度。□

知识类型和数据限制在挖掘前使用。其它类型的限制可以在挖掘后使用，以便过滤发现的规则。然而，这可能使得挖掘过程非常低效，代价昂贵。维/层限制在 6.3.2 小节已讨论，而兴趣度限制在本章已广泛讨论。现在，让我们把注意力放在规则限制上。

“什么类型的规则限制可以在挖掘过程中使用，以缩小规则搜索空间？”你可能会问。“更特殊地，什么类型的规则限制可以‘推进’到挖掘过程，并且仍然保证回答挖掘查询的完全性？”

对于频繁项集挖掘，规则限制可以分为如下五类：（1）反单调的，（2）单调的，（3）简洁的，（4）可变的，（5）不可变的。对于每一类，我们将使用一个例子展示它的特性，并解释如何将这类限制用在挖掘过程中。

第一类限制是反单调性。考虑例 6.9 的规则限制”sum($I.price$) < 100”。假定我们使用类似于 Apriori 的方法（逐层），对于每次迭代 k，探查 k-项集。其价格和不小于 100 的任何项集都可以由搜索空间剪去，因为向该项集中进一步添加项将会使它更贵，因此不可能满足限制。换一句话说，如果一个项集不满足该规则限制，它的任何超集也不可能满足该规则限制。如果一个规则具有这一性质，则称它是**反单调的**。根据反单调规则限制进行剪枝可以用于类-Apriori 算法的每一次迭代，以帮助提高整个挖掘过程的性能，而保证数据挖掘任务的完全性。

注意，Apriori 性质是说频繁项集的任何非空子集也必然是频繁的，它也是反单调的。如果给定的项集不满足最小支持度，则它的任何超集也不可能满足。这个性质用于 Apriori 算法的每次迭代，以减少考察的候选项集的个数，从而压缩关联规则的搜索空间。

反单调限制的其它例子包括”min($J.Price$) ≥ 500”和”$count(I) \leq 10$”等。任何违反这些限制的项集都可以丢弃，因为向这种项集中添加更多的项不可能满足限制。诸如”avg($I.price$) ≤ 100”的限制不是反单调的。对于一个不满足该限制的项集，通过添加一些（便宜的）项得到的超集可能满足该限制。因此，将这种限制推进到挖掘过程之中，将不保证挖掘任务的完全性。表 6.5 的第二列给出刻画反单调性特性的基于 SQL 原语限制列表。为简化我们的讨论，只给出了存在性操作符（例如，=、∈，但没有≠、∉）和带等号的比较（或包含）操作符（例如，≤、⊆）。

第二类限制是单调性。如果例 6.9 中的规则限制是“sum($I.price$) ≥ 100”，则基于限制的处理方法将很不相同。如果项集 I 满足该限制，即，集合中的单价和不少于 100，进一步添加更多的项到 I 将增加价格，并且总是满足该限制。因此，在项集 I 上进一步检查该限制是多余的。换言之，如果一个项集满足这个规则限制，它的所有超集也满足。如果一个规则具有这一性质，则称它是**单调的**。类似的规则单调限制包括“min($I.price$) ≤ 10”，“count(I) ≥ 10”等。表 6.5 的第三列给出刻画单调性特性的基于 SQL 原语限制列表。

表 6.5　常用的基于 SQL 的限制的特性

限制	反单调的	单调的	简洁的
$v \in S$	否	是	是
$S \supseteq V$	否	是	是
$S \subseteq V$	是	否	是
$min(S) \leq v$	否	是	是
$min(S) \geq v$	是	否	是
$max(S) \leq v$	是	否	是
$max(S) \geq v$	否	是	是
$count(S) \leq v$	是	否	弱
$count(S) \geq v$	否	是	弱
$sum(S) \leq v$ （$\forall a \in S, a \geq 0$）	是	否	否
$sum(S) \geq v$ （$\forall a \in S, a \geq 0$）	否	是	否
$range(S) \leq v$	是	否	否
$range(S) \geq v$	否	是	否
$avg(S) \theta v$, $\theta \in \{=, \leq, \geq\}$	可变的	可变的	否
$support(S) \geq \xi$	是	否	否
$support(S) \leq \xi$	否	是	否

第三类是简洁性限制。对于这类限制，我们可以列出、并且仅仅列出所有确保满足该限制的集合。即，如果一个规则限制是**简洁的**，我们可以直接精确地产生满足它的集合，甚至在支持计数开

始之前。这避免了产生-测试方式的过大开销。换言之，这种限制是计数前可剪枝的。例如，例 6.9 中的限制“min(*J.price*) ≥ 500”是简洁的。这是因为我们能够准确无误地产生满足该限制的所有项集。特殊地，这种集合至少包含一个项，其单价不低于$500。它是这种形式：$S_1 \cup S_2$，其中 $S_1 \neq \varnothing$ 是集合中价格不低于$500 的项的子集；而 S_2可能为空，是集合中价格不超过$500 的项的子集。因为有一个精确“公式”，产生满足简洁限制的所有集合，在挖掘过程中不必迭代地检验规则限制。表 6.5 的第四列给出刻画简洁性特性的基于 SQL 原语限制列表。

第四类限制是**可变的限制**。有些限制不属于以上三类。然而，如果项集中的项以特定的次序排列，则对于频繁项集挖掘过程，限制可能成为单调的或反单调的。例如，限制“avg(*I.price*)”既不是反单调的，也不是单调的。然而，如果事务中的项以单价的递增序添加到项集中，则该限制就成了反单调的，因为如果项集 *I* 违反了该限制（即，平均单价大于$100），更贵的商品进一步添加到该项集中不会使它满足该限制。类似地，如果事务中的项以单价的递减序添加到项集中，则该限制就成了单调的，因为如果项集 *I* 违反了该限制（即，平均单价不超过$100），添加更便宜的商品到当前项集将使得平均单价不大于$100。除表 6.5 给出的“avg(*S*) ≤ *v*”和“avg(*S*) ≥ *v*”外，还有一些其它可变的限制，如“*variance(S)* ≥ *v*”和“*standard_variation(S)* ≥ *v*”等。

注意，以上讨论并不意味每种限制都是可变的。例如，“sum(*S*) θ *v*”不是可变的，其中，θ∈{≤, ≥}并且 *S* 中的元素可以是任意实数。因此，还有第五类限制，称作**不可变的限制**。一个好消息是：尽管有一些难处理的限制是不可变的，大部分使用 SQL 内部聚集的简单 SQL 表达式都属于前四类之一，对于它们可以使用有效的限制挖掘方法。

6.7 总结

- 大量数据之间的关联关系的发现在选择购物、决策分析和商务管理方面是有用的。一个流行的应用领域是**购物篮分析**，通过搜索经常一块购买的商品的集合（或序列），研究顾客的购买习惯。**关联规则挖掘**首先找出**频繁**项集（项的集合，如 *A* 和 *B*，满足最小支持度阈值，或任务相关元组的百分比），然后，由它们产生形如 $A \Rightarrow B$ 的**强**关联规则。这些规则也满足最小置信度阈值（预定义的、在满足 *A* 的条件下满足 *B* 的概率）。
- 根据不同的标准，关联规则可以分成若干类型，如：
  (1) 根据规则所处理的值的类型，关联规则可以分为**布**尔的和**量化的。**布尔关联规则表现离散(分类）对象之间的联系。量化关联规则是多维关联规则，涉及动态离散化的数值属性。它也可能涉及分类属性。
  (2) 根据规则中数据涉及的维，关联规则可以分成**单维**和**多维的**。单维关联规则涉及单个谓词或维，如 *buys*；而多维关联规则涉及多个（不同的）谓词或维。单维关联规则展示的是**维内联系**（即，同一个属性或维内的关联）；而多维关联规则展示的是**维间联系**（即，属性/维之间的关联）。
  (3) 根据规则涉及的抽象层，关联规则可以分为**单层**和**多层**的。在单层关联规则中，项或谓词的挖掘不考虑不同的抽象层；而多层关联规则考虑多个抽象层。
  (4) 根据对关联挖掘的不同扩充，关联挖掘可以扩充为**相关分析**和**最大频繁模式**（“最大模式”）与**频繁闭项集**挖掘。相关分析指出相关项的存在与否。最大模式是一个频繁模式 *p*，使得 *p* 的任何真超集都不是频繁的。频繁闭项集是指：项集 *c* 是闭的，如果不存在 *c* 的真超集 *c'*，使得包含 *c* 的子模式的每个事务也包含 *c'*。
- **Apriori 算法**是一种有效的关联规则挖掘算法，它逐级探查，进行挖掘。**Apriori 性质**：频繁项集的所有非空子集都必须是频繁的。在第 *k* 次迭代，它根据频繁 *k*-项集，形成频繁(*k*+1)-项集候选，并扫描数据库一次，找出完整的频繁(*k*+1)-项集 L_{k+1}。

  涉及散列和事务压缩的变形可以用来使得过程更有效。其它变形涉及划分数据（在每一部分上挖掘，然后合并结果）和数据选样（在数据子集上挖掘）。这些变形可以将数据扫描次数减少到一或两次。
- **频繁模式增长**（**FP-增长**）是一种不产生候选的挖掘频繁项集方法。它构造一个高度压缩的数据结构（FP-树），压缩原来的事务数据库。不是使用类 Apriori 方法的产生-测试策略，它聚焦于频繁模式（段）增长，避免了高代价的候选产生，获得更好的效率。

- **多层关联规则**可以根据每个抽象层上的最小支持度阈值如何定义，使用多种策略挖掘。当在较低层使用**递减的支持度**时，剪枝方法包括层交叉按单项过滤，层交叉按 *k*-项集过滤。冗余的（后代）关联规则可以删除，不向用户提供，如果根据其对应的祖先规则，它们的支持度和置信度接近于期望值的话。
- 挖掘**多维关联规则**可以根据对量化属性处理分为若干类。第一，量化属性可以根据预定义的概念分层静态离散化。数据方非常适合这种方法，因为数据方和量化属性都可以利用概念分层。第二，可以挖掘**量化关联规则**，其量化属性根据分箱动态离散化，“临近的”关联规则可以用聚类组合。第三，可以挖掘**基于距离的关联规则**，其中区间根据聚类定义。
- 并非所有的强关联规则都是有趣的。对于统计相关的项，可以挖掘**相关规则**。
- **基于限制的挖掘**允许用户聚焦，按提供的元规则（即，模式模板）和其它挖掘限制搜索规则。这种挖掘促进了说明性数据挖掘查询语言和用户界面的使用，并对挖掘查询优化提出了巨大挑战。规则限制可以分五类：反单调的、单调的、简洁的、可变的和不可变的。前四类限制可以在关联挖掘中使用，指导挖掘过程，导致更有功效和更有效率的挖掘。
- 关联规则不应当直接用于没有进一步分析或领域知识的预测。它们不必指示因果关系。然而，对于进一步探查，它们是有帮助的切入点。这使得它们成为理解数据的流行工具。

习题

6.1 Apriori 算法使用子集支持度性质的先验知识。

(a) 证明频繁项集的所有非空子集必须也是频繁的。

(b) 证明项集 *s* 的任意非空子集 *s'*的支持度至少和 *s* 的支持度一样大。

(c) 给定频繁项集 *l* 和 *l* 的子集 *s*，证明规则"*s' ⇒ (l-s')*"的置信度不可能大于"*s ⇒ (l-s)*"的置信度。其中，*s'*是 *s* 的子集。

(d) Apriori 的一种变形将事务数据库 *D* 中的事务划分成 *n* 个不重叠的部分。证明在 *D* 中是频繁的任何项集至少在 *D* 的一个部分中是频繁的。

6.2 6.2.2 小节介绍了由频繁项集产生关联规则的方法。提出一个更有效的方法。解释它为什么比 6.2.2 小节的方法更有效。（提示：考虑将习题 6.1(b)和 6.1(c)的性质结合到你的设计中）

6.3 数据库有 4 个事务。设 *min_sup* = 60%，*min_conf* = 80%。

TID	date	items_bought
T100	10/15/99	{K,A,D,B}
T200	10/15/99	{D,A,C,E,B}
T300	10/19/99	{C,A,B,E}
T400	10/22/99	{B,A,D}

(a) 分别使用 Apriori 和 FP-增长算法找出频繁项集。比较两种挖掘过程的有效性。

(b) 列出所有的强关联规则（带支持度 *s* 和置信度 *c*），它们与下面的元规则匹配，其中，*X* 是代表顾客的变量，$item_i$是表示项的变量（例如，"*A*"、"*B*"等）：

$$\forall\ x \in transaction,\ buys(X,\ item_1) \wedge buys(X,\ item_2) \Rightarrow buys(X,\ item_3) \quad [s,\ c]$$

6.4 数据库有 4 个事务。设 *min_sup* = 60%，*min_conf* = 80%。

cust_ID	TID	items_bought（以 brand-item_category 形式）
01	T100	{King's-Carb, Sunset-Milk, Dairyland-Cheese, best-Bread}
02	T200	{Best-Cheese, Dairyland-Milk, Goldenfarm-Apple, tasty-Pie, Wonder-Bread}
01	T300	{Westcoast-Apple, Dairyland-Milk, Wonder-Bread, Tasty-Pie}
03	T400	{Wonder-Bread, Sunset-Milk, Dairyland-Cheese}

(a) 在 *item_category* 粒度（例如，$item_i$可以是“*Milk*”），对于下面规则模板

$$\forall\ x \in transaction,\ buys(X,\ item_1) \wedge buys(X,\ item_2) \Rightarrow buys(X,\ item_3) \quad [s,\ c]$$

对最大的 *k*，列出频繁 *k* -项集和包含最大的 *k* 的频繁 *k*-项集的所有强关联规则。

(b) 在 brand-item_category 粒度（例如，可以是“*Sunset-Milk*”），对于下面的规则模板

$\forall\ x \in customer,\ buys(X,\ item_1) \wedge buys(X,\ item_2) \Rightarrow buys(X,\ item_3)$

对最大的 k, 列出频繁 k -项集。注意：不打印任何规则。

6.5 假定一个大型存储具有分布在 4 个站点的事务数据库。每个成员数据库中的事务具有相同的格式 $T_j: \{i_1, \ldots, i_m\}$；其中，$T_j$是事务标识符，而 $i_k (1 \le k \le m)$是事务中购买的商品标识符。提出一个有效的算法，挖掘全局关联规则（不考虑多层关联规则）。你可以给出你的算法的要点。你的算法不必将所有的数据移到一个站点，并且不造成过度的网络通讯开销。

6.6 假定大型事务数据库 DB 的频繁项集已经存储。讨论：如果新的事务集ΔDB（渐增地）加进，在相同的最小支持度阈值下，如何有效地挖掘（全局）关联规则?

6.7 假定描述 Big-University 大学学生的数据关系已被泛化为表 6.6 的泛化关系 R。

设概念分层如下：

status: {*freshman, sophomore, junior, senior*} ∈ *undergraduate*

{*M.Sc, M.A, Ph.D*} ∈ *graduate*

major: {*physics, chemistry, math*} ∈ *science*

{*CS, engineering*} ∈ *appl_science*

age: {16...20, 21-25} ∈ *young*

{26...30, over_30} ∈ *old*

nationality: {*Asia, Europe, Latin_America*}∈ *foreign*

{Canada, U.S.A.} ∈North_America

表 6.6 习题 6.7 的泛化关系

major	status	age	nationality	gpa	count
French	M.A	over_30	Canada	2.8...3.2	3
CS	junior	16...20	Europe	3.2...36	29
physics	M.S	26...30	Latin_America	3.2...3.6	18
engineering	Ph.D	26...30	Asia	3.6...4.0	78
philosophy	Ph.D	26...30	Europe	3.2...3.6	5
French	senior	16...20	Canada	3.2...3.6	40
chemistry	junior	21...25	U.S.A.	3.6...4.0	25
CS	senior	16...20	Canada	3.2...3.6	70
philosophy	M.S	over_30	Canada	3.6...4.0	15
French	junior	16...20	U.S.A.	2.8...3.2	8
philosophy	junior	26...30	Canada	2.8...3.2	9
philosophy	M.S	26...30	Asia	3.2...3.6	9
French	junior	16...20	Canada	3.2...3.6	52
math	senior	16...20	U.S.A.	3.6...4.0	32
CS	junior	16...20	Canada	3.2...3.6	76

philosophy	Ph.D	26...30	Canada	3.6...4.0	14
philosophy	senior	26...30	Canada	2.8...3.2	19
French	Ph.D	over_30	Canada	2.8...3.2	1
engineering	junior	21...25	Europe	3.2...3.6	71
math	Ph.D	26...30	Latin_America	3.2...3.6	7
chemistry	junior	16...20	U.S.A.	3.6...4.0	46
engineering	junior	21...25	Canada	3.2...3.6	96
French	M.S	over_30	Latin_America	3.2...3.6	4
philosophy	junior	21...25	U.S.A.	2.8...3.2	8
math	junior	16...20	Canada	3.6...4.0	59

设最小支持度阈值为 2%，最小置信度阈值为 50%（每一层）。

(a) 画出 *status, major, age, nationality* 的概念分层。

(b) 对所有层使用一致的支持度，对于下面的规则模板

$$\forall S \in R \quad P(S,x) \wedge Q(S,y) \Rightarrow gpa(S, z) \qquad [s, c]$$

其中，$P, Q \in \{status, major, age, nationality\}$。找出 R 中的多层强关联规则。

(c) 使用层交叉单项过滤，找出 R 中的多层强关联规则。其中，递减的支持度 10%用于如下规则模板的最低抽象层：

$$\forall S \in R \quad P(S,x) \wedge Q(S,y) \Rightarrow gpa(S, z) \qquad [s, c]$$

不要挖掘交叉层规则。

6.8 提出并给出挖掘多层关联规则的**层共享**挖掘方法的要点。其中，每个项用它的层位置编码，一次初始数据库扫描收集每个概念层的每个项的计数，识别频繁和子频繁项集。将用该方法挖掘多层关联规则与挖掘单层关联规则的花费进行比较。

6.9 证明：包含项 h 和其祖先 $\hat{h}$ 的项集 H 的支持度与项集 $H-\hat{h}$ 的支持度相同。解释如何将它用于层交叉关联规则挖掘。

6.10 在挖掘层交叉关联规则时，假定发现项集"{IBM dssktop computer, printer}"不满足最小支持度。这一信息可以用来剪去诸如"{IBM desktop computer, b/w printer}"的"后代"项集的挖掘吗？给出一个一般规则，解释这一信息如何用于对搜索空间剪枝。

6.11 提出一种挖掘混合维关联规则（多维关联规则，有重复谓词）的方法。

6.12 给出一个短例子，表明强关联规则中的项可能实际上是负相关的。

6.13 下面的相依表汇总了超级市场的事务数据。其中，*hot dog* 表示包含热狗的事务，$\overline{hotdog}$ 表示不包含热狗的事务，*humburgers* 表示包含汉堡包的事务，$\overline{humburgers}$ 表示不包含汉堡包的事务。

	hot dog	$\overline{hotdog}$	Σ_{row}
humburgers	2,000	500	2,500
$\overline{humburgers}$	1,000	1,500	2,500
Σ_{col}	3,000	2,000	5,000

(a) 假定发现关联规则"*hot dog* $\Rightarrow$ *humburgers*"。给定最小支持度阈值 25%，最小置信度阈值 50%，该关联规则是强的吗？

(b) 根据给定的数据，买 *hot dog* 独立于买 *humburgers* 吗？如果不是，二者之间存在何种相关联系？

6.14　序列模式可以用类似于关联规则挖掘的方法挖掘。设计一个有效的算法，由事务数据库挖掘多层序列模式。这种模式的一个例子如下：“买 PC 的顾客在三个月内将买 Microsoft 软件”，在其上，可以下钻，发现该模式的更详细的版本，如“买 Pentium Pro 的顾客在三个月内将买 Microsoft Office ”。

6.15　证明下面表中的每一项正确地刻画了它对应的关于频繁项集挖掘的规则限制。

	规则限制	反单调性	单调性	简洁性
(a)	$v \in S$	否	是	是
(b)	$S \subseteq V$	是	否	是
(c)	$min(S) \leq v$	否	是	是
(d)	$range(S) \leq v$	是	否	否
(e)	$avg(S) \geq v$	可变的	可变的	否

6.16　商店里每种商品的价格是非负的。商店经理只关心如下形式的规则：“一件免费商品可能触发在同一事务中$200 的总购物”。陈述如何有效地挖掘这种规则。

6.17　商店里每种商品的价格是非负的。对于以下每种情况，识别它们提供的限制类型，并简略讨论如何有效地挖掘这种关联规则。

(a)　至少包含一件 Nintendo 游戏。

(b)　包含一些商品，它们的单价和小于$150。

(c)　包含一件免费商品，并且其它商品的单价和至少是$200。

(d)　所有商品的平均价格在$100 和$500 之间。

文献注释

关联规则挖掘首先由 Agrawal, Imielinski 和 Swami[AIS93b] 提出。6.2.1 小节讨论的 Aprior*i* 算法由 Agrawal 和 Srikant [AS94] 提出。使用类似的剪枝方法的算法变形独立地由 Mannila, Toivonen 和 Verkamo[MTV94]开发。结合这些工作的联合出版物稍后出现在 Aga\rawal, Mannila, Skrikant, Toivonen 和 Verkamo[AMS+96]。产生关联规则的方法在 Agrawal 和 Srikant[AS94a]中介绍。6.2.3 小节的 *Apriori* 的变形包括如下引文。使用 *hash* 表提高关联规则挖掘效率被 Park,Chen 和 Yu[PCY95]研究。扫描和事务压缩技术在 Agrawal 和 Srikant[AS94b],Han 和 Fu[HF95],以及 Park, Chen 和 Yu[PCY95a]中介绍。划分技术由 Savasere, Omiecinski 和 Navathe[SON95]提出。选样方法在 Toivonen[Toi96]中讨论。动态项集计数在 Brin, Motwani, Ullman 和 Tsur[BMUT97]中给出。关联规则挖掘有许多扩充，包括序列模式挖掘(Agrawal 和 Srikant[AS95]),espisodes 挖掘(Mannila, Toivonen 和 Verkamo[MTV97]），挖掘空间关联规则（Kopeski 和 Han[KH95]），挖掘有环的关联规则（Özden, Ramaswamy 和 Silberschatz[ORS98]），挖掘否定的关联规则（Savasere, Omiecinski 和 Navathe[SON98]),挖掘事务间关联规则(Lu, Han 和 Feng[LHF98])和日历购物篮分析(Ramaswamy, Mahajan 和 Silberschatz[RMS98]）。最大模式的挖掘在 Bayaedo[Bay98]中介绍。频繁闭合项集的挖掘由 Pasquier, Bastile, Taouil 和 Lakhal[PBTL99]提出，而有效的挖掘算法由 Pei, Han 和 Mao[PHM00]提出。冰山查询在 Fang, Shivakumar, Garcia-Molina 等[FSGM+98] 中介绍，而 Beyer 和 Ramakrishnan[BR99]开发了冰山查询的有效计算方法。频繁项集的深度优先产生由 Agrawal,Aggarwal 和 Prasad[AAP00]提出。挖掘频繁模式而不产生候选的方法由 Han, Pie 和 Yin[HPY00]提出。

多层关联规则挖掘在 Han 和 Fu[HF95],Srikant 和 Agrawal[SA95]中研究。在 Srikant 和 Agrawal [SA]中，这种挖掘以广义关联规则的形式研究，并提出 R-兴趣度度量，以删除冗余规则。

6.4.3 小节介绍的根据规则聚类挖掘量化关联规则的 ARCS 系统由 Lent,Swami 和 Widom[LSW97]提出。基于 *x*-单调和直线区间挖掘量化规则的技术由 Fukuda, Morimoto, Morishita 和 Tokuyama[FMMT96]，Yoda, Fukuda, Morimoto 等[YFM+97]提出。挖掘量化关联规则的非基于栅格的

技术使用部分完全性度量，由 Srikant 和 Agrawal[SA96]提出。6.4.3 小节介绍的挖掘区间数据上的（基于距离的）关联规则由 Miller 和 Yang[MY97]提出。使用量化属性的静态离散化和数据方，挖掘多维关联规则被 Kamber, Han 和 Chiang[KHC97]研究。

在数据挖掘中规则的统计独立性被 Piatetski_Shapiro[PS91]研究。强关联规则的兴趣度问题被 Chen, Han 和 Yu[CHY96]，Brin, Motwani 和 Silverstein[BMS97], Aggarwal 和 Yu[AY99]讨论。推广关联到相关的有效方法在 Brin, Motwani 和 Silverstein[BMS97]，并简略总结 6.5.2 小节中。评估关联规则兴趣度的支持度-置信度框架的其它替代在 Brin, Motwani, Ullman 和 Tsur[BMUT97]，Ahmed, EI-Makky 和 Taha[AEMT00]中提出。Silverstein, Brin, Motwani 和 Ullman[SBMU98]研究了挖掘事务数据库因果关系结构的问题。

使用元规则作为语法或语义过滤器，定义单维关联规则的感兴趣形式由 Klemettinen, Mannila, Ronkainen 等[KMR+94]提出。元规则制导的挖掘在 Shen, Ong, Mitbander 和 Zaniolo[SOMZ96]中提出；那里，元规则后件指定用于满足元规则前件的数据的动作（如，贝叶斯聚类）。关联规则元规则制导的挖掘基于关系的方法在 Fu 和 Han[HF95]中研究。基于数据方的方法在 Kamber, Han 和 Chiang[KHC97]中研究。6.4.2 小节基于限制的关联规则挖掘在 Ng, Lakshmanan, Han 和 Pang[NLHP98]，Lakshmanan, Ng, Han 和 Pang [LNHP99], Pei 和 Han[PH00]中研究。挖掘受限的相关集的有效方法在 Grahne, Lakshmanan 和 Wang[GLW00]中给出。涉及在挖掘中使用模板或谓词限制的其它思想在[AK93, DT93, HK91, LHC97, St96, SVA97]中讨论。

本章提供的关联规则挖掘语言基于 Han, Fu, Wang 等[HFW+96]提出的数据挖掘查询语言 DMQL 的扩充，结合了由 Meo,Paila 和 Ceri[MPC96]提出的挖掘单维关联规则的类 SQL 的操作。预计今后的版本将遵循 Microsoft 公司提出的 DM OLE DB[Mic00]语法。

挖掘关联规则的有效渐增更新由 CheungHan, Ng 和 Wong[CHNW96]提出。在 Apriori 框架下，并行和分布关联规则挖掘被 Park,Chen 和 Yu[PCY95b]，Agrawal 和 Shafer[AS96]，Cheung, han, Ng 等[CHN+96]研究。另一种并行关联规则挖掘方法使用垂直数据库设计探查项集聚类，在 Zaki, Parthasarathy, Ogihara 和 Li[ZPOL97]中提出。

第七章 分类和预测

数据库内容丰富，蕴藏大量信息，可以用来作出智能的商务决策。分类和预测是两种数据分析形式，可以用于提取描述重要数据类的模型或预测未来的数据趋势。然而，分类是预测分类标号（或离散值），而预测建立连续值函数模型。例如，可以建立一个分类模型，对银行贷款的安全或风险进行分类；而可以建立预测模型，给定潜在顾客的收入和职业，预测他们在计算机设备上的花费。许多分类和预测方法已被机器学习、专家系统、统计和神经生物学方面的研究者提出。大部分算法是内存算法，通常假定数据量很小。最近的数据挖掘研究建立在这些工作之上，开发了可规模化的分类和预测技术，能够处理大的、驻留磁盘的数据。这些技术通常考虑并行和分布处理。

本章，你将学习数据分类的基本技术，如判定树归纳、贝叶斯分类和贝叶斯网络、神经网络。数据仓库技术与分类的集成，以及基于关联的分类也在本章讨论。本章还介绍其它分类方法，如 *k*-最临近分类、基于案例的推理、遗传算法、粗糙集和模糊逻辑技术。预测方法，包括线性的、非线性的、广义线性回归也将简要讨论。你将学会修改、扩充和优化这些技术，将它们应用到大型数据库的分类和预测。

7.1 什么是分类？什么是预测？

数据分类是一个两步过程（图 7.1）。第一步，建立一个模型，描述预定的数据类或概念集。通过分析由属性描述的数据库元组来构造模型。假定每个元组属于一个预定义的类，由一个称作**类标号属性**的属性确定。对于分类，数据元组也称作样本、实例或对象。为建立模型而被分析的数据元组形成**训练数据集**。训练数据集中的单个元组称作**训练样本**，并随机地由样本群选取。由于提供了每个训练样本的类标号，该步也称作**有指导的学习**（即，模型的学习在被告知每个训练样本属于哪个类的“指导”下进行）。它不同于**无指导的学习**（或**聚类**），那里每个训练样本的类标号是未知的，要学习的类集合或数量也可能事先不知道。聚类是第 8 章的主题。

通常，学习模型用分类规则、判定树或数学公式的形式提供。例如，给定一个顾客信用信息的数据库，可以学习分类规则，根据他们的信誉度优或相当好来识别顾客（图 7.1(a)）。该规则可以用来为以后的数据样本分类，也能对数据库的内容提供更好的理解。

第二步（图 7.1(b)），使用模型进行分类。首先评估模型（分类法）的预测准确率。本章的 7.9 节介绍评估分类准确率的多种方法。**保持（holdout）方法**是一种使用类标号样本**测试集**的简单方法。这些样本随机选取，并独立于训练样本。模型在给定测试集上的**准确率**是正确被模型分类的测试样本的百分比。对于每个测试样本，将已知的类标号与该样本的学习模型类预测比较。注意，如果模型的准确率根据训练数据集评估，评估可能是乐观的，因为学习模型倾向于过分适合数据（即，它可能并入训练数据中某些异常，这些异常不出现在总体样本群中）。因此，使用测试集。

如果认为模型的准确率可以接受，就可以用它对类标号未知的数据元组或对象进行分类。（这种数据在机器学习也称为“未知的”或“先前未见到的”数据）。例如，在图 7.1(a)通过分析现有顾客数据学习得到的分类规则可以用来预测新的或未来顾客的信誉度。

“预测和分类有何不同？”**预测**是构造和使用模型评估无标号样本，或评估给定样本可能具有的属性值或值区间。在这种观点下，分类和回归是两类主要预测问题；其中，分类是预测离散或标称值，而回归用于预测连续或有序值。然而，我们的观点是：预测类标号为分类，预测连续值（例如，使用回归方法）为预测。这种观点在数据挖掘界广泛接受。

分类和预测具有广泛的应用，包括信誉证实、医疗诊断、性能预测和选择购物。

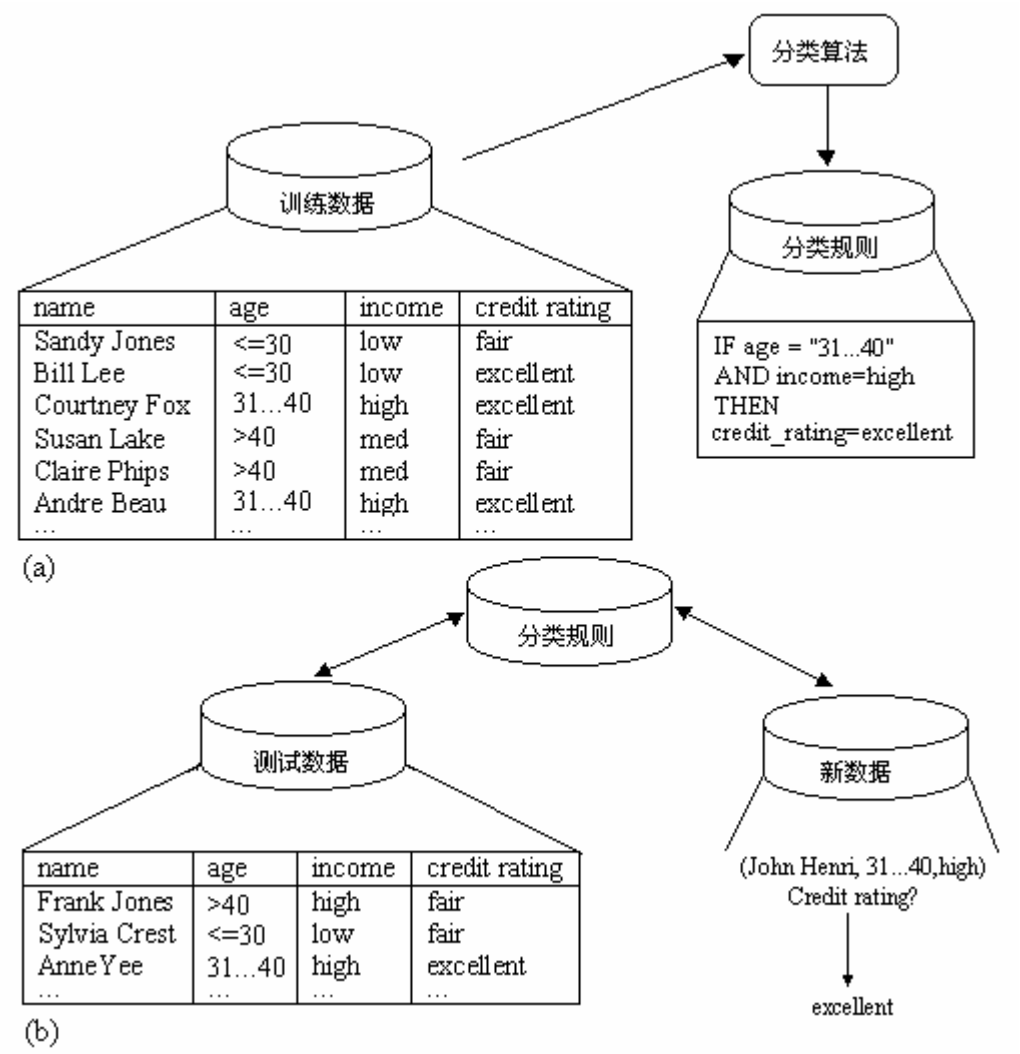


图 7.1　数据分类过程：(a) 学习：用分类算法分析训练数据。这里，类标号属性是 *credit_rating*，学习模型或分类法以分类规则形式提供。　(2) 分类：测试数据用于评估分类规则的准确率。如果准确率是可以接受的，则规则用于新的数据元组分类

例 7.1　假定我们有一个 AllElectronics 的邮寄清单数据库。邮寄清单用于分发介绍新产品和降价信息材料。数据库描述顾客的属性，如他们的姓名、年龄、收入、职业和信誉度。顾客可以按他们是否在 AllElectronics 购买计算机分类。假定新的顾客添加到数据库中，你想将新计算机的销售信息通知顾客。将促销材料分发给数据库中的每个新顾客的费用可能很高。一个更有效的方法是只给那些可能买新计算机的顾客寄材料。为此，可以构造和使用分类模型。

另外，假定你想预测在一个财政年度，一个顾客将在 AllElectronics 进行的主要购买数量。由于预测的值是有序的，为此可以构造一个预测模型。□

7.2　关于分类和预测的问题

本节介绍分类和预测数据的预处理问题。分类方法的比较和评估标准也在本节介绍。

7.2.1 准备分类和预测数据

可以对数据使用下面的预处理，以便提高分类和预测过程的准确性、有效性和可规模性。

- **数据清理**：是旨在消除或减少数据噪音（例如，使用平滑技术）和处理遗漏值（例如，用该属性最常出现的值，或根据统计，用最可能的值替换遗漏值）的数据预处理。尽管大部分分类算法都有处理噪音和遗漏值的机制，但该步骤有助于减少学习时的混乱。
- **相关性分析**：数据中许多属性可能与分类和预测任务不相关。例如，记录银行贷款星期几签署的数据可能与应用的成功不相关。此外，其它属性可能是冗余的。因此，可以进行相关分析，删除学习过程中不相关或冗余属性。在机器学习，这一过程称为特征选择。包含这些属性将减慢和误导学习步骤。

  理想地，用在相关分析上的时间，加上从“压缩的”结果子集上学习的时间，应当少于由原来的数据集合上学习所花的时间。因此，这种分析可以帮助提高分类的有效性和可规模性。
- **数据变换**：数据可以泛化到较高层概念。概念分层可以用于此目的。对于连续值属性，这一步非常有用。例如，属性 *income* 的数值值可以泛化为离散的区间，如 *low, medium* 和 *high*。类似地，标称值，如 *street*，可以泛化到高层概念，如 *city*。由于泛化压缩了原来的训练数据，学习时的输入/输出操作将减少。

  数据也可以规范化，特别是在学习阶段使用神经网络或涉及距离度量的方法时。**规范化**涉及将属性的所有值按比例缩放，使得它们落入较小的指定区间，如-1.0 到 1.0，或 0.0 到 1.0。例如，在使用距离度量的方法中，这可以防止具有较大初始域的属性（如 *income*）相对于具有较小初始域的属性（如二进位属性）权重过大。

数据清理、相关分析和数据变换已在本书的第 3 章详细介绍。相关分析还在第 5 章介绍过。

7.2.2 比较分类方法。

分类和预测方法可以根据下列标准进行比较和评估：

- **预测的准确率**：这涉及模型正确地预测新的或先前未见过的数据的类标号的能力。
- **速度**：这涉及产生和使用模型的计算花费。
- **强壮性**：这涉及给定噪音数据或具有遗漏值的数据，模型正确预测的能力。
- **可规模性**：这涉及给定大量数据，有效地构造模型的能力。
- **可解释性**：这涉及学习模型提供的理解和洞察的层次。

这些问题的讨论贯穿本章。数据库研究界对数据挖掘的分类和预测的贡献一直强调可规模性，特别是对判定树归纳。

7.3 用判定树归纳分类

“什么是判定树？”**判定树**是一个类似于流程图的树结构；其中，每个内部结点表示在一个属性上的测试，每个分枝代表一个测试输出，而每个树叶结点代表类或类分布。树的最顶层结点是根结点。一棵典型的判定树如图 7.2 所示。它表示概念 *buys_computer*，即，它预测 AllElectronics 的顾客是否可能购买计算机。内部结点用矩形表示，而树叶用椭圆表示。

为了对未知的样本分类，样本的属性值在判定树上测试。路径由根到存放该样本预测的叶结点。判定树容易转换成分类规则。

在 7.3.1 小节， 我们介绍学习判定树的基本算法。在判定树构造时，许多分枝可能反映的是训练数据中的噪音或局外者。树剪枝试图检测和剪去这种分枝，以提高在未知数据上分类的准确性。

树剪枝在 7.3.2 小节介绍。由判定树提取分类规则在 7.3.3 小节讨论。基本判定树算法的加强在 7.3.4 小节给出。大型数据库判定树归纳的可规模性问题在 7.3.5 小节讨论。7.3.6 小节介绍判定树归纳与诸如数据方等数据仓库机制的集成，允许多个粒度层的判定树挖掘。判定树已在由医疗到游戏理论和商务等应用领域广泛使用。判定树是一些商业规则归纳系统的基础。

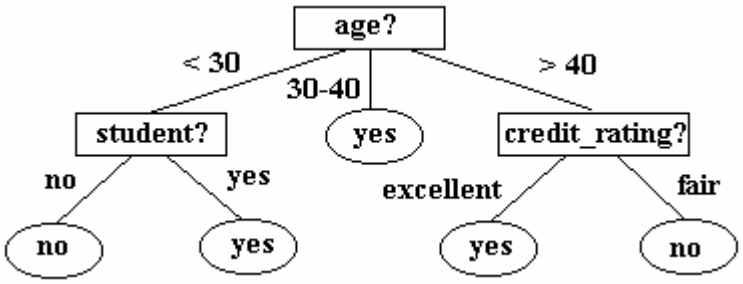


图 7.2　概念 *buys_computer* 的判定树，指出 *AllElectronics* 的顾客是否可能购买计算机。每个内部（非树叶）结点表示一个属性上的测试，每个树叶结点代表一个类（*buys_computer = yes*, 或 *buys_computer = no*）

算法：Generate_decision_tree。由给定的训练数据产生一棵判定树。

输入：训练样本 *samples*，由离散值属性表示；候选属性的集合 *attribute_list*。

输出：一棵判定树。

方法：

(1) 创建结点 N;
(2) **if** *samples* 都在同一个类 C **then**
(3)　　return N 作为叶结点，以类 C 标记;
(4) **if** *attribut_list* 为空 **then**
(5)　　return N 作为叶结点，标记为 *samples* 中最普通的类;　　//majority voting
(6) 选择 *attribute_list* 中具有最高信息增益的属性 *test_attribute*;
(7) 标记结点 N 为 *test_attribute*;
(8) **for each** *test_attribute* 中的未知值 a_i　//partition the samples
(9)　　由结点 N 长出一个条件为 *test_attribute* = a_i 的分枝;
(10)　　设 s_i 是 *samples* 中 *test_attribute* = a_i 的样本的集合; //a partition
(11)　　**if** s_i 为空 **then**
(12)　　加上一个树叶，标记为 *samples* 中最普通的类;
(13)　　**else**　加　上　一　个　由　Generate_decision_tree(s_i, *attribute_list-test_attribute*)返回的结点;

图 7.3　由训练样本归纳判定树的基本算法

7.3.1　判定树归纳

判定树归纳的基本算法是贪心算法，它以自顶向下递归的划分-控制方式构造判定树。算法在图 7.3 中，是一种著名的判定树算法 ID3 版本。算法的扩展将在 7.3.2 到 7.3.6 小节讨论。算法的基本策略如下：

- 树以代表训练样本的单个结点开始（步骤 1）。
- 如果样本都在同一个类，则该结点成为树叶，并用该类标号（步骤 2 和 3）。

- 否则，算法使用称为信息增益的基于熵的度量作为启发信息，选择能够最好地将样本分类的属性（步骤 6）。该属性成为该结点的“测试”或“判定”属性（步骤 7）。在算法的该版本中，所有的属性都是分类的，即离散值。连续属性必须离散化。
- 对测试属性的每个已知的值，创建一个分枝，并据此划分样本（步骤 8-10）。
- 算法使用同样的过程，递归地形成每个划分上的样本判定树。一旦一个属性出现在一个结点上，就不必该结点的任何后代上考虑它（步骤 13）。
- 递归划分步骤仅当下列条件之一成立停止：

  (a) 给定结点的所有样本属于同一类（步骤 2 和 3）。
  (b) 没有剩余属性可以用来进一步划分样本（步骤 4）。在此情况下，使用**多数表决**（步骤 5）。这涉及将给定的结点转换成树叶，并用样本中的多数所在的类标记它。替换地，可以存放结点样本的类分布。
  (c) 分枝 *test_attribute* = a_i 没有样本（步骤 11）。在这种情况下，以 *samples* 中的多数类创建一个树叶（步骤 12）。

属性选择度量

在树的每个结点上使用**信息增益**度量选择测试属性。这种度量称作属性选择度量或分裂的优劣度量。选择具有最高信息增益（或最大熵压缩）的属性作为当前结点的测试属性。该属性使得对结果划分中的样本分类所需的信息量最小，并反映划分的最小随机性或 “不纯性”。这种信息理论方法使得对一个对象分类所需的期望测试数目最小，并确保找到一棵简单的（但不必是最简单的）树。

设 S 是 s 个数据样本的集合。假定类标号属性具有 m 个不同值，定义 m 个不同类 C_i（$i = 1, \ldots, m$）。设 s_i 是类 C_i 中的样本数。对一个给定的样本分类所需的期望信息由下式给出：

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{7.1}$$

其中，p_i 是任意样本属于 C_i 的概率，并用 s_i / s 估计。注意，对数函数以 2 为底，因为信息用二进位编码。

设属性 A 具有 v 个不同值$\{a_1, \ldots, a_v\}$。可以用属性 A 将 S 划分为 v 个子集$\{S_1, \ldots, S_v\}$；其中，S_j 包含 S 中这样一些样本，它们在 A 上具有值 a_j。如果 A 选作测试属性（即，最好的划分属性），则这些子集对应于由包含集合 S 的结点生长出来的分枝。设 s_{ij} 是子集 S_j 中类 C_i 的样本数。根据 A 划分子集的熵或期望信息由下式给出：

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + ... + s_{mj}}{s} I(s_{1j}, ..., s_{mj}) \tag{7.2}$$

项 $\frac{s_{1j} + ... + s_{mj}}{s}$ 充当第 j 个子集的权，并且等于子集（即，A 值为 a_j）中的样本个数除以 S 中的样本总数。熵值越小，子集划分的纯度越高。注意，对于给定的子集 S_j，

$$I(s_{1j}, s_{2j}, ..., s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}) \tag{7.3}$$

其中，$p_{ij} = \frac{s_{ij}}{|S_j|}$，是 S_j 中的样本属于 C_i 的概率。

在 A 上分枝将获得的编码信息是

$$Gain(A) = I(s_1, s_2, ..., s_m) - E(A) \tag{7.4}$$

换言之，*Gain(A)*是由于知道属性 *A* 的值而导致的熵的期望压缩。

算法计算每个属性的信息增益。具有最高信息增益的属性选作给定集合 *S* 的测试属性。创建一个结点，并以该属性标记，对属性的每个值创建分枝，并据此划分样本。

例 7.2　判定树归纳。表 7.1 给出了取自 AllElectronics 顾客数据库数据元组训练集。（该数据取自[Qui86]）。类标号属性 *buys_computer* 有两个不同值（即，{*yes, no*}），因此有两个不同的类($m = 2$)。设类 C_1对应于 *yes*，而类 C_2对应于 *no*。类 *yes* 有 9 个样本，类 *no* 有 5 个样本。为计算每个属性的信息增益，我们首先使用(7.1)式，计算对给定样本分类所需的期望信息：

$$I(s_1, s_2) = I(9,5) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

表 7.1　AllElectronics 顾客数据库训练数据元组

RID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31...40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31...40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31...40	medium	no	excellent	yes
13	31...40	high	yes	fair	yes
14	>40	medium	no	excellent	no

下一步，我们需要计算每个属性的熵。让我们从属性 *age* 开始。我们需要观察 *age* 的每个样本值的 *yes* 和 *no* 分布。我们对每个分布计算期望信息。

对于 *age* = "<=30"：$s_{11} = 2$　$s_{21} = 3$　$I(s_{11}, s_{21}) = 0.971$

对于 *age* = "31...40"：$s_{12} = 4$　$s_{22} = 0$　$I(s_{12}, s_{22}) = 0$

对于 *age* = ">40"：$s_{13} = 3$　$s_{23} = 2$　$I(s_{13}, s_{23}) = 0.971$

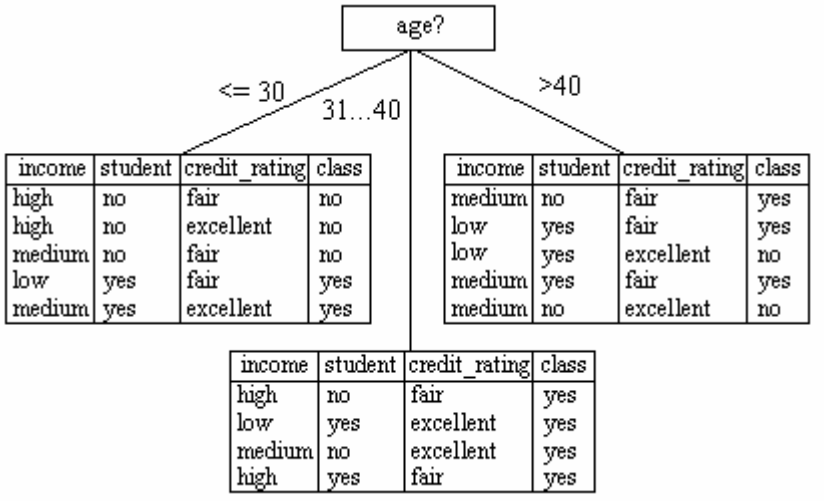
使用(7.2)式，如果样本按 *age* 划分，对一个给定的样本分类所需的期望信息为：

$$E(age) = \frac{5}{14} I(s_{11}, s_{21}) + \frac{4}{14} I(s_{12}, s_{22}) + \frac{5}{14} I(s_{13}, s_{23}) = 0.694$$

因此，这种划分的信息增益是

$$gain(age) = I(s_1, s_2) - E(age) = 0.246$$

类似地，我们可以计算 *Gain(income)* = 0.029, *Gain(student)* =0.151 和 *Gain(credit_rating)* = 0.048。由于 *age* 在属性中具有最高信息增益，它被选作测试属性。创建一个结点，用 *age* 标记，并对于每个属性值，引出一个分枝。样本据此划分，如图 7.4 所示。注意，落在分区 *age* = “31...40” 的样本都属于同一类。由于它们都属于同一类 *yes*，因此要在该分枝的端点创建一个树叶，并用 *yes* 标记。算法返回的最终判定树如图 7.2 所示。□



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

图 7.4：属性 *age* 具有最高信息增益，因此成为判定树根的测试属性。
由每个 *age* 引出分枝，样本据此划分

总而言之，判定树归纳算法已在广泛的应用领域用于分类。这种系统不使用领域知识。判定树归纳的学习和分类步骤通常很快。

7.3.2 树剪枝

当判定树创建时，由于数据中的噪音和局外者，许多分枝反映的是训练数据中的异常。剪枝方法处理这种过分适应数据问题。通常，这种方法使用统计度量，剪去最不可靠的分枝，这将导致较快的分类，提高树独立于测试数据正确分类的可靠性。

“树剪枝如何做？”有两种常用的剪枝方法。

在**先剪枝**方法中，通过提前停止树的构造（例如，通过决定在给定的结点上不再分裂或划分训练样本的子集）而对树“剪枝”。一旦停止，结点成为树叶。该树叶可能持有子集样本中最频繁的类，或这些样本的概率分布。

在构造树时，统计意义下的度量，如χ^2、信息增益等，可以用于评估分裂的优劣。如果在一个结点划分样本将导致低于预定义阈值的分裂，则给定子集的进一步划分将停止。然而，选取一个适当的阈值是困难的。较高的阈值可能导致过分简化的树，而较低的阈值可能使得树的化简太少。

第二种方法是**后剪枝**方法，它由“完全生长”的树剪去分枝。通过删除结点的分枝，剪掉树结点。代价复杂性剪枝算法是后剪枝方法的一个实例。最下面的未被剪枝的结点成为树叶，并用它先前分枝中最频繁的类标记。对于树中每个非树叶结点，算法计算该结点上的子树被剪枝可能出现的期望错误率。然后，使用每个分枝的错误率，结合沿每个分枝观察的权重评估，计算不对该结点剪枝的期望错误率。如果剪去该结点导致较高的期望错误率，则保留该子树；否则剪去该子树。逐渐产生一组被剪枝的树之后，使用一个独立的测试集评估每棵树的准确率，就能得到具有最小期望错误率的判定树。

我们可以根据编码所需的二进位位数，而不是根据期望错误率，对树进行剪枝。“最佳剪枝树”使得编码所需的二进位最少。这种方法采用最小描述长度（MDL）原则。由该原则，最简单的解是最期望的。不象代价复杂性剪枝，它不需要独立的样本集。

也可以交叉使用先剪枝和后剪枝，形成组合式方法。后剪枝所需的计算比先剪枝多，但通常产生更可靠的树。

7.3.3 由判定树提取分类规则

“我可以由我的判定树得到分类规则吗？如果能，怎么做？”可以提取判定树表示的知识，并以 IF-THEN 形式的分类规则表示。对从根到树叶的每条路经创建一个规则。沿着给定路经上的每个属性-值对形成规则前件（“IF”部分）的一个合取项。叶结点包含类预测，形成规则后件（“THEN”部分）。IF-THEN 规则易于理解，特别是当给定的树很大时。

例 7.3　由判定树产生分类规则。沿着由根结点到树叶结点的路经，图 7.2 的判定树可以转换成 IF-THEN 分类规则。由图 7.2 提取的规则是：

IF *age* = ”<=30” AND *student* = “*no*” THEN *buys_computer* = “*no*”
IF *age* = ”<=30” AND *student* = “*yes*” THEN *buys_computer* = “*yes*”
IF *age* = ”31...40” THEN *buys_computer* = “*yes*”
IF *age* = ”>40” AND *credit_rating* = “*excellent*” THEN *buys_computer* = “*no*”
IF *age* = ”>40” AND *credit_rating* = “*fair*” THEN *buys_computer* = “*yes*”

C4.5（ID3 算法的后继版本）使用训练样本估计每个规则的准确率。由于这将导致对规则的准确率的乐观估计，C4.5 使用一种悲观估计来补偿偏差。替换地，也可以使用一组独立于训练样本的测试样本来评估准确性。

通过删除规则前件中无助于改进规则评估准确性的条件，可以对规则“剪枝”。对于每一类，类中规则可以按它们的精确度定序。由于一个给定的样本可能不满足任何规则前件，通常将一个指定主要类的省缺规则添加到规则集中。

7.3.4 基本判定树归纳的加强

“对基本判定树归纳的加强有哪些？”业已提出了许多对 7.3.1 小节判定树归纳基本算法的加强。本小节，我们将讨论若干主要加强，其中一些结合到 ID3 的后继算法 C4.5 中。

7.3.1 小节的判定树归纳基本算法要求所有的属性是分类的或离散化的。可以修改该算法，允许属性具有整个离散区间或连续值。在这种属性 A 上的测试导致两个分枝，对应于条件 $A \leq V$ 和 $A > V$，其中 V 是 A 的某个数值值。给定 A 的值 v，确定 V 时考虑 v-1 个可能的分割。通常，考虑每对相邻值的中间值。如果这些值已预先排序，则只需要扫描一次这些值。

信息增益度量有倾斜，它倾向于适合具有许多值的属性。已经提出了一些替代的方法，如增益率，它考虑每个属性值的概率。还有一些其它选择度量，包括 Gini 索引，χ^2 相依表统计和 G-统计。

已经提出了许多方法来处理遗漏的属性值。例如，属性 A 的遗漏值或未知值可以用 A 的最常见值替代。替换地，属性 A 的外观上的信息增益也可以根据 A 的值未知的样本百分比减少。这样，具有缺少值的样本“片段”可以在测试结点被划分到多个分枝。其它方法可能寻找 A 的最可能值，或使用 A 和其它属性的已知联系。

通过重复地将数据划分成越来越小的部分，判定树归纳可能面临碎片、重复和复制问题。**碎片**是指一个给定分枝中的样本数太小，没有统计意义。解决该问题的一种方法是将分类属性值分组。树结点可以测试一个属性值是否属于给定的集合，如 $A_i \in \{a_1, a_2, \ldots, a_n\}$。另一种替代是创建二叉判定树，其每个分枝拥有一个属性上的布尔测试。二叉树导致较少的数据碎片。一些实验研究发现二叉判定树比传统的判定树更精确。当一个属性沿树的一个给定分枝重复测试时，就出现**重复**。**复制**是复制树中已存在的子树。**属性**（特征）**构造**是防止这三个问题的一种方法。通过由给定的属

性创建新的属性，改进给定属性的受限表示。属性构造作为数据变换的一种形式，已在第 2 章讨论过。

业已提出了判定树归纳的**增量**版本。当给定新的训练数据时，增量方法重新构造由先前的训练数据学习获得的判定树，而不是“盲目地”通过学习构造一棵新树。

还有一些对基本判定树归纳的加强，它们关注可规模性和与数据仓库技术的集成。这些分别在 7.3.5 和 7.3.6 小节讨论。

7.3.5 判定树归纳的可规模性

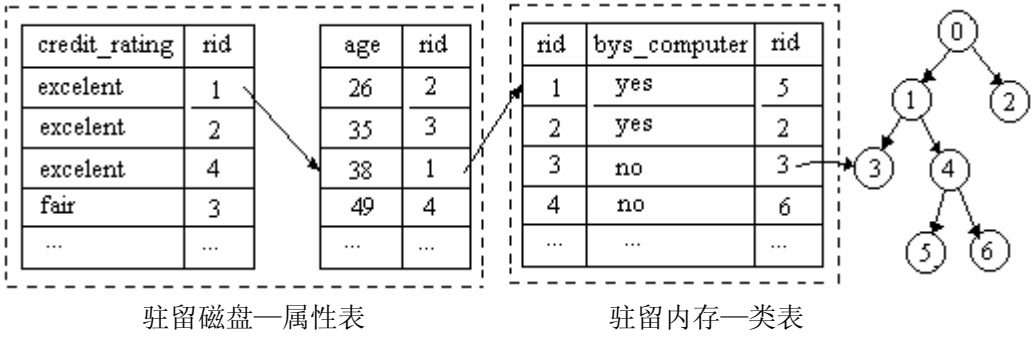
“判定树归纳的可规模性如何？”已有的判定树算法，如 ID3 和 C4.5，对于相对小的数据集是很有效的。当这些算法用于非常大的、现实世界数据库的挖掘时，有效性和可规模性就成了关注的问题。大部分判定树算法都限制训练样本驻留主存。在数据挖掘应用中，包含数以百万计样本的非常大的训练集是很普通的。因此，这一限制就制约了这些算法的可规模性。由于训练样本在主存和高速缓存换进换出，判定树的构造可能变得效率低下。

由大型数据库构造判定树的早期策略包括对连续属性离散化，在每个结点对数据选样。然而，这些仍然假定训练集可以放在主存。一种替代的方法是：首先，将样本划分成子集，使得每个子集可以放在内存；然后，由每个子集构造一棵判定树；最后，输出的分类法将由每个子集得到的分类法组合在一起。尽管该方法可以用于大数据集的分类，其分类的准确性不如一次使用所有的数据的方法高。

最近，已经提出了一些判定树算法，它们强调可规模性。由非常大的训练集进行判定树归纳的算法包括 SLIQ 和 SPRINT；它们都能处理分类属性和连续值属性。这两种算法都使用了预排序技术，对非常大，而不能放入内存的驻留磁盘的数据集进行预排序。两种算法都定义使用新的数据结构，以利于树的构造。SLIQ 使用若干驻留磁盘的属性表和单个驻留主存的类表。对于表 7.2 的样本数据，SLIQ 产生的属性表和类表如图 7.5 所示。每一个属性具有一个属性表，在 *RID*（记录标识）建立索引。每个元组由一个从每个属性表的一个表目到类表的一个表目（存放给定元组的类标号）的链接表示。而类表表目链接到它在判定树中对应的叶子结点。类表驻留在主存，因为判定树的构造和剪枝时，经常访问它。类表的大小随训练集中元组数目成比例增长。当类表不能放在主存时，SLIQ 的性能下降。

表 7.2：类 *buys_computer* 的样本数据

RID	*credit_rating*	*age*	*buys_computer*
1	Excellent	38	yes
2	Excellent	26	yes
3	Fair	35	no
4	Excellent	49	no



驻留磁盘—属性表　　驻留内存—类表

图 7.5：对于表 7.2 样本数据，SLIQ 使用的属性表和类表

SPRINT 使用不同的属性表数据结构，存放类和 *RID* 信息，如图 7.6 所示。当结点分裂时，属性表被相应划分，并在结果子女中分布。当表划分时，表中记录的次序维持不变。因此，划分表不需要重新排序。SPRINT 的设计易于并行，这就进一步增强了可规模性。

credit_rating	buys_computer	id
Excellent	Yes	
Excellent	Yes	
Excellent	No	
fair	No	
...	...	..

ge	buys_computer	id
6	yes	
5	no	
8	yes	
9	no	
..	...	.

图 7.6　对于表 7.2 的样本数据，　SPRINT 使用的属性表数据结构

当 SLIQ 和 SPRINT 处理的驻留磁盘的数据太大，不能一次装入内存时，SLIQ 的可规模性受限于它所使用的常驻内存的数据结构。SPRINT 消除了所有的内存限制，但仍然需要使用正比例于训练集的散列树。随着训练集的增长，这可能变得代价昂贵。

雨林（RainForest）是用于可规模化的判定树归纳的框架。该方法适合有大量可用的内存，并用于任意判定树归纳算法。它使用一个 AVC-集（属性-值类标号），指示每个属性类分布。据称，雨林的速度超过 SPRINT。

7.3.6 集成数据仓库技术和判定树归纳

判定树归纳可以与数据仓库技术集成，用于数据挖掘。本小节，我们讨论多维数据方方法和面向属性的归纳如何与判定树归纳集成，以利于交互的多层挖掘。一般地，这里介绍的技术也可以用于其它形式的学习。

数据方方法可以与判定树归纳集成，提供交互的判定树的多层挖掘。数据方和存放在概念分层中的知识可以用于在不同的抽象层归纳判定树。此外，一旦导出判定树，概念分层可以用来泛化或特化树的结点，可以在属性上进行上卷或下钻，并对新的特定抽象层的数据重新分类。这种交互特点使得用户可以将他们的注意力集中在他们感兴趣的树区域或数据。

面向属性的归纳（AOI）使用概念分层，通过以高层概念替换低层概念泛化训练数据（第 5 章）。当我们将 AOI 与判定树归纳集成时，泛化到很低的（特定的）概念层可能导致非常大而茂盛的树。对非常高的概念层的泛化可能导致判定树没什么用；这里，由于过度泛化，一些有趣、重要的子概念丢失了。应当泛化到由领域专家设定，或由用户指定的阈值控制的某个中间概念层。这样，AOI 的使用可能产生更易理解的、较小的分类树，从而得到的树比直接在低层、非泛化的数据集上操作的方法（如 SLIQ 或 SPRINT）产生的树更易于解释。

对判定树的典型批评是，由于递归地划分，一些数据子集可能变得太小，使得进一步划分它们就失去统计意义。这种“无意义”的数据子集的最大尺寸可以统计地确定。为处理这一问题，可以引进一个**例外阈值**。如果给定子集中的样本数少于该阈值，该子集的进一步划分停止。替换地，创建一个叶结点，存放该子集和该子集样本的类分布。

由于大型数据库中的数据量大、发散，假定每个树叶包含属于一个公共类的样本可能是不合理的。这一问题可以通过使用**准确率**或**分类阈值**解决。如果属于给定结点的任意类的样本百分比超过该阈值，在给定结点上的进一步划分将终止。

数据挖掘查询语言可以容易地用于说明增强的判定树归纳方法。假定数据挖掘任务是根据顾客的收入和职业，预测 30 多岁的顾客的信用风险，可以用如下数据挖掘查询来说明：

```
mine classfication
analyze credit_risk
in relevance to income,occupation
from Customer_db
where (age>=30)  and  (age<40)
display as rules
```

上面用 DMQL 表达的查询在 *Customer_db* 上执行关系查询，提取任务相关的数据。不满足 **where** 子句条件的元组将被忽略，并且仅收集 **in relevance to** 子句中说明的属性和类标号属性（*credit_risk*）。然后，AOI 在这些数据上操作。由于该查询并未说明所用的概念分层，因此使用省缺的概念分层。可以设计一个图形用户界面，使得用户通过这种数据挖掘查询语言说明数据挖掘任务更加容易。借助于这种办法，用户可以指导自动的数据挖掘过程。

7.4 贝叶斯分类

“什么是贝叶斯分类？”贝叶斯分类是统计学分类方法。它们可以预测类成员关系的可能性，如给定样本属于一个特定类的概率。

贝叶斯分类基于贝叶斯定理，在下面介绍。分类算法的比较研究发现，一种称作朴素贝叶斯分类的简单贝叶斯分类算法可以与判定树和神经网络分类算法相媲美。用于大型数据库，贝叶斯分类也已表现出高准确率与高速度。

朴素贝叶斯分类假定一个属性值对给定类的影响独立于其它属性的值。该假定称作类条件独立。做此假定是为了简化所需计算，并在此意义下称为“朴素的”。 贝叶斯信念网络是图形模型。不象贝叶斯朴素分类，它能表示属性子集间的依赖。贝叶斯信念网络也可以用于分类。

7.4.1 小节回顾基本的概率概念和贝叶斯定理。然后，你将在 7.4.2 小节学习朴素贝叶斯分类。贝叶斯信念网络在 7.4.3 小节介绍。

7.4.1 贝叶斯定理

设 X 是类标号未知的数据样本。设 H 为某种假定，如，数据样本 X 属于某特定的类 C。对于分类问题，我们希望确定 $P(H|X)$——给定观测数据样本 X，假定 H 成立的概率。

$P(H|X)$ 是**后验概率**，或条件 X 下，H 的后验概率。例如，假定数据样本世界由水果组成，用它们的颜色和形状描述。假定 X 表示红色和圆的，H 表示假定 X 是苹果，则 $P(H|X)$ 反映当我们看到 X 是红色并是圆的时，我们对 X 是苹果的确信程度。*P(H)*是**先验概率**，或 H 的先验概率。对于我们的例子，它是任意给定的数据样本为苹果的概率，而不管数据样本看上去如何。后验概率 $P(H|X)$ 比先验概率 *P(H)*基于更多的信息（如，背景知识）。*P(H)*是独立于 X 的。

类似地，$P(X|H)$ 是条件 H 下，X 的后验概率。即，它是已知 X 是苹果，X 是红色并且是圆的的概率。*P(X)*是 X 的先验概率。使用我们的例子，它是由我们的水果集取出一个数据样本是红的和圆的的概率。

“如何计算这些概率？”正如我们下面将看到的，*P(X)*，*P(H)*和 $P(X|H)$ 可以由给定的数据计算。**贝叶斯定理**是有用的，它提供了一种由 *P(X)*，*P(H)*和 $P(X|H)$ 计算后验概率 $P(H|X)$ 的方法。贝叶斯定理是：

$$P(H|X)=\frac{P(X|H)P(H)}{P(X)} \tag{7.5}$$

下一小节，你将学习如何在朴素贝叶斯分类中使用贝叶斯定理。

7.4.2 朴素贝叶斯分类

朴素贝叶斯分类，或简单贝叶斯分类的工作过程如下：

1. 每个数据样本用一个 n 维特征向量 $X=\{x_1,x_2,...,x_n\}$ 表示，描述由属性 $A_1,A_2,...,A_n$ 对样本的 n 个度量。

2. 假定有 m 个类 $C_1,C_2,...,C_m$。给定一个未知的数据样本 X（即，没有类标号），分类法将预测 X 属于具有最高后验概率（条件 X 下）的类。即，朴素贝叶斯分类将未知的样本分配给类 C_i，当且仅当：

$$P(C_i \mid X) > P(C_j \mid X) \quad 1 \le j \le m \quad j \ne i.$$

这样，我们最大化 $P(C_i \mid X)$。其 $P(C_i \mid X)$ 最大的类 C_i 称为最大后验假定。根据贝叶斯定理((7.5)式)，

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)} \tag{7.6}$$

3. 由于 $P(X)$ 对于所有类为常数，只需要 $P(X \mid C_i)P(C_i)$ 最大即可。如果类的先验概率未知，则通常假定这些类是等概率的；即，$P(C_1)=P(C_2)=...=P(C_m)$。并据此对只 $P(C_i \mid X)$ 最大化。否则，我们最大化 $P(X \mid C_i)P(C_i)$。注意，类的先验概率可以用 $P(C_i)=s_i/s$ 计算；其中，s_i 是类 C 中的训练样本数，而 s 是训练样本总数。

4. 给定具有许多属性的数据集，计算 $P(X \mid C_i)$ 的开销可能非常大。为降低计算 $P(X \mid C_i)$ 的开销，可以做**类条件独立**的朴素假定。给定样本的类标号，假定属性值条件地相互独立。即，在属性间，不存在依赖关系。这样，

$$P(X \mid C_i) = \prod_{k=1}^{n} p(x_k \mid C_i) \tag{7.7}$$

概率 $P(x_1 \mid C_i)$, $P(x_2 \mid C_i)$, . . . , $P(x_n \mid C_i)$ 可以由训练样本估值，其中，

(a) 如果 A_k 是分类属性，则 $P(x_k \mid C_i)=s_{ik}/s_i$；其中 s_{ik} 是在属性 A_k 上具有值 x_k 的类 C_i 的训练样本数，而 s_i 是 C_i 中的训练样本数。

(b) 如果是连续值属性，则通常假定该属性服从高斯分布。因而，

$$P(x_k \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x-\mu_{C_i})^2}{2\sigma_{C_i}^2}} \tag{7.8}$$

其中，给定类 C_i 的训练样本属性 A_k 的值，$g(x_k,\mu_{C_i},\sigma_{C_i})$ 是属性 A_k 的**高斯密度函数**，而 μ_{C_i},σ_{C_i} 分别为平均值和标准差。

5. 为对未知样本 X 分类，对每个类 C_i，计算 $P(X \mid C_i)P(C_i)$。样本 X 被指派到类 C_i，当且仅当：

$$P(X \mid C_i)P(C_i) > P(X \mid C_j)P(C_j) \quad 1 \le j \le m \quad j \ne i.$$

换言之，X 被指派到其 $P(X \mid C_i)P(C_i)$ 最大的类 C_i。

“贝叶斯分类的效率如何？”理论上讲，与其它所有分类算法相比，贝叶斯分类具有最小的出错率。然而，实践中并非总是如此。这是由于对其应用的假定（如，类条件独立性）的不正确性，

以及缺乏可用的概率数据造成的。然而，种种实验研究表明，与判定树和神经网络分类算法相比，在某些领域，该分类算法可以与之媲美。

贝叶斯分类还可以用来为不直接使用贝叶斯定理的其它分类算法提供理论判定。例如，在某种假定下，可以证明正如朴素贝叶斯分类一样，许多神经网络和曲线拟合算法输出最大的后验假定。

例 7.4　使用朴素贝叶斯分类预测类标号：给定与例 7.2 判定树归纳相同的训练数据，我们希望使用朴素贝叶斯分类预测一个未知样本的类标号。训练数据在表 7.1 中。数据样本用属性 *age, income, student* 和 *credit_rating* 描述。类标号属性 *buys_computer* 具有两个不同值（即，{yes, no}）。设 C_1对应于类 *buys_computer* = "*yes*"，而 C_2对应于类 *buys_computer* = "*no*"。我们希望分类的未知样本为：

$$X = (age = "<= 30", income = "medium", student = "yes", credit\_rating = "fair").$$

我们需要最大化 $P(X|C_i)P(C_i)$，i = 1,2。每个类的先验概率 $P(C_i)$ 可以根据训练样本计算：

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

为计算 $P(X|C_i)$，　i = 1,2。我们计算下面的条件概率:

$P(age = "<30" \mid buys\_computer = "yes")$ = 2/9 = 0.222

$P(age = "<30" \mid buys\_computer = "no")$ = 3/5 = 0.600

$P(income = "medium" \mid buys\_computer = "yes")$ = 4/9 = 0.444

$P(income = "medium" \mid buys\_computer = "no")$ = 2/5 = 0.400

$P(student = "yes" \mid buys\_computer = "yes")$ = 6/9 = 0.667

$P(student = "yes" \mid buys\_computer = "no")$ = 1/5 = 0.200

$P(credit\_rating = "fair" \mid buys\_computer = "yes")$ = 6/9 = 0.667

$P(credit\_rating = "fair" \mid buys\_computer = "no")$ = 2/5 = 0.400

使用以上概率，我们得到：

$$P(X \mid buys\_computer = "yes") = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$
$$P(X \mid buys\_computer = "no") = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019$$
$$P(X \mid buys\_computer = "yes")\, P(buys\_computer = "yes") = 0.044 \times 0.643 = 0.028$$
$$P(X \mid buys\_computer = "no")\, P(buys\_computer = "no") = 0.019 \times 0.357 = 0.007$$

因此，对于样本 X，朴素贝叶斯分类预测 *buys_computer* =" *yes*"。□

7.4.3　贝叶斯信念网络

朴素贝叶斯分类假定类条件独立。即，给定样本的类标号，属性的值可以条件地相互独立。这一假定简化了计算。当假定成立时，与其它所有分类算法相比，朴素贝叶斯分类是最精确的。然而，在实践中，变量之间的依赖可能存在。**贝叶斯信念网络**说明联合概率分布。它允许在变量的子集间定义类条件独立性。它提供一种因果关系的图形，可以在其上进行学习。这种网络也被称作**信念网络**、**贝叶斯网络**和**概率网络**。为简洁计，我们称它为信念网络。

信念网络由两部分定义。第一部分是有向无环图，其每个结点代表一个随机变量，而每条弧代表一个概率依赖。如果一条弧由结点 Y 到 Z，则 Y 是 Z 的**双亲**或**直接前驱**，而 Z 是 Y 的**后继**。给定其双亲，每个变量条件独立于图中的非后继。变量可以是离散的或连续值的。它们可以对应于数据中给定的实际属性，或对应于一个相信形成联系的“隐藏变量”（如，医疗数据中的综合病症）。

图 7.7(a)给出了一个 6 个布尔变量的简单信念网络，取自[RN95]。弧表示因果知识。例如，得肺癌受其家族肺癌史的影响，也受其是否吸烟的影响。此外，该弧还表明：给定其双亲 *FamilyHistory* 和 *Smoker*，变量 *LungCancer* 条件地独立于 *Emphysema*。这意味，一旦 *FamilyHistory* 和 *Smoker* 的值已知，变量 *Emphysema* 并不提供关于 *LungCancer* 的附加信息。

定义信念网络的第二部分是每个属性一个条件概率表（CPT）。变量 *Z* 的 CPT 说明条件分布 *P(Z | Parents(Z))*；其中，*Parents(Z)*是 *Z* 的双亲。图 7.7(b)给出了 *LungCancer* 的 CPT。对于其双亲值的每个可能组合，表中给出了 *LungCancer* 的每个值的条件概率。例如，由左上角和右下角，我们分别看到

$$P(LungCancer = \text{“}yes\text{”} \mid FamilyHistory = \text{“}yes\text{”}, Smoker = \text{“}yes\text{”}) = 0.8$$
$$P(LungCancer = \text{“}no\text{”} \mid FamilyHistory = \text{“}no\text{”}, Smoker = \text{“}no\text{”}) = 0.9$$

对应于属性或变量 $Z_1,..,Z_n$ 的任意元组 $(z_1,..,z_n)$ 的联合概率由下式计算：

$$P(z_1,...,z_n) = \prod_{i=1}^{n} P(z_i \mid parents(Z_i))$$

其中，$P(z_i \mid parents(Z_i))$ 的值对应于 Z_i 的 *CPT* 中的表目。

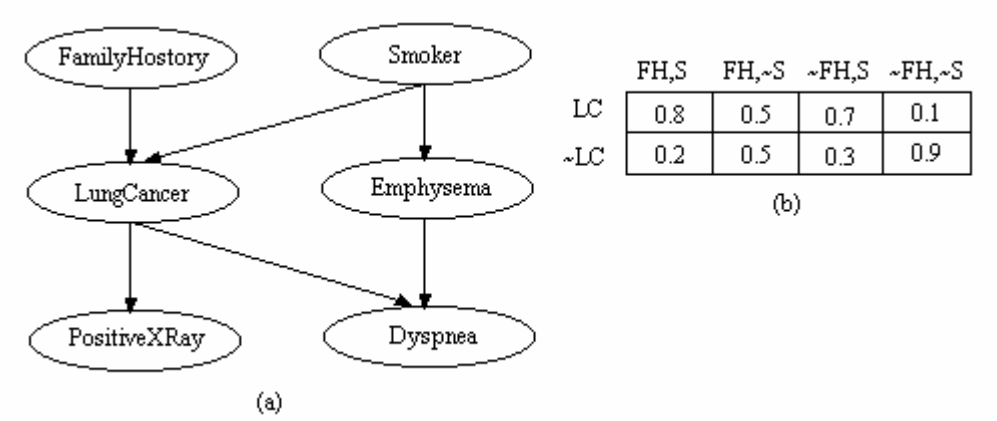


图 7.7　(a) 一个简单的贝叶斯信念网络　(b) 变量 *LungCance（LC）*值的条件概率表，给出其双亲结点 *FamilyHistory* 和 *Smoke* 的每个可能值的组合的条件概率

网络结点可以选作“输出”结点，对应于类标号属性。可以有多个输出结点。学习推理算法可以用于网络。分类过程不是返回单个类标号，而是返回类标号属性的概率分布；即，预测每个类的概率，

7.4.4　训练贝叶斯信念网络

“贝叶斯信念网络如何学习？”在学习或训练信念网络时，许多情况都是可能的。网络结构可能预先给定，或由数据导出。网络变量可能是可见的，或隐藏在所有或某些训练样本中。隐藏数据的情况也称为遗漏值或不完全数据。

如果网络结构已知并且变量是可见的，训练网络是直接了当的。该过程由计算 CPT 项组成，与朴素贝叶斯分类涉及的计算概率类似。

当网络结构给定，而某些变量是隐藏的时，则可使用梯度下降方法训练信念网络。目标是学习 CPT 项的值。设 *S* 是 *s* 个训练样本 $X_1, X_2, \ldots, X_s$ 的集合，$w_{i,jk}$ 是具有双亲 $U_i = u_{ik}$ 的变量 $Y = y_{ij}$ 的 CPT 项。例如，如果 $w_{i,jk}$ 是图 7.7(b)左上角的 CPT 项，则 Y_i 是 *LungCancer*；y_{ij} 是其值“*yes*”；U_i 列出 Y_i 的双亲结点{*FamilyHistory, Smoker*}；而 u_{ik} 列出双亲结点的值{“*yes*”, “*yes*”}。$w_{i,jk}$ 可以看作权，类似于神经网络（7.5 节）中隐藏单元的权。权的集合记作 *w*。这些权被初始化为随机概率值。梯度下降策略采用贪心爬山法。在每次迭代中，修改这些权，并最终收敛到一个局部最优解。

基于 w 的每个可能设置都等可能的假定，该方法搜索能最好地对数据建模 $w_{i,jk}$ 值。目标是最大化 $p_w(S)=\prod_{d=1}^{s}P_w(X_d)$。这通过按 $ln\ P_w(S)$ 梯度来做，使得问题更简单。给定网络结构和 $w_{i,jk}$ 的初值，该算法按以下步骤处理：

1. **计算梯度**：对每个 i，j，k，计算

$$\frac{\partial \ln P_w(S)}{\partial w_{ijk}}=\sum_{d=1}^{s}\frac{P(Y_i=y_{ij},U_i=u_{ik}\mid X_d)}{w_{ijk}} \tag{7.10}$$

(7.10)式右端的概率要对 S 中的每个样本 X_d 计算。为简洁计，我们简单地称此概率为 p。当 Y_i 和 U_i 表示的变量对某个 X_d 是隐藏的时，则对应的概率 p 可以使用贝叶斯网络推理的标准算法（如，商用数值软件包 Hugin 提供的那些(*http://www.hugin.dk*)），由样本的观察变量计算。

2. **沿梯度方向前进一小步**：用下式更新权值

$$w_{ijk}\leftarrow w_{ijk}+(l)\frac{\partial \ln P_w(S)}{\partial w_{ijk}} \tag{7.11}$$

其中，l 是表示步长的**学习率**，而 $\frac{\partial \ln P_w(S)}{\partial w_{ijk}}$ 由(7.10)式计算。学习率被设置为一个小常数。

3. **重新规格化权值**：由于权值 $w_{i,jk}$ 是概率值，它们必须在 0.0 和 1.0 之间，并且对于所有的 i,k，$\sum_j w_{ijk}$ 必须等于 1。在权值被(7.11)式更新后，可以对它们重新规格化来保证这一条件。

有一些算法，由给定可观察变量的训练数据学习网络结构。该问题是离散优化问题。请参阅本章的文献注释。

7.5 后向传播分类

“什么是后向传播？”后向传播是一种神经网络学习算法。神经网络最早是由心理学家和神经学家提出的，旨在寻求开发和测试神经的计算模拟。粗略地说，**神经网络**是一组连接的输入/输出单元，其中每个连接都与一个权相相联。在学习阶段，通过调整神经网络的权，使得能够预测输入样本的正确类标号来学习。由于单元之间的连接，神经网络学习又称**连接者学习**。

神经网络需要很长的训练时间，因而对于有足够长训练时间的应用更合适。它需要大量的参数，这些通常主要靠经验确定，如网络拓扑或“结构”。由于人们很难解释蕴涵在学习权之中的符号含义，神经网络常常因其可解释性差而受到批评。这些特点使得神经网络在数据挖掘的初期并不看好。

然而，神经网络的优点包括其对噪音数据的高承受能力，以及它对未经训练的数据的分类能力。此外，最近已提出了一些由训练过的神经网络提取规则的算法。这些因素推动了神经网络在数据挖掘分类方面的应用。

最流行的神经网络算法是 80 年代提出的后向传播算法。在 7.5.1 小节，你将学习多层前馈网络，后向传播算法在这种类型的网络上运行。7.5.2 小节讨论定义网络拓扑。后向传播算法在 7.5.3 小节介绍。由训练的神经网络提取规则在 7.5.4 小节讨论。

7.5.1 多路前馈神经网络

后向传播算法在**多路前馈**神经网络上学习。这种神经网络的一个例子如图 7.8 所示。输入对应于对每个训练样本度量的属性。输入同时提供给称作**输入层**的单元层。这些单元的加权输出依次同时地提供给称作**隐藏层**的“类神经元的”第二层；该隐藏层的加权输出可以输入到另一个隐藏层；如此下去。隐藏层的数量是任意的，尽管实践中通常只用一层。最后一个隐藏层的加权输出作为构成**输出层**的单元的输入。输出层发布给定样本的网络预测。

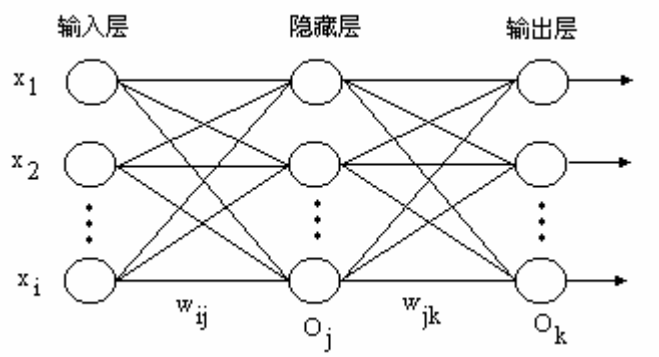


图 7.8　一个多层前馈神经网络。训练样本 $X = \{x_1, x_2, \ldots, x_i\}$ 馈入输入层。每层之间存在加权连接；其中，w_{ij} 表示由某层的单元 j 到前一层的单元 i 的权

隐藏层和输出层的单元，有时称作 neurodes（源于符号生物学），或**输出单元**。图 7.8 所示的多层神经网络具有两层输出单元。因此，我们称之为**两层**神经网络。类似地，包含两个隐藏层的网络称作三层神经网络，如此等等。网络是**前馈的**，如果其权都不回送到输入单元，或前一层的输出单元。网络是**全连接的**，如果每个单元都向下一层的每个单元提供输入。

给定足够多的隐藏单元，线性阈值函数的多层前馈神经网络可以逼近任何函数。

7.5.2　定义网络拓扑

“如何设计神经网络拓扑？”在开始训练之前，用户必须说明输入层的单元数、隐藏层数（如果多于一层）、每一隐藏层的单元数和输出层的单元数，以确定网络拓扑。

对训练样本中每个属性的值进行规格化将有助于加快学习过程。通常，对输入值规格化，使得它们落入 0.0 和 1.0 之间。离散值属性可以重新编码，使得每个域值一个输入单元。例如，如果属性 A 的定义域为(a_0, a_1, a_2)，则可以分配三个输入单元表示 A。即，我们可以用 I_0, I_1, I_2作为输入单元。每个单元初始化为 0。如果 $A = a_0$，则 I_0置为 1；如果 $A = a_1$，I_1置 1；如此下去。一个输出单元可以用来表示两个类（值 1 代表一个类，而值 0 代表另一个）。如果多于两个类，则每个类使用一个输出单元。

对于“最好的”隐藏层单元数，没有明确的规则。网络设计是一个实验过程，并可能影响结果训练网络的准确性。权的初值也可能影响结果的准确性。一旦网络经过训练，并且其准确率不能被接受，则通常用不同的网络拓扑或使用不同的初始权值，重复训练过程。

7.5.3　后向传播

“后向传播如何工作？”后向传播通过迭代地处理一组训练样本，将每个样本的网络预测与实际知道的类标号比较，进行学习。对于每个训练样本，修改权，使得网络预测和实际类之间的均方误差最小。这种修改“后向”进行。即，由输出层，经由每个隐藏层，到第一个隐藏层（因此称作后向传播）。尽管不能保证，一般地，权将最终收敛，学习过程停止。算法在图 7.9 中给出。每一步的解释如下：

初始化权：网络的权被初始化为很小的随机数（例如，由-1.0 到 1.0，或由-0.5 到 0.5）。每个单元有一个偏置，下面解释。偏置也类似地初始化为小随机数。

每个样本 X按以下步骤处理。

向前传播输入：在这一步，计算隐藏层和输出层每个单元的净输入和输出。首先，训练样本提供给网络的输入层。注意，对于输入层的单元 j，它的输出等于它的输入；即，对于单元 j，$O_j = I_j$。然后，隐藏层和输出层的每个单元的净输入用其输入的线性组合计算。为帮助解释这一点，图 7.10 给出了一个隐藏层或输出层单元。事实上，单元的输入是连接它的前一层的单元的输出。为计算它

的净输入，连接该单元的每个输入乘以其对应的权，然后求和。给定隐藏层或输出层的单元 j，到单元 j 的净输入 I_j 是：

$$I_j = \sum_i w_{ij} O_i + \theta_j \tag{7.12}$$

其中，w_{ij} 是由上一层的单元 i 到单元 j 的连接的权；O_i 是上一层的单元 i 的输出；而 θ_j 是单元 j 的偏置。偏置充当阈值，用来改变单元的活性。

隐藏层和输出层的每个单元取其净输入，然后将赋活函数作用于它，如图 7.10 所示。该函数用符号表现单元代表的神经元活性。使用 logistic 或 simoid 函数。给定单元 j 的净输入 I_j，则单元 j 的输出 O_j 用下式计算：

$$O_j = \frac{1}{1+e^{-I_j}} \tag{7.13}$$

该函数又称挤压函数，因为它将一个较大的输入值域映射到较小的区间 0 到 1。logistic 函数是非线性的和可微的，使得后向传播算法可以对线性不可分的问题建模。

算法：后向传播。使用后向传播算法的神经网络分类学习。
输入：训练样本 *samples*，学习率 *l*，多层前馈网络 *network*。
输出：一个训练的、对样本分类的神经网络。
方法：

1) 初始化 *network* 的权和偏置。
2) **while** 终止条件不满足 {
3) **for** *samples* 中的每个训练样本 X {
4) // 向前传播输入
5) **for** 隐藏或输出层每个单元 j {
6) $I_j = \sum_i w_{ij} O_i + \theta_j$；// 相对于前一层 i，计算单元 j 的净输入
7) $O_j = 1/(1+e^{-I_j})$；} // 计算单元 j 的输出
8) // 后向传播误差
9) **for** 输出层每个单元 j
10) $Err_j = O_j(1-O_j)(T_j - O_j)$；// 计算误差
11) **for** 由最后一个到第一个隐藏层，对于隐藏层每个单元 j
12) $Err_j = O_j(1-O_j)\sum_k Err_k w_{kj}$；// 计算关于下一个较高层 k 的误差
13) **for** *networ* 中每个权 w_{ij} {
14) $\Delta w_{ij} = (l) Err_j O_i$； // 权增值
15) $w_{ij} = w_{ij} + \Delta w_{ij}$； } // 权更新
16) **for** *networ* 中每个偏差 θ_j {
17) $\Delta \theta_j = (l) Err_j$； // 偏差增值
18) $\theta_j = \theta_j + \Delta \theta_j$； } // 偏差更新
19) }}

图 7.9 后向传播算法

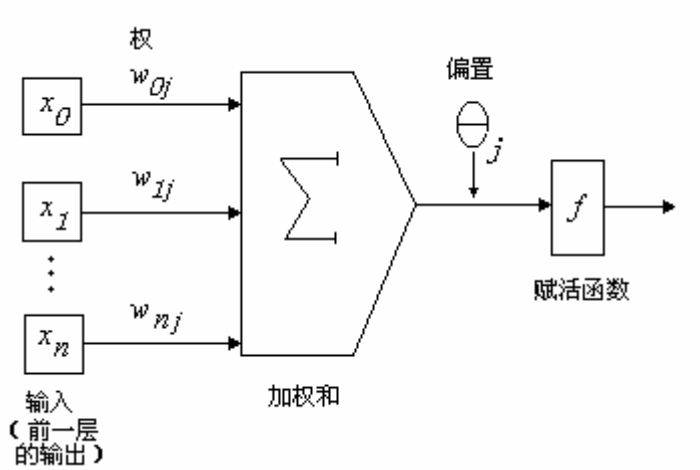


图 7.10　一个隐藏或输出单元 j：j 的输入是来自前一层的输出。这些与对应的权相乘，以形成加权和。加权和加到与单元 j 相联的偏置上。一个非线性的赋活函数用于净输入

后向传播误差：通过更新权和反映网络预测误差的偏置，向后传播误差。对于输出层单元 j，误差 Err_j 用下式计算

$$Err_j = O_j(1-O_j)(T_j - O_j) \tag{7.14}$$

其中，O_j 是单元 j 的实际输出，而 T_j 是 j 基于给定训练样本的已知类标号的真正输出。注意，O_j (1- O_j) 是 logistic 函数的导数。

为计算隐藏层单元 j 的误差，考虑下一层中连接 j 的单元的误差加权和。隐藏层单元 j 的误差是

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{kj} \tag{7.15}$$

其中，w_{kj} 是由下一较高层中单元 k 到单元 j 的连接权，而 Err_k 是单元 k 的误差。

更新权和偏差，以反映传播的误差。权由下式更新，其中，Δw_{ij} 是权 w_{ij} 的改变。

$$\Delta w_{ij} = (l) Err_j O_i \tag{7.16}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{7.17}$$

“（7.16）式中的‘l’是什么？”变量 l 是**学习率**，通常取 0 和 1 之间的值。后向传播使用梯度下降法搜索权值的集合。这些权值可以对给定的分类问题建模，使得样本的网络类预测和实际的类标号距离平方的平均值最小。学习率帮助避免陷入判定空间的局部最小（即，权值看上去收敛，但不是最优解），并有助于找到全局最小。如果学习率太小，学习将进行得很慢。如果学习率太大，可能出现在不适当的解之间摆动。一个调整规则是将学习率设置为 $1/t$。其中，t 是已对训练样本集迭代的次数。

偏置由下式更新。其中，$\Delta\theta_j$ 是偏置 θ_j 的改变。

$$\Delta\theta_j = (l) Err_j \tag{7.18}$$

$$\theta_j = \theta_j + \Delta\theta_j \tag{7.19}$$

注意，这里我们每处理一个样本就更新权和偏置，这称作**实例更新**。替换地，权和偏置的增量可以累积到变量中，使得可以在处理完训练集中的所有样本之后再更新权和偏置。后一种策略称作

周期更新，扫描训练集的一次迭代是一个**周期**。理论上，后向传播的数学推导使用周期更新，而实践中实例更新更常见，因为它通常产生更准确的结果。

终止条件。训练停止，如果

- 前一周期所有的Δw_{ij}都太小，小于某个指定的阈值，或
- 前一周期未正确分类的样本百分比小于某个阈值，或
- 超过预先指定的周期数。

实践中，权收敛可能需要数十万个周期。

例 7.5　通过后向传播算法学习的样本计算。图 7.11 给出了一个多层前馈神经网络。设学习率为 0.9。该网络的初始权值和偏置值以及第一个训练样本 $X = \{1,0,1\}$（其类标号为 1）在表 7.3 中给出。

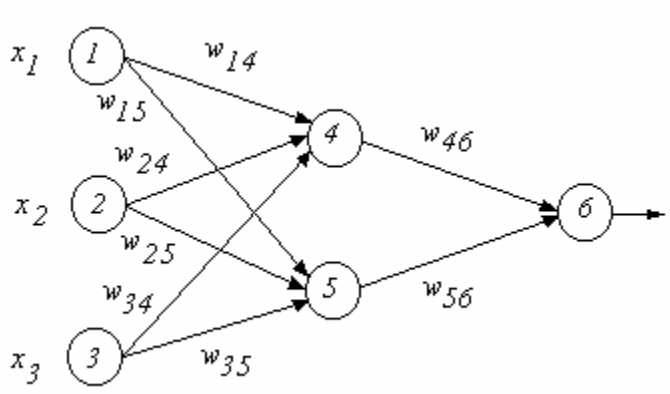


图 7.11　多层前馈神经网络的一个例子

表 7.3：初始输入、权值和偏差值

1	2	3	14	15	24	25	34	35	46	56	4	5	6
			.2	-0.3	.4	.1	-0.5	.2	-0.3	-0.2	-0.4	.2	.1

给定第一个训练样本 X，该例展示后向传播计算。首先将样本提供给网络，计算每个单元的净输入和输出。这些值在表 7.4 中。计算每个单元的误差，并后向传播。误差值在表 7.5 中，权和偏置的更新在表 7.6 中。□

表 7.4：净输入和输出的计算表

单元 j	净输入 Ij	输出 Oj
4	0.2+0-0.5-0.4 = -0.7	$1+(1+e^{0.7})$ = 0.33
5	-0.3+0+0.2+0.2 = 0.1	$1+(1+e^{-0.1})$ = 0.525
6	(-0.3)(0.332)-(0.2)(0.525)+0.1 = -0.105	$1+(1+e^{-0.105})$ = 0.474

7.5：计算每个结点的误差表

单元 j	Err_j

6	(0.474)(1-0.474)(1-0.474) =
5	0.1311
4	(0.525)(1-0.525)(0.1311)(-0.2) =
	-0.0065
	(0.332)(1-0.332)(0.1311)(-0.3) =
	-0.02087

7.6：计算权和偏置的更新

权或偏差	新值
w_{46}	-0.3 + (0.9)(0.1311)(0.332) =
w_{56}	-0.261
w_{14}	-0.2 + (0.9)(0.1311)(0.525) =
w_{15}	-0.138
w_{24}	0.2 + (0.9)(-0.0087)(1) = 0.192
w_{25}	-0.3 + (0.9)(0.0065)(1) =
w_{34}	-0.306
w_{35}	0.4 + (0.9)(-0.0087)(0) = 0.4
θ_6	0.1 + (0.9)(-0.0065)(0) = 0.1
θ_5	-0.5 + (0.9)(-0.0087)(1) =
θ_4	-0.508
	0.2 + (0.9)(-0.0065)(1) = 0.194
	0.1+(0.9)(0.1311) = 0.218
	0.2+(0.9)(-0.0065) = 0.194
	-0.4+(0.9)(-0.0087) = -0.408

业已提出了一些后向传播算法的变形和替代，用于神经网络分类。这些可能涉及网络拓朴和学习率或其它参数的动态调整，或使用不同的误差函数。

7.5.4 后向传播和可解释性

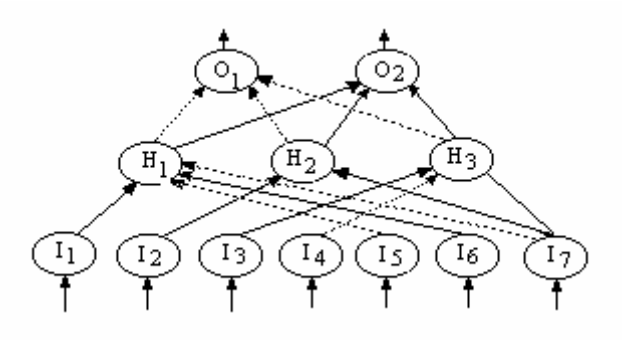
“如何‘理解’后向传播神经网络的学习结果？”神经网络的主要缺点是其知识的表示。用加权链连接单元的网络表示的知识很难被人理解。这激发了提取隐藏在训练的神经网络中的知识，并象征地解释这些知识的研究。方法包括由网络提取规则和灵敏度分析。

业已提出了各种规则提取算法。通常，这些方法对训练给定神经网络所用的过程、网络的拓朴结构和输入值的离散化加以限制。

全连接的网络很难处理。然而，由神经网络提取规则的第一步通常是**网络剪枝**。该步通过剪去对训练网络影响最小的加权链简化网络结构。例如，如果删除一个加权链不导致网络的分类精确度下降，则应当删除该加权链。

一旦训练网络已被剪枝，一些方法将进行链、单元或活跃值的聚类。例如，在一种方法中，使用聚类发现给定训练的两层神经网络中每个隐藏单元的共同活跃值的集合（图 7.12）。分析每个隐藏单元的这些活跃值。导出涉及这些活跃值与对应输出单元值组合的规则。类似地，研究输入值和活跃值的集合，导出描述输入和隐藏单元层联系的规则。最后，两个规则的集合可以结合在一起，形成 IF-THEN 规则。其它算法可能导出其它形式的规则，包括 M-of-N 规则（其中，为应用规则的后件，规则前件中给定的 N 个条件中的 M 个条件必须为真），具有 M-of-N 测试的判定树、模糊规则和有穷自动机。

灵敏度分析用于评估一个给定的输入变量对网络输出的影响。改变该变量的输入，而其它输入变量为某固定值。其间，监测网络输出的改变。由这种形式的分析得到的知识是形如“IF X 减少 5% THEN Y 增加 8%”的规则。



识别每个隐藏结点 H_i 的共同活跃值集合：

H_1: (-1, 0, 1)
H_2: (0, 1)
H_3: (-1, 0.24, 1)

导出与输出结点 O_j 的共同活跃值相关的规则：

IF ($H_2 = 0$ AND $H_3 = -1$) OR
($H_1 = -1$ AND $H_2 = 1$ AND $H_3 = -1$) OR
($H_1 = -1$ AND $H_2 = 0$ AND $H_3 = 0.24$)
THEN $O_1 = 1$, $O_2 = 0$
ELSE $O_1 = 0$, $O_2 = 1$

导出与输入 I_i 到输出结点 O_j 相关的规则：

IF ($I_2 = 0$ AND $I_7 = 0$) THEN $H_2 = 0$
IF ($I_4 = 1$ AND $I_6 = 1$) THEN $H_3 = -1$
IF ($I_5 = 0$) THEN $H_3 = -1$
...

得到关于输入和输出类的规则：

IF ($I_2 = 0$ AND $I_7 = 0$ AND $I_4 = 1$ AND $I_6 = 1$)
THEN class=1
IF ($I_2 = 0$ AND $I_7 = 0$ AND $I_5 = 0$) THEN class = 1

图 7.12　规则可以由训练神经网络提取

7.6　基于源于关联规则挖掘概念的分类

“源于关联规则挖掘的思想可以用于分类吗？”关联规则挖掘是数据挖掘研究的一个重要的、高度活跃的领域。本书的第 6 章介绍了许多关联规则挖掘算法。最近，数据挖掘技术业已将关联规则挖掘用于分类问题。本节，我们按历史次序研究三种方法。前两种方法，ARCS 和关联分类使用关联规则分类。第三种方法 CAEP 挖掘“显露模式”，它考虑挖掘关联规则使用的支持度概念。

第一种方法基于聚类挖掘关联规则，然后使用规则进行分类。ARCS 或关联规则聚类系统（6.4.3 小节）挖掘形如 $A_{quant1} \wedge A_{quant2} \Rightarrow A_{cat}$ 的关联规则；其中，A_{quant1}，A_{quant2} 是在量化属性区间上的测试（区间动态地确定），而 A_{cat} 为给定训练数据的分类属性指定一个类标号。关联规则画在 2-D 栅格上。算法扫描栅格，搜索规则的矩形聚类。用这种办法，出现在一个规则聚类内的量化属性的相邻区间可以结合。由 ARCS 产生的聚类关联规则用于分类，其准确率可与 C4.5 媲美。一般地，当数据中存在局外者时，实验发现 ARCS 比 C4.5 稍微精确一点。ARCS 的准确性与离散化程度有关。从可规模性

来说，不论数据库多大，ARCS 需要的存储容量为常数。相比之下，C4.5 具有指数运行时间，要求整个数据库（乘以某个因子）全部装入内存。

第二种方法称作**关联分类**。它挖掘形如 $condset \Rightarrow y$ 的规则；其中，*condset* 是项（或属性-值对）的集合，而 y 是类标号。满足最小支持度的规则是**频繁的**；这里，规则具有**支持度** s，如果给定数据集中的样本 $s\%$包含 *condset* 并且属于类 y。满足最小置信度的规则是**精确的**；这里，规则的**置信度**为 c，如果给定数据集中包含 *condset* 的样本 $c\%$属于类 y。如果一个规则项集具有相同的 *condset*，则选择具有最高置信度的规则作为**可能规则**（PR），代表该集合。

关联分类方法由两步组成。第一步是找出所有频繁的、精确的 PR 集合。这些是类关联规则(*CAR*)。其 *condset* 包含 k 个项的规则项称作 ***k*-规则项**。算法使用迭代方法，类似于 6.2.1 小节介绍的 Apriori 使用的方法，先验知识用于裁减规则搜索。第二步使用一种启发式方法构造分类。这里，发现的规则根据支持度和置信度按递减的优先次序组织。算法可能需要多次扫描数据集，这依赖于找到的最长规则的长度。对一个新的样本进行分类时，满足该样本的第一个规则用于对它分类。分类法也包含省缺规则，它具有最低的优先次序，用来为不被分类法中其它规则满足的新样本指定一个省缺的类。一般地，经验表明，上述关联分类方法在许多数据集上比 C4.5 更精确。以上两步都具有线性可规模性。

第三种方法 **CAEP**（通过聚集显露模式分类）使用项集支持度挖掘**显露模式**（**EP**），而 EP 用于构造分类。粗略地说，**EP** 是一个项集（项的集合），其支持度由一个类到另一个类显著增加。两个支持度的比称作 EP 的**增长率**。例如，假定我们有顾客数据集，包含类 *buys_computer* = "*yes*"或 C_1和 *buys_computer* = "*no*"或 C_2。项集{*age* ="*<=30*", *students* ="*no*"}是一个典型的 EP，其支持度由在 C_1中的 0.2%增长到在 C_2中的 57.6%，增长率 $\frac{57.6\%}{0.2\%} = 288$。注意，一个项或者是分类属性上的简单相等测试，或者是检查数值属性是否在某个区间的测试。每个 EP 是一个多属性上的测试，并且可能在区分一个类的实例与另一个类的实例方面非常强。例如，如果一个新样本 X包含在上面的 EP 中，我们可以说 X属于 C_2的几率为 99.6%。一般地，EP 的区分能力大约正比于它的增长率和它在目标类的支持度。

"CAEP 如何使用 EP 建立分类法？"对于每个类 C，CAEP 找出满足给定支持度和增长率阈值的 EP；这里，增长率按所有的非 C类样本的集合对所有的 C类样本目标集合来计算。"基于边界"的算法可以用于计算。在对一个新样本 X分类时，对于每个类 C，对出现在 X中的类 C的 EP 的区分能力聚集，得到 C的得分，然后对得分规格化。具有最大规格化得分的类决定 X的类标号。

业已发现，在许多数据集上，CAEP 比 C4.5 和基于关联的分类更精确。它在主要感兴趣的类占少数的数据集上也运行良好。它在数据量和维数上都是可规模化的。一种替代的分类法称作 JEP 分类法，基于**跳跃显露模式**（JEP）提出。**JEP** 是一种特殊类型的 EP，定义为这样的项集，其支持度由在一个数据集中的 0 陡峭地增长到另一个数据集中的非 0。这两种分类法被认为是互补的。

7.7 其它分类方法

本节，我们给出一些其它分类方法的简略介绍。这些方法包括 k-最临近分类、基于案例的推理、遗传算法、粗糙集和模糊集方法。一般地说，与本章前面介绍的方法相比，这些方法在商品化的数据挖掘系统中较少用于分类。例如，最临近分类存储所有样本，当由非常大的数据集学习时，这可能带来困难。此外，基于案例的推理、遗传算法和粗糙集分类还在原型阶段。然而，这些方法日趋流行，因此我们把它们包含在这里。

7.7.1 *k*-最临近分类

最临近分类基于类比学习。训练样本用 n 维数值属性描述。每个样本代表 n 维空间的一个点。这样，所有的训练样本都存放在 n 维模式空间中。给定一个未知样本，***k*-最临近分类法**搜索模式空间，找出最接近未知样本的 k 个训练样本。这 k 个训练样本是未知样本的 k 个"近邻"。"临近性"

用欧几里德距离定义。其中，两个点 $X = (x_1, x_2, \ldots, x_n)$和 $Y = (y_1, y_2, \ldots, y_n)$的欧几里德距离是：

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{7.20}$$

未知样本被分配到 k 个最临近者中最公共的类。当 $k = 1$ 时，未知样本被指定到模式空间中与之最临近的训练样本的类。

最临近分类是**基于要求的**或**懒散的学习法**；即，它存放所有的训练样本，并且直到新的（未标记的）样本需要分类是才建立分类。这与诸如判定树归纳和后向传播这样的**急切学习法**形成鲜明对比，后者在接受待分类的新样本之前构造一个一般模型。当与给定的无标号样本比较的可能的临近者（即，存放的训练样本）数量很大时，懒散学习法可能招致很高的计算开销。这样，它们需要有效的索引技术。正如所预料的，懒散学习法在训练时比急切学习法快，但在分类时慢，因为所有的计算都推迟到那时。与判定树归纳和后向传播不同，最临近分类对每个属性指定相同的权。当数据中存在许多不相关属性时，这可能导致混淆。

最临近分类也可以用于预测。即，返回给定的未知样本实数值预测。在此情况下，分类返回未知样本的 k 个最临近者实数值标号的平均值。

7.7.2 基于案例的推理

基于案例的推理（CBR）分类法是基于要求的。不象最临近分类法将训练样本作为欧氏空间的点存放，CBR 存放的样本或“案例”是复杂的符号描述。CBR 的商务应用包括诸如顾客服务台问题求解；那里，案例描述产品有关的诊断问题。CBR 还被用在诸如工程和法律领域；那里，案例分别是技术设计和合法规则。

当给定一个待分类的新案例时，基于案例的推理首先检查是否存在一个同样的训练案例。如果找到一个，则返回附在该案例上的解。如果找不到同样的案例，则基于案例的推理将搜索具有类似于新案例成分的训练案例。概念上讲，这些训练案例可以视为新案例的邻接者。如果案例用图描绘，这涉及搜索类似于新案例的子图。基于案例的推理试图组合临近的训练案例，提出新案例的解。如果解之间出现不相容，可能需要退回搜索其它解。基于案例的推理可能使用背景知识和问题求解策略，以便提出可行的组合解。

基于案例的推理存在的挑战包括找到一个好的相似矩阵（例如，为匹配子图），开发对训练案例索引的有效技术和组合解的方法。

7.7.3 遗传算法

遗传算法试图结合自然进化的思想。一般地，遗传学习开始如下：创建一个由随机产生的规则组成的初始**群体**。每个规则可以用一个二进位串表示。作为一个简单的例子，假定给定的训练集用两个布尔属性 A_1和 A_2描述，并且有两个类 C_1和 C_2。规则“IF A_1 AND NOT A_2 THEN C_2”可以用二进位串“100”编码；其中，最左边的两个二进位分别代表属性 A_1和 A_2，而最右边的二进位代表类。类似地，规则“IF NOT A_1 AND NOT A_2 THEN C_1”可以用“001”编码。如果一个属性具有 k（$k > 2$）个值，则可以用 k 个二进位对该属性的值编码。类可以用类似的形式编码。

根据适者生存的原则，形成由当前群体中最适合的规则组成新的群体，以及这些规则的子女。典型地，规则的**适合度**用它对训练样本集的分类准确率评估。

子女通过使用诸如交叉和变异等遗传操作来创建。在**交叉**操作中，来自规则对的子串交换，形成新的规则对。在**变异**操作中，规则串中随机选择的位被反转。

由先前的规则群体产生新的规则群体的过程继续，直到群体 P“进化”，P 中的每个规则满足预先指定的适合度阈值。

遗传算法易于并行，并且业已用于分类和其它优化问题。在数据挖掘，它们可能用于评估其它算法的适合度。

7.7.4 粗糙集方法

粗糙集理论可以用于分类，发现不准确数据或噪音数据内在的结构联系。它用于离散值属性。因此，连续值属性必须在处理前离散化。

粗糙集理论基于给定训练数据内部的**等价类**的建立。形成等价类的所有数据样本是不加区分的。即，对于描述数据的属性，这些样本是等价的。给定现实世界数据，通常有些类不能被可用的属性区分。粗糙集可以用来近似或“粗略地”定义这种类。给定类 C 的粗糙集定义用两个集合近似：C 的**下近似**和 C 的**上近似**。C 的下近似由一些这样的数据样本组成，根据关于属性的知识，它们毫无疑问属于 C。C 的上近似由所有这样的样本组成，根据关于属性的知识，它们不可能被认为不属于 C。类 C 的下近似和上近似如图 7.13 所示。其中，每个矩形区域代表一个等价类。判定规则可以对每个类产生。通常，使用判定表表示这些规则。

粗糙集也可以用于特征归约（那里，可以识别和删除无助于给定训练数据分类的属性）和相关分析（那里，根据分类任务评估每个属性的贡献或意义）。找出可以描述给定数据集中所有概念的最小属性子集（归约）问题是 NP-难处理的。然而，业已提出了一些降低计算强度的算法。例如，有一种方法使用**识别矩阵**存放每对数据样本属性值之间的差别。不是在整个训练集上搜索，而是搜索矩阵，检测冗余属性。

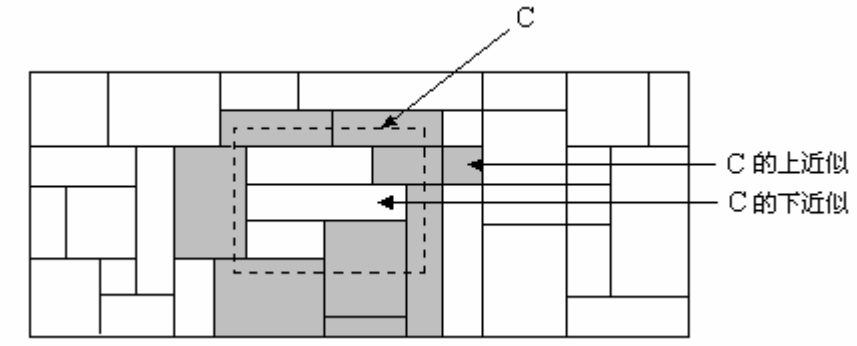


图 7.13　类 C 的样本集的使用 C 的上、下近似集的粗糙集近似。矩形区域表示等价类

7.7.5 模糊集方法

基于规则的分类系统有一个缺点：对于连续属性，它们有陡峭的截断。例如，考虑下面关于顾客信用申请批准的规则。该规则本质上是说：工作两年或多年，并且具有较高收入（即，多于 50K）的顾客申请将被批准。

$$IF \quad (year\_employed >= 2) \wedge (income >= 50K) \ THEN \ credit = \text{“}approved\text{”}. \tag{7.21}$$

根据规则(7.21)，一个至少工作两年的顾客将得到信用卡，如果他的收入是\$50K；但是，如果他的收入是\$49K，他将得不到。这种苛刻的阈值看来可能不公平。替换地，可以将模糊逻辑引入系统，允许定义“模糊”阈值或边界。模糊逻辑使用 0.0 和 1.0 之间的真值表示一个特定的值是一个给定类成员的程度，而不是用类或集合的精确截断。因而，使用模糊逻辑，我们可以断言：在某种程度上，\$49K 的收入是高的，尽管没有\$50K 的收入高。

对于数据挖掘系统进行分类，模糊逻辑是有用的。它提供了在高抽象层处理的便利。一般地，模糊逻辑在基于规则的系统中的使用涉及：

- 将属性值转换成模糊值。图 7.14 展示如何将连续属性 *income* 的值映射到离散分类{*low*, *medium*, *high*}上，以及如何计算模糊成员关系或真值。通常，模糊逻辑系统在这一步提供图形工具，支持用户。
- 对于给定的新样本，可以使用多个模糊规则。每个可用规则为分类的成员关系贡献一票。通常，对每个预测分类的真值进行求和。

- 组合上面得到的和，得到一个系统返回的值。这一过程可以这样做：用每个分类的真值和加权，并乘以每个分类的平均真值。所涉及的计算可能更复杂，这取决于模糊成员关系图的复杂性。

  模糊逻辑系统已用于许多分类领域，包括健康和财经。

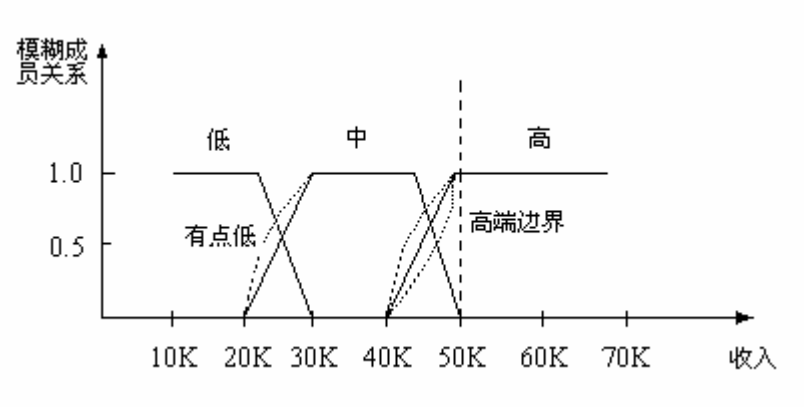


图 7.14　　收入的模糊值

7.8　预测

“如果我们想预测一个连续的值，而不是一个分类标号，怎么办？”连续值的预测可以用回归统计技术建模。例如，我们可能希望开发一个模型，预测具有 10 年工作经验大学毕业生的工资，或一种给定价格的新产品的可能销售量。这类问题可以用回归分析统计技术建模。许多问题可以用线性回归解决，并且更多的可以对变量进行变换，使得非线性问题可以转换为线性的来加以处理。受篇幅限制，我们不能给出回归处理的全部细节。本节直观地介绍该问题。通过本节学习，你将熟悉线性回归、多元回归和非线性回归的思想，以及广义线性模型。

有一些软件包解决回归问题。例如 SAS(*http://www.sas.com*)、SPSS(*http://www.spss.com*)和 S-Plus(*http://www.mathsoft.com*)。

7.8.1　线性和多元回归

“什么是线性回归？”在**线性回归**中，数据用直线建模。线性回归是最简单的回归形式。双变量回归将一个随机变量 Y（称作**响应变量**）视为另一个随机变量 X（称为**预测变量**）的线性函数。即：

$$Y = \alpha + \beta X \tag{7.22}$$

其中，Y的方差为常数；α和β是**回归系数**，分别表示直线在 Y 轴的截断和直线的斜率。这些系数可以用**最小平方法**求解，这使得实际数据与该直线的估计之间误差最小。给定 s 个样本或形如(x_1, y_1)，(x_2, y_2),...，(x_s, y_s)的数据点，回归系数α和β可以用下式计算：

$$\beta = \frac{\sum_{i=1}^{s}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{s}(x_i - \bar{x})^2} \tag{7.23}$$

$$\alpha = \bar{y} - \beta\bar{x} \tag{7.24}$$

其中，$\bar{x}$ 是 x_1，x_2，...，x_s 的平均值，而 $\bar{y}$ 是 y_1，y_2，...，y_s 的平均值。与其它复杂的回归方法相比，线性回归常常给出很好的近似。

例 7.6　使用最小平方法的线性回归。表 7.7 给出了一组年薪数据。其中，X 表示大学毕业后工作的年数，而 Y 是对应的收入。这些数据点如图 7.15 所示，暗示我们 X 和 Y 之间存在线性关系。我们用方程 $Y = \alpha + \beta X$ 表示年薪和工作年数之间的关系。

给定以上数据，计算出 $\bar{x}$ = 9.1，$\bar{y}$ = 55.4。将这些值代入(7.23)和(7.24)式，得到

$$\beta = \frac{(3-9.1)(30-55.4)+(8-9.1)(57-55.4)+...+(16-9.1)(83-55.4)}{(3-9.1)^2+(8-9.1)^2+...+(16-9.1)^2} = 3.5$$

$$\alpha = 55.4-(3.7)(9.1) = 23.6$$

这样，我们得到方程 Y = 23.6+ 3.5X。使用该方程，我们可以预测有 10 年工作经验的大学毕业生的年薪为$58.6K。□

表 7.7：年薪数据

X 工 作 年数	Y 年薪（单位：$1K）
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

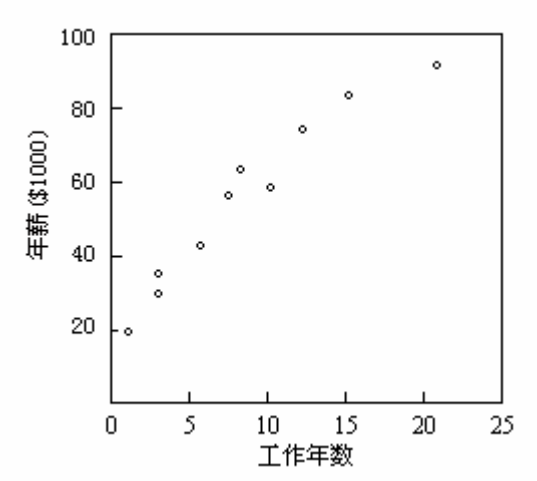


图 7.15　例 7.6 的表 7.7 中数据的图示。尽管这些点不在一条直线上，但总体模式表现出 X（工作年数）和 Y（年薪）间的线性关系

多元回归是线性回归的扩展，涉及多个预测变量。响应变量 Y 可以是一个多维特征向量的线性函数。基于两个预测属性或变量 X_1 和 X_2 的多元回归模型的例子是

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 \tag{7.25}$$

最小平方法可以用在这里求解 α，β_1 和 β_2。

7.8.2 非线性回归

“如何对不呈现线性依赖的数据建模？例如，如果给定的响应变量和预测变量间的关系可以用多项式函数表示，会怎么样？”通过在基本线性模型上添加多项式项，**多项式回归**可以用于建模。通过对变量进行变换，我们可以将非线性模型转换成线性的，然后用最小平方方法求解。

例 7.7 多项式回归模型转换为线性回归模型。考虑下式给出的三次多项式关系

$$Y = \alpha + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \tag{7.26}$$

为将该方程转换成线性的，我们定义如下新变量：

$$X_1 = X \qquad X_2 = X^2 \qquad X_3 = X^3 \tag{7.27}$$

使用上面的定义，方程(7.26)可以转换成线性形式，结果为$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$。它可以用最小平方法求解。□

在习题 7.9 中，要求你找出将涉及幂函数的非线性模型转换成线性回归模型所需的变换。

有些模型是难处理的（如，指数项和的形式）并且不能转换成线性模型。对于这些情况，可能通过对更复杂的公式进行计算，得到最小平方估计。

7.8.3 其它回归模型

线性回归用于对连续值函数进行建模。它被广泛使用，主要是由于它的简洁性。“它也能用来预测分类标号吗？”**广义线性模型**提供了将线性回归用于分类响应变量的理论基础。与线性回归不同，在广义线性模型中，响应变量 Y 的方差是 Y 的平均值的函数。而在线性回归中，Y 的方差为常数。广义线性模型的常见形式包括**对数回归**和**泊松回归**。对数回归将某些事件发生的概率看作预测变量集的线性函数。计数数据常常呈现泊松分布，并通常使用泊松回归建模。

对数线性模型近似离散的多维概率分布。可以使用它们估计与数据方单元相关的概率值。例如，假定给定属性 *city*,*item*,*year* 和 *sales* 的值。在对数线性方法中，所有的属性必须是分类的，因此连续值属性(如 *sales*)必须首先离散化。然后，使用该方法，根据 *city* 和 *item*，*city* 和 *year*，*city* 和 *sales* 的 2-D 方体，*item*,*year* 和 *sales* 的 3-D 方体估计给定属性的 4-D 基本方体中每个单元的概率。在这种方法中，一种迭代技术可以用来由低阶的数据方建立高阶的数据方。这种技术具有很好的可规模性，允许许多维。除预测之外，对数线性模型对于数据压缩（由于较低阶的方体的全部也比基本方体占用的空间少）和数据平滑（由于较低阶方体的单元估计比较高阶方体面临较少的选样变化）也是有用的。

7.9 分类的准确性

估计分类法的准确率是重要的，这使得我们可以估计一个给定的分类法对未来的数据（即，未经分类法处理的数据）正确标号的准确率。例如，如果先前的数据用于训练分类法，以便预测顾客的购物行为，我们希望评估该分类法预测未来顾客购物行为的准确率。准确率估计也可以用来比较分类法。在 7.9.1 小节，我们讨论评估分类法准确率的技术，包括保持（holdout）和 *k*-折交叉确认(k-fold cross-validation)方法。7.9.2 介绍两种提高分类法准确率的策略：装袋(bagging)和推进(boosting)。7.9.3 小节讨论关于分类法准确率和选择的其它问题。

7.9.1 评估分类法的准确率

由于学习算法（或模型）对数据的过分特化，使用训练数据得到分类法，然后评估分类法可能错误地导致过于乐观的估计。保持和 *k*-折交叉确认是两种基于给定数据随机选样划分的、常用的评估分类法准确率的技术。

在**保持**方法中，给定数据随机地划分成两个独立的集合：训练集和测试集。通常，三分之二的数据分配到训练集，其余三分之一分配到测试集。使用训练集导出分类法，其准确率用测试集评估（图 7.16）。评估是保守的，因为只有一部分初始数据用于导出的分类法。**随机子选样**是保持方法的一种变形，它将保持方法重复 *k* 次。总体准确率估计取每次迭代准确率的平均值。

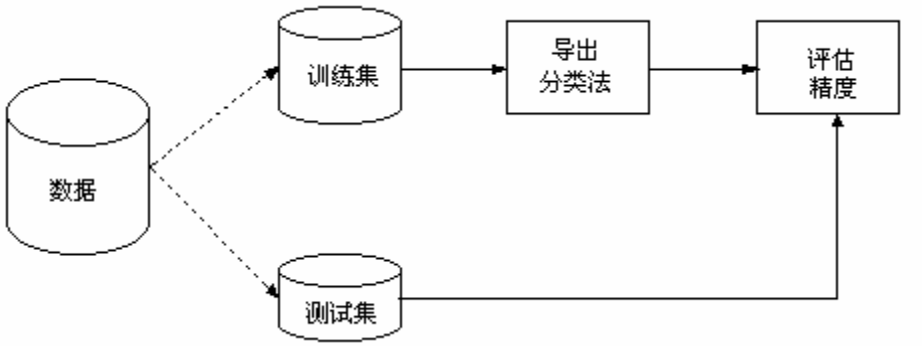


图 7.17：用保持方法评估分类法的准确率

在 ***k*-折交叉确认**中，初试数据被划分成 *k* 个互不相交的子集或“折” $S_1, S_2, \ldots, S_k$，每个折的大小大致相等。训练和测试进行 *k* 次。在第 *i* 次迭代，S_i 用作测试集，其余的子集都用于训练分类法。即，第一次迭代的分类法在子集 $S_2, \ldots, S_k$ 上训练，而在 S_1 上测试；第二次迭代的分类法在子集 S_1，$S_3, \ldots, S_k$ 上训练，而在 S_2 上测试；如此下去。准确率估计是 *k* 次迭代正确分类数除以初始数据中的样本总数。在**分层交叉确认**中，折被分层，使得每个折中样本的类分布与在初始数据中的大致相同。

评估分类法准确率的其它方法包括**解靴带**（bootstrapping）和**留一**。前者使用一致的、带放回的选样，选取给定的训练实例；后者是 *k*-折交叉确认，这里 *k* 为初始样本数 s。一般地，建议使用调整的 10-折交叉确认，因为它具有相对低的偏置和方差。

使用这些技术评估分类法的准确率增加了总体运行时间，但对于由多个分类法中选择仍然是有用的。

7.9.2 提高分类法的准确率

在前一小节，我们研究了评估分类法准确率的方法。在 7.3.2 小节，我们看到剪枝如何用于判定树归纳，帮助提高结果判定树的准确率。存在改进分类法准确率的一般技术吗？

答案是肯定的。**装袋**（或解靴带聚集）和**推进**是两种这样的技术（图 7.17）。每个都将 *T* 个学习得到的分类法 $C_1, C_2, \ldots, C_T$ 组合起来，旨在创建一个改进的分类法 *C**。

“这些方法如何工作？“假定你是一个病人，希望根据你的症状进行诊断。你可能选择看多个医生，而不是一个。如果某种诊断比其它诊断出现的次数多，你可能将它作为最终或最好的诊断。现在，将医生换成分类法，你就可以直观地理解装袋。假定你根据医生以前诊断的准确率，对每个医生的诊断“值”或价值赋予一个权值，则最终的诊断是加权的诊断的组合。这就是推进的基本思想。让我们进一步考察这两种技术。

给定 *s* 个样本的集合 *S*，装袋过程如下。对于迭代 *t*（ *t* = 1,2,...,*T*），训练集 S_t 采用放回选样，由原始样本集 *S* 选取。由于使用放回选样，*S* 的某些样本可能不在 S_t 中，而其它的可能出现多次。由每个训练集 S_t 学习，得到一个分类法 C_t。为对一个未知的样本 *X* 分类，每个分类法 C_t 返回它的类预测，算作一票。装袋的分类法 *C**统计得票，并将得票最高的类赋予 *X*。通过取得票的平均值，而不是多数，装袋也可以用于连续值的预测。

在推进中，每个训练样本赋予一个权。学习得到一系列分类法。学习得到分类法 C_t 后，更新权，使得随后的分类法 C_{t+1} “更关注” C_t 的分类错误。最终的推进分类法 C^* 组合每个分类法的表决，这里每个分类法的表决是其准确率的函数。推进算法也可以扩充到连续值预测。

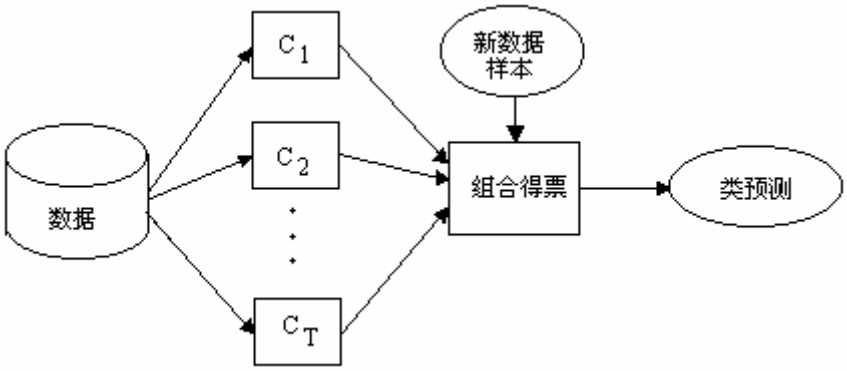


图 7.17　提高分类的准确率：分袋和推进都产生一系列分类法 C_1, C_2,...，C_T。使用选票策略组合给定未知样本的类预测

7.9.3　准确率确定分类法够吗？

除准确率外，分类法还可以根据其速度、鲁棒性（例如，在噪音数据上的准确性）、可规模性、可解释性进行比较。可规模性可以通过计算给定分类算法在渐增的数据集上的 I/O 操作次数评估。可解释性是主观的，尽管我们可以在评估它时使用诸如结果分类法的复杂性（例如，判定树的结点数，或神经网络的隐藏单元数）等客观度量。

“有无准确率度量的替代？”假定你已经训练了一个分类法，将医疗数据分类为"*cancer*"或"*non_cancer*"。90%的准确率使得该分类法看上去相当准确，但是如果实际只有 3-4%的训练样本是"*cancer*"，怎么样？显然，90%的准确率是不能接受的——该分类法只能正确地标记"*non_cancer*"样本。替换地，我们希望能够评估该分类法能够识别样本"*cancer*"（称作**正样本**）的情况和它识别样本"*non_cancer*"（称作**负样本**）的情况。为此，我们可以分别使用**灵敏性**和**特效性**度量。此外，我们可以使用**精度**评估标记为"*cancer*"，实际是"*cancer*"的样本的百分比。这些度量定义为

$$sensitivity = \frac{t\_pos}{pos} \tag{7.28}$$

$$specificity = \frac{t\_neg}{neg} \tag{7.29}$$

$$precision = \frac{t\_pos}{(t\_pos + f\_pos)} \tag{7.30}$$

其中，*t_pos* 是**真正样本**（被正确地按此分类的"*cancer*"样本）数，*pos* 是正（"*cancer*"）样本数，*t_neg* 是**真负样本**（被正确地按此分类的"*non_cancer*"样本）数，*neg* 是负（"*non_cancer*"）样本数，而 *f_pos* **假正样本**（被错误地标记为"*cancer*"的"*non_cancer*'样本）数。可以证明正确率是灵敏性和特效性度量的函数：

$$accuracy = sensitivity\frac{pos}{(pos+neg)} + specificity\frac{neg}{(pos+neg)} \tag{7.31}$$

“还有其它情况，准确性可能不合适吗？”在分类问题中，通常假定所有对象都是唯一可分类的；即，每个训练样本能够，并仅能够属于一个类。然而，由于大型数据库中的数据非常发散，假定所有的对象都唯一可分类并非总是合理的。假定每个对象属于多个类是可行的。这样，如何度量大型数据库上分类的准确率呢？准确率度量是不合适的，因为它没考虑样本属于多个类的可能性。

不返回类标号，而返回类分布概率是有用的。这样，准确率度量可以采用**二次猜测**：一个类预测是正确的，如果它与最可能的或次可能的类一致。尽管这在某种程度上确实考虑了对象的非唯一分类，但它不是完全解。

7.10 总结

- 分类和预测是数据分析的两种形式，可以用于提取描述重要数据类的模型或预测未来的数据趋势。**分类**预测分类标号（类），而**预测**建立连续值函数模型。
- 分类和预测准备阶段的预处理可能涉及**数据清理**（减少噪音，或处理丢失的值）、**相关性分析**（删除不相关或冗余属性）和**数据变换**（如，泛化数据到较高的概念层，或对数据规范化）。
- 预测的准确率、计算速度、鲁棒性、可规模性和可解释性是评估分类和预测方法的五条标准。
- ID3 和 C4.5 是判定树归纳的贪心算法。每种算法都使用一种信息论度量，为树中每个非树叶结点选择测试属性。**剪枝**算法试图通过剪去反映数据中噪音的分枝，提高准确率。通常，早期的判定树算法假定数据是驻留内存的——对大型数据库上的数据挖掘是一种限制。其后，提出了一些可规模化的算法，来解决这一问题，如 SLIQ、SPRINT 和雨林算法。判定树容易转换成 IF-THEN 分类规则。
- **朴素贝叶斯分类**和**贝叶斯信念网络**基于后验概率的贝叶斯定理。不象贝叶斯分类（其假定类条件独立），贝叶斯信念网络允许在变量子集之间定义类条件独立性。
- **后向传播**是一种用于分类的神经网络算法，使用梯度下降方法。它搜索一组权，这组权可以对数据建模，使得数据样本的网络类预测和实际类标号间的平均平方距离最小。可以由训练的神经网络提取规则，帮助改进学习网络的可理解性。
- **关联挖掘**技术在大型数据库中搜索频繁出现的模式，可以用于分类。
- 最近相邻分类法和基于案例的分类法是**基于要求的**分类方法，它们在模式空间存放所有的训练样本。因此，它们都需要有效的索引技术。在**遗传算法**中，规则群体通过交叉和变异操作“进化”，直到群体中所有的规则都满足指定的阈值。**粗糙集理论**可以用来近似地定义类，这些类根据可用的属性是不可区分的。**模糊集**方法用成员程度函数替换连续值属性的“脆弱的”陡峭阈值。
- 线性、非线性和广义线性**回归**模型都可以用于预测。许多非线性问题都可以通过预测变量上的变换，转换成线性问题。
- 数据仓库技术，如面向属性的归纳和多维数据方的使用，都可以与分类方法集成，以支持快速**多层挖掘**。分类任务可以使用数据挖掘查询语言说明，促进交互数据挖掘。
- **分层的 k-折交叉确认**是一种推荐的评估分类法准确率的方法。**装袋**和**推进**方法通过学习和组合一系列分类法，可用于提高分类的整体准确率。**灵敏性、特效性**和**精度**是对准确性度量的替换，特别是当感兴趣的主类为少数时。
- 已有许多关于不同分类方法的比较，并且该问题仍然是一个研究课题。尚未发现有一种方法对所有数据都优于其它方法。如准确性、训练时间、鲁棒性、可解释性和可规模性必须考虑，并且可能涉及折衷，使得寻求更好方法进一步复杂化。实验研究表明，许多算法的准确性非常类似，其差别是统计不明显的，而训练时间可能显著不同。一般地，大部分神经网络和涉及样条的统计分类与大部分判定树方法相比，趋向于计算量大。

习题

7.1 简述判定树分类的主要步骤。

7.2 在判定树归纳中，为什么树剪枝是有用的？

7.3 给定判定树，你有选择：（a）将判定树转换成规则，然后对结果规则剪枝，或（b）对判定树剪枝，然后将剪枝后的树转换成规则。相对于（b），（a）的优点是什么？

7.4　为什么朴素贝叶斯分类称为“朴素”的？简述朴素贝叶斯分类的主要思想。

7.5　比较急切分类（如，判定树、贝叶斯、神经网络）相对于懒散分类（如，*k* -最临近、基于案例的推理）的优缺点。

7.6　下表由雇员数据库的训练数据组成。数据已泛化。对于给定的行，*count* 表示 *department*, *status*, *age* 和 *salary* 在该行上具有给定值的元组数。

department	*status*	*age*	*salary*	*count*
sales	senior	31...35	46K...50K	3
sales	junior	26...30	26K...30K	4
systems	junior	31...35	31K...35K	4
systems	junior	21...25	46K...50K	2
systems	senior	31...35	66K...70K	5
systems	junior	26...30	46K...50K	3
marketing	senior	41...45	66K...70K	3
marketing	senior	36...40	46K...50K	1
secretary	junior	31...35	41K...45K	4
secretary	senior	46...50	36K...40K	4
	junior	26...30	26K...30K	6

设 *status* 是类标号属性。

(a) 你将如何修改 ID3 算法，以便考虑每个泛化数据元组（即，每一行）的 *count*？

(b) 使用你修改过的 ID3 算法，构造给定数据的判定树。

(c) 给定一个数据样本，它在属性 *department*, *salary* 和 *age* 上的值分别为“*systems*”，“46..50K”和“20...24”。该样本 *status* 的朴素贝叶斯分类是什么？

(d) 为给定的数据设计一个多层前馈神经网络。标记输入和输出层结点。

(e) 使用上面得到的多层前馈神经网络，给定训练实例(*sales*, *senior*, 31...35, 46K...50K)，给出后向传播算法一次迭代后的权值。指出你使用的初始权值和偏置以及学习率。

7.7　给定 *k* 和描述每个样本的属性数 *n*，写一个 *k*-最临近分类算法。

7.8　下表给出课程数据库中学生的期中和期末考试成绩。

X 期中考试	*Y* 期末考试
72	84
50	63
81	77
74	78
94	90

86	75
59	49
83	79
65	77
33	52
88	74
81	90

(a)对数据做图。X和 Y看上去具有线性联系吗？

(b)使用最小平方法，求由学生的期中成绩预测学生的期末成绩的方程式。

(c)预测期中成绩为 86 分的学生的期末成绩。

7.9 通过对预测变量的变换，有些非线性回归模型可以转换成线性的。指出如何将非线性回归方程 $Y = \alpha X^{\beta}$ 转换成可以用最小平方法求解的线性回归方程。

7.10 什么是推进？陈述它为何能够提高判定树归纳的准确性。

7.11 证明准确率是灵敏性和特效性度量的函数。即，证明（7.31）式。

7.12 当一个数据对象可以同时属于多个类时，很难评估分类的准确率。陈述在这种情况下，你将使用何种标准比较在相同数据上建立的不同分类法。

文献注释

机器学习观点的分类在许多书中都有介绍，如 Weiss 和 Kulikowski[WK91]，Michie, Spiegelhalter和Taylor[MST94],Langley[Lan96]和Mitchell[Mit91]。Weiss和Kulikowski [WK91]除介绍了分类法性能评估的实用技术外，还比较了许多不同领域的分类和预测。这些书的大部分都介绍了本章讨论的每个基本分类方法。包含机器学习方面的原始论文可以在Michalksi, Carbonell 和 Mitchell[MCM83,MCM86]，Kodratoff 和 Michalski[KM90]，Shavlik 和 Dietterich[SD90]，Michalski 和 Tecuci[MT94]中找到。关于数据挖掘应用的机器学习介绍，见 Michalski, Bratko 和 Kubat[MBK98]。

C4.5 算法由 J. R. Quinlan[Qui93]在书中介绍。该书给出了关于判定树归纳的许多问题的很好介绍，正如由 Murthy[Mur98]的关于判定树归纳的全面综述一样。判定树归纳的其它算法包括 C4.5 前驱 ID3(Quinlan[Qui86])、CART(Breiman, Friedman, Olshen 和 Stone[BFOS84])、FACT(Loh 和 Vanichsetakul [LV88])，QUEST(Loh 和 Shih[LS97])，PUBLIC(Rastogi 和 Shim[RS98])，CHAID（Kass[Kas80]，Magidson[Mag94]）。ID3 的增量版本包括 ID4（Schlimmer 和 Fisher[SF86a]）和 ID5(Utgoff[Utg88])。此外，INFERULE(Uthurusamy, Fayyad 和 Spangler[UFS91])由非决定的数据学习，构造判定树。KATE(Manago 和 Kodratoff[MK91])由复杂的结构化数据学习，构造判定树。强调在数据挖掘中可规模性的判定树算法包括 SLIQ(Mehta, Agrawal 和 Rissanen[MAR96])，SPRINT(Shafer, Agrawal 和 Mehta[SAM96])，雨林(Gehrke, Ramakrrishnan 和 Ganti[GRG98]), BOAT (Gehrke, Ganti, Ramakrishnan 和 Loh[GGRL99])以及 Kamber, Winstone< Gong 等[KWG+97]。早期方法的介绍包括[Cat91, CS93a, CS93b]。判定树归纳属性选择度量比较见 Buntine 和 Niblett[BN92],Murthy[Mur98] 和 Shih[Shi00]。这些度量的详细讨论见 Kononenko 和 Hong[KH97]。属性(或特征)构造在 Liu 和 Motoda[LM98a, LM98b]中介绍。具有属性构造的系统例子包括 Langley, Simon, Bradshaw 和 Zytko 的 BACON[LSBZ87]，Schimmer 的 Stagger[Sch87]，Pagallo 的 FRINGE[Pag89]，Bloedorn 和 Michalski 的 AQ17-DCI[BM98]。

有许多判定树剪枝算法,包括代价复杂性剪枝(Breiman, Friedman, Olshen 和 Stone[BFOS84])，减少错误剪枝(Quinlan[Qui87])和悲观估计剪枝(Quinlan[Qui86])。PUBLIC(rastogi 和 Shim[RS98])将判定树构造和剪枝集成在一起。基于 MDL 的剪枝方法可以在 Quinlan 和 Rivest[QR89]，Mehta, Agrawal 和 Sissanen[MAS95]，以及 Rastogi 和 Shim[RS98]中找到。其它方法包括 Niblett 和

Bratko[NB86]，Hosking, Pednault 和 Sadan[HPS97]。剪枝方法的实验比较见 Mingers[Min89], Malerba, Floriana 和 Semeraro[MFS95]。关于简化判定树的综述，见 Breslow 和 Aha[BA97]。

关于由判定树提取规则,见 Quinlan[Qui87, Qui93]。也可以直接由训练数据推导规则，而不是通过由判定树提取产生规则。规则推导算法包括 CN2(Clark 和 Niblett[CN89]),AQ15(Hong, Mozetic 和 Machalski[HMM86])，ITRULE(Smyth 和 Goodman[SG92])，FOIL(Quinlan[Qui90])和 Swap-1(Weiss 和 Indurkhya[WI98])。规则精炼策略由给定的规则集识别最有趣的规则，可以在 Major 和 Mangano[MM95]中找到。面向属性的归纳与判定树归纳的集成在 Kamber, Winstone, Gong 等[KWG+97]中提出。7.3.6 小节介绍的准确率或分类阈值在 Agrawal, Ghosh< Imielinski 等[AGI+92]和 Kamber 等[KWG+97]中。

贝叶斯分类的全面介绍可以在 Duda 和 Hart[DH73]，模式识别的典型教科书，以及诸如 Weiss 和 Kulikowski[WK91]和 Mitchell[Mit97]等机器学习教材中找到。关于类条件独立性不成立时，朴素贝叶斯分类的预测能力分析，见 Domingos 和 Pazzani[DP96]。对于朴素贝叶斯分类法，连续属性值内核稠密估计，而不是高斯估计的实验报告见 John[Joh97]。关于贝叶斯信念网络的介绍，见 Heckerman[Hec96]。在信念网络上推理的算法可以在 Russell 和 Norvig[RN95],以及 Jensen[Jen96]中找到。7.4.4 小节介绍的训练贝叶斯信念网络的梯度下降法在 Russell, Binder, Koller 和 Kanazawa[RBKK95]给出。图 7.8 给出的例子取自 Russell 等[RBKK95]。学习具有隐藏变量的信念网络的替换策略包括 EM 算法(Lauritzen[Lau95])。由给定可观察变量的训练数据学习信念网络的解在[CH92, Bun94, HGC95]中提出。

后向传播算法在 Rumelhart, Hinton 和 Williams[RHW86]中提出。自那以后，业已提出许多变形，包括替换的误差函数(Hanson 和 Burr[HB88]，网络拓朴的动态调整(Fahlman 和 Lebiere[FL90]，Le Cun, Denker 和 Solla[LDS90])，学习率和要素参数的动态调整(Jacobs[Jac88])。其它变形在 Chauvin 和 Rumelhart[CR95]中讨论。神经网络的书籍包括[RM86, HN90, HKP91, Fu94, CR95, Bis95, Rip95]。许多机器学习书籍，如[WK91, Mit97]，也包含后向传播算法的很好解释。有许多由神经网络提取规则的技术，如[SN88, Gal93, TS93, Fu94, Avn95, LSL95, CS96b, LGT97]。7.5.4 小节介绍的规则提取方法基于 Lu, Setiono 和 Liu[LSL95]。由神经网络提取规则技术的批评可以在 Craven 和 Shavlik[CS97]中找到。Roy[Roy00]提出神经网络的理论基础与关于连接者学习如何模型人脑的假定有裂缝。神经网络在工业、商务和科学方面的应用概览在 Widrow, Rumelhart 和 Lehr[WRL94]中。

7.6 节介绍的 ARCS 在 Lent, Swami 和 Widom[LSW97]中提出，也在第 6 章介绍。关联分类由 Liu, Hsu 和 Ma [LHM98]提出。CAEP 分类法使用显露模式，由 Dong 和 Li[DL99]提出。JEP 分类法使用跳跃显露模式，在 Li, Dong 和 Rmamohanarao[LDR00]中介绍。Meretaski 和 Wüthrich [MW99]提出通过挖掘长项集构造贝叶斯分类法。

最临近方法在许多分类统计教科书中讨论，如 Duda 和 hart[DH73]，James[Jam85]。附加的信息可以在 Cover 和 Hart[CH67]以及 Fukunaga 和 Hummels[FH87]中找到。基于案例的推理（CBR）文献包括教材[RS89, Kol93, Lea96]以及[AP94]。关于遗传算法的书籍见[Gol89, Mic92, Mit96]。粗糙集介绍在 Pawlak[Paw91]中。数据挖掘中粗糙集的详尽总结包括[Zia91, CPS98]。粗糙集业已用于许多应用的特征归约和专家系统，包括[Zia91, LP97, Swi98]。降低寻找归约的计算强度的算法已在[SR92]中提出。模糊逻辑的一般介绍可以在[Zad65, BS97, CPS98]中找到。

有许多好教材包含回归技术，如[Jam85, Dob90, JW92, Dev95, HC95,NKNW96, Agr96]。PressTeukolsky, Vetterling 和 Flannery[PTVF96]的书和附带的原代码包含许多统计过程，如线性和多元回归的最小平方法。新近的非线性回归模型包括投影追击(projection pursuit)和 MARS (Friedman[Fri91])。对数线性模型在计算机科学界也被称作乘法模型。由计算机科学角度讨论对数线性模型见 Pearl[Pea88]。回归树(Breiman, Friedman, Olshen 和 Stone[BFOS84])在性能上常常可以与其它回归方法媲美，特别是当预测变量间存在许多较高阶的依赖时。

数据清理和数据变换方法在 Kennedy, Lee, Van Roy 等[KLV+98]，Weiss 和 Indurkhya [WI98]，以及本书第 3 章讨论。涉及评估分类法准确率的问题在 Weiss 和 Kulikowski[WK91]中介绍。根据 Kohavi[Koh95]的理论和实验研究，与保持、交叉验证、留一(Stone[Sto74])和解靴带(Efron 和 Tibshirani[ET93])方法相比，评估分类法准确率的调整的 10-折交叉验证优先推荐。装带在 Briman[Bre96]中提出。Freund 和 Schapire[FS97]的推进技术业已用于许多不同的分类法，包括判定树归纳(Quinlan[Que96])和朴素贝叶斯分类(Elkan[Elk97])。灵敏性、指定性和精度在 Frakes 和 Baeza-Yates[FBY92]中讨论。

加州大学 Irvine 分校（UCI）维护了一个数据集的机器学习知识库，用于分类算法的开发和测试。关于它的信息，见 *http://www.ics.uci.edu/~mlearn/MLRepository.html*。

没有一种分类算法对所有的数据类型和定义域都优于其它分类算法。分类方法的实验比较包括[Qui88, SMT91, BCP93, CM94, MST94, BU95, LLS00]。

第八章　聚类分析

设想要求对一个数据对象的集合进行分析，但与分类不同的是，它要划分的类是未知的。聚类(clustering)就是将数据对象分组成为多个类或簇(cluster)，在同一个簇中的对象之间具有较高的相似度，而不同簇中的对象差别较大。相异度是基于描述对象的属性值来计算的。距离是经常采用的度量方式。聚类分析源于许多研究领域，包括数据挖掘，统计学，生物学，以及机器学习。

在本章中，大家将了解基于大数据量上进行操作而对聚类方法提出的要求，将学习如何计算由各种属性和不同的类型来表示的对象之间的相异度。还将学习几种聚类技术，它们可以分为如下几类：划分方法（partitioning method），层次方法(hierarchical method)，基于密度的方法(density-based method)，基于网格的方法(grid-based method)，和基于模型的方法(model-based method)。本章最后讨论如何利用聚类方法进行孤立点分析(outlier detection)。

8．1 什么是聚类分析？

将物理或抽象对象的集合分组成为由类似的对象组成的多个类的过程被称为聚类。由聚类所生成的簇是一组数据对象的集合，这些对象与同一个簇中的对象彼此相似，与其他簇中的对象相异。在许多应用中，一个簇中的数据对象可以被作为一个整体来对待。

聚类分析是一种重要的人类行为。早在孩提时代，一个人就通过不断地改进下意识中的聚类模式来学会如何区分猫和狗，或者动物和植物。聚类分析已经广泛地用在许多应用中，包括模式识别，数据分析，图像处理，以及市场研究。通过聚类，一个人能识别密集的和稀疏的区域，因而发现全局的分布模式，以及数据属性之间的有趣的相互关系。

“聚类的典型应用是什么？”在商业上，聚类能帮助市场分析人员从客户基本库中发现不同的客户群，并且用购买模式来刻画不同的客户群的特征。在生物学上，聚类能用于推导植物和动物的分类，对基因进行分类，获得对种群中固有结构的认识。聚类在地球观测数据库中相似地区的确定，汽车保险持有者的分组，及根据房子的类型，价值，和地理位置对一个城市中房屋的分组上也可以发挥作用。聚类也能用于对 Web 上的文档进行分类，以发现信息。作为一个数据挖掘的功能，聚类分析能作为一个独立的工具来获得数据分布的情况，观察每个簇的特点，集中对特定的某些簇作进一步的分析。此外，聚类分析可以作为其他算法（如分类等）的预处理步骤，这些算法再在生成的簇上进行处理。

数据聚类正在蓬勃发展，有贡献的研究领域包括数据挖掘，统计学，机器学习，空间数据库技术，生物学，以及市场营销。由于数据库中收集了大量的数据，聚类分析已经成为数据挖掘研究领域中一个非常活跃的研究课题。

作为统计学的一个分支，聚类分析已经被广泛地研究了许多年，主要集中在基于距离的聚类分析。基于 k-means(k-平均值)，k-medoids(k-中心)和其他一些方法的聚类分析工具已经被加入到许多统计分析软件包或系统中，例如 S-Plus，SPSS，以及 SAS。在机器学习领域，聚类是无指导学习(unsupervised learning)的一个例子。与分类不同，聚类和无指导学习不依赖预先定义的类和训练样本。由于这个原因，聚类是通过观察学习，而不是通过例子学习。在概念聚类（conceptual clustering）中，一组对象只有当它们可以被一个概念描述时才形成一个簇。这不同于基于几何距离来度量相似度的传统聚类。概念聚类由两个部分组成：（1）发现合适的簇；（2）形成对每个簇的描述。在这里，追求较高类内相似度和较低类间相似度的指导原则仍然适用。

在数据挖掘领域，研究工作已经集中在为大数据量数据库的有效且高效的聚类分析寻找适当的方法。活跃的研究主题集中在聚类方法的可伸缩性，方法对聚类复杂形状和类型的数据的有效性，高维聚类分析技术，以及针对大的数据库中混合数值和分类数据的聚类方法。

聚类是一个富有挑战性的研究领域，它的潜在应用提出了各自特殊的要求。数据挖掘对聚类的典型要求如下：

- 可伸缩性：许多聚类算法在小于 200 个数据对象的小数据集合上工作得很好；但是，一个大规模数据库可能包含几百万个对象，在这样的大数据集合样本上进行聚类可能会导致有偏的结果。我们需要具有高度可伸缩性的聚类算法。

- 处理不同类型属性的能力：许多算法被设计用来聚类数值类型的数据。但是，应用可能要求聚类其他类型的数据，如二元类型(binary)，分类/标称类型（categorical/nominal），序数型（ordinal）数据，或者这些数据类型的混合。
- 发现任意形状的聚类：许多聚类算法基于欧几里得或者曼哈顿距离度量来决定聚类。基于这样的距离度量的算法趋向于发现具有相近尺度和密度的球状簇。但是，一个簇可能是任意形状的。提出能发现任意形状簇的算法是很重要的。
- 用于决定输入参数的领域知识最小化：许多聚类算法在聚类分析中要求用户输入一定的参数，例如希望产生的簇的数目。聚类结果对于输入参数十分敏感。参数通常很难确定，特别是对于包含高维对象的数据集来说。这样不仅加重了用户的负担，也使得聚类的质量难以控制。
- 处理“噪声”数据的能力：绝大多数现实中的数据库都包含了孤立点，缺失，或者错误的数据。一些聚类算法对于这样的数据敏感，可能导致低质量的聚类结果。
- 对于输入记录的顺序不敏感：一些聚类算法对于输入数据的顺序是敏感的。例如，同一个数据集合，当以不同的顺序交给同一个算法时，可能生成差别很大的聚类结果。开发对数据输入顺序不敏感的算法具有重要的意义。
- 高维度（high dimensionality）：一个数据库或者数据仓库可能包含若干维或者属性。许多聚类算法擅长处理低维的数据，可能只涉及两到三维。人类的眼睛在最多三维的情况下能够很好地判断聚类的质量。在高维空间中聚类数据对象是非常有挑战性的，特别是考虑到这样的数据可能分布非常稀疏，而且高度偏斜。
- 基于约束的聚类：现实世界的应用可能需要在各种约束条件下进行聚类。假设你的工作是在一个城市中为给定数目的自动提款机选择安放位置，为了作出决定，你可以对住宅区进行聚类，同时考虑如城市的河流和公路网，每个地区的客户要求等情况。要找到既满足特定的约束，又具有良好聚类特性的数据分组是一项具有挑战性的任务。
- 可解释性和可用性：用户希望聚类结果是可解释的，可理解的，和可用的。也就是说，聚类可能需要和特定的语义解释和应用相联系。应用目标如何影响聚类方法的选择也是一个重要的研究课题。

记住这些约束，我们对聚类分析的学习将按如下的步骤进行。首先，学习不同类型的数据，以及它们对聚类方法的影响。接着，给出了一个聚类方法的一般分类。然后我们详细地讨论了各种聚类方法，包括划分方法，层次方法，基于密度的方法，基于网格的方法，以及基于模型的方法。最后我们探讨在高维空间中的聚类和孤立点分析（outlier analysis）。

8．2 聚类分析中的数据类型

在这一节中，我们研究在聚类分析中经常出现的数据类型，以及如何对其进行预处理。假设要聚类的数据集合包含 n 个数据对象，这些数据对象可能表示人，房子，文档，国家等。许多基于内存的聚类算法选择如下两种有代表性的数据结构：

- 数据矩阵（Data matrix，或称为对象属性结构）：它用 p 个变量（也称为属性）来表现 n 个对象，例如用年龄，身高，性别，种族等属性来表现对象“人”。这种数据结构是关系表的形式，或者看为 n*p 维（n 个对象*p 个属性）的矩阵。
  (8.1 p338?)

- 相异度矩阵（dissimilarity matrix，或称为对象-对象结构）：存储 n 个对象两两之间的近似性，表现形式是一个 n*n 维的矩阵。

  (8.2 p338?)
  在这里 d(i,j)是对象 i 和对象 j 之间相异性的量化表示，通常它是一个非负的数值，当对象 i 和 j 越相似，其值越接近 0；两个对象越不同，其值越大。既然 d(i,j) = d(j,i)，而且 d(i,i)=0，我们可以得到形如(8.2)的矩阵。关于相异度，我们在这一节中会进行详细探讨。

数据矩阵经常被称为二模（two-mode）矩阵，而相异度矩阵被称为单模（one-mode）矩阵。这

是因为前者的行和列代表不同的实体，而后者的行和列代表相同的实体。许多聚类算法以相异度矩阵为基础。如果数据是用数据矩阵的形式表现的，在使用该类算法之前要将其转化为相异度矩阵。

你可能想知道如何来估算相异度。在本节中，我们讨论如何计算用各种类型的属性来描述的对象的相异度，相异度将被用来进行对象的聚类分析。

8.2.2 区间标度（Interval-Scaled）变量

本节讨论区间标度变量和它们的标准化，然后描述距离度量，它通常用于计算用该类变量描述的对象的相异性。距离的度量包括欧几里得距离，曼哈顿距离，以及明考斯基距离。

“什么是区间标度变量？” 区间标度变量是一个线性标度的连续度量。典型的例子包括重量和高度，经度和纬度坐标，以及大气温度。

选用的度量单位将直接影响聚类分析的结果。例如，将高度的度量单位由“米”改为“英尺”，或者将重量的单位由“千克”改为“磅”，可能产生非常不同的聚类结构。一般而言，所用的度量单位越小，变量可能的值域就越大，这样对聚类结果的影响也越大。为了避免对度量单位选择的依赖，数据应当标准化。标准化度量值试图给所有的变量相等的权重。当没有关于数据的先验知识时，这样做是十分有用的。但是，在一些应用中，用户可能想给某些变量较大的权重。例如，当对篮球运动员进行聚类时，我们可能愿意给高度变量较大的权重。

“怎样将一个变量的数据标准化？”为了实现度量值的标准化，一种方法是将原来的度量值转换为无单位的值。给定一个变量 f 的度量值，可以进行如下的变换：

1．计算平均的绝对偏差（mean absolute deviation）S_f:

$S_f = (|x_{1f}-m_f|+|x_{2f}-m_f|+\ldots+|x_{nf}-m_f|)/n$ （8.3 p339?）

这里的 $x_{1f},\ldots,x_{nf}$ 是 f 的 n 个度量值，m_f 是 f 的平均值，即

$m_f =(|x_{1f}+x_{2f}+\ldots+x_{nf})/n$

2．计算标准化的度量值，或 z-score：

$z_{if} = (x_{if} - m_f) / s_f$ (8.4 p840 ?)

这个平均的绝对偏差 s_f 比标准的偏差对于孤立点具有 更好的鲁棒性。在计算平均绝对偏差时，度量值与平均值的偏差没有被平方，因此孤立点的影响在一定程度上被减小了。虽然存在更好的对偏差的度量方法，例如中值绝对偏差（median absolute deviation），但采用平均绝对偏差的优点在于孤立点的 z-score 值不会太小，因此孤立点仍可以被发现。

在特定的应用中，数据标准化可能有用，也可能没用。因此是否和如何进行标准化的选择被留给了用户。在第三章里数据预处理的规范化技术中也讨论了标准化的方法。

“好的，”现在你可能会问，“我已经对数据进行了标准化处理，我该如何计算对象间的相异度？”在标准化处理后，或者在某些应用中不需要标准化，对象间的相异度（或相似度）是基于对象间的距离来计算的。最常用的距离度量方法是欧几里得距离，它的定义如下：

d(I,j) = （？p340，8.5）

这里的 i=（xi1，xi2，…,xip）和 j=(xj1,xj2,…xjp)是两个 p 维的数据对象。

另一个著名的度量方法是曼哈顿距离，其定义如下：

d（I，j）= |xi1-xj1|+|xi2-xj2|+…+|xip-xjp| (8.6 p340 ?)

上面的两种距离度量方法都满足对距离函数的如下数学要求：

1．d（i，j）≥0：距离是一个非负的数值。

2．d（i，i）=0：一个对象与自身的距离是 0。

3．d（i，j）= d（j，i)：距离函数具有对称性。

4．d（i，j）≤ d（i，h）+d（h，j)：从对象 I 到对象 j 的直接距离不会大于途径任何其他对象的距离。

明考斯基距离是欧几里得距离和曼哈顿距离的概化，它的定义如下：

D（I，j）=（|xi1-xj1|q+|xi2-xj2|q+…+|xip-xjp|q）1/q (8.7 p341 ?)

这里的 q 是一个正整数。当 q=1 时，它表示曼哈顿距离；当 a=2 表示欧几里得距离。

如果对每个变量根据其重要性赋予一个权重，加权的欧几里得距离可以计算如下：

d（I，j）= w1|xi1-xj1|2+ (?p341,8.8)

8.2.3 二元变量（binary variable）

本小节介绍如何计算用二元变量描述的对象间的相似度。

一个二元变量只有两个状态：0 或 1，0 表示该变量为空，1 表示该变量存在。例如，给出一个描述病人的变量 smoker，1 表示病人抽烟，而 0 表示病人不抽烟。像处理区间标度变量一样来对待二元变量会误导聚类结果，所以要采用特定的方法来计算其相异度。

“那么，我怎样计算两个二元变量之间的相似度？”一个方法涉及对给定的数据计算相异度矩阵。如果假设所有的二元变量有相同的权重，我们得到一个两行两列的可能性表 8.1。在表中，q 是对对象 i 和 j 值都为 1 的变量的数目，r 是在对象 i 中值为 1，在对象 j 中值为 0 的变量的数目，s 是在对象 i 中值为 0，在对象 j 中值为 1 的变量的数目，t 是在对象 i 和 j 中值都为 0 的变量的数目。变量的总数是 p，p=q+r+s+t。

表 8.1 二元变量的可能性表

	对象 j		
			总和
对象 I			

(? p341)

“对称的二元变量和不对称的二元变量之间的区别是什么？”如果它的两个状态有相同的权重, 那么该二元变量是对称的，也就是两个取值 0 或 1 没有优先权。例如，属性“性别”就是这样的一个例子，它有两个值：“女性”和“男性”。基于对称二元变量的相似度称为恒定的相似度，即当一些或者全部二元变量编码改变时，计算结果不会发生变化。对恒定的相似度来说，评价两个对象 i 和 j 之间相异度的最著名的系数是简单匹配系数，其定义如下：

d(I,j) = (r+s) / (q+r+s+t) （8.9 p342 ?）

如果两个状态的输出不是同样重要，那么该二元变量是不对称的。例如一个疾病检查的肯定和否定的结果。根据惯例，我们将比较重要的输出结果，通常也是出现几率较小的结果编码为 1（例如，HIV 阳性），而将另一种结果编码为 0（例如 HIV 阴性）。给定两个不对称的二元变量，两个都取值 1 的情况（正匹配）被认为比两个都取值 0 的情况（负匹配）更有意义。因此，这样的二元变量经常被认为好像只有一个状态。基于这样变量的相似度被称为非恒定的相似度。对非恒定的相似度，最著名的评价系数是 Jaccard 系数，在它的计算中，负匹配的数目被认为是不重要的，因此被忽略。

D(I,j) = (r+s) / (q+r+s) （8.10）

当对称的和非对称的二元变量出现在同一个数据集中，在 8.2.4 节中描述的混合变量方法可以被应用。

例 8.1 二元变量之间的相异度：假设一个病人记录表（表 8.2）包含属性 name（姓名）, gender（性别）, fever（发烧）, cough（感冒）, test-1, test-2, test-3, 和 test-4，这里的 name 是对象标识，gender 是对称的二元变量，其余的属性都是非对称的二元变量。

表 8.2 大部分为二元属性的关系变量
(p343 ?)

对非对称属性，值 Y(yes)和 P(positive)被置为 1，值 N(no 或者 negative)被置为 0。假设对象（病人）之间的距离只基于非对称变量来计算。根据 Jaccard 系数公式（8.10），三个病人 Jack,Mary,和 Jim 两两之间的相异度如下：

d(jack,mary) = (0+1)/(2+0+1) = 0.33 (8.11 p343)
d(jack,jim) = (1+1)/ (1+1+1) = 0.67 (8.12 p343 ?)
d(jim,mary)=(1+2)/(1+1+2) = 0.75 (8.13 p343 ?)
上面的值显示 Jim 和 Mary 不可能有相似的疾病，因为他们有着最高的相异度。在这三个病人中，Jack 和 Mary 最可能有类似的疾病。

8.2.4 标称型、序数型和比例标度型变量

本节讨论如何计算用标称（Nominal），序数（Ordinal）和比例标度（Ratio-Scaled）变量描述的对象之间的相异度。

标称变量

标称变量是二元变量的推广，它可以具有多于两个的状态值。例如，map_color 是一个标称变量，它可能有五个值：红色 ,黄色，绿色，粉红色，和蓝色。

假设一个标称变量的状态数目是 M。这些状态可以用字母，符号，或者一组整数（如 1，2，…,M）来表示。要注意这些整数只是用于数据处理，并不代表任何特定的顺序。

“如何计算标称变量所描述的对象之间的相异度？”两个对象 i 和 j 之间的相异度可以用简单匹配方法来计算：

d(I,j) = (p-m)/p (8.14 p343)

这里 m 是匹配的数目，即对 i 和 j 取值相同的变量的数目；而 p 是全部变量的数目。我们可以通过赋权重来增加 m 的影响，或者赋给有较多状态的变量的匹配更大的权重。

通过为每个状态创建一个二元变量，可以用二元变量来表示标称变量。对一个有特定状态值的对象，对应该状态值的二元变量值置为 1，而其余的二元变量值置为 0。例如，为了对标称变量 map_color 进行编码，对应于上面所列的五种颜色分别创建一个二元变量。如果一个对象是黄色，那么 yellow 变量被赋值为 1，而其余的四个变量被赋值为 0。对于这种形式的编码，可以采用在 8.2.2 节中讨论的方法来计算相异度。

序数型变量

一个离散的序数型变量类似于标称变量，除了序数型变量的 M 个状态是以有意义的序列排序的。序数型变量对记录那些难以客观度量的主观评价是非常有用的。例如，职业的排列经常按某个顺序，例如助理，副手，正职。一个连续的序数型变量看起来象一个未知刻度的连续数据的集合，也就是说，值的相对顺序是必要的，而其实际的大小则不重要。例如，在某个比赛中的相对排名（例如金牌，银牌，和铜牌）经常比事实的度量值更为必需。将区间标度变量的值域划分为有限个区间，从而将其值离散化，也可以得到序数型变量。一个序数型变量的值可以映射为排序。例如，假设一个变量 f 有 Mf 个状态，这些有序的状态定义了一个序列 1，…,Mf。

“怎样处理序数型变量？”在计算对象的相异度时，序数型变量的处理与区间标度变量非常类似。假设 f 是用于描述 n 个对象的一组序数型变量之一，关于 f 的相异度计算包括如下步骤：

1. 第 i 个对象的 f 值为 xif，变量 f 有 Mf 个有序的状态，对应于序列 1，…，Mf。用对应的 rif 代替 xif，rif∈ {1,…,Mf}。
2. 既然每个序数型变量可以有不同数目的状态，我们经常必须将每个变量的值域映射到 [0 .0, 1.0]上，以便每个变量都有相同的权重。这一点可以通过用 zif 代替 rif 来实现。
   Zif = (rif –1) / (Mf-1) (8.15 p344)
3. 相异度的计算可以采用 8.2.1 节所描述的任意一种距离度量方法，采用 zif 作为第 i 个对象的 f 值。

比例标度型变量

比例标度型变量在非线性的刻度取正的度量值，例如指数，近似地遵循如下的公式

（？8.16 p345）

这里的 A 和 B 是正的常数。典型的例子包括细菌数目的增长，或者放射性元素的衰变。

“如何计算用比例标度型变量描述的对象之间的相异度？”目前有三种方法：

- 采用与处理区间标度变量同样的方法。但是，这种作法通常不是一个好的选择，因为刻度可能被扭曲了。
- 对比例标度型变量进行对数变换，例如对象 i 的 f 变量的值 xif 被变换为 yif，yif = log(xif)。变换得到的 yif 值可以采用在 8.2.1 节中描述的方法来处理。需要注意的是，对一些比例标度型变量，可以采用 log-log 或者其他形式的变换，具体的做法取决于定义和应用。
- 将 xif 看作连续的序数型数据，将其秩作为区间标度的值来对待。

尽管选用哪种方法取决于实际的应用，但后两种方法是比较有效的。

8.2.5 混合类型的变量

在 8.2.1 到 8.2.3 节中讨论了计算由同种类型变量描述的对象之间的相异度的方法，变量的类型可能是区间标度变量, 对称二元变量,不对称二元变量,标称变量,序数型变量 或者比例标度型变量。但是在许多真实的数据库中，对象是被混合类型的变量描述的。一般来说，一个数据库可能包含上面列出的全部六种类型。

“那么，我们怎样计算用混合类型变量描述的对象之间的相异度？”一种方法是将变量按类型分组，对每种类型的变量进行单独的聚类分析。如果这些分析得到兼容的结果，这种方法是可行的。但是，在实际的应用中，这种情况是不大可能的。

一个更可取的方法是将所有的变量一起处理，只进行一次聚类分析。一种技术将不同类型的变量组合在单个相异度矩阵中，把所有有意义的变量转换到共同的值域区间[0.0, 1.0]上。

假设数据集包含 p 个不同类型的变量，对象 i 和 j 之间的相异度 d(i,j)定义为

d(I,j) = (? 8.17 p346)

如果 x_{if} 或 x_{jf} 缺失（即对象 i 或对象 j 没有变量 f 的度量值），或者 $x_{if}=x_{jf}=0$，且变量 f 是不对称的二元变量，则指示项（？）=0；否则，指示项（？）=1。变量 f 对 i 和 j 之间相异度的计算方式与其具体类型有关：

- 如果 f 是二元或标称变量：如果 $x_{if}=x_{jf}$，$d_{ij}^{(f)}$（？）=0；否则 $d_{ij}^{(f)}$（？）=1。
- 如果 f 是区间标度变量：$d_{ij}^{(f)}$（？）=（？），这里的 h 包含了所有有变量 f 值的对象。
- 如果 f 是序数型或者比例标度型变量：计算秩 r_{if} 和 z_{if}=（？），将 z_{if} 作为区间标度变量值对待。

这样，当描述对象的变量是不同类型时，对象之间的相异度也能够进行计算。

8.3 主要聚类方法的分类

目前在文献中存在大量的聚类算法。算法的选择取决于数据的类型， 聚类的目的和应用。如果聚类分析被用作描述或探查的工具，可以对同样的数据尝试多种算法，以发现数据可能揭示的结果。

大体上，主要的聚类算法可以划分为如下几类：

划分方法（partitioning methods）：给定一个 n 个对象或元组的数据库，一个划分方法构建数据的 k 个划分，每个划分表示一个聚类，并且 k<=n。也就是说，它将数据划分为 k 个组，同时满足如下的要求：（1）每个组至少包含一个对象；（2）每个对象必须属于且只属于一个组。注意在某些模糊划分技术中第二个要求可以放宽。在参考文献中列出了对于该类技术的参照。

给定 k，即要构建的划分的数目，划分方法首先创建一个初始划分。然后采用一种迭代的重定位技术，尝试通过对象在划分间移动来改进划分。一个好的划分的一般准则是：在同一个类中的对象之间的距离尽可能小，而不同类中的对象之间的距离尽可能大。还有许多其他划分质量的评判准则。

为了达到全局最优，基于划分的聚类会要求穷举所有可能的划分。实际上，绝大多数应用采用了以下两个比较流行的启发式方法：（1）k-means 算法，在该算法中，每个簇用该簇中对象的平均值来表示。（2）k-medoids 算法，在该算法中，每个簇用接近聚类中心的一个对象来表示。这些启发

式聚类方法对在中小规模的数据库中发现球状簇很适用。为了对大规模的数据集进行聚类，以及处理复杂形状的聚类，基于划分的方法需要进一步的扩展。8.4 节对基于划分的聚类方法进行了深入的研究。

层次的方法（hierarchical methods）：层次的方法对给定数据集合进行层次的分解。根据层次的分解如何形成，层次的方法可以被分为凝聚的或分裂的方法。凝聚的方法，也称为自底向上的方法，一开始将每个对象作为单独的一个组，然后继续地合并相近的对象或组，直到所有的组合并为一个（层次的最上层），或者达到一个终止条件。分裂的方法，也称为自顶向下的方法，一开始将所有的对象置于一个簇中。在迭代的每一步中，一个簇被分裂为更小的簇，直到最终每个对象在单独的一个簇中，或者达到一个终止条件。

层次的方法的缺陷在于，一旦一个步骤（合并或分裂）完成，它就不能被撤消。这个严格规定是有用的，由于不用担心组合数目的不同选择，计算代价会较小。但是，该技术的一个主要问题是它不能更正错误的决定。有两种方法可以改进层次聚类的结果：（1）在每层划分中，仔细分析对象间的联接，例如 CURE 和 Chameleon 中的做法。（2）综合层次凝聚和迭代的重定位方法。首先用自底向上的层次算法，然后用迭代的重定位来改进结果。例如在 BIRCH 中的方法。8.5 节讨论了层次的聚类方法。

基于密度的方法：绝大多数划分方法基于对象之间的距离进行聚类。这样的方法只能发现球状的簇，而在发现任意形状的簇上遇到了困难。随之提出了基于密度的另一类聚类方法，其主要思想是：只要临近区域的密度（对象或数据点的数目）超过某个阈值，就继续聚类。也就是说，对给定类中的每个数据点，在一个给定范围的区域中必须包含至少某个数目的点。这样的方法可以用来过滤“噪音”数据，发现任意形状的簇。

DBSCAN 是一个有代表性的基于密度的方法，它根据一个密度阈值来控制簇的增长。OPTICS 是另一个基于密度的方法，它为自动的，交互的聚类分析计算一个聚类顺序。
基于密度的聚类方法将在 8.6 节中进行详细的讨论。

基于网格的方法（grid-based methods）：基于网格的方法把对象空间量化为有限数目的单元，形成了一个网格结构。所有的聚类操作都在这个网格结构（即量化的空间）上进行。这种方法的主要优点是它的处理速度很快，其处理时间独立于数据对象的数目，只与量化空间中每一维的单元数目有关。

STING 是基于网格方法的一个典型例子。CLIQUE 和 WaveCluster 这两种算法既是基于网格的，又是基于密度的。对基于网格方法的详细讨论将在 8.7 节中进行。

基于模型的方法（model-based methods）：基于模型的方法为每个簇假定了一个模型，寻找数据对给定模型的最佳匹配。一个基于模型的算法可能通过构建反映数据点空间分布的密度函数来定位聚类。它也基于标准的统计数字自动决定聚类的数目，考虑“噪音”数据和孤立点，从而产生健壮的聚类方法。基于模型的聚类方法将在 8.8 节予以讨论。

一些聚类算法集成了多种聚类方法的思想，所以有时将某个给定的算法划分为属于某类聚类方法是很困难的。此外，某些应用可能有特定的聚类标准，要求综合多个聚类技术。

在接下来的章节中，我们将详细讨论上述的五类聚类方法，同时也将介绍综合了多种聚类方法思想的算法。作为与聚类分析密切相关的孤立点分析将在 8.9 节中讨论。

8.4 划分方法（partitioning methods）

给定一个包含 n 个数据对象的数据库，以及要生成的簇的数目 k，一个划分类的算法将数据对象组织为 k 个划分（k≤n），其中每个划分代表一个簇。通常会采用一个划分准则（经常称为相似度函数，similarity function），例如距离，以便在同一个簇中的对象是“相似的”，而不同簇中的对象是“相异的”。

算法：k-means。

输入：簇的数目 k 和包含 n 个对象的数据库。
输出：k 个簇，使平方误差最小。
方法：

（1）任意选择 k 个对象作为初始的簇中心；
（2）repeat
（3）　　根据与每个中心的距离，将每个对象赋给“最近”的簇；
（4）　　重新计算每个簇的平均值；
（5）until 不再发生变化

图 8．1 k-means 算法
（p349？）

8.4.1 典型的划分方法：k-Means 和 k-Medoids

最著名也是最常用的划分方法是 k-means，k-medoids 和它们的变种。

基于质心（centroid）的技术：k-Means 方法

k-means 算法以 k 为参数，把 n 个对象分为 k 个簇，以使类内具有较高的相似度，而类间的相似度最低。相似度的计算根据一个簇中对象的平均值（被看作簇的重心）来进行。

“k-means 算法是怎样工作的？”k-means 算法的处理流程如下。首先，随机地选择 k 个对象，每个对象初始地代表了一个簇中心。对剩余的每个对象，根据其与各个簇中心的距离，将它赋给最近的簇。然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛。有代表性地，平方误差准则被采用，其定义如下：

（8．　18　p349 ？）

这里的 E 是数据库中所有对象的平方误差的总和，p 是空间中的点，表示给定的数据对象，m_i 是簇 C_i 的平均值（p 和 m_i 都是多维的）。这个准则试图使生成的结果簇尽可能的紧凑和独立。图 8.1 给出了 K-means 过程的概述。

这个算法尝试找出使平方误差函数值最小的 k 个划分。当结果簇是密集的，而簇与簇之间区别明显时，它的效果较好。对处理大数据集，该算法是相对可伸缩的和高效率的，因为它的复杂度是 O（nkt），n 是所有对象的数目，k 是簇的数目，t 是迭代的次数。通常地，k<<n，且 t<<n。这个算法经常以局部最优结束。

但是，k-means 方法只有在簇的平均值被定义的情况下才能使用。这可能不适用于某些应用，例如涉及有分类属性的数据。要求用户必须事先给出 k（要生成的簇的数目）可以算是该方法的一个缺点。K-means 方法不适合于发现非凸面形状的簇，或者大小差别很大的簇。而且，它对于“噪音”和孤立点数据是敏感的，少量的该类数据能够对平均值产生极大的影响。

例子 8.2　假设有一个分布在空间中的对象集合，如图 8.2(a)所示。给定 k=3，即用户要求将这些对象聚为三类。根据图 8.1 中的算法，我们任意选择三个对象作为三个初始的簇中心，簇中心在图中用“+”来标注。根据与簇中心的距离，每个对象被分配给最近的一个簇。这样的分布形成了如图 8.2(a)中虚线所描绘的图形。

这样的分组会改变聚类中心，也就是说，每个聚类的平均值会根据类中的对象重新计算。依据这些新的聚类中心，对象被重新分配到各个类中。这样的重新分配形成了图 8.2（b）中虚线所描绘的轮廓。

以上的过程重复，产生图 8.2(c)的情况。最后，当没有对象的重新分配发生时，处理过程结束。聚类的结果被返回。

图 8.2 基于 k-means 方法的一组对象的聚类（簇中心在图中用“+”来标注）
（p350 ？）

k-means 方法有很多变种。它们可能在初始 k 个平均值的选择，相异度的计算，计算聚类平均值的策略上有所不同。经常会产生较好的聚类结果的一个有趣策略是首先采用层次的自底向上算法决定结果簇的数目，及找到初始的簇，然后用迭代的重定位来改进聚类结果。

k-means 方法的一个变体是 k-modes 方法，它扩展了 k-means 方法，用模式来代替类的平均值，采用新的相异性度量方法来处理分类性质的数据，采用基于频率的方法来修改聚类的模式。K-means 和 k-modes 方法可以综合起来处理有数值类型和分类类型属性的数据，这就是 k-prototypes 方法。

EM（Expectation Maximization，期望最大）算法以另一种方式对 k-means 方法进行了扩展。它不把对象分配给一个确定的簇，而是根据对象与簇之间隶属关系发生的概率来分派对象。换句话说，在簇之间没有严格的界限。因此，新的平均值基于加权的度量值来计算。

“怎样增强 k-means 算法的可扩展性？”最近提出的一种方法是识别数据的三种区域：可以压缩的区域，必须在主存中的区域，可废弃的区域。如果一个对象与某个簇的隶属关系是确定的，则它是可废弃的，。如果一个对象不是可废弃的，但属于某个较小的子簇，那么它是可压缩的。一个数据结构——聚类特征（clustering feature）用来汇总那些可废弃的或者可压缩的对象。如果一个对象既不是可废弃的，又不是可压缩的，那它就应该被保存在主存中。为了达到可扩展性，这个迭代的算法只包含可压缩的对象和必须在主存中的对象的聚类特征，从而将一个基于二级存储的算法变成了基于主存的算法。

基于有代表性的对象的技术：k-medoids 方法

既然一个有极大值的对象可能相当程度上扭曲数据的分布，k-means 算法对于孤立点是敏感的。

大家可能想知道，“如何修改这个算法来消除这种敏感性？”。不采用簇中对象的平均值作为参照点，可以选用簇中位置最中心的对象，即 medoid。这样划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的。这是 k-medoids 方法的基础。

k-medoids 聚类算法的基本策略是：首先为每个簇随意选择选择一个代表对象；剩余的对象根据其与代表对象的距离分配给最近的一个簇。然后反复地用非代表对象来替代代表对象，以改进聚类的质量。聚类结果的质量用一个代价函数来估算，该函数评估了对象与其参照对象之间的平均相异度。为了判定一个非代表对象 O_{random} 是否是当前一个代表对象 O_j 的好的替代，对于每一个非代表对象 p，下面的四种情况被考虑：

- 第一种情况：p 当前隶属于代表对象 O_j。如果 O_j 被 O_{random} 所代替，且 p 离 O_I 最近，i≠j，那么 p 被重新分配给 O_I。
- 第二种情况：p 当前隶属于代表对象 O_j。如果 O_j 被 O_{random} 代替，且 p 离 O_{random} 最近，那么 p 被重新分配给 O_{random}。
- 第三种情况：p 当前隶属于 O_I，i≠j。如果 O_j 被 O_{random} 代替，而 p 仍然离 O_I 最近，那么对象的隶属不发生变化。
- 第四种情况：p 当前隶属于 O_I，i≠j。如果 O_j 被 O_{random} 代替，且 p 离 O_{random} 最近，那么 p 被重新分配给 O_{random}。

1. 重新分配给 O_I　2. 重新分配给 O_{random}　3. 不发生变化　4. 重新分配给 O_{random}

数据对象

簇中心

替代前

替代后

图 8.3 k-medoids 聚类代价函数的四种情况.(P352 ?)

图 8.3 描述了上述的四种情况。每当重新分配发生时，square-error 所产生的差别对代价函数有影响。因此，如果一个当前的代表对象被非代表对象所代替，代价函数计算 square-error 值所产生的差别。替换的总代价是所有非代表对象所产生的代价之和。如果总代价是负的，那么实际的 square-error 将会减小，O_j 可以被 O_{random} 替代。如果总代价是正的，则当前的代表对象是可接受的，在本次迭代中没有变化发生。一个典型的 k-medoids 算法描述在图 8.4 中给出。

PAM（Partitioning around Medoids，围绕 k-medoids 的划分）是最早提出的 k-medoids 算法

之一。它试图对 n 个对象给出 k 个划分。最初随机选择 k 个代表对象后，该算法反复地试图找出更好的代表对象。所有可能的对象对被分析，每个对中的一个对象被看作是代表对象，而另一个不是。对可能的各种组合，估算聚类结果的质量。一个对象 0_j 被可以产生最大 square-error 值减少的对象代替。在一次迭代中产生的最佳对象的集合成为下次迭代的代表对象。当 n 和 k 的值较大时，这样的计算代价相当高。

“哪个方法更健壮：k-means 或者 k-medoids？”当存在“噪音”和孤立点数据时，k-medoids 方法比 k-means 方法更健壮，这是因为 medoid 不象平均值那么容易被极端数据影响。但是，k-medoids 方法的执行代价比 k-means 算法高。此外这两种方法都要求用户指定结果簇的数目 k。

算法：k-medoids，
输入：结果簇的数目 k，包含 n 个对象的数据库
输出：k 个簇，使得所有对象与其最近代表对象的相异度总和最小。
方法：

（1）随机选择 k 个对象作为初始的代表对象；
（2）repeat
（3）　　指派每个剩余的对象给离它最近的代表对象所代表的簇；
（4）　　随意地选择一个非代表对象 0_{random}；
（5）　　计算用 0_{random} 代替 0_j 的总代价 S；
（6）　　如果 S<0，则用 0_{random} 替换 0_j，形成新的 k 个代表对象的集合；
（7）until 不发生变化

图 8.4 k-medoids 算法　　（p353 ？）

8.4.2 大规模数据库中的划分方法：从 k-medoids 到 CLARANS

“k-medoids 算法在大数据集合上的效率如何？”典型的 k-medoids 算法，如 PAM，对小的数据集合非常有效，但对大的数据集合没有良好的可伸缩性。为了处理较大的数据集合，可以采用一个基于样本的方法 CLARA（Clustering LARge Applications）。

CLARA 的主要思想是：不考虑整个数据集合，选择实际数据的一小部分作为数据的样本。然后用 PAM 方法从样本中选择代表对象。如果样本是以非常随机的方式选取的，它应当足以代表原来的数据集合。从中选出的代表对象很可能与从整个数据集合中选出的非常近似。CLARA 抽取数据集合的多个样本，对每个样本应用 PAM 算法，返回最好的聚类结果作为输出。如同人们希望的，CLARA 能处理比 PAM 更大的数据集合。每步迭代的复杂度现在是 $O(ks^2+k(n-k))$，s 是样本的大小，k 是簇的数目，而 n 是所有对象的总数。CLARA 的有效性取决于样本的大小。要注意 PAM 在给定的数据集合中寻找最佳的 k 个代表对象，而 CLARA 在抽取的样本中寻找最佳的 k 个代表对象。如果任何取样得到的代表对象不属于最佳的代表对象，CLARA 不能得到最佳聚类结果。例如，如果对象 O_i 是最佳的 k 个代表对象之一，但它在取样的时候没有被选择，那 CLARA 将永远不能找到最佳聚类。这是为了效率而做的折中。如果样本发生偏斜，基于样本的一个好的聚类不一定代表了整个数据集合的一个好的聚类。

“我们怎样改进 CLARA 的聚类质量和可伸缩性？”作为 k-medoids 类的算法，CLARANS(Clustering Large Application based upon RANdomized Search)将采样技术和 PAM 结合起来。但是，与 CLARA 不同，CLARANS 没有在任一给定的时间局限于任一样本。CLARA 在搜索的每个阶段有一个固定的样本，而 CLARANS 在搜索的每一步带一定随机性地抽取一个样本。聚类过程可以被描述为对一个图的搜索，图中的每个节点是一个潜在的解，也就是说，k 个代表对象的集合。在替换了一个代表对象后得到的聚类结果被称为当前聚类结果的邻居。随机尝试的邻居的数目被用户定义的一个参数加以限制。如果一个更好的邻居被发现，也就是说它有更小的 square-error 值，CLARANS 移到该邻居节点，处理过程重新开始。否则当前的聚类达到了一个局部最优。如果找到了一个局部最优，CLARANS 从随机选择的节点开始寻找新的局部最优。

实验显示 CLARANS 比 PAM 和 CLARA 更有效。通过采用一个轮廓系数——对象的一个属性，定义该对象在多大程度上真的属于某个簇，能够发现最“自然的”的结果簇数目。CLARANS 能够

探测孤立点。但是 CLARANS 算法的计算复杂度大约是 $O(n^2)$，n 是对象的数目。而且，它的聚类质量取决于所用的抽样方法。通过采用空间数据结构，例如 R*-tree，及一些调节技术，CLARANS 的性能可以得到进一步的提高。

8.5 层次方法

一个层次的聚类方法将数据对象组成一棵聚类的树。根据层次分解是自底向上，还是自顶向下形成，层次的聚类方法可以进一步分为凝聚（agglomerative）和分裂(divisive)层次聚类。一个纯粹的层次聚类方法的聚类质量受限于如下特点：一旦一个合并或分裂被执行，就不能修正。最近的研究集中于凝聚层次聚类和迭代重定位方法的集成。

凝聚的

分裂的

图 8.5 在对象集合{a,b,c,d,e}上的凝聚和分裂层次聚类 （p355？）

8.5.1 凝聚的和分裂的层次聚类

一般来说，有两种类型的层次聚类方法：

- 凝聚的层次聚类：这种自底向上的策略首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到所有的对象都在一个簇中，或者某个终结条件被满足。绝大多数层次聚类方法属于这一类，它们只是在簇间相似度的定义上有所不同。
- 分裂的层次聚类：这种自顶向下的策略与凝聚的层次聚类不同，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到每个对象自成一簇，或者达到了某个终结条件，例如达到了某个希望的簇数目，或者两个最近的簇之间的距离超过了某个阈值。

例8.3 图 8.5 描述了一个凝聚的层次聚类算法 AGENES(Agglomerative NESting)和一个分裂的层次聚类算法 DIANA(Divisive ANAlysis)在一个包含五个对象的数据集合上的处理过程。最初，AGNES 将每个对象作为一个簇，然后这些类根据某些准则被一步步地合并。例如，如果簇 C_1 中的一个对象和簇 C_2 中的一个对象之间的距离是所有属于不同簇的对象间距离中最小的，C_1 和 C_2 可能被合并。这是一个 single-link 方法，以便每个聚类可以被簇中所有对象代表，簇间的相似度由两个簇中距离最近的数据点间的相似度来确定。聚类的合并过程反复进行直到所有的对象最终合并形成一个簇。

在 DIANA 方法的处理过程中，所有的对象初始都放在一个簇中。根据一些原则，这个簇被分裂，簇的分裂过程反复进行直到最终每个新的簇只包含一个对象。

在凝聚或者分裂的层次聚类方法中，用户能定义希望得到的簇数目作为一个结束条件。

四个广泛采用的簇间距离度量方法如下：

- 最小距离：$d_{min}(C_i,C_j) = \min_{p\in Ci,p'\in Cj} |p-p'|$
- 最大距离：$d_{max}(C_i,C_j) = \max_{p\in Ci,p'\in Cj} |p-p'|$
- 平均值的距离：$d_{mean}(C_i,C_j) = | m_I - m_j |$
- 平均距离：$d_{avg}(C_i,C_j) = \sum_{p\in Ci} \sum_{p'\in Cj} |p-p'|$

这里|p-p’|是两个对象 p 和 p’之间的距离，m_I 是簇 C_i 的平均值，而 n_I 是簇 Cj 中对象的数目。

“层次聚类的困难之处在那里？”层次聚类方法尽管简单，但经常会遇到合并或分裂点选择的困难。这样的决定是非常关键的，因为一旦一组对象被合并或者分裂，下一步的处理将在新生成的簇上进行。已做的处理不能被撤消，聚类之间也不能交换对象。如果在某一步没有很好地选择合并或分裂的决定，可能会导致低质量的聚类结果。而且，这种聚类方法不具有很好的可伸缩性，因为合并或分裂的决定需要检查和估算大量的对象或簇。

改进层次方法的聚类质量的一个有希望的方向是将层次聚类和其他的聚类技术进行集成，形成

多阶段聚类。在下面的章节中介绍了一些这类的方法。第一个方法称为 BIRCH，它首先用树结构对对象进行层次划分，然后采用其他的聚类算法对聚类结果进行求精。第二个方法称为 CURE，它采用固定数目的代表对象来表示每个簇，然后依据一个定义的分数向着聚类中心对它们进行收缩。 第三个方法 ROCK 基于簇间的互联性进行合并。第四个方法 Chameleon 在层次聚类中发现动态模型。

根
第一级

图 8。6 CF 树结构

8．5．2 BIRCH：利用层次方法的平衡迭代约减和聚类（Balanced Iterative Reducing and Clustering Using Hierarchies）

BITCH 是一个综合的层次聚类方法。它引入了两个概念：聚类特征和聚类特征树(CF tree)，它们用于概括聚类描述。这些结构辅助聚类方法在大型数据库中取得高的速度和伸缩性。BIRCH 方法对增量或动态聚类非常有效。让我们详细讨论一下上面提到的结构。一个聚类特征(CF)是一个三元组，对对象子类的信息给出了总结性描述。假设某个子类中有 N 个 d 维的点或对象$\{o_I\}$，则该子类的 CF 定义如下

CF = (N,LS,SS), (8.19 p357 ?)

这里 N 是子类中点的数目，LS（？）是 N 个点的线性和（即？），SS 是数据点的平方和（即？）。

聚类特征是对给定子类的统计描述：从统计学的观点来看，子类的第零个，第一个，及第二个要素。它记录了计算聚类和有效利用存储的关键数值，因为它概括了关于子类的信息，而不是存储所有的对象。

一个 CF 树是高度平衡的树，它为层次聚类存储了聚类特征。图 8.6 给出了一个例子。根据定义，树中的非叶节点有后代或“孩子”，它们存储了其孩子的 CF 的总和，即概括了关于其孩子的聚类信息。一个 CF 树有两个参数：分支因子 B，和阈值 T。分支因子定义了每个非叶节点的最大孩子数目，而阈值参数给出了存储在树的叶子节点中的子类的最大直径。这两个参数影响了结果树的大小。

“BIRCH 算法是怎样工作的？”它包括两个阶段：

阶段一：BIRCH 扫描数据库，建立一个初始存放于内存的 CF 树，它可以被看作数据的多层压缩，试图保留数据内在的聚类结构。

阶段二：BIRCH 采用某个聚类算法对 CF 树的叶节点进行聚类。

在阶段一，随着对象被插入，CF 树被动态地构造。所以这个方法支持增量聚类。一个对象被插入到最近的叶子条目（子类）。如果在插入后存储在叶子节点中的子类的直径大于阈值，那么该叶子节点及可能有其他节点被分裂。新对象插入后，关于该对象的信息向着树根传递。通过修改阈值，CF 树的大小可以改变。如果存储 CF 树需要的内存大小大于主存的大小，可以定义一个较小的阈值，并重建 CF 树。重建过程从旧树的叶子节点建造一个新树。这样，重建树的过程不需要重读所有的对象。这类似于 B+树构建中的插入和节点分裂。因此，为了建树，只需读一次数据。采用一些启发式规则和方法，通过额外的数据扫描来处理孤立点和改进 CF 树的质量。CF 树建好后，可以在阶段二被采用任何聚类算法，例如典型的划分方法，。

BIRCH 试图利用可用的资源来生成最好的聚类结果。给定有限的主存，一个重要的考虑是最小的 I/O 时间。BIRCH 采用了一种多阶段聚类技术：数据集合的单遍扫描产生了一个基本的聚类，一或多遍的额外扫描可以进一步地改进聚类质量。这个算法的计算复杂性是 O（n），这里的 n 是对象的数目。

“BIRCH 的有效性如何？”实验显示该算法具有对对象数目的线性伸缩性，及较好的聚类质量。但是，既然 CF 树的每个节点由于大小限制只能包含有限数目的条目，一个 CF 树节点并不总是对应于用户所认为的一个自然聚类。而且，如果簇不是球形的，BIRCH 不能很好地工作，因为它用了半径或直径的概念来控制聚类的边界。

8．5．3 CURE：利用代表点聚类(clustering using representative)

绝大多数聚类算法或者擅长处理球形和相似大小的聚类，或者在存在孤立点时变得比较脆弱。CURE 解决了偏好球形和相似大小的问题，在处理孤立点上也更加健壮。CURE 采用了一种新的层次聚类算法，该算法选择了位于基于质心和基于代表对象方法之间的中间策略。它不用单个质心或对象来代表一个簇，而是选择了数据空间中固定数目的具有代表性的点。一个簇的代表点通过如下方式产生：首先选择簇中分散的对象，然后根据一个特定的分数或收缩因子向簇中心“收缩”或移动它们。在算法的每一步，有最近距离的代表点对（每个点来自于一个不同的簇）的两个簇被合并。

每个簇有多于一个的代表点使得 CURE 可以适应非球形的几何形状。簇的收缩或凝聚可以有助于控制孤立点的影响。因此，CURE 对孤立点的处理更加健壮，而且能够识别非球形和大小变化较大的簇。对于大规模数据库，它也具有良好的伸缩性，而且没有牺牲聚类质量。

针对大数据库，CURE 采用了随机取样和划分两种方法的组合：一个随机样本首先被划分，每个划分被部分聚类。这些结果簇然后被聚类产生希望的结果。

下面的步骤描述了 CURE 算法的核心：

1．从源数据对象中抽取一个随机样本 S。
2．将样本 S 划分为一组分块。
3．对每个划分局部地聚类。
4．通过随机取样剔除孤立点。如果一个簇增长得太慢，就去掉它。
5．对局部的簇进行聚类。落在每个新形成的簇中的代表点根据用户定义的一个收缩因子 a 收缩或向簇中心移动。这些点描述和捕捉到了簇的形状。
6．用相应的簇标签来标记数据。

让我们来看一个例子。

例8.4　假设有一组点（或对象）位于一个长方形的区域。图 8.7(a)描述了这些对象的一个随机样本。这些对象被分为两个部分，每个部分分别基于最小平均距离进行局部聚类。如图 8.7(b)所示，所形成的局部聚类结果被虚线标出。每个簇代表用“+”标记。这些局部聚类结果被进一步的聚类，产生图 8.7(c)中实线所描出的两个簇。每个新的簇通过朝簇中心移动其代表点进行收缩或凝聚。这些代表点描述了每个簇的形状。这样，排除了孤立点后，初始的对象被划分为两个簇，如图 8.7(d)所示。

（p360　？）

图 8.7　用 CURE 对一组对象进行聚类。（a）对象的一个随机样本。（b）对象被划分和局部聚类。每个簇的代表点被“+”标记。（c）局部聚类被进一步地聚类。对每个新簇，代表点向簇中心收缩或凝聚。（d）最后的结果簇是非球形的。

在孤立点存在的情况下，CURE 可以产生高质量的聚类，支持复杂形状和不同大小的聚类。该算法要求整个数据库的一遍扫描。给定 n 个对象，CURE 的复杂度是 O（n）。“CURE 对用户给出的参数的敏感度如何，例如样本大小，希望的聚类的数目，及收缩因子α？”敏感度分析显示，尽管一些参数变化可能不影响聚类质量，一般来说，参数设置确实对聚类结果有显著的影响。

CURE 不处理分类属性。ROCK 是一个可选的凝聚的层次聚类算法，适用于分类属性。它通过将集合的互连性与用户定义的互连性模型相比较来度量两个簇的相似度。两个簇 C1 和 C2 的互连性被定义为两个簇间交叉邻接（cross link）的数目，$link(p_i, p_j)$是两个点 p_I 和 p_j 共同的邻居的数目。换句话说，簇间相似度是基于来自不同簇而有相同邻居的点的数目。

ROCK 首先根据相似度阈值和共同邻居的概念从给定的数据相似度矩阵构建一个稀疏的图，然后在这个稀疏图上运行一个层次聚类算法。

8.5.4 Chameleon（变色龙）：一个利用动态模型的层次聚类算法

chameleon 是一个在层次聚类中采用动态模型的聚类算法。在它的聚类过程中，如果两个簇间

的互连性和近似度与簇内部对象间的互连性和近似度高度相关，则合并这两个簇。基于动态模型的合并过程有利于自然的和相似的聚类的发现，而且只要定义了相似度函数就可应用于所有类型的数据。

Chameleon 的产生是基于对两个层次聚类算法 CURE 和 ROCK 的缺点的观察。CURE 及其相关的方案忽略了关于两个不同簇中的对象的互连性的信息，而 ROCK 及其相关的方案强调对象间互连性，却忽略了关于对象间近似度的信息。

“chameleon 怎样工作的呢？”图 8.8 中描述了 chameleon 的主要思想。Chameleon 首先通过一个图划分算法将数据对象聚类为大量相对较小的子类，然后用一个凝聚的层次聚类算法通过反复地合并子类来找到真正的结果簇。它既考虑了互连性，又考虑了簇间的近似度，特别是簇内部的特征，来确定最相似的子类。这样它不依赖于一个静态的，用户提供的模型，能够自动地适应被合并的簇的内部特征。

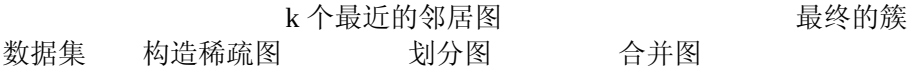


图 8.8 chameleon：基于 K 个最近的邻居和动态建模的层次聚类。见[KHK99] (p361 ?)

下面我们详细讨论 chameleon 算法。如图 8.8 所示，chameleon 基于通常采用的 k 个最近的邻居图方法来描述它的对象。K 个最近的邻居图中的每个点代表一个数据对象，如果一个对象是另一个对象的 k 个最类似的对象之一，在这两个点（对象）之间存在一条边。K 个最近邻居图 Gk 动态地捕捉邻域的概念：一个对象的领域半径被对象所在区域的密度所决定。在一个密集区域，邻域的定义范围相对狭窄；在一个稀疏区域，它的定义范围相对较宽。与采用基于密度的全局邻域方法相比，如 DBSCAN（在 8.6 节中描述），该方法能产生更自然的聚类结果。而且，区域的密度作为边的权重被记录下来。就是说，一个密集区域的边趋向于比稀疏区域的边有更大的权重。

Chameleon 通过两个簇的相对互连性 RI（Ci，Cj）和相对近似性 RC（Ci，Cj）来决定簇间的相似度：

- 如果两个簇 Ci 和 Cj 之间的相对互连性 RI（Ci，Cj）已经针对两个簇的内部互连性进行标准化，则称为 Ci 和 Cj 之间的绝对互联性，即，

  （ 8.20　p362 ?）

这里 $EC_{\{Ci,\ Cj\}}$是包含 Ci 和 Cj 的簇必须切断的边的数目，以便该簇分裂为 Ci 和 Cj。类似地，EC_{Ci}（或 EC_{Cj}）是它的最小等分线的大小（即将图划分为两个大致相等的部分需要切断的边的加权总和）

- 如果两个簇 Ci 和 Cj 之间的相对封闭性已经针 Ci 和 Cj 的内部封闭性进行标准化，则 RC(Ci,Cj) 称为 Ci 和 Cj 之间的绝对封闭性。它的定义如下：

  （8.21　p362 ?）

这里（？）是连接 Ci 和 Cj 节点的边的平均权重，（？）是 Ci（或 Cj）的最小等分线的边的平均权重。

可以看出 Chameleon 跟 CURE 和 DBSCAN 相比，在发现高质量的任意形状的聚类结果方面有更强的能力。但是在最坏的情况下，高维数据的处理代价可能对 n 个对象需要 $O(n^2)$的时间。

8.6 基于密度的方法

为了发现任意形状的聚类结果，提出了基于密度的聚类方法。这类方法将簇看作是数据空间中由低密度区域分割开的高密度对象区域。

8.6.1 DBSCAN:一个基于密度和高密度的连结区域的聚类算法

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个基于密度的聚类算法。该算法将具有足够高密度的区域划分为簇，并可以在带有“噪音”的空间数据库中发现任意形状的聚类。它定义簇为密度相连的点的最大集合。

基于密度的聚类的基本想法涉及一些新的定义。我们先给出这些定义，然后用一个例子加以说明。

- 一个给定对象周围半径ε内的区域称为该对象的ε –邻域。
- 如果一个对象的ε–邻域至少包含最小数目 MinPts 的对象，那么该对象称为核心对象。
- 给定一个对象集合 D，如果 p 是在 q 的ε–邻域内，而 q 是一个核心对象，我们说对象 p 从对象 q 出发是直接密度可达的，。
- 如果存在一个对象链 $p_1,p_2,\ldots,p_n$，$p_1=q$， $p_n=p$，对 $p_i\in$ D，$1\leqslant i\leqslant n$，p_{i+1} 是从 p_i 关于ε和 MinPts 直接密度可达的，则对象 p 是从对象 q 关于ε和 MinPts 密度可达的(density-reachable)。
- 如果对象集合 D 中存在一个对象 o，使得对象 p 和 q 是从 o 关于ε和 MinPts 密度可达的，那么对象 p 和 q 是关于ε和 MinPts 密度相连的（density-connected）。

密度可达性是直接密度可达性的传递闭包，这种关系是非对称的。只有核心对象之间是相互密度可达的。不过，密度相连性是一个对称的关系。

图 8.9 在基于密度的聚类中密度可达和密度相连。见[EKSX96] (p364 ?)

例8.5 考虑图 8.9，给定ε为圆的半径，MinPts=3。基于上述的定义：

- 在被标记的点中，M，P，O，及 R 是核心对象，因为它们在ε-领域内包含了至少三个对象。
- Q 是从 M 直接密度可达的。M 是从 P 直接密度可达的，反之亦然。
- 因为 Q 是从 M 直接密度可达的，M 是从 P 直接密度可达的，所以 Q 是从 P 间接密度可达的，但是，P 并不是从 Q 密度可达的，因为 Q 不是一个核心对象。类似地，R 和 S 是从 O 密度可达的，而 O 是从 R 密度可达的。
- O，R 和 S 都是密度互连的。

一个基于密度的簇是基于密度可达性的最大的密度相连对象的集合。不包含在任何簇中的对象被认为是“噪音”。

“DBSCAN 如何进行聚类？”DBSCAN 通过检查数据库中每个点的ε-邻域来寻找聚类。如果一个点 p 的ε-邻域包含多于 MinPts 个点，则创建一个以 p 作为核心对象的新簇。DBSCAN 然后反复地寻找从这些核心对象直接密度可达的对象，这个过程可能涉及几个密度可达簇的合并。当没有新的点可以被添加到任何簇时，该过程结束。

如果采用空间索引，DBSCAN 的计算复杂度是 O（nlogn），这里 n 是数据库中对象的数目。否则，计算复杂度是 O（n^2）。该算法对用户定义的参数是敏感的，DBSCAN 在下面的章节中会进一步的讨论，同时将与另一个基于密度的聚类算法 OPTICS 进行比较。

8.6.2 OPTICS：通过对象排序识别聚类结构（Ordering Points to Identify the Clustering Structure）

尽管 DBSCAN（8.6.1 节中描述的基于密度的聚类算法）能根据给定输入参数ε和 MinPts 对对象进行聚类，它仍然将选择能产生可接受的聚类结果的参数值的责任留给了用户。事实上，这也是许多其他聚类算法存在的问题。对于真实的，高维的数据集合而言，参数的设置通常是依靠经验，难以确定。绝大多数算法对参数值是非常敏感的：设置的细微不同可能导致差别很大的聚类结果。而且，真实的高维数据集合经常分布不均，全局密度参数不能刻画其内在的聚类结构。

为了解决这个难题，提出了 OPTICS 聚类分析方法。OPTICS 没有显式地产生一个数据集合簇，它为自动和交互的聚类分析计算一个簇次序（cluster ordering ）。这个次序代表了数据的基于密度的聚类结构。它包含了信息，等同于从一个广域的参数设置所获得的基于密度的聚类。

考察 DBSCAN，我们可以看到，对一个恒定的 MinPts 值，关于高密度的（即较小的ε值）的聚类结果被完全包含在根据较低密度所获得的密度相连的集合中。记住参数ε是距离，它是邻域的半径。因此，为了生成基于密度聚类的集合或次序，我们可以扩展 DBSCAN 算法来同时处理一组距离参数值。为了同时构建不同的聚类，对象应当以特定的顺序来处理。这个次序选择根据最小的ε值密度可达的对象，以便高密度的聚类能被首先完成。基于这个想法，每个对象需要存储两个值——核心距离（core-distance）和可达距离（reachability-distance）：

- 一个对象 p 的核心距离是使得 p 成为核心对象的最小ε。如果 p 不是核心对象，p 的核心距离没有定义。
- 一个对象 q 关于另一个对象 p 的可达距离是 p 的核心距离和 p 与 q 的欧几里得距离之间的较大值。如果 p 不是一个核心对象，p 和 q 之间的可达距离没有定义。

p 的核心距离　　可达距离（

可达距离（

图 8.10 OPTICS 术语。见[ABK99]（p366 ?）

例8.6　图 8.10 描述了核心距离和可达距离的概念。假设ε=6，MinPts=5。p 的核心距离是 p 与第四个最近的数据对象之间的距离ε'。q_1关于 p 的可达距离是 p 的核心距离（即ε'=3mm），因为它比从 p 到 q_1的欧几里得距离要大。q_2关于 p 的可达距离是从 p 到 q_2的欧几里得距离，它大于 p 的核心距离。

“这些值怎样使用？”OPTICS 算法创建了数据库中对象的一个次序，额外存储了每个对象的核心距离和一个适当的可达距离。一种算法被提出基于 OPTICS 产生的次序信息来抽取聚类。这样的信息对于关于小于在生成该次序中采用的距离ε的任何距离ε'的基于密度的聚类都是足够的。

一个数据集合的聚类次序可以被图形化地描述，以有助于它的理解。例如，图 8.11 是一个简单的二维数据集合的可达性图表，它给出了数据如何被聚类的概述。也有在多个层次上观察高维数据的聚类结构的方法被研究。

可达距离

无定义

对象的簇次序

图 8.11 OPTICS 中的簇次序。图来自[ABK99]。

由于 OPTICS 算法与 DBSCAN 在结构上的等价性，OPTICS 算法具有和 DBSCAN 相同的时间复杂度，即当空间索引被采用，复杂度为 O（nlongn）。

8．6．3 DENCLUE：基于密度分布函数的聚类

DENCLUE（DENsity-based CLUstEring）是一个基于一组密度分布函数的聚类算法。该算法主要基于下面的想法：（1）每个数据点的影响可以用一个数学函数来形式化地模拟，它描述了一个数据点在邻域内的影响，被称为影响函数（influence function）；（2）数据空间的整体密度可以被模拟为所有数据点的影响函数的总和；（3）然后聚类可以通过确定密度吸引点(density attractor)来得到，这里的密度吸引点是全局密度函数的局部最大值。

假设 x 和 y 是 d 维特征空间 F^d 中的对象。数据对象 y 对 x 的影响函数是一个函数 $f^y_B: F^d \rightarrow R^+_0$，它是根据一个基本的影响函数 f_B 来定义的：

$f^y_B(x) = f_B(x, y)$　　（8.22　p367 ?）

原则上，影响函数可以是一个任意的函数，它由某个领域内的两个对象之间的距离来决定。距离函数 d（x，y）应当是自反的和对称的，例如欧几里得距离函数（8.2.1 节）。它用来计算一个方波影

响函数（square wave influence function）：

f_{Square}（x，y）=　　（8.23　p367 ?）

或者一个高斯（Gaussian）影响函数：

（8.24　p367　?）

密度　　密度

数据集　　方波　　高斯

图 8.12 对一个 2-D 数据集的可能的密度函数（？p368）

在一个对象 x（$x \in F^d$）上的密度函数被定义为所有数据点的影响函数的总和。给定 n 个对象，$D=\{x1, \ldots, xn\} \subset F^d$，在 x 上的密度函数定义如下：

（8.25　p368 ?）

例如，根据高斯影响函数得出的密度函数是

（8．26　p368 ？）

根据密度函数，我们能够定义该函数的梯度和密度吸引点（全局密度函数的局部最大值）。一个点 x 是被一个密度吸引点 x*密度吸引的，如果存在一组点 $x_0, x_1, \ldots, x_k$，$x_0=x$，$x_k=x^*$，对 $0<I<k$，x_{i-1} 的梯度是在 x_I 的方向上。对一个连续的，可微的影响函数，一个用梯度指导的爬山算法能用来计算一组数据点的密度吸引点。图 8.12 显示了一个二维数据集，高斯密度函数和密度吸引点。

基于这些概念，中心定义的簇(center-defined cluster)和任意形状的簇(arbitrary-shape cluster)能够被形式化地定义。一个根据密度吸引点 x*中心定义的聚类是一个被 x*密度吸引的子集 C，在 x*的密度函数不小于一个阈值 ξ；否则（即如果它的密度函数值小于 ξ），它被认为是孤立点。一个任意形状的簇是一组子集 C，每一个是密度吸引的，有不小于阈值 ξ 的密度函数值，从每个区域到另一个都存在一条路径 P，该路径上每个点的密度函数值都不小于 ξ。中心定义和任意形状的簇的例子在图 8. 13 中给出。

图 8.13 中心定义的簇（顶部）和任意形状的簇（底部）的例子。见[HK98]. (p369　?)

“DENCLUE 与其它聚类算法相比有什么主要的优点？”主要有如下一些：（1）它有一个坚实的数学基础，概括了其他的聚类方法，包括基于划分的，层次的，及基于位置的方法。（2）对于有大量“噪音”的数据集合，它有良好的聚类特性。（3）对高维数据集合的任意形状的聚类，它给出了简洁的数学描述。（4）它使用了网格单元，只保存了关于实际包含数据点的网格单元的信息。它以一个基于树的存取结构来管理这些单元，因此比一些有影响的算法（如 DBSCAN）速度要快。但是，这个方法要求对密度参数 σ 和噪音阈值 ξ 进行仔细的选择，因为这样的参数选择可能显著地影响聚类结果的质量。

8.7 基于网格的方法

基于网格的聚类方法采用一个多分辨率的网格数据结构。它将空间量化为有限数目的单元，这些单元形成了网格结构，所有的聚类操作都在网格上进行。这种方法的主要优点是处理速度快，其处理时间独立于数据对象的数目，仅依赖于量化空间中每一维上的单元数目。

基于网格方法的有代表性的例子包括 STING，它利用了存储在网格单元中的统计信息；WaveCluster，它用一种小波转换方法来聚类对象；CLIQUE，它是在高维数据空间中基于网格和密度的聚类方法。

8.7.1 STING：统计信息网格(STatistical INformation Grid)

STING 是一个基于网格的多分辨率聚类技术，它将空间区域划分为矩形单元。针对不同级别的分辨率，通常存在多个级别的矩形单元，这些单元形成了一个层次结构：高层的每个单元被划分为

多个低一层的单元。关于每个网格单元属性的统计信息（例如平均值，最大值，和最小值）被预先计算和存储。这些统计变量可以方便下面描述的查询处理使用。

图 8.14 显示了 STING 聚类的一个层次结构。高层单元的统计变量可以很容易地从低层单元的变量计算得到。这些统计变量包括：属性无关的变量 count；属性相关的变量 m（平均值），s（标准偏差），min（最小值），max（最大值），以及该单元中属性值遵循的分布类型 distribution，例如正态的，均衡的，指数的，或无（如果分布未知）。当数据被装载进数据库，最底层单元的变量 count，m，s，min，和 max 直接进行计算。如果分布的类型事先知道，distribution 的值可以由用户指定，也可以通过假设检验来获得。一个高层单元的分布类型可以基于它对应的低层单元多数的分布类型，用一个阈值过滤过程来计算。如果低层单元的分布彼此不同，阈值检验失败，高层单元的分布类型被置为 none。

“这些统计信息怎样用于回答查询？”统计变量的使用可以以自顶向下的基于网格的方法。首先，在层次结构中选定一层作为查询处理的开始点。通常，该层包含少量的单元。对当前层次的每个单元，我们计算置信度区间（或者估算其概率），用以反映该单元与给定查询的关联程度。不相关的单元就不再考虑。低一层的处理就只检查剩余的相关单元。这个处理过程反复进行，直到达到最底层。此时，如果查询要求被满足，那么返回相关单元的区域。否则，检索和进一步的处理落在相关单元中的数据，直到它们满足查询要求。

“与其它聚类算法相比，STING 有什么优点？”STING 有几个优点：（1）由于存储在每个单元中的统计信息描述了单元中数据的与查询无关的概要信息，所以基于网格的计算是独立于查询的；（2）网格结构有利于并行处理和增量更新；（3）该方法的效率很高：STING 扫描数据库一次来计算单元的统计信息，因此产生聚类的时间复杂度是 O（n），n 是对象的数目。在层次结构建立后，查询处理时间是 O（g），这里 g 是最底层网格单元的数目，通常远远小于 n。

第一层
第（i-1）层
第 i 层

图 8.14 STING 聚类的层次结构 (p371 ?)

由于 STING 采用了一个多分辨率的方法来进行聚类分析，STING 聚类的质量取决于网格结构的最底层的粒度。如果粒度比较细，处理的代价会显著增加；但是，如果网格结构最底层的粒度太粗，将会降低聚类分析的质量。而且，STING 在构建一个父亲单元时没有考虑孩子单元和其相邻单元之间的关系。因此，结果簇的形状是（isothetic），即所有的聚类边界或者是水平的，或者是竖直的，没有斜的分界线。尽管该技术有快速的处理速度，但可能降低簇的质量和精确性，

8.7.2 WaveCluster：采用小波变换聚类

WaveCluster 是一种多分辨率的聚类算法，它首先通过在数据空间上强加一个多维网格结构来汇总数据，然后采用一种小波变换来变换原始的特征空间，在变换后的空间中找到密集区域。

在该方法中，每个网格单元汇总了一组映射到该单元中的点的信息。这种汇总信息适合于在内存中进行多分辨率小波变换使用，以及随后的聚类分析。

“什么是小波变换？”小波变换是一种信号处理技术，它将一个信号分解为不同频率的子波段。通过应用一维小波变换 n 次，小波模型可以应用于 n 维信号。在进行小波变换时，数据被变换以在不同的分辨率层次保留对象间的相对距离。这使得数据的自然聚类变得更加容易区别。通过在新的空间中寻找高密度区域，可以确定聚类。小波变换在第三章中也进行了讨论，它们用于通过压缩来缩减数据。对该技术的参考文献在文献目录中列出。

“为什么小波变换对聚类是有用的？”它主要有如下的优点：

- 它提供了没有监控的聚类。它采用了 hat-shape 过滤，强调点密集的区域，而忽视在密集区域外的较弱的信息。这样，在原始特征空间中的密集区域成为了附近点的吸引点（attractor），距离较远的点成为抑制点（inhibitor）。这意味着数据的聚类自动地显示出来，并“清理”了周围的区域。这样，小波变换的的另一个优点是能够自动地排除孤立点。

- 小波变换的多分辨率特性对不同精确性层次的聚类探测是有帮助的。例如，图 8.15 显示了一个二维特征空间的例子，图中的每个点代表了空间数据集中一个对象的属性或特征值。图 8.16 显示了不同分辨率的小波变换结果，从细的尺度到粗的尺度。在每一个层次，显示了原始数据分解得到的四个子波段。在左上像限中显示的子波段强调了每个数据点周围的平均邻域。右上像限内的子波段强调了数据的水平边。左下像限中的子波段强调了垂直边，而右下像限中的子波段强调了转角。
- 基于小波的聚类速度很快，计算复杂度是 O（n），这里 n 是数据库中对象的数目。这个算法的实现可以并行化。

图 8.15 二维特征空间的例子 (p373 ?)

图 8.16 图 8.15 中特征空间的多种分辨率的结果：（a）1 级（高分辨率）(b)2 级（中分辨率）(c)3 级（低分辨率）。见[SCZ98]。 (p373 ?)

WaveCluster 是一个基于网格和密度的算法。它符合一个好的聚类算法的许多要求：它能有效地处理大数据集合，发现任意形状的簇，成功地处理孤立点，对于输入的顺序不敏感，不要求诸如结果簇的数目，邻域的半径等输入参数的定义。在实验分析中，WaveCluster 在效率和聚类质量上优于 BIRCH，CLARANS，和 DBSCAN。实验分析也发现 WaveCluster 能够处理最多 20 维的数据。

8.7.3 CLIQUE：聚类高维空间

CLIQUE（Clustering In QUEst）聚类算法综合了基于密度和基于网格的聚类方法。它对于大型数据库中的高维数据的聚类非常有效。CLIQUE 的核心想法：

- 给定一个多维数据点的大集合，数据点在数据空间中通常不是均衡分布的。CLIQUE 区分空间中稀疏的和拥挤的区域，以发现数据集合的全局分布模式。
- 如果一个单元中的包含的数据点超过了某个输入参数，则该单元是密集的。在 CLIQUE 中，相连的密集单元的最大集合定义为簇。

“CLIQUE 如何工作？”CLIQUE 分两步进行多维聚类：

第一步，CLIQUE 将 n 维数据空间划分为互不相交的长方形单元，识别其中的密集单元。该工作对每一维进行。例如，图 8.17 显示了关于 age 和 salary，vocation 维的密集的长方形单元。代表这些密集单元的相交子空间形成了一个候选搜索空间，其中可能存在更高维度的密集单元。

图 8.17 关于 age 和 salary，vocation 维的密集单元,代表这些密集单元的相交子空间形成了一个候选搜索空间，其中可能存在更高维度的密集单元。

(p375 ?)

“为什么 CLIQUE 将更高维密集单元的搜索限制在子空间密集单元的交集中？”这种候选搜索空间的确定采用基于关联规则挖掘[23]中的先验特性（apriori property）。一般来说，该特性在搜索空间中利用数据项的先验知识以裁减空间。CLIQUE 所采用的特性如下：如果一个 k 维单元是密集的，那么它在 k-1 维空间上的投影也是密集的。也就是说，给定一个 k 维的候选密集单元，如果我们检查它的 k-1 维投影单元，发现任何一个不是密集的，那么我们知道第 k 维的单元也不可能是密集的。因此，我们可以从（k-1）维空间中发现的密集单元来推断 k 维空间中潜在的或候选的密集单元。通常，最终的结果空间比初始空间要小很多。然后对检查密集单元决定聚类。

在第二步，CLIQUE 为每个簇生成最小化的描述。对每个簇，它确定覆盖相连的密集单元的最大区域，然后确定最小的覆盖。

“CLIQUE 的有效性如何？” 因为高密度的聚类存在于那些子空间中，CLIQUE 自动地发现最高维的子空间，对元组的输入顺序不敏感，无需假设任何规范的数据分布。它随输入数据的大小线性地扩展，当数据的维数增加时具有良好的可扩展性。但是，由于方法大大简化，聚类结果的精确性可能会降低。

---

[23] 关联规则讨论参见第 6 章。特别地，先验特性的描述见 6.2.1 节。

8.8 基于模型的聚类方法

基于模型的聚类方法试图优化给定的数据和某些数学模型之间的适应性。这样的方法经常是基于这样的假设：数据是根据潜在的概率分布生成的。基于模型的方法主要有两类：统计学方法和神经网络方法。在本节中对每种方法的例子都给出了描述。

统计学方法

概念聚类是机器学习中的一种聚类方法，给出一组未标记的对象，它产生一个分类模式。与传统的聚类不同，概念聚类除了确定相似对象的分组外，还向前走了一步，为每组对象发现了特征描述，即每组对象代表了一个概念或类。因此，概念聚类是一个两步的过程：首先，进行聚类，然后给出特征描述。在这里，聚类质量不再只是单个对象的函数，而且加入了如导出的概念描述的简单性和一般性等因素。

概念聚类的绝大多数方法采用了统计学的途径，在决定概念或聚类时使用概率度量。概率描述用于描述导出的概念。

图 8.18 (p377 ?)

COBWEB 是一种简单，流行的增量概念聚类算法。它的输入对象用（分类属性，值）来描述。COBWEB 以一个分类树的形式创建层次聚类。

“但是，分类树是什么？它跟决策树一样吗？”图 8.18 显示了一棵对动物数据的分类树，它基于[Fis87]。分类树与决策树不同。分类树中的每个节点对应一个概念，包含该概念的一个概率描述，概述了被分在该节点下的对象。概率描述包括概念的概率和形如 P（Ai= Vij| Ck）的条件概率，这里 Ai=Vij 是一对属性和值，Ck 是概念类（计数被累计和存储在每个节点中，用于概率的计算）。这就与决策树不同，决策树标记分支，而非节点，而且采用逻辑描述符，而不是概率描述符[24]。在分类树某个层次上的兄弟节点形成了一个划分。为了用分类树对一个对象进行分类，采用了一个部分匹配函数来沿着最佳匹配节点的路径在树中向下移动。

COBWEB 采用了一个启发式估算值——分类效用（category utility）来指导树的构建。分类效用（Category Utility，CU）定义如下：

（8.27　p378 ?）

这里 n 是在树的某个层次上形成一个划分{C1，C2，…，Cn}的节点，概念，或“种类”的数目。尽管我们没有空间来显示这个推导过程，分类效用奖励类内相似性和类间相异性，

- 概率 P(Ai=Vij|Ck)表示类内相似性。该值越大，共享该属性-值的类成员比例越大，该属性-值对类成员的预见性就越大。
- 概率 P（Ck|Ai=Vij）表示类间相异性。该值越大，共享该属性-值却在其它类中的对象就越少，该属性-值对类的预见性就越大。

图 8。18 分类树。参见[Fis87]

让我们看一下 COBWEB 怎样工作。COBWEB 将对象增量地加入到分类树中。“给定一个新的对象，”你想知道，“COBWEB 怎样决定将其加入分类树的位置？”COBWEB 沿着一条适当的路径向下，修改计数，寻找可以分类该对象的最好节点。这个决策基于将对象临时置于每个节点，计算结果划分的分类效用。产生最高分类效用的方案应当是一个好的选择。

“但如果对象不属于树中现有的任何概念怎么办？如果为给出的对象新建一个节点更好怎么办？”这是一个很好的想法。事实上，COBWEB 也计算为给定对象创建一个新的节点
所产生的分类效用。它与基于现存节点的计算相比较。根据产生最高分类效用的划分，对象被置于一个已存在的类，或者为它创建一个新类。要注意 COBWEB 可以自动修正划分中类的数目。它不需要用户提供这样的输入参数。

上面提到的两个操作符对于对象的输入顺序非常敏感。为了降低它对输入顺序的敏感度，

[24] 决策树描述参见第 7 章。

COBWEB 有两个额外的操作符：合并（merging）和分裂（splitting）。当一个对象被加入，两个最好的候选节点可以考虑合并为单个类。此外，COBWEB 考虑在现有的分类中分裂最佳的候选节点的孩子。这些决定基于分类效用。合并和分裂操作符使得 COBWEB 执行一种双向的搜索，例如，一个合并可以抵消一个以前的分裂。

“COBWEB 的局限性是什么？”COBWEB 有若干局限性。首先，它基于这样一个假设：在每个属性上的概率分布是彼此独立的。由于属性间经常是相关的，这个假设并不总是成立。此外，聚类的概率分布描述使得更新和存储聚类相当昂贵。因为时间和空间复杂度不只依赖于属性的数目，而且取决于每个属性的值的数目，所以当属性有大量的取值时情况尤其严重。而且，分类树对于偏斜的输入数据不是高度平衡的，它可能导致时间和空间复杂性的剧烈变化。

CLASSIT 是 COBWEB 的扩展，用以处理连续性数据的增量聚类。它在每个节点中为每个属性存储一个连续的正常的分布（即平均值和标准偏差），采用一个修正的分类效用函数，计算出在连续属性上的一个整数值。但是，它与 COBWEB 存在类似的问题，因此不适用于聚类大规模的数据。

在产业界，AutoClass 是一个比较流行的聚类方法，它采用 Bayesian 统计分析来估算结果簇的数目。将概念聚类方法应用到数据挖掘中需要进行额外的研究。参考文献中给出了相关的参考书目。

神经网络方法

神经网络方法将每个簇描述为一个模型（exemplar）。模型作为聚类的“原型”，不一定对应一个特定的数据例子或对象。根据某些距离函数，新的对象可以被分配给模型与其最相似的簇。被分配给一个簇的对象的属性可以根据该簇的模型的属性来预测。

在本节中，我们讨论神经网络聚类的两个比较著名的方法。第一个是有竞争学习（competitive learning），第二个是自组织特征图，这两种方法都涉及有竞争的神经单元。

有竞争学习采用了若干个单元的层次结构（或者人造的“神经元”），它们以一种“winner-take-all”的方式对系统当前处理的对象进行竞争。图 8.19 显示了一个竞争学习系统的例子。每个圆圈代表一个单元。在一个簇中获胜的单元成为活动的（以填满的圆圈表示），而其它是不活动的（以空的圆圈表示）。各层之间的连接是激发（excitatory）——在某个给定层次中的单元可以接收来自低一层次所有单元的输入。在一层中活动单元的布局代表了高一层的输入模式。在某个给定层次中，一个簇中的单元彼此竞争，对低一层的输出模式做出反应。一个层次内的联系是抑制（inhibitory），以便在任何簇中只有一个单元是活动的。获胜的单元修正它与簇中其它单元连接上的权重，以便未来它能够对与当前对象相似或一样的对象做出较强的反应。如果我们将权重看作定义一个模型，那么新的对象被分配给有最近模型的簇。结果簇的数目和每个簇中单元的数目是输入参数。

层 3 抑制簇

激发联接

层 2 抑制簇

层 1 输入单元

输入模式

图 8.19 有竞争学习的结构。其层数可以任意。见[RZ85] (p380 ?)

在聚类过程结束时，每个簇可以被看作一个新的“feather”，它发现了对象的一些规律性。这样，产生的结果簇可以被看作一个低层特征向高层特征的映射。

对于自组织特征图（self-organizing feature map，SOMs）,聚类也是通过若干个单元竞争当前对象来进行的。权重向量最接近当前对象的单元成为获胜的或活动的单元。为了更接近输入对象，获胜单元及其最近的邻居的权重进行调整。SOMs 假设在输入对象中存在一些拓扑结构或顺序，单元将最终在空间中呈现这种结构。单元的组织形成一个特征图。SOMs 被认为类似于大脑的处理过程，对在二或三维空间中直观化高维数据是有用的。

神经网络聚类方法与实际的大脑处理有很强的理论联系。由于较长的处理时间和数据的复杂性，需要进行进一步的研究来使它适用于大规模的数据库。

8.9 孤立点(outlier)分析

“孤立点是什么？”经常存在一些数据对象，它们不符合数据的一般模型。这样的数据对象被称为孤立点，它们与数据的其它部分不同或不一致。

孤立点可能是度量或执行错误所导致的。例如，一个人的年龄为-999 可能是对未记录的年龄的缺省设置所产生的。另外，孤立点也可能是固有的数据可变性的结果。例如，一个公司的首席执行官的工资自然远远高于公司其他雇员的工资，成为一个孤立点。

许多数据挖掘算法试图使孤立点的影响最小化，或者排除它们。但是由于一个人的“噪音”可能是另一个人的信号，这可能导致重要的隐藏信息的丢失。换句话说，孤立点本身可能是非常重要的，例如在欺诈探测中，孤立点可能预示着欺诈行为。这样，孤立点探测和分析是一个有趣的数据挖掘任务，被称为孤立点挖掘。

孤立点挖掘有着广泛的应用。像上面所提到的，它能用于欺诈监测，例如探测不寻常的信用卡使用或电信服务。此外，它在市场分析中可用于确定极低或极高收入的客户的消费行为，或者在医疗分析中用于发现对多种治疗方式的不寻常的反应。

孤立点挖掘可以描述如下：给定一个 n 个数据点或对象的集合，及预期的孤立点的数目 k，发现与剩余的数据相比是相异的，例外的，或不一致的头 k 个对象。孤立点挖掘问题可以被看作两个子问题：（1）定义在给定的数据集合中什么样的数据可以被认为是不一致的；（2）找到一个有效的方法来挖掘这样的孤立点。

孤立点的定义非常重要。如果采用一个回归模型，剩余量的分析可以给出对数据“极端”的很好的估计。但是，当在时间序列数据中寻找孤立点时，它们可能隐藏在趋势的，周期性的，或者其他循环变化中，这项任务非常棘手。当分析多维数据时，不是任何特别的一个，而是维值的组合可能是极端的。对于非数值型的数据（如分类数据），孤立点的定义要求特殊的考虑。

“采用数据直观化方法来进行孤立点探测如何？”你可能想知道。既然人眼在发现数据的不一致上是非常迅速和有效的，可能看起来这是一个明显的选择。但是，这不适用于包含周期性曲线的数据，这时例外的值可能正好是现实中有效的值。数据直观化方法对于探测有很多分类属性的数据，或高维数据中的孤立点效率很低，这是因为人眼只擅长于处理两到三维的数值型数据。

在本节中，我们探讨基于计算机的孤立点探测方法。它们可以被分为三类：统计学方法，基于距离的方法，和基于偏移的方法，每类方法都会进行讨论。注意，当聚类算法将孤立点作为噪音剔除时，它们可以被修改，包括孤立点探测作为执行的副产品。一般说来，用户必须检查以确定发现的每个孤立点均是事实上的孤立点。

8.9.1 基于统计的孤立点探测

统计的方法对给定的数据集合假设了一个分布或概率模型（例如一个正态分布），然后根据模型采用不一致性检验（discordancy test）来确定孤立点。该检验要求数据集参数（例如假设的数据分布），分布参数（例如平均值和方差），和预期的孤立点的数目。

“不一致性检验如何进行？”一个统计学的不一致性检验检查两个假设：一个工作假设（working hypothesis）和一个替代假设（alternative hypothesis）。一个工作假设 H 是一个命题：n 个对象的整个数据集合来自一个初始的分布模型 F，即

H：Oi ∈F，I=1，2，…，n

如果没有显著的证据支持拒绝这个假设，它就被保留。不一致性检验验证一个对象 Oi 关于分布 F 是否显著地大（或者小）。依据可用的关于数据的知识，不同的统计量被提出来用作不一致性检验。假设某个统计量被选择用于不一致性检验，对象 Oi 的该统计量的值为 Vi，然后分布 T 被构建。显著性概率 SP（Vi）=Prob(T>Vi)被估算。如果某个 SP(Vi)是足够的小，那么 Oi 是不一致的，工作假设被拒绝。替代假设被采用，它声明 Oi 来自于另一个分布模型 G。既然 Oi 可能在一个模型下是孤立点，在另一个模型下是非常有效的值，那么结果非常依赖于模型 F 的选择。

替代分布在决定检验的能力（即当 Oi 真的是孤立点时工作假设被拒绝的概率）上是非常重要的。有许多不同的替代分布：

- 固有的替代分布（inherent alternative distribution）：在这种情况下，所有对象来自分布 F 的工作假设被拒绝，而所有对象来自另一个分布 G 的替代假设被接受：

H': Oi ∈G, I= 1,2,…,n （p383 ?）

F 和 G 可能是不同的分布，或者是参数不同的相同分布。对 G 分布的形式存在约束以便它有产生孤立点的可能性。例如，它可能有不同的平均值或者离差，或者更长的尾部。

- 混合替代分布（mixture alternative distribution）：混合替代分布认为不一致的值不是 F 分布中的孤立点，而是来自其他分布的污染物。在这种情况下，替代假设是：
  H': Oi∈(1-λ)F+λG, I=1, 2, …, n （p383 ?）
- 滑动替代分布(slippage alternative distribution)：这个替代分布声明所有的对象（除了少量外）根据给定的参数独立地来自初始的模型 F，而剩余的对象是来自修改过的 F 的独立的观察，这个 F 的参数已经变化了。

探测孤立点有两类基本的过程：

- block 过程：或者所有被怀疑的对象都被作为孤立点对待，或者都被作为一致的而接受。
- 连续的过程：该过程的一个例子是 inside-out 过程。它的主要思想是：最不可能是孤立点的对象首先被检验。如果它被发现是孤立点，那么所有更极端的值都被认为是孤立点；否则，下一个极端的对象被检验，依次类推。这个过程往往比 block 过程更为有效。

“在孤立点探测上统计学方法的有效性如何？”一个主要的缺点是绝大多数检验是针对单个属性的，而许多数据挖掘问题要求在多维空间中发现孤立点。而且，统计学方法要求关于数据集合参数的知识，例如数据分布。但是在许多情况下，数据分布可能是未知的。当没有特定的检验时，统计学方法不能确保所有的孤立点被发现，或者观察到的分布不能恰当地被任何标准的分布来模拟。

8.9.2 基于距离的孤立点探测

为了解决统计学方法带来的一些限制，引入了基于距离的孤立点的概念。

“什么是基于距离的孤立点？”如果至少数据集合 S 中对象的 p 部分与对象 o 的距离大于 d，对象 o 是一个基于距离的带参数 p 和 d 的孤立点，即 DB(p,d)。换句话说，不依赖于统计检验，我们可以将基于距离的孤立点看作是那些没有足够邻居的对象，这里的邻居是基于距给定对象的距离来定义的。与基于统计的方法相比，基于距离的孤立点探测归纳了多个标准分布的不一致性检验的思想。基于距离的孤立点探测避免了过多的计算，而大量的计算正是使观察到的分布适合某个标准分布，及选择不一致性检验所需要的。

对许多不一致性检验来说，如果一个对象 o 根据给定的检验是一个孤立点，那么对恰当定义的 p 和 d，o 也是一个 DB(p,d) 孤立点。例如，如果离平均值偏差 3 或更大的对象被认为是孤立点，假设一个正态分布，那么这个定义能够被一个 DB(0.9988,0.13σ)孤立点所概括[25]。

目前已经开发了若干个高效的挖掘基于距离的孤立点的算法，具体如下：

基于索引的算法： 给定一个数据集合，基于索引的算法采用多维索引结构，例如 R 树或 k-d 树，来查找每个对象 o 在半径 d 范围内的邻居。设 M 是一个孤立点的 d 邻域内的最大对象数目。因此，一旦对象 o 的 M+1 个邻居被发现，o 就不是 孤立点。这个算法在最坏情况下的复杂度为 $O(k*n^2)$，这里 k 是维数，n 是数据集合中对象的数目。当 k 增加时，基于索引的算法具有良好的扩展性。但是，复杂度估算只考虑了搜索时间，即使建造索引的任务本身就是计算密集的。

嵌套-循环算法：嵌套-循环算法和基于索引的算法有相同的计算复杂度，但它避免了索引结构的构建，试图最小化 I/O 的次数。它把内存的缓冲空间分为两半，把数据集合分为若干个逻辑块。通过精心选择逻辑块装入每个缓冲区域的顺序，I/O 效率能够改善。

基于单元(cell-based)的算法：为了避免 $O(n^2)$的计算复杂度，为驻留内存的数据集合开发了基于单元的算法。它的复杂度是 $O(c^k+n)$，这里 c 是依赖于单元数目的常数，k 是维数。在该方法中，数据空间被划分为边长等于 d/2(?)的单元。每个单元有两层围绕着。第一层的厚度是一个单元，而第二层的厚度是（p385 ?）。该算法一个单元一个单元地对孤立点记数，而不是一个对象一个对象地进行。

[25] 参数 p 和 d 使用正态曲线概率函数加以计算，以满足概率条件（。。。），即（。。。）。（注意其解决方案不是唯一的。）半径为 0.13 的 d 邻域表示一个 3σ ±0.13 单位（即，[2.87,3.13]）的范围。有关偏离的完整的证明见[KN97]。

对一个给定的单元，它累计三个计数——单元中对象的数目，单元和第一层中对象的数目，及单元和两个层次中的对象的数目。让我们把这些计数分别称为 cell_count，cell_+_1_layer_count，cell_+_2_layers_count。

“在该方法中怎样确定孤立点？”设 M 是一个孤立点的 d 邻域中可能存在的孤立点的最大数目。

- 在当前单元中的一个对象 o 被认为是孤立点，仅当 cell_+_1_layer_count 小于或等于 M。如果这个条件不成立，那么该单元中所有的对象可以从进一步的考察中移走，因为它们不可能是孤立点。
- 如果 cell_+_2_layers_count 小于或等于 M，那么单元中所有的对象被认为是孤立点。否则，如果这个计数大于 M，那么单元中的某些对象有可能是孤立点。为了探测这些孤立点，一个对象一个对象的处理被采用，对单元中的每个对象 o，o 的第二层中的对象被检查。对单元中的对象，只有那些 d 邻域内有不超过 M 个点的对象是孤立点。一个对象的 d 邻域由这个对象的单元，它的第一层的全部，和它的第二层的部分组成。

一个该算法的变形是关于 n 呈线性的，确保不会要求对数据集合进行超过三遍的扫描。它可以被用于大的磁盘驻留的数据集合，但对于高维数据不能很好地伸缩。

基于距离的孤立点探测要求用户设置参数 p 和 d。寻找这些参数的合适设置可能涉及多次的试探和错误。

8.9.3 基于偏离的孤立点探测

基于偏离的孤立点探测（deviation-based outlier detection）不采用统计检验或基于距离的度量值来确定异常对象。相反，它通过检查一组对象的主要特征来确定孤立点。与给出的描述偏离的对象被认为是孤立点。这样，该方法中的 deviation 典型地用于指孤立点。在本节中，我们研究基于偏离的孤立点探测的两种技术。第一种顺序地比较一个集合中的对象，而第二种采用了一个 OLAP 数据立方体方法。

序列异常技术

序列异常技术（sequential exception technique）(模仿了人类从一系列推测类似的对象中识别异常对象的方式。它利用了隐含的数据冗余。给定 n 个对象的集合 S，它建立一个子集合的序列，{S1,S2,…,Sm}，这里 2≤m≤n，满足

Sj-1　Sj, Sj S　(p386 ?)

序列中子集合间的相异度被估算。这个技术引入了下列的关键术语：

- 异常集(exception set)：它是偏离或孤立点的集合，被定义为某类对象的最小子集，这些对象的去除会产生剩余集合的相异度的最大减少。
- 相异度函数(dissimilarity function)：该函数不要求对象之间的度量距离。它可以是满足如下条件的任意函数：当给定一组对象时，如果对象间相似，返值就较小。对象间的相异度越大，函数返回的值就越大。一个子集的相异度是根据序列中先于它的子集增量计算的。给定一个 n 个对象的子集合$\{x_1,\ldots,x_n\}$，可能的一个相异度函数是集合中对象的方差：

  （8．28　p386？）

  这里（？）是集合中 n 个数的平均值。对于字符串，相异度函数可能是模式字符串的形式（例如，包含通配符），它可以用来覆盖目前所见的所有模式。当覆盖 Sj-1 中所有字符串的模式不能覆盖在 Sj 中，却不在 Sj-1 中的任一字符串时，相异度增加。
- 基数函数(cardinality function)：这一般是给定的集合中对象的数目。
- 平滑因子(smoothing factor)：这是一个为序列中的每个子集计算的函数。它估算从原始的数据集合中移走子集合可以带来的相异度的降低程度。该值由集合的势依比例决定。平滑因子值最大的子集是异常集。

一般的寻找异常集的任务可以是 NP 完全的（即，不可解的）。一个顺序的方法在计算上是可行的，能够用一个线性的算法实现。

“该方法如何工作？”不考虑其补集来估算当前子集的相异度，该算法从集合中选择了一个子集合的序列来分析。对每个子集合，它确定其与序列中前一个子集合的相异度差异。

“序列中子集合的顺序不影响结果吗？”为了减轻输入顺序对结果的任何可能的影响，以上的处理过程可以被重复若干次，每一次采用子集合的一个不同的随机顺序。在所有的迭代中有最大平滑因子值的子集合成为异常集。

OLAP 数据立方体技术

孤立点探测的 OLAP 方法在大规模的多维数据中采用数据立方体来确定反常区域。这种技术在第二章中有详细的描述。为了提高效率，孤立点的探测过程与立方体的计算是重叠的。这个方法是一种发现驱动的探索形式，预先计算的指示数据异常的值被用来在集合计算的所有层次上指导用户进行数据分析。如果一个立方体的单元值显著地不同与根据统计模型得到期望的值，该单元值被认为是一个例外，并采用可视化的提示来表示，例如背景颜色反映每个单元的异常程度。用户可以选择对那些标为异常的单元进行钻取。一个单元的度量值可能反映了发生在立方体更低层次上的异常，这些异常从当前的层次是不可见的。

这个模型考虑了涉及一个单元所属的所有维的度量值中的变化和模式。例如，假设你有一个销售数据的数据立方体，正在查看按月汇总的销售额。在可视化提示的帮助下，你注意到与其他月份相比，十二月的销售额有增长。这可能看起来是时间维上的一个异常。但是，通过向下查看十二月中每个项目的销售额，你发现每个项目的销售额都有相似的增长。因此，如果考虑项目维，十二月份总销售额的增长就不是一个异常。该模型考虑了隐藏在数据立方体集合分组操作后面的异常情况。对这样的异常，由于搜索空间很大，特别是当存在许多涉及多层概念层次的维的时候，人工探测是非常困难的。

8.10 总结

- 一个簇是一组数据对象的集合，在同一个簇中的对象彼此类似，而不同簇中的对象彼此相异。将一组物理或抽象对象分组为类似对象组成的多个簇的过程被称为聚类。
- 聚类分析有很广泛的应用，包括市场或客户识别，模式识别，生物学研究，空间数据分析，Web 文档分类，及许多其他方面。聚类分析可以用作独立的数据挖掘工具，来获得对数据分布的了解，也可以作为其它数据挖掘算法的预处理步骤。
- 聚类的质量是基于对象相异度来评估的，相异度可以对多种类型的数据来计算，包括区间标度变量，二元变量，标称变量，序数型变量，和比例标度型变量，或者这些变量类型的组合。
- 在数据挖掘中，聚类分析是一个活跃的研究领域。许多聚类算法已经被开发出来。具体可以分为划分方法，层次方法，基于密度的方法，基于网格的方法，及基于模型的方法。
- 划分方法首先得到初始的 k 个划分的集合，这里的参数 k 是要构建的划分的数目；然后它采用迭代重定位技术，试图通过将对象从一个簇移到另一个来改进划分的质量。有代表性的划分方法包括 k-means，k-medoids，CLARANS，和对它们的改进。
- 层次方法创建给定数据对象集合的一个层次性的分解。根据层次分解的形成过程，这类方法可以被分为自底向上的，或自顶向下的。为了弥补合并或分裂的严格性，凝聚的层次方法的聚类质量可以通过分析每个层次划分中的对象链接（例如 CURE 和 Chameleon），或集成其它的聚类技术（例如迭代重定位，BIRCH）来改进。
- 基于密度的方法基于密度的概念来聚类对象。它或者根据邻域对象的密度（例如 DBSCAN），或者根据某种密度函数（例如 DENCLUE）来生成聚类结果。OPTICS 是一个基于密度的方法，它生成数据聚类结构的一个扩充的顺序。
- 基于网格的方法首先将对象空间量化为有限数目的单元，形成网格结构，然后在网格结构上进行聚类。STING 是基于网格方法的一个有代表性的例子，它基于存储在网格单元中的统计信息聚类。CLIQUE 和 WaveCluster 是两个既基于网格，又基于密度的聚类算法。
- 基于模型的方法为每个簇假设一个模型，发现数据对模型的最好匹配。有代表性的基于模型的方法包括统计学方法（例如 COBWEB，CLASSIT，和 AutoClass），或神经网络方法（例如有竞争学习和自组织特征图）。
- 一个人的“噪音”可能是另一个人的信号。孤立点探测和分析对于欺诈探测，定制市场，医疗分析，及许多其它的任务是非常有用的。基于计算机的孤立点分析方法包括基于统计学方法，

基于距离的方法，和基于偏差的方法。

习题

8.1 简单地描述如何计算由如下类型的变量描述的对象间的相异度：

（a） 不对称的二元变量

（b） 标称变量

（c） 比例标度型（ratio-scaled）变量

（d） 数值型的变量

8.2 给定对如下的年龄变量的度量值：

18，22，25，42，28，43，33，35，56，28

通过如下的方法进行变量标准化：

（a） 计算年龄的平均绝对偏差

（b） 计算头四个值的 z-score

8.3 给定两个对象，分别表示为（22，1，42，10），（20，0，36，8）：

（a） 计算两个对象之间的欧几里得距离

（b） 计算两个对象之间的曼哈顿距离

（c） 计算两个对象之间的明考斯基距离，q=3

8.4 如下的表包含了属性 name，gender，trait-1，trait-2，trait-3，及 trait-4，这里的 name 是对象的 id，gender 是一个对称的属性，剩余的 trait 属性是不对称的，描述了希望找到笔友的人的个人特点。假设有一个服务是试图发现合适的笔友。

图（？ 390）

对不对称的属性的值，值 P 被设为 1，值 N 被设为 0。

假设对象（潜在的笔友）间的距离是只基于不对称变量来计算的。

（a） 给定 Kevan,Caroline,和 Erk，给出对象之间的可能性矩阵。

（b） 计算对象间的简单匹配系数。

（c） 计算对象间的 Jaccard 系数。

（d） 你认为哪两个人将成为最佳笔友？哪两个会是最不能相容的？

（e） 假设我们将对称变量 gender 包含在我们的分析中。基于 Jaccard 系数，谁将是最和谐的一对？为什么？

8.5 什么是聚类？简单描述如下的聚类方法：划分方法，层次方法，基于密度的方法，基于网格的方法，及基于模型的方法。为每类方法给出例子。

8.6 假设数据挖掘的任务是将如下的八个点（用(x,y)代表位置）聚类为三个类。

A1(2,10),A2(2,5),A3(8,4),B1(5,8),B2(7,5),B3(6,4),C1(1,2),C2(4,9)

距离函数是 Euclidean 函数。假设初始我们选择 A1,B1,和 C1 为每个聚类的中心，用 k-means 算法来给出

（a） 在第一次循环执行后的三个聚类中心

（b） 最后的三个簇

8.7 用一个图表来描述当给定一个常数 MinPts 时，关于一个较高密度（即，邻域半径取一个较低的值）的基于密度的聚类结果如何被完全包含在根据较低的密度所获得的密度相连的集合中。

8.8 人眼在判断聚类方法对二维数据的聚类质量上是快速而有效的。你能设计一个数据可视化方法来使数据聚类可视化和帮助人们判断三维数据的聚类质量吗？对更高维数据又如何？

8.9 给出一个特定的聚类方法如何被综合使用的例子，例如，什么情况下一个聚类算法被用作另一个算法的预处理步骤。

8.10 聚类被广泛地认为是一种重要的数据挖掘方法，有着广泛的应用。对如下的每种情况给出一个应用例子：

（a）采用聚类作为主要的数据挖掘方法的应用

（b）采用聚类作为预处理工具，为其它数据挖掘任务作数据准备的应用

8.11 数据立方体和多维数据库以层次的或聚集的形式包含分类的，序数型的，和数值型的数据。基于你已经学习的关于聚类方法的知识，设计一个可以有效和高效地在大数据立方体中发现簇的聚类方法。

8.12 假设你将在一个给定的区域分配一些自动取款机以满足需求。住宅区或工作区可以被聚类以便每个簇被分配一个 ATM。但是，这个聚类可能被一些因素所约束，包括可能影响 ATM 可达性的桥梁，河流和公路的位置。其它的约束可能包括对形成一个区域的每个地域的 ATM 数目的限制。给定这些约束，怎样修改聚类算法来实现基于约束的聚类？

8.13 为什么孤立点挖掘是重要的？简单地描述基于统计的孤立点探测，基于距离的孤立点探测，和基于偏离的孤立点探测的方法。

文献注释

若干本教科书中都讨论了聚类方法，例如 Hartigan[Har75]，Jain 和 Dubes[JD88]，及 Kaufman 和 Rousseeuw[KR90]。Jain，Murty 和 Flynn 关于聚类的一个最近的综述在[JMF99]中可以被找到。合并不同类型的变量到单个相异度矩阵中的方法在[KR90]中有介绍。

关于划分方法，k-means 算法首先由 MacQueen[Mac67]提出。K-medoids 算法 PAM 和 CLARA 由 Kaufman 和 Rousseeuw[KR90]提出。K-modes（k-模，聚类分类数据）和 k-prototype（k-原型，聚类混合数据）算法由 Huang[Hua98]提出，而 EM（Expectation Maximization，最大期望）算法由 Lauritzen[Lau95]提出。CLARANS 算法由 Ng 和 Han[NH94]提出。Ester，Kriegel，和 Xu[EKX95]提出了采用高效的空间存取方法，例如 R*树和调焦技术，来进一步改进 CLARANS 的性能。另一个基于 k-means 的可扩展的聚类算法由 Bradley，Fayyad，和 Reina[BFR98]提出。

凝聚的层次聚类（例如 AGNES）和分裂的层次聚类（例如 DIANA）由 Kaufman 和 Rousseeuw[KR90]提出。改进层次聚类方法的聚类质量的一个引起关注的方向是综合层次聚类和基于距离的迭代重定位，或其它非层次的聚类方法。例如，由 Zhang，Ramakrishnan，和 Linvy[ZRL96]提出的 BIRCH 在采用其他技术之前首先用 CF 树进行层次聚类。层次聚类也能被成熟的连结分析，转换，或者最近邻居分析来执行，例如 Guha，Rastogi，和 Shim[GRS98]提出的 CURE，Guha，Rastogi，和 Shim[GRS99]提出的 ROCK（聚类分类属性），及 Karypis，Han，和 Kumar[KHK99]提出的 Chameleon。

关于基于密度的聚类方法，Ester，Kriegel，Sander，和 Xu 在[EKSX96]提出了 DBSCAN。Ankerst，Breunig，Kriegel，和 Sander[ABKS99]开发了一个聚类排序方法 OPTICS，它方便了基于密度的聚类，而不用担心参数定义。基于一组密度分布函数的 DENCLUE 算法由 Hinneburg 和 Keim[HK98]提出。

一个基于网格的多分辨率方法 STING 由 Wang，Yang，和 Muntz[WYM97]提出，它在网格单元中收集统计信息。WaveCluster 由 Sheikholeslami，Chatterjee，和 Zhang[SCZ98]提出，是一个多分辨率的聚类方法。它通过小波变换来转换原始的特征空间。Agrawal，Gehrke，Gunopulos，和 Raghavan[AGGR98]提出的 CLIQUE 是一个综合了基于密度和基于网格方法的聚类算法，用于聚类高维数据。

关于基于模型的聚类方法，请参照 Shavlik 和 Dietterich[SD90]。概念聚类首先由 Michalski 和 Stepp[MS83]提出。统计的聚类方法的其它例子包括 Fisher[Fis87]提出的 COBWEB，Gennari，Langley，和 Fisher[GLF89]提出的 CLASSIT，及 Cheeseman 和 Stutz[CS96a]提出的 AutoClass。神经网络方法的研究包括 Rumelhart 和 Zipser[RZ85]提出的竞争学习和 Kohonen[Koh82]提出的 SOM（self-organizing feature maps）。

聚类分类数据的可扩展方法被广泛研究，包括 Gibson，Kleinberg，和 Raghavan[GKR98]，Guha，Rastogi，和 Shim[GRS99]，及 Ganti，Gehrke，和 Ramakrishnan[GGR99]。此外，也有许多其它的聚类范型。例如，模糊聚类方法在 Kaufman 和 Rousseeuw[KR90]，及 Bezdek 和 Pal[BP92]中进行了讨论。

孤立点探测和分析可以分为三类方法：基于统计的方法，基于距离的方法，和基于偏离的方法。Barnett 和 Lewis[BL94]中描述了统计的方法和不一致性检验。基于距离的孤立点探测在 Knorr 和 Ng[KN97，KN98]中有描述。基于偏离的孤立点探测的顺序方法在 Arning，Agrawal，和 Raghavan[AAR96]中提出。Sarawagi，Agrawal，和 Megiddo[SAM98]提出了一个发现驱动的方法，在大规模的多维数据中采用 OLAP 数据立方体来确定异常情况。Jagadish，Koudas，和 Muthukrishnan[JKM99]提出了一个高效的在时间序列数据库中挖掘异常的方法。

第九章 复杂类型数据的挖掘

前面所讨论的数据挖掘技术，主要面对的是以结构化数据为主的关系数据库，事务数据库，和数据仓库。随着数据处理工具，先进数据库技术，以及万维网（WWW）技术的迅速发展，大量的形式各样的复杂类型的数据（如结构化与非结构化，超文本与多媒体）不断涌现。因此数据挖掘面临的一个重要的课题就是针对复杂类型数据的挖掘，这包括复杂对象，空间数据，多媒体数据，时间序列数据，文本数据，和 Web 数据[26]。

本章主要讨论复杂信息的挖掘技术，包括对基本数据挖掘技术（如特征，关联，分类，和聚类）的扩展；对复杂数据类型提出一些新的技术；以及在复杂的信息中实施知识挖掘的方法。本章组织如下：9.1 节介绍基于复杂数据对象的多维分析和描述性挖掘；9.2 节描述空间数据挖掘；9.3 节讨论多媒体数据挖掘；9.4 介绍时间序列数据挖掘；9.5 给出文本数据库的挖掘；9.6 介绍了 Web 挖掘技术。由于对复杂数据的挖掘研究尚在起步阶段，本章讨论只涉及一些基本问题。我们希望今后能有更多的论著介绍复杂类型数据的挖掘技术。

9.1 复杂数据对象的多维分析和描述性挖掘（descriptive mining）

许多用于多维分析的商品化数据仓库和 OLAP 工具，其主要局限在于，维和度量（measure）只限定于特定的数据类型。大部分数据立方体的维限定为非数字数据，度量只能为简单的聚集值。为引入针对复杂数据的数据挖掘和多维数据分析，本节将介绍复杂数据对象的概化（generalization），以及用于对象数据库中 OLAP 和挖掘的对象立方体的构造。

复杂结构化数据（complex structured data）的存取方法在对象关系和面向对象数据库系统已有研究。在这些系统中，大量复杂数据对象组织为*类*，类又按*类/子类*的层次加以组织。类中的每个对象具有：（1）一个对象标识；（2）一组属性，它们可以具有复杂的数据结构，如集合（set）值或列表（list）值数据，类复合层次（class composition hierarchies），多媒体数据等等；（3）一组方法，用于说明与对象类相关的计算程序或规则。

为在对象关系和面向对象数据库中引入概化和归纳（induction），需重点讨论对象数据库中每一组成(component)的概化方法，以及概化数据用于多维分析和数据挖掘的方法。

9．1．1 结构数据概化

对象关系和面向对象数据库的主要特征就是对**复杂结构数据**（如集合值和列表值数据，和具有嵌套结构的数据）的存储，访问和建模。

“如何对这些数据进行概化？”首先来考虑集合值和列表值属性的概化。

一个**集合值属性**（set-valued attribute）可以是同构类型，也可以是异构类型。通常，集合值数据概化方法有：（1）将集合中的每一个值概化为其对应的更高级别的概念；或者（2）导出集合的一般特征，如集合中元素的个数，集合中类型或值的区间分布，或数字数据的加权平均。而且，同一概化可以基于不同的概化操作，得到不同的概化路径(generalization path)。在此情况下，概化结果为一个异构集合。

例 9．1 假设某人的业余爱好为一集合值属性，包含一组值{网球，曲棍球，国际象棋，小提琴，任天堂游戏}。这一集合可以概化为一组高级别概念，如{体育，音乐，电子游戏}，或概化为数字 5（即集合中有 5 个爱好）。而且，每个概化值可以连带一个记数，用于指明属于该概化值的个数，如{体育（3），音乐（1），电子游戏（1）}，其中体育（3）表明有 3 种体育项目，如此等等。

集合值属性可以概化为集合值属性或单值属性；若单值属性形成一个格(lattice)或“层次”，或概化有不同的概化路径，则它可以概化为一个集合值属性；进一步地，在概化集合值属性上的概化应遵循集合中每一值的概化路径。

列表值或序列值属性(list-valued or sequence-valued attribute)的概化方法类似集合值属性，所不同的是概化中要保持元素的次序。列表中的每一个值可以概化为其对应的高级别概念。或者，把一

[26] 有关这些复杂数据的简单介绍见 1.3.4。

个列表概化为一般特征，如列表长度，列表元素类型，值区间，数字值的加权平均，或删除列表中不重要的元素。一个列表可以概化为列表，集合，或单一值。

例 9．2 考虑如下有关个人教育记录的列表或序列：“((电子工程本科，U.B.C., Dec., 1990), (计算机工程硕士，马里兰大学，May, 1993), (计算机科学博士，UCLA, Aug., 1997))”。通过去掉不太重要的描述（属性）可以把该列表概化为一个新的列表，如“((本科，U.B.C., Dec., 1990)，…”，并且/或者只保留列表中最重要的元组，如“(计算机科学博士，UCLA, 1997)”。

复杂的结构值属性可以包括集合，元组，列表，树，记录等等，以及它们的组合，即其中的一个结构可以以任意深度嵌套在另一个结构中。通常，一个结构值属性可以有几种概化方法，例如：（1）保持原本结构不变，对其中的每一个属性加以概化；（2）把原结构扁平化，对扁平化的结构做概化；（3）用高级别的概念或聚集概化低级别的结构；（4）概化出原结构的类型或概貌。

9．1．2 空间和多媒体数据概化中的聚集和近似计算

聚集和近似计算(approximation)是概化的另一个重要的方面，它对具有大量值，复杂结构的空间或多媒体数据的属性尤为重要。

以空间数据为例。我们通常需要将一些具体的地理上的点概化为一些聚合区域，如根据土地的用途可概化为商业区，居民区，工业区，或农业区等。这种概化需要通过一些空间操作，如空间并或空间聚类方法，把一组地理区域加以合并。聚集和相似计算是实现这种形式概化的重要技术手段。在**空间合并（spatial merge）**中，不仅需要合并出具有同一类别的相似类型的区域，而且需要计算出总的面积，平均密度，或其它的聚集函数，这其中还要考虑忽略那些不重要的类型各异的分散的区域。其它一些空间操作，如空间并(spatial union)，空间重叠(spatial overlapping)，和空间交(spatial intersection)，它们需要把一些分散的小的区域合并为大的聚合区域，这些操作也要使用空间聚集和近似计算来完成概化处理。

例 9．3 假设我们有几片用于各种农业用途的土地，例如分别用于蔬菜，谷物和水果种植。这几片土地则可以通过空间合并操作合并或聚集为一大片的农业用地。然而这样一片农业用地中可能包含了高速公路，房屋，小的店铺，等等。如果这片土地的主要用途是农业，则其中分散的用于其它目的的区域可以忽略，即整个区域可以通过近似计算归结为一片农业区域。■

多媒体数据库包含复杂的文本，图形，图象，视频，地图，声音，音乐，和其它形式的音频/视频信息。多媒体数据通常以可变长度的位串存储，并且为便于数据的引用，数据片段要相互链接或建立多维方式的索引。

多媒体数据的概化可通过对这类数据的基本特征和（/或）一般模式的识别和抽取加以完成。抽取这类信息的方式很多。对图象数据，通过聚集和近似计算可提取的信息可以有尺寸(size)，颜色(color)，形状(shape)，质地（texture），方位(orientation)，和图象中所包含对象或区域的位置和结构。对音乐数据，其音调可以通过近似计算找出重复出现的模式片段，而其风格可以基于音调，节拍，或主要演奏乐器总结得出。对一篇文章，其概化结果可以是文章的摘要或篇章结构（例如，目录，出现频率较高的主题和索引）。

通常，从空间和多媒体数据中提取隐含存在的知识，从而对这些数据加以概化是一件具有挑战性的工作。必须把空间数据库和多媒体数据库技术（如空间数据的访问和分析技术，基于内容的图象检索和多维索引方法）与数据概化和数据挖掘技术结合起来使用，才能取得满意结果。针对此类数据的挖掘技术将在以下小节中讨论。

9．1．3 对象标识和类/子类层次的概化

“如果对象标识的作用是唯一标识对象，那么如何对其进行概化？”初看起来，对象标识似乎不可被概化。因为对象标识即使在数据结构重组后也保持不变。然而由于面向对象数据库中的对象按类组织，类又组织为类/子类层次结构，因此对象的概化可以基于相关的层次结构来完成。这样对象标识可以按如下步骤加以概化：首先，对象标识概化为对象所属的最底层子类的标识。然后子类标识可以沿类/子类层次向上概化为高一级别的类/子类标识。同样，类或子类可以顺着类/子类层次结构向上被概化为其对应的超类。

“对象的继承特性可以被概化吗？”由于面向对象数据库组织为类/子类层次，对象类的某些属性或成员并不明确地在类中说明，而是从对象的高一级别类中继承得来。有些面向对象数据库系统

允许多重继承（multiple inheritance），即当类/子类结构呈类格时，一些特性可以从不止一个超类中继承而来。对象的继承特性可以由面向对象数据库中的查询处理推导得出。从数据概化的角度看，没有必要区分数据直接来自类还是继承于超类。只要查询处理能够把有关的数据集合得到，数据挖掘处理时会对两类数据（继承的和直接的）一视同仁，并据此加以概化。

方法是面向对象数据库的重要组成部分。对象的很多特征数据可以通过应用方法导出。由于方法通常定义为计算过程/函数或一组演绎规则，因此对方法本身不存在概化问题。但是可以对由方法导出的数据加以概化。即一旦由方法导出了一组数据，则可以对这些数据加以概化。

9．1．4 类复合层次概化

一个对象的属性可以定义为另一个对象，而该对象的属性又可以定义为对象，如此便形成了类复合层次（class composition hierarchy）结构。关于类复合层次的概化可视为在一组嵌套的结构化数据（如果嵌套是递归的，其嵌套层次可能无限）之上的概化。

原则上，对复合对象（composite object）的引用要在类复合层次上遍历一段引用路径。但在大多数情况下，遍历路径越长，其初始对象与被引用的复合对象间的语义相关性越弱。例如，对象类student(学生)的属性 vehicles_owned(拥有汽车)可引用另一对象 car(轿车)，而 car 可能包含属性auto_dealer(经销商)，它可能要引用有关经销商的上司和子女方面的属性。显然，在学生和他所购买的车辆的经销商的上司的子女之间不可能存在什么有兴趣的关联。因此，一组对象上的概化必须限定在对有限的紧密相关的构成属性上的概化。即，要发现感兴趣的知识，其概化必须在类复合层次中与当前类有紧密语义关联的对象上进行，而不是那些相隔较远，语义联系较弱的对象上。

9．1．5 对象立方体的构造与挖掘

在对象数据库中，数据的概化与多维分析不适用单个对象，而是面对一组对象。由于某个类的一组对象可能共享许多属性和方法，并且每个属性和方法的概化可能使用一系列的概化操作，这时一个很重要的问题是如何使类中不同的属性和方法的概化处理相互协作利用。

“对一组对象如何处理基于类的概化？”对基于类的概化，第 5 章中有关关系数据库的特征挖掘中介绍的面向属性的归纳方法，可以加以扩展用于对象数据库中数据特征挖掘。基于概化的数据挖掘过程可视为一组在不同属性上基于类的概化操作的序列。概化可以连续进行，直到结果类中所包含的概化对象数目较少，并且可以概括为一个抽象层次较高的简练而一般的规则。为高效实现这一概化，对复杂对象类的多维属性的概化可以转化为对每一属性（维）的概化，既概化每一属性为简单值数据，并据此构造一个多维数据立方体，称为**对象立方体**。 一旦有了对象立方体，其多维分析和数据挖掘就可比照关系数据立方体的方法进行。

值得注意的是，从应用角度看，并不是总可以把一组值概化为单值数据。例如对属性 keyword（关键字），它可能包含一些有关书籍的关键字，把这样的一组关键字概化为一个单一值是没有意义的。在这里，很难构造一个包含 keyword 维的对象立方体。在下一节讨论空间数据立方体的构造时，会指出在此方面的一些进展。不过，在对象立方体构造和基于对象的数据挖掘中，如何有效处理集合值数据，仍是一个具有挑战性的研究课题。

9．1．6 对规划数据库的概化挖掘

为说明概化在复杂数据库挖掘中所起的重要作用，本节给出一个案例，具体讲述的是采用分而治之（divide-and-conquer）策略，在规划数据库（plan database）中挖掘有意义的成功行为模式。

一个规划通常由一个可变的行为序列组成。一个规划数据库，或简称为规划库（planbase），则为若干计划的集合。规划挖掘（plan mining）就是从规划库中挖掘出有意义的模式或知识。规划挖掘有很多用途，例如可从飞行数据库中发现商务乘客旅行模式，或在汽车修理数据库中的行为序列中找出有意义的模式。规划挖掘有别于序列模式挖掘，后者是指在一个很详细的层次上挖掘出大量的出现频繁的序列模式。而规划挖掘是指从规划库中提取重要的或有意义的概化（序列）模式。

下面以搭乘飞机旅行为例说明规划挖掘的过程。

例 9．4 一个乘飞机旅行规划库：假设乘飞机旅行规划库如表 9.1 所示，存储着旅客飞行的序列数据，其中每一个记录对应序列数据库中的一个行为（action），具有相同规划号的记录序列是一

个有关行为序列的计划。列 departure（出发）和 arrival（到达）给出的是相应机场的代码。表 9.2 给出了每个机场的信息。

从类似表 9.1 的规划库中可以挖掘出许多模式。例如，我们可以挖掘出从靠近大西洋的城市，途径芝加哥的 ORD 机场（ORD 可能是几个主要航线的重要枢纽），飞往中西部城市的最多的航班。注意充当航线上集散中心（如位于洛杉矶的 LAX，芝加哥的 ORD，和纽约的 JFK）的机场可以很容易地从表 9.2 中基于 airport_size（机场规模）导出。然而在数据库中可能有上百个这样的集散中心。不加选择的挖掘可能导致挖出大量的规则，它们缺乏足够的支持，没有清晰的总体画面。

表 9。1 旅行规划数据库：旅行规划库

P401

表 9。2 机场信息表

P402

也许人们要问“如何挖掘规划库？”。其实我们想挖掘的是很少数量的一般（序列）模式，它可以覆盖决大部分规划，我们可以据此序列分开来做进一步的搜索。挖掘此类模式的关键是要能把规划库中的规划概化为足够高级别的概念。如图 9。1 所示的关于飞行规划库的多维数据库模型，可以有助于此类规划的概化。由于低级别信息缺乏足够的共性来形成简明的规划，因此要按以下步骤处理：（1）基于多维模型按不同的方向概化规划库；（2）保证概化规划具有共同的，感兴趣的序列模式，它有足够的支持度；（3）据此导出高级别简洁的规划。

下面举例说明。将规划库中具有相同规划号的元组合并，得到如下的行为序列（以机场代码表示）：

ALB-JFK-ORD-LAX-SAN

SPI-ORD-JFK-SYR

。。。

这些序列看上去差异很大。然而它们可以在多个维上加以概化。当基于 airport_size 维概化时，可得到一些有意思的序列模式如 S-L-L-S，其中 L 代表大的机场（即集散中心），S 代表相对小型的机场，具体如表 9。3 所示。

基于大量飞行旅行规划的概化结果可产生一些相当一般但十分规则的模式。为此一般对概化序列使用合并（merge）和或选（optional）操作，前者是把连续相同的符号合并为一个，并使用传递闭包记号“+”来表示这同一类型行为的一个序列，而后者使用记号“[]”来表示方括弧“[]”中的对象或行为是可选的。表 9。4 给出了针对表 9。3 中的规划实施合并后的结果。

Airport_size(机场规模)

沿概念层次上升

Airline(航线)　　location（位置 ）

图 9。1 数据库的多维视图 P403

表 9。3 规划库的多维概化 P403

表 9。4 合并规划中连续相同的行为 P404

通过合并相同的行为，可以得到概化的序列模式，如模式（9。1）：

$[S]—L^{+}—[S]$　[98.5%]　　(9.1)

该模式表明有 98.5%的旅行规划具有模式$[S]—L^{+}—[S]$，其中[S]表示行为 S 是可选的，L^{+}表示一个或多个 L。换句话说，这一旅行模式含义是，首先可能由一个小的机场出发，途径一个或多个大型机场，最后达到一个大型（也可能小型）机场。

在得到一个有足够支持度的序列模式后，可以用来对原有的规划库分片。然后可以对每一个分片进一步挖掘，得到共性特征。例如，从以分片的规划库可以得出：

P404

其含义是对由小机场 x 直飞大机场 y 的飞行，x 和 y 同属一个区域的概率为 75%。■

此例展示了一种分而治之的策略，即首先通过对规划库的多维概化找出有兴趣的高级别的简练的规划序列，然后基于挖掘出的模式对规划库分片，进一步发现子规划库的有关特征。这种挖掘方法可适用许多其它应用。例如，在 Weblog 挖掘中，我们可以研究 Web 的一般访问模式，以便识别

出热点 Web 门户，和公共路径，然后再进行具体的子模式的挖掘。

规划挖掘技术可以在几个方面得到进一步的扩展。例如，类似关联规则挖掘的最小支持阈值可用于确定概化的级别并保证模式有足够的覆盖率。在规划挖掘中还可以引入其它操作，如小于（less_than）。其它方面包括从子规划库中抽取关联信息，或挖掘含有多维属性的序列模式，例如，包含机场规模和国家的两个属性的模式。这种包含多维的挖掘，首先也得先求出没个维的概化结果，然后再求得组合的序列模式。

9．2 空间数据库挖掘

空间数据库存储了大量与空间有关的数据，例如地图，遥感或医学图象数据，VLSI 芯片设计数据等。空间数据库有许多与关系数据库所不同的显著特征。空间数据库包含了拓扑和/或距离信息，通常按复杂的，多维空间索引结构组织数据，其访问是通过空间数据的访问方法，经常需要空间推理，地理计算，和空间知识表示技术。

空间数据挖掘是指对空间数据库中非明确存在的知识，空间关系，或其它有意义的模式等的提取。空间数据挖掘需要综合数据挖掘与空间数据库技术，它可用于对空间数据的理解，空间关系和空间与非空间数据间关系的发现，空间知识库的构造，空间数据库的重组，和空间查询的优化。空间数据挖掘在地理信息系统，“地理市场”（geomarketing），遥感，图象数据库探测，医学图象处理，导航，交通控制，环境研究，以及许多使用空间数据的领域中有广泛的应用价值。由于空间数据的大数据量和空间数据类型和空间访问方法的复杂性，空间数据挖掘面临的主要挑战是研究高效的空间数据挖掘技术。

“空间数据挖掘使用统计技术方法如何？”统计空间数据分析已经是空间数据分析中常用的方法。统计方法可以很好地处理数字型数据，并可以对空间现象提出现实的模型。然而它存在的问题也很多，比如统计方法通常假设空间分布的数据间是统计上独立的，但现实是空间对象间是相互关联的；大部分统计模型只有具有相当丰富领域知识和统计方面经验的统计专家才用得起来；统计方法不适用符号值，或不完整或非确定的数据，对大规模数据库其计算代价也十分昂贵。空间数据挖掘将对传统的空间分析方法加以扩展，重点解决其高效性，可伸缩性，与数据库系统的紧密结合，改进与用户的交互，以及新的知识的发现。

9．2．1 空间数据立方体构造和空间 OLAP

“可以构造出空间数据仓库吗？”是的，象关系数据一样，我们可以把空间数据集成起来构成一个数据仓库以便空间数据挖掘的处理。空间数据仓库是面向主题的，集成的，随时间变化的，并且是非易失性的空间和非空间数据的集合，用于支持空间数据挖掘和空间数据有关的决策支持处理。

下面举例说明。

例 9．5 在英属哥伦比亚（BC）分布着 3000 个气象探测器，每一个记录了指定区域的每日气温和降雨量，并将数据传送到全省的气象总站。通过建立空间数据仓库，可以支持空间 OLAP，用户可以在地图上按月，按地区，按温度和降雨量的不同组合观察气象变化模式，可以动态地沿任何一维下钻（drill down）和上卷(roll up)，发现希所望的模式，诸如“1999 年夏 Fraser 峡谷的湿热地区”。■

构造和使用空间数据仓库存在几个挑战性的问题。首先是从异构数据源和系统中把空间数据集成起来的问题。空间数据通常存储在行行色色的工业企业和政府机构中，数据格式各异。数据格式不仅有特定的结构有关（例如，基于光栅/向量空间数据，面向对象模型/关系模型，各式各样的空间存储和索引结构，等等），而且与特定厂家有关（例如，ERSI，MapInfo, Intergraph 等等）。有关异构空间数据的集成与交换已有很多的研究工作，这为空间数据集成和空间仓库构造铺平了道路。

第二个问题是如何在空间数据仓库中实现快速而灵活的联机分析处理。第二章中介绍的星型模式很适合空间数据仓库，因为它提供了简洁而有组织的仓库结构，便于 OLAP 操作。但在空间数据仓库中，维和度量都包含空间成分。

在空间数据立方体中有三种类型的维：

■**非空间维**只包含非空间数据。如例 9。5 中可构造数据仓库的非空间维温度和降雨量，因为它们每一个只包含非空间数据，其概化也是非空间的（如气温的“热”，降雨量的“湿”）。

■**空间-非空间维**是指初始数据是空间数据，但其概化值，在一定的抽象级别则是非空间的。例如，空间维 city 取自美国地图的地理数据。假设此维的一个空间值，比如西雅图，概化为字符串“pacific_northwest(西北_太平洋)”。 虽然“pacific_northwest”是一个空间概念，但不是一个空间值（因为，在此例中，它为一字符串）。因此它是一个非空间维。

■**空间-空间维**是指无论初始数据还是所有高一级别的概化数据都是空间维的。例如，equi_temperature_region 维包含空间数据，对其所有概化，如 0-5_degree(摄氏)，5-10_degree 等的地区，也是由空间数据组成。

地区维表　　BC 气象事实表　　温度维表

时间维表　　降雨量维表

图 9。2 BC_weather 空间数据仓库的星模式及对应的 BC 气象监测图 P407

空间数据立方体中有两类不同的度量。

■数字度量仅包含数字数据。例如，空间数据仓库中的一个度量可以为某地区的月收入，通过上卷可计算出按年，按郡等的收入。数字度量可近一步划分为如第二章所述的分布的(distributive)，代数的，整体的(holistic)。

■空间度量包含一组指向空间对象的指针。例如，在例 9。5 的空间数据立方体中的概化（或上卷）中，具有相同温度和降雨量的地区被组合为同一个单元，所形成的度量包含了指向这一地区的一组指针。

非空间立方体仅包含非空间维和数字化的度量。若一个空间数据立方体包含空间维但不含空间度量，其 OLAP 操作，如上钻或转轴 (pivoting)，可以以非空间数据立方体的方式实现。

“那么空间数据立方体中的空间度量会是怎样？”这一提问会带来一些有关实现方面的挑战性问题，如以下例子中所看到的。

例 9．6 例 9。5 的 BC_weather 空间数据仓库的星模式如图 9。2 所示。它包含四个维：地区，气温，时间，降雨量，三个度量：region_map（地区_地图），area（面积）,count（计数）。每一维的概念层次可由用户或专家建立，或由数据的聚类分析自动生成。图 9。3 展示了 BC_weather 仓库中每一维的概念层次。

地区名维
检测位置<市区<城市<省

时间维
小时<天<月<季度

气温维
（冷，温和，炎热）⊂ all （气温）
。。。。。。 ⊂ 冷
。。。。。。 ⊂ 温和
。。。。。。 ⊂ 炎热

降雨量维
（干燥，适量，湿）⊂ all （降雨量）
。。。。。。。 ⊂ 干燥
。。。。。。。 ⊂ 适量
。。。。。。。 ⊂ 湿

图 9。3 BC_weather 数据仓库每一维的概念层次 P408

图 9。4 不同的上卷操作后的概化地区图 P408

三个度量中，area 和 count 是数字度量，可以按非空间数据立方体的计算方法加以计算。region_map 是空间度量，表示一组指向有关地区的空间指针。由于不同的空间 OLAP 操作作用于 region_map 的不同集合，因此具有挑战性的问题是如何灵活动态地计算大量地区的合并操作。例如，两个不同的基于 BC 气象地图数据（如图 9。2）的上卷操作，可产生两个不同的地区地图，如图 9。4 所示，其中每一个都是对图 9。2 中大量小的（监测）地区的合并结果。◆

“可否预先计算出所有可能的空间合并，并存储到对应的空间数据立方体的单元中？”答案是——可能不行。与数字度量不同，其每个聚集值仅需要几个字节的空间，BC 合并后地区地图需要

上兆存储空间。这样我们面临的是一个两难的选择，即在联机计算代价和存储计算度量所需额外空间之间的选择：一方面空间聚集计算中需要预先计算来减少大量无用计算开销，另一方面聚集空间结果的大量存储负担又减弱着这一需要。

在空间数据立方体的构造中至少有三种可供选择的空间度量的计算方法。

■在空间数据立方体中收集存储有关的空间对象指针，但不执行空间度量的预计算。其实现方法可以是在有关的立方体单元中存储一个指向空间对象指针集合的指针，必要时可在空闲时执行有关空间对象的空间合并（或其它计算）。这一方法在如下的情况下不失为好的选择：当仅需要空间结果显示（即无须真的空间合并），或者在任一指针集合中没有太多可以合并的地区，或者联机空间合并计算速度很快（近来，针对快速空间 OLAP 开发出了一些高效的空间合并方法）。由于 OLAP 的结果经常用于联机空间分析和挖掘，因此人们一般还是主张将一些空间上邻接的地区预先合并，这样可以加速此类的分析。

■在空间数据立方体中预先计算并存储一个粗略近似的空间度量结果。在假定所需存储空间有限的情况下，若只需要对空间合并结果的粗略浏览或大致估算，此方法不失为好的选择。例如，最小边界矩形（MBR），可由两个点表示，它可以作为合并地区的粗略估算。这类预计算的结果较小，并可快速展示给用户。若对特定单元需要更高的精度，应用可以选择预计算质量较高的结果，或在空闲时加以计算。

■在空间数据立方体中有选择地预先计算一些空间度量。这是一个较为灵活的选择。问题是，“应选择立方体的哪一部分预先计算？”选择可以在 cuboid 级进行，即或者对选择的 cuboid 的每一个单元预计算并存储每一个可合并的空间地区，或者 cuboid 没被选择则不做任何预计算。通常 cuboid 由很多空间对象组成，因此它可能涉及很多可合并空间对象的预计算和存储，不过这其中有些可能很少用到。因此选择要在较细粒度的级别上进行：检查 cuboid 中的每一组可合并的空间对象，判定是否需要预计算。这里判定需要考虑的因素包括合并区域的实用性（如访问频率或访问优先级），共享性，以及空间和联机计算时间的代价权衡。

有了空间数据立方体和空间 OLAP 有效支持，基于概化的描述性空间挖掘，如空间特征和判别(discrimination)，可以得到和好的解决。

9．2．2 空间关联分析

“如何挖掘空间关联规则？”与事务型和关系型数据库的关联规则挖掘一样，空间数据库中也可以挖掘关联规则。空间关联规则形如 A⇒B[s%，c%]，其中 A 和 B 空间和非空间谓词的集合，s%表示规则的支持度，c%表示规则的的可信度。例如，下面是一个空间关联规则的例：

P410

此规则表明 80%靠近体育中心的学校同时也靠近公园，并且有 0。5%的数据符合这一规则。

各种各样的空间谓词可以用来构成空间关联规则。例如有关距离信息（如 close_to（临近）far_away（远离）），拓扑关系（如 intersect(交),overlap（重叠）,disjoin（分离）），和空间方位（如 left_of(左边),west_of（西部））。

由于空间关联规则的挖掘需要在大量的空间对象中计算多种空间关系，因此其代价是很高的。一种称为逐步求精的挖掘优化方法可用于空间关联的分析。该方法首先用一种快速的算法粗略地对一个较大的数据集进行一次挖掘，然后在裁减过的数据集上用代价较高的算法进一步改进挖掘的质量。

为保证裁减过的数据集能够满足后续使用的高质量挖掘算法对数据集的需要，很重要的一点是前期采用的粗略挖掘算法必须满足超集覆盖特性（superset coverage property）：即它保持了所有潜在的答案。换句话说，它应当允许假正测试（false positive test），即可以包括一些不属于结果集的数据集；不应当允许假负测试（false negative test），即它可能排除一些潜在的答案。

为了挖掘与空间谓词 close_to 有关的空间关联规则，我们可以通过以下方法首先收集一些满足最小支持阈值的候选数据：

■使用一定的近似空间计算算法，例如，可以用最小边界矩形（minimum bounding rectangle）结构（它仅涉及两个空间点，不象多边形那样有一组点）；

■计算放宽后的空间谓词，如 g_close_to，表示概化的 close_to，它包括了 close_to,touch,和 intersect 的结果。

如果两个空间对象紧密相邻，那么其最小边界矩形也一定相邻，即满足 g_close_to。但反过来

则不一定成立：如果最小边界矩形紧密相邻，两个空间对象可能相邻也可能不相邻。这样最小边界矩形剪裁对相邻来说是一个假正测试：只有通过初始测试的数据才需用计算代价更高的算法做进一步的处理。通过这一预处理，只有在近似阶段频繁出现的模式，才可能被更精细、更复杂的空间计算方法加以处理。

9．2．3 空间聚类方法

空间数据聚类是要在一个较大的多维数据集中根据距离的计算找出簇，或稠密区域。其实第八章对空间聚类方法已有全面的介绍，因为聚类分析通常考虑的就是空间数据聚类的例子和应用。因此对空间聚类感兴趣的读者可参见第 8 章。

9．2．4 空间分类和空间趋势分析

空间分类指分析空间对象导出与一定空间特征有关的分类模式，如郊区，高速公路，河流的邻接。

例 9．7 空间分类：假设需要根据平均家庭收入把地区按贫富分类。为此要找出决定一个地区分类的重要的空间上的因素。空间对象有许多特性，如有大学，有州际高速公路，靠近湖泊或海洋，等等。这些特性可用于有关的分析，找出有意义的分类模式。此类分类模式可以表示为决策树或规则的形式，如第 7 章所述。

空间趋势分析处理的是另一类问题：根据某空间维找出变化趋势。通常，趋势分析考虑的是时间上的变化，如在时间序列数据中时态模式的变化。空间趋势分析中空间替代了时间，研究的是空间上的非空间与空间数据的变化。例如，当离城市中心越来越远时，我们要分析经济形势的变化趋势，或离海洋越来越远时，气候与植物的变化趋势。对此类分析，一般要在空间数据结构和空间访问方法之上，使用回归和相关分析方法。

还有很多应用其模式是随时间和空间一起变化的。例如，高速路和城市中的交通流量是与时间和空间都有关的。气象模式也是与时间和空间紧密相关的。虽然在空间分类和空间趋势分析方面有一些研究，但时空数据挖掘的研究远不够充分。空间分类和趋势分析的方法与应用，特别是与时间有关的方法与应用，需要在未来做更进一步的研究。

9．2．5 光栅数据库挖掘

空间数据库系统通常处理的是由点，线，多边形（区域），和其组合如网络或分片（partition）组成的向量数据。这类数据的典型例子包括地图，设计图，蛋白质分子链的 3-D 排列。然而大量存在的空间数据是数字光栅（图象）形式的数据，如卫星图象，遥感数据，计算机 X 线断层摄影图象等。研究光栅或图象数据库中的数据挖掘方法是十分重要的。光栅和图象数据的挖掘方法将在下面有关多媒体数据挖掘的章节中加以介绍。

9．3 多媒体数据挖掘

"什么是多媒体数据库？"多媒体数据库是指存储和管理大量多媒体对象的数据库，如音频数据，图象数据，视频数据，序列数据，以及超文本数据，包含文本，文本标记(text markup)，和链接(linkage)。由于音频视频设备，CD-ROMs，和因特网的流行和普及，多媒体数据库系统变的日益常见。典型的多媒体数据库系统包括 NASA's EOS（地球观测系统），各种图象和音频视频数据库，人类基因数据库，和因特网数据库。

本节有关多媒体数据挖掘主要考虑的是图象数据的挖掘。序列数据挖掘的研究在 9。4 节和第 10 章有关生物信息中数据挖掘应用一节中介绍。超文本数据挖掘在 9。6 节万维网挖掘中讨论。本节介绍一些多媒体数据挖掘的方法，包括多媒体数据中的相似搜索，多维分析，分类和预测分析，多媒体数据的关联挖掘。

9．3．1 多媒体数据的相似搜索

"在多媒体数据库中搜索相似数据，既可以基于数据描述，也可以基于数据内容？"此言不错。对多媒体数据相似搜索，主要考虑两种多媒体标引和检索系统：（1）基于描述的检索系统，主要是在图象描述之上建立标引和执行对象检索，如关键字，标题，尺寸，创建时间等；（2）基于内容的检索系统，它支持基于图象内容的检索，如颜色构成，质地，形状，对象，和小波变换等。基于描述的检索若手工完成是很费力的。若自动完成，检索结果质量通常较差；例如，对图象赋予关键字可以是很灵活随意的事情。基于内容的检索使用视觉的特征标引图象并基于特征相似检索对象，这在很多应用中都是需要的。

在基于内容的检索系统中,通常有两种查询：基于图象样本的查询（image sample-based queries）和图象特征描述查询(image feature specification queries)。**图象样本查询**是指找出所有与给定图象样本相似的图象。其做法是把从样本中提取的**特征向量(feature vector)**（或特征标识（signature））与已经提取出并在图象数据库中已经索引过的图象特征向量相比较。基于这一比较结果，可以得到与样本图象近似的图象。图象特征描述查询是指给出图象的特征描述或概括，如颜色，结构，或形状，把其转换为特征向量，与数据库中已有的图象特征向量匹配。基于内容的检索的有广泛的用途，包括医疗诊断，气象预报，TV 制作，针对图象的 Web 搜索引擎，以及电子商务等。一些系统如 QBIC（Query By Image Content,按图象内容查询），同时支持样本查询和图象特征描述查询。也有系统同时支持基于内容和基于描述的检索。

人们已经提出了几种在图象数据库中基于图象特征标识的相似检索方法：

■基于颜色直方图的特征标识（color histogram-based signature）：此方法中，图象的特征标识包括了基于的图象颜色构成的颜色直方图，这其中忽略的图象的尺度（scale）或方位(orientation)。由于此方法中并不包含任何有关形状，位置，或质地的信息，因此具有相似颜色构成的两幅图象可以包含极为不同的形状或质地，这样在语义上可以是完全不相关的。

■多特征构成的特征标识（multifeature composed signature）：此方法中，图象的特征标识由多个特征的组成：颜色直方图，形状，位置，和结构。通常，可以对每一个特征定义其距离函数，然后将各结果综合导出总的结果。多维的基于内容的检索通常使用一个或几个探测特征，来搜索包含同样特征的图象。因此它可用于相似图象的搜索。

■基于小波的特征标识（wavelet-based signature）：本方法使用了图象的小波系数作为起特征标识。小波可以在一个唯一统一的框架内[27]表示形状，结构和位置等信息。这将改进效率并减少对多个特征搜索的需要（与第二种方法不同）。然而，由于此方法对整个图象只计算一个特征标识，它可能无法识别出虽包含相同对象但对象位置或尺寸不同的图象。

■带有区域粒度的小波特征标识（wavelet-based signature with region-based granularity）：此方法中，特征标识的计算和比较是在区域粒度上进行，而不是在整个图象上。这是基于如下的结论：相同的图象可能包含相同的区域，但一幅图象中一个区域可以是另一幅图象中的匹配区域的变换或伸缩的结果。因此，查询图象 Q 和目标图象 T 之间的相似计算可定义在由 Q 和 T 相匹配的区域所覆盖的两幅图象的面积碎片上进行。这种基于区域相似的搜索可以找出这样的图象，它们包含相似对象，但这些对象可以是经过变换或伸缩过的。

9．3．2 多媒体数据的多维分析

“可否为多媒体数据分析构造数据立方体？”为进行大型多媒体数据库的多维分析，可以按传统的从关系数据中构造数据立方体的方法，去设计和构造出多媒体数据立方体。多媒体数据立方体可包含针对多媒体信息的维和度量，如颜色，质地，和形状。

这里先看一个称为 MultiMediaMiner 的多媒体数据挖掘系统原型，它是在 DBMiner 系统上扩展了处理多媒体数据的功能。MultiMediaMiner 系统测试用的一个样本数据库构成如下：每个图象包含了两个描述子：特征描述子和轮廓描述子（layout descriptor）。原始图象并不直接存储在数据库中，数据库中存储的是描述子信息。描述信息包括图象文件名，图象 URL，图象类型（如 gif, jpeg, bmp, avi, mpeg 等），一组引用这一图象的 Web 页面（即父 URL），一组关键字，以及用户界面对图象和视频浏览的说明。特征描述子是一组针对每一个可视特征的向量。主要的向量是颜色向量，包含可

[27] 小波分析见 3.4.3 节的讨论。

达 512 色（R%G%B 为 8%8%8）的颜色直方图，一个 MFC（Most Frequent Color）向量，和一个 MFO（Most Frequent Orientation）向量。MFC 和 MFO 分别对五个最常用的颜色和五个最常见方位包含了五个颜色中心值和五个边方位形心。边方位为 0^0，22.5^0，45^0，67.5^0，90^0，等等。轮廓描述子包含颜色轮廓向量和边轮廓向量。无论原来尺寸大小，所有图象均被赋予一个 8%8 的栅格。对 64 个单元中的每一个的最常用颜色存储在颜色轮廓向量，对每一个单元中的每一方位的边数存储在边轮廓向量。其它尺寸的栅格，如 4%4，2%2 和 1%1，可以很容易地导出。

Image Excavator(图象挖掘器)是 MultiMediaMiner 的组成部分，它利用图象的上下文信息，如 Web 页面中的 HTML 标记，可以推导出关键字。通过遍历联机目录结构，如 Yahoo!目录，可以建立对应目录结构的关键字层次，从中可以方便地找到图象。在多媒体数据立方体中它可以作为关键字维的概念层次。

“一个多媒体数据立方体可以有什么样的维？”多媒体数据立方体可以有很多维。下面是一些这方面的例子：图象的尺寸或视频的字节；图象或视频的建立时间（或最新更新时间）；图象或视频的格式类型；按秒计算的祯序列时间；图象或视频的因特网域；引用图象或视频的页的因特网域（父 URL）；边-方位维；等等。很多数字维的概念层次可以自动加以定义。对其它维，如因特网域或颜色，可预先加以定义。

多媒体数据立方体的建立有助于多媒体数据的基于视觉内容的多维分析，和多种知识的挖掘，包括汇总，比较，分类，关联，和聚类。MultiMediaMiner 的分类模块和其输出如图 9。5 所示。

图 9。5 MultiMediaMiner 分类模块的输出 P415

多媒体数据立方体对多媒体数据的多维分析是很有用的模型。然而，必须注意到实现一个维数很大的数据立方体是极其困难的。这一困难情况对多媒体数据立方体而言尤为如此。在多媒体数据立方体中，我们要考虑颜色，方位，结构，关键字，等等多维属性。但是这其中的很多属性是集合值而不是单值。例如，一图象可能对应一组关键字。它可能包含一组对象，每一对象都对应一组颜色。在设计数据立方体时若以每一个关键字作为一维，或以每一种颜色作为一维，这将导致维数过多。若不如此，则会使对图象的建模范围过于粗糙，有限，和不精确。如何设计出既能满足效率的要求，又能有足够的表达能力的多媒体数据立方体，是个亟待研究的问题。

9．3．3 多媒体数据的分类和预测分析

分类和预测分析已经用于多媒体数据挖掘，尤其在科学研究中，如天文学，地震学，和地理科学的研究。目前图象数据挖掘应用[28]中决策树分类是最基本的数据挖掘方法。

例 9．8　以天文学家认真分类过的天空图象为训练集，我们可以构造出模型用于识别星系，星星，以及其它恒星体，基于的特性如大小，面积，密度，图象的瞬间和方位。基于这一模型可以对由望远镜和太空探测器收集的大量的图象进行处理，以发现新的天体。人们已经成功地运用这一方法去识别金星上的火山。

数据预处理在图象数据挖掘中是相当重要的，它包括数据清洗，数据聚焦（ data focusing），和特征抽取（feature extractor）。除了在模式识别中使用的标准的方法如边探测和 Hough 变换外，还可以探索新的技术，如把图象分解为特征向量，或采用概率模型处理不确定性。由于图象数据量是很大的，需要很强的处理的能力，因此需要使用并行和分布处理技术。

图象数据的分类和聚类与图象分析和科学数据挖掘有紧密的联系，因此图象分析技术和科学数据分析方法可以用于图象数据的挖掘过程。

9．3．4 多媒体数据中的关联规则挖掘

“多媒体数据中可以挖掘什么样的关联？”在图象和视频数据库中可以挖掘涉及多媒体对象的关联规则。至少包含以下三类：

■**图象内容和非图象内容特征间的关联**：如规则“如果照片的上半部分的 50%是兰色，那它很可能是天空”，属于此类，因为它把图象的内容和关键字*天空*关联在一起。

[28] 分类方法，包括基于决策树的分类，可参见第七章的讨论。

■**与空间关系无关的图象内容的关联**：如规则“如果一幅图片包含两个兰色正方形，那么它很可能也包含一个红色圆形”，属于此类，因为关联考虑的都是图象内容。

■**与空间关系有关的图象内容的关联**：如规则“如果一个红色矩形是在两个黄色正方形之间，那么很可能在下面存在一个大的椭圆形对象”，属于此类，因为它把图象中对象与空间关系关联在一起。

要挖掘多媒体对象间的关联，我们可以把每一个图象看作一个事务，从中找出不同图象间出现频率高的模式。

“多媒体数据库中的关联规则挖掘与事务数据库中的有什么不同？”有一些细微差异。首先，一个图象可以包含多个对象，每个对象可以有许多特征，如颜色，形状，结构，关键字，和空间位置。这样可能存在大量的关联。在很多情况下，两个图象的某个特征在某一分辨率级别下是相同的，但在更细的分辨率下则是不同的。因此，需要一种改进分辨率（progressive resolution）的逐步求精的方法。即，我们可以首先在一个相对较粗的分辨率下挖掘出现频率高的模式，然后对那些通过最小支持度阈值的图象做进一步的更细分辨率下的挖掘。这是由于在粗一级分辨率下不频繁出现的模式，不可能在细一级的分辨率下出现。这种多级分辨率挖掘策略极大地降低了总体数据挖掘的代价，而又不损失数据挖掘结果的质量和完整。由此得出一种在大规模多媒体数据库中挖掘关联的高效的方法论。

第二，由于包含多个重复出现对象的图片是图象分析中的一个重要特征，在关联分析中不应忽视同一对象的重复出现问题。例如，一幅包含两个金色圆形的图片与只有一个圆形的图片是截然不同的。这与事务数据库中的情形完全不同，如一个人买一加仑牛奶和买两加仑通常都视为“buy_milk(买牛奶)”这同一事实。因此多媒体关联及其度量的定义，如支持度和可信度，都需要做相应的调整。

第三，在多媒体对象间通常存在着重要的空间关系，如之上，之下，之间，附近，左边等。这些特征对挖掘对象关联和相关性非常有用。空间关系，与其它基于内容的多媒体特征，如颜色，形状，结构和关键字等，一起可以形成有趣的关联。这样空间数据挖掘方法和拓扑空间关系特性对多媒体挖掘显得十分重要。

9．4 时序和序列数据的挖掘

“什么是时序（time_series）数据库？什么是序列（sequence）数据库？”**时序数据库**是指有随时间变化的值或时间组成的数据库。值通常是在等时间间隔测得的数据。很多应用中时序数据库很普遍，如股票市场的每日波动，动态产品加工过程，科学实验，医学治疗，等等。时序数据库也是一种序列数据库。然而序列数据库是指由有序事件序列组成的数据库，它可以有时间标记，也可以没有。例如，Web 页面遍历序列是一种序列数据，但可能不是时序数据。本节将介绍时序数据库和序列数据库的挖掘的几种重要内容，包括趋势分析，相似搜索，序列模式挖掘和与时间有关数据的周期模式挖掘。

9．4．1 趋势分析

时序变量 Y，表示股票市场中一股票的每日收盘价，它可以表示为时间 t 的函数，即 Y=F（t）。此函数可以图示为一个时序图，如图 9。6 所示，它描述了一个点随时间变化的情况。

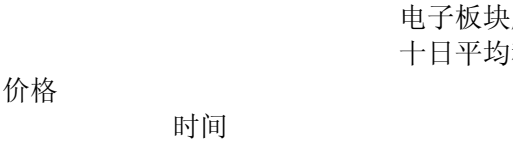


图 9。6 时序数据：电子板块的股票价格随时间的变动 P418

“如何处理时序数据？”目前一般有四种主要的变化成分用于特化时序数据：

■长期或趋势变化(trend movement)：它用于反映一般变化方向，其时序图是在较长时间间隔上的数据变化。这种变化反映为一种趋势曲线，或趋势线。例如，图 9。6 的趋势曲线由图中的虚线表示。确定趋势曲线或趋势线的典型方法包括加权移动平均方法和最小二乘法，详见下面的讨论。

■循环运动或循环变化(cyclic movement or cyclic variations)：主要指循环性，即趋势线或曲线在

长期时间内呈摆动迹象，它可以是也可以不是周期性的。即在等时间间隔之间，循环不需要沿着同样的模式演进。

■季节性运动或季节性变化(seasonal movements or seasonal variations)：它反映的是每年都重复出现的事件，如情人节前巧克力和鲜花会的销量突然上升，或在圣诞节节前储藏商品的销售会突然增加。换句话说，季节性运动是指同一或近似同一的模式，在连续几年的有关月份期间重复出现。

■非规则或随机变化（irregular or random movements）：它反映的是随机或偶然事件零星时序变化，如劳工需求，洪水，或企业内发生的人事变动等。

以上有关趋势的，循环的，重复的，和非规则的运动，可以分别用变量 T，C，S，I 表示。时序分析也可以指将时序分解为以上四个基本运动的分析。时序变量 Y 可以表示为四个变量的积（即 Y=T%C%S%I），或四变量之和。其选择通常是凭经验的。

“对 Y 给定一组值（即，$y_1,y_2,y_3,\ldots$），如何确定数据的趋势？”确定趋势的常见方法是按如下的算术平均式序列，计算 **n 阶的移动平均值（a moving average of order n）**：

P419

移动平均可以降低数据集中的变化总量。因此用移动平均替代时序，可以减少不希望出现的波动，故它也称为平滑的时序(smoothing of time series)。如果在序列（9。3）中使用加权算术平均，则称为 **n 阶的加权移动平均(weighted moving average of order n)**。

例 9．9 给定九个值的序列，我们可计算出 3 阶的移动平均，及其权重为（1，4，1）的加权移动平均。计算结果可以记录在按表格中，其中移动平均的每一个值是相邻三值平均，加权移动平均的每一个值是相邻的三值加权平均。

初始数据：
3 阶的移动平均：
3 阶的加权（1，4，1）移动平均：

第一个加权平均值计算为 　（P420）。加权平均通常对中间元素赋予较大的权重，以便抵消平滑效果。◆

移动平均会丢失系列中的头尾数据，由此有时会生成在原始数据中不会出现的循环或其它变化趋势，并且它可能受一些极端数据的影响。对极端数据的影响，可通过采用如图 9.9 所示的适当权重的加权移动平均的方法降低其负面影响。

采用适当的阶数的加权移动平均，可以消除数据中的循环，重复和非规则的模式，而只保留趋势变化。

“还有其它计算趋势的方法吗？”答案是肯定的。其中之一是所谓的**徒手法**（free-hand method），它是基于用户的判断画一根近似曲线或直线去吻合一组数据。这一方法代价很大，且只对大规模数据挖掘可靠。另一种是**最小二乘法**，其中以最吻合的曲线 C 作为最小二乘曲线，即曲线具有 P420 的最小值，其中 d_i 的偏差或误差是指点（x_i, y_i）的值 y_i 与对应曲线 C 的值之间的差值。

“对季节性波动，是否有调整数据的方法？”在许多商业交易中，存在预期的季节性波动，如圣诞节期间的旺销。因此，对趋势和循环数据分析，很重要的一点是识别此类重复性变化，并对数据“反季节化（deseasonalize）”。为此，引入季节指数（seasonal index）的概念，用一组数字表示一年中某些月份某变量的相关值。例如，如十月，十一月，十二月的销售分别是全年平均月销量的 80%，120%，140%，那么 80，120，140 就是本年度的季节指数。若原始的每月数据由对应的季节指数去除，其结果数据被称为是反季节化的，或者是对季节变量调整过的。

反季节化数据可以针对趋势做进一步的调整，即按对应的趋势值去除这些数据。而且，合适的移动平均可以平滑掉的非规则的变化，而只剩下循环变化做进一步的分析。如果循环呈现周期或近似周期，则可以按引入季节性指数的同样方法引入**循环指数**（cyclic index）。

最后，非规则或随机变化可以通过针对趋势，季节和循环变化的数据调整，加以估计。一般地，小偏差出现的频率高，大偏差出现的频率低，遵从正态分布。

在实际中，首先掌握时序图和量化估算出长期趋势，重复变化，和循环变化的规律的，将获益匪浅。原因是它有助于选择合适的方法去做分析和有助于全面理解结果数据。

通过对趋势，循环，季节和非规则成分的运动的系统分析，使人们可以在较合理的情况下，制定出长期或短期的预测（即预报时序）。

9．4．2 时序分析中的相似搜索

“什么是相似搜索（similarity search）？”通常数据库查询是要找出符合查询的精确数据，相似搜索与之不同，它是找出与给定查询序列最接近的数据序列。**子序列匹配**（subsequence matching）是找出与给定序列相似的所有数据序列，而**整体序列匹配**（whole sequence matching）是找出彼此间相似的序列。对金融市场的分析（如股票数据分析），医疗诊断（如心电图分析），和科学与工程数据库（如能量消耗分析）等，时序分析中的相似搜索大有用武之地。

数据变换(data transformation)：从时间域(time domain)到频率域(frequency domain)

对时序数据的相似分析，通常采用欧氏距离作为相似计算的依据。

“那么为什么需要变换数据？”许多信号分析的技术需要数据来自频率域。因此，有关距离的正交变换经常用于从时间域到频率域的数据变换。通常，使用独立于数据的变换，其变换矩阵是预先确定的，与输入数据无关。两个常见的独立于数据的变换是离散傅立叶变换（DFT）和离散小波变换（DWT）[29]。由于在时间域中两个信号的距离与频率域中欧氏距离类似，所以 DTF 可以出色发挥，在头系数（first few coefficients）表现突出。通过仅保持 DFT 的头几个（即，“最强的”）系数，我们可以计算出实际距离的下限。

“一旦数据经过变换，比如 DFT，如何进行相似搜索？”为提高访问效率，可以用头几个傅立叶系数构造一个多维索引。当相似查询提交给系统，可以利用索引检索出与查询序列保持一定最小距离的序列。通过计算时间域序列和未满足查询的序列间的实际距离，可以进行必要的后处理(postprocessing)。

“子序列匹配如何进行？”对子序列匹配，每一序列首先被分割为长度为 w 的窗口“片段”。每个序列影射为特征空间的中一个“线索（trail）”。对子序列分析，把每个序列的线索划分为“子线索(subtrail)”，每一个由最小边界矩形表示。利用多片组装算法（multipiece assembly algorithm）可以搜索更长的匹配序列。

增强相似搜索方法，处理偏移和振幅中的间隙(gap)和差异(differences)

大部分实际应用并不一定要求匹配的子序列在时间轴上完全一致。换句话说，若子序列对具有同样的形状，但在序列内存在间隙或在偏移或振幅（offsets or amplitudes） 中存在差异，我们也可以认为它们是匹配的。这在许多相似序列分析中尤为有用，如股票市场的分析和心电图分析。

“如何改进相似搜索，使其能够在存在这种差异的情况下仍能判断其相似性？”一种改进的相似模型，是允许用户或专家说明一些参数，如滑动窗口(sliding window)尺寸，相似范围的宽度，最大间隙，匹配片段（matching fraction），等等。如图 9.7，给出了两个时间序列，其中的间隙被移去。结果序列按偏移转移和振幅调整加以规范处理，使得振幅和偏移量不同的序列得以匹配。当一个子序列与另一个子序列处于某一宽度$\in$的范围内（其中$\in$是一个小的数字，可由用户或专家指定），这两个子序列是相似和可匹配。两序列是相似的，当它们之间存在足够多的非重叠的符合时间次序的相似子序列对。

基于以上讨论，能够处理存在间隙和偏移与振幅差异的相似搜索的执行步骤如下：

1．原子匹配（atomic matching）：找出所有无间隙的较小相同窗口对。
2．窗口结合(window stitching)：把相同窗口结合，形成大的相似子序列对，其中允许在原子匹配间有间隙。
3．子序列排序(subsequence ordering)：线性排列子序列匹配，以判定是否存在足够大的相似片段。

通过以上的处理步骤，可以求得形状相似但有间隙或在偏移或振幅中有差异的相似序列。

P423

（1）原始序列
序列 S
序列 T

（2）移去间隙
序列 S
序列 T

（3）偏移迁移
序列 S

（4）振幅放大
序列 S

[29] 离散傅立叶变换和小波变换见 3。4。3 节的讨论。

序列 T　　序列 T

（5）子序列匹配

图 9。7 在时序数据中的子序列匹配：原始序列形状相同，但需要调整以处理存在于间隙，偏移，振幅中差异。这些调整允许子序列在一定宽度∈的范围内匹配。

相似搜索的索引方法

“是否存在高效的实现方法？”为在大型数据库中改进相似搜索的效率，人们提出了各种了索引的方法。例如，R-树，R*-树用于存储最小边界矩形以加速相似搜索。另外，提出了∈kdB 树用于在高维点上提高空间相似连接的速度，还提出了后缀树（ suffix tree）等。

有关时间序列的查询语言

“如何给出相似搜索的请求？”设计和开发功能强大的查询语言，以利时间序列的相似查询说明，是一件很重要的事情。时间序列查询语言应该不仅能够描述简单的相似查询，如“找出与给定子序列 Q 相似的所有序列”，而且还能描述复杂的查询，如“找出与类 A 中某序列相似，但与类 B 中的任一序列不相似的所有序列”。而且，它应能够支持各种类型的查询，如范围查询（range query），所有对查询（all-pair query），和最邻近查询(nearest neighbor　query)等。

另一种有意思的时间序列查询语言是形态定义语言（shape definition language）。它允许用户以人类可读的序列串或宏的形式定义或查询时间序列的总体形状，其中忽略一些细节。

例 9．10 模式 up,Up,UP 可用于说明坡度上升的程度。宏，如 spike 可用于表示一个序列如(SteepUps（上陡）,flat（平坦）,SteepDown（下陡）)，其中 SteepUps 定义为（{Up,UP},{UP,UP},{Up,UP}），其含义是一个 SteepUps 由三个陡峭的斜坡组成，每一个或对应 Up，或 UP。SteepDown 的定义类似。□

类似的形态定义语言对序列相似搜索可以提高用户在说明形态查询方面的灵活性。

9．4．3 序列模式挖掘

“什么是序列模式挖掘？”**序列模式挖掘**（sequence pattern mining）是指挖掘相对时间或其它模式出现频率高的模式。一个序列模式的例子是“九个月以前购买奔腾 PC 的客户很可能在一个月内订购新的 CPU 芯片”。由于很多商业交易，电传记录，天气数据，和产品处理都是时间序列数据，在针对目标市场，客户吸引，气象预报等的数据分析中，序列模式挖掘是很有用途的。

序列模式挖掘的情形和参数

许多有关序列模式挖掘的研究主要针对符号模式（symbolic pattern），因为数字曲线模式通常属于统计时序分析中的趋势分析和预测范畴，见 9.1 节讨论。

对序列模式挖掘，存在一些参数，其取值如何，将严重影响挖掘结果。第一个参数是时间序列的持续时间（duration）T。持续时间可以是数据库中的整个序列，或由用户选择的一个子序列，如 1999 年。序列模式挖掘因此是限制在特定的持续时间内的挖掘。持续时间还可定义为一组分割的序列，如每年，或股票大跌后的每周，或火山喷发前后的每两周等。在这些情形中，可以发现周期模式(periodic pattern)。

第二个参数是事件重叠窗口（event folding window），w。在指定时间周期内出现的一组事件，可以视为某一分析中一起出现的事件。若 w 设为与持续时间 T 相同的值，则找出的是与时间无关的模式——即是一些基本的相关模式，如“在 1999，购买 PC 的顾客也购买数字相机”（这里不反映先购买那一个）。若 w 取值为 0（即，没有事件序列折叠），则找出的序列模式中的每个事件出现在不同的时间值，如“购买了 PC 的顾客，可能接着买内存芯片，再买 CD-ROM”。若 w 设为之间的值（如同一月内发生的交易，或 24 小时滑动窗口内），则考虑同一周期内出现的交易，分析中序列被折叠。

第三个参数是被发现的模式中时间之间的时间间隔（interval）int。此参数可取如下的值：

■int=0：表示没有时间间隔；即，所找出的是严格连续的序列，如序列模式 $a_{i-1}a_ia_{i+1}$，其中 a_i 在时间 i 出现的事件。事件折叠窗口 w 同此情形。例如，如事件折叠窗口设为一周，将找出连续几周内出现的模式。DNA 分析通常需要发现没有间隔的连续序列。

■min_interval[int[max_interval：表示要找出最小间隔为 min_interval 而最大间隔为 max_interval 的模式。例如，模式“如果某人租了影片 A，很可能 30 天内租影片 B”蕴涵 int≤30(天)。

■int=c≠0：用户可以找出具有确定间隔 int 的模式。例如，查询“每当道琼斯下降超过 5%，两天后会发生什么事情？”，将搜索间隔 int=2（天）的序列模式。

用户可以在要挖掘的序列模式上指定约束，方法是提供“模式模板”，其形式可以是系列片段（serial episode），并行片段(parallel episode)，或正则表达式。系列片段是一组在总序列中出现的事件，而并行片段是一组与出现次序无关紧要的事件。设记号（E,t）表示发生在时间 t 的事件类型 E。考虑数据（A，1），（C，2），（B，5），具有宽度为 2 的事重折叠窗口，其中系列片段 A→B 和并行片段 A&B 都出现此数据中。用户还可以正则表达式说明约束，如（A|B）C*（D|E），它表示用户希望查出这样的模式：事件 A 和 B 先出现（但它们二者的出现次序无关紧要），之后是一个或一组事件 C，再之后是事件 D 和 E（D、E 无先后）。注意，其它事件可以出现在正则表达式说明的序列中。

序列模式挖掘的方法

关联规则挖掘中采用的 Apriori 特性可以用于序列模式的挖掘，因为若长度为 k 的序列模式是非频繁的，其超集（长度为 k+1）不可能是频繁的。因此，序列模式挖掘的大部分方法都采用了类 Apriori 算法的变种，虽然所考虑的参数设置和约束都有所不同。另一种挖掘此类模式的方法是基于数据库投影的序列模式生长(database project based sequential pattern growth)技术，类似无候选的频繁模式挖掘(frequent pattern)方法，频繁模式增长法（FP-growth）。

9．4．4 周期分析

“什么是周期分析？”周期分析（periodicity analysis）是指对周期模式的挖掘，即在时序数据库中找出重复出现的模式。周期模式可以应用于许多重要的领域。例如季节，潮汐，行星轨道，每日能源消耗，每日交通模式，和每周特定时间的所有 TV 节目。、

如前一节所指出的，周期模式挖掘可视为以一组分片序列为持续时间的序列模式挖掘，如每年，某事件出现前后的每一位置等等。

周期模式挖掘的问题可分为三类：

■挖掘**全周期模式（full periodic pattern）**，这里每一时间点都影响着（精确或近似）时序上的循环行为。如一年中的每一天都对一年中的季节循环起着作用。

■挖掘**部分周期模式(partial periodic pattern)**，它描述在部分时间点的时序周期。例如，Sandy 在平日的早晨 7：00 至 7：30 阅读纽约时报，但在其它时间则没有什么规律。部分周期是一种比全周期较为松散的形式，在现实世界也更为常见一些。

■挖掘**循环或周期关联规则(cyclic or periodic association rule)**，这种规则是周期出现的事件的关联规则。周期关联规则的一个例子是“基于每天的营业记录，若周末下午茶在 3：00-5：00pm，则晚餐最佳营业时间为 7：00-9：00。

全周期分析的技术已在信号分析和统计中得到研究。如 FFT（快速傅立叶变换）方法已广泛用于时间域到频率域的数据转换，以便于此类分析。

“全周期模式挖掘方法可否用于部分周期模式挖掘？”高效的部分周期模式挖掘已在最近的数据挖掘中研究中得到重视。全周期模式挖掘的大部分方法相对于部分周期模式挖掘或者不适用，或者代价太大，原因是部分周期模式在同一周期内混杂有周期事件和非周期事件。例如，FFT 不能用于部分周期挖掘，因为它把时序看作不可分离的数据流。一些周期探测方法不能覆盖部分周期模式，除非部分模式的的周期，长度，和选时（timing）是明确说明的。以新闻阅读为例，我们需要明确说明诸如“以 24 小时为一周期，找出 Sandy 在 7：00 以后半小时内的有规律的活动”。把此类方法简单适用于部分周期模式的挖掘问题是不足取的，因为它需要处理的是周期，长度，和选时的大量组合。

有关部分周期模式和循环关联规则挖掘的大部分研究都应用了 Apriori 特性启发式和采用了变通的 Apriori 挖掘方法。在序列模式和周期模式挖掘中可以引入约束。Apriori 特性，各种改进的 Apriori 算法，以及约束挖掘（mining constraint）的使用在第六章中已有讨论。、

9．5 文本数据库挖掘

前面对数据挖掘的大部分研究主要针对的是结构数据，如关系的，交易性，和数据仓库数据。然而在现实世界中，可获取的大部分信息是存储在文本数据库（或文档数据库）中的。它由来自各种数据源（如新闻文章，研究论文，书籍，数字图书馆，电子邮件消息，和 Web 页面）的大量文档组成。由于电子格式的信息量的飞速增长，如电子出版物，电子邮件，CD-ROM，和万维网（它也可以被视为一个巨大的，互连的，动态的数据库）等，文本数据库得到迅速的发展。

文档数据库中存储最多的数据是所谓的半结构化数据（semistructure data），它既不是完全无结构的也不是完全结构的。例如，一个文档可能包含结构字段，如标题，作者，出版日期，长度，分类，等等，也可能包含大量的非结果化的文本成分，如摘要和内容。在最近数据库领域研究中已有大量有关半结构化数据的建模和实现方面的研究。而且，信息检索技术，如文本标引(text index)方法，已经被用来处理非结构化文档。

传统的信息检索技术已不适应日益增加的大量文本数据处理的需要。典型的，大量文档中只有很少一部分与某一个体或用户相关。而不清楚文档中的内容，就很难形成有效的查询，从数据中分析和提取有用信息。用户需要有关的工具完成不同文档的比较，以及文档重要性和相关性排列，或找出多文档的模式或趋势。因此，文档挖掘就成为数据挖掘中一个日益流行而重要的研究课题。

9．5．1 文本数据分析和信息检索

“什么是信息检索？”**信息检索**（IR）是与数据库系统并行发展很多年的一个领域。与数据库系统不同，信息检索研究的主要不是结构数据的查询和事务处理的问题，而是研究的大量文本文档的信息组织和检索。典型的信息检索问题是基于用户的输入（如关键字或样例文档）定位相关的文档。典型的信息检索系统有联机图书馆目录系统和联机文档管理系统。

由于信息检索和数据库系统处理的是不同类型的数据，数据库中的一些常见问题并不出现在信息检索系统中，如并发控制，恢复，事务管理，和更新。同样信息检索中的一些问题也不出现在传统的数据库系统中，如非结构化文档，基于关键字的近似搜索，和相关表示等。

文本检索的基本评价

“假设一个文本检索系统按一定查询格式的输入检索出了一组文档。那么我们如何判断该系统输出的结果是‘准确’或‘正确’的？”设与某查询相关的一组文档记为{Relevant}，由系统检索出的一组文档记为{Retrieved}。既相关又被检索出的一组文档记为{Relevant}∩{Retrieved}，如图 9。8 所示的 Venn 图。这里有两个判断文本检索质量的基本度量：

■查准率（precision）：它是所检索到的实际与查询相关的文档的百分比（即，反映“正确性”）。其形式定义如下

P429

■查全率(recall)：它是与查询相关的，并且实际被检索出的文档的百分比。其形式定义如下

P429

相关文档　　相关并被检索　　检索到的文档

所有文档

图 9。8 相关文档集和被检索文档之间的关系

基于关键字和基于相似的检索

“信息检索有哪些方法?”大部分信息检索系统都支持基于关键字（keyword-based）和/或相似(similarity based)检索。在**关键字检索**中，文档被看作字符串，可以用一组关键字加以识别。用户提供一个关键字或一组由关键字构成的表达式，如“汽车 and 修理店”，“茶 or 咖啡”，“数据库系统 but not Oracle”等。好的信息检索系统在处理此类查询时应考虑其同义词问题(synonymy problem)。例如，给定关键字 car（汽车），其同义词 automobile 和 vehicle 同样应加以考虑在内。基于关键字的检索有两个主要的困难问题。第一个是同义词问题：一个关键字如软件产品，可能并不在文档中出现，但文档确实是与软件产品有关的。第二个是多义词问题(polysemy problem)：同一个关键字，如

挖掘，可能在不同的上下文中有不同的含义。

相似检索是指基于一组共同的关键字找出相似的文档。此类检索的输出应基于相关度，其相关度的计算基于与关键字近似性，关键字的出现频率等。注意，在很多情况下，是很难给出一组关键字之间的精确的相关度计算结果，如数据挖掘和数据分析两词之间的距离。

“基于关键字和基于相似的信息检索方法的工作原理如何？”文本检索系统通常用到称为无用词表(stop list)的一组词来加工文档。无用词表被认为是一组“无关的”词。例如，a, the, of , for, with 等都是无用词，虽然其出现频率都很高。文档不同，无用词表也会不同。例如，数据库系统可能在报纸中是一个重要的关键字。但对一组有关数据库系统会议上的研究论文，可能被视为无用词。

一组不同的词可能有相同的词根。文本检索系统需要识别出具有相同语法构成的词，收集每组词的公共词根。例如，drug, drugged, drugs，有共同的词根 drug ，这一组词可视为是同一个词的不同变种。

“如何加工文档使其方便信息检索？”初始有一组文档 d 和一组词 t，我们可以把每一个文档视为是一个在 d 维空间$\Re^t$上的一个向量 v。v 的第 j 个坐标是一个数字，反映的是第 j 个词与所给文档的关联度：当文档不含有该词时，其值设为 0，否则设为一个非零值。有很多方法可以定义向量中的值。例如，可以简单定义只要第 j 个词出现在文档中则 v_j=1，或者 v_j 设为词频值(term frequency)，即词 t_j 在文档中出现的次数，或是**相对词频(relative term frequency)值**，即词频相对于所有词在文档中出现的总次数。

例 9．11 词频矩阵：在表 9。5 中，每一行表示一个词，每一列表示一个文档向量，其中每一项，frequency_matrix(i,j)，表示词 t_i 在文档 d_j 中出现的次数。□

“那么如何确定两个文档是相似的？”由于相似文档具有相似的相对词频，因此我们可以基于频率表中的相对词频，计算一组文档的相似性，或文档与查询（一般为一组关键字）的相似性。

表 9。5 表示每一文档词频的词频矩阵 P431

词/文档

此外，还有很多计算文档相似的方法。具有代表性的是余弦计算法(cosine measure)，如下面的定义。设 v_1 和 v_2 为两个文档向量。其余弦相似度定义为：

P431

其中内积 $v_1 \bullet v_2$ 为标准向量点积，定义为 P431，分母中的$|v_1|$定义为 P431。

“如何使用相似矩阵？”通过使用文档的相似矩阵，我们可以构造出文档的基于相似性的标引。基于文本的查询于是可以表示为向量，用于在文档中搜索接近的文档。然而对仍何一个非平凡的文档数据库，词的数目 T 和文档数目 D 通常很大。如此的高维数会导致低效的计算，因为结果频率表大小为 T×D。而且高维数还会导致非常大的稀疏矩阵，这样会增加寻找词之间关系的难度（如同义词）。为克服这些问题，人们提出了潜在语义标引（latent semantic indexing）方法，可以有效降低分析用的频率表的大小。

潜在语义标引

“潜在语义标引是如何减少词频率矩阵大小的？”潜在语义标引方法使用了矩阵理论中的著名的技术奇异值分解（singular value decomposition，SVD)。给定 T 个词和 D 个文档的词频矩阵 T×D，SVD 方法删除一些行和列，使矩阵减小为 K×K，对大量文档，K 一般取值为几百（如 200）。为使信息丢失最小化，只忽略频率矩阵中意义最小的部分。

通过 SVD 的矩阵变换方法是相当复杂的，这超出了本书的讨论范围。但是，一些著名的 SVD 算法从一些软件包可免费获得，如 MALTAB（www.mathworks.com）和 LAPACK(www.netlib.org/lapack++).

一般地，潜在语义标引方法包括如下的基本步骤：

1．建立一个词频矩阵，frequent_matrix。
2．计算 frequent_matrix 的奇异值分解，方法是把矩阵分裂为三个小的矩阵 U，S，V，其中 U 和 V 是正交矩阵（即 $U^TU=I$），S 是奇异值的对角矩阵。矩阵 S 大小为 K×K，是原频率矩阵的消减矩阵。
3．对每一文档 d，用排除了 SVD 中消除的词的新的向量替换原有的向量。
4．保存所有向量集合，用高级多维索引技术为其创建索引。

通过奇异值分解和多维索引，变换后的文档向量可用于比较两文档的相似性或找出查询的头 N 个匹

配结果。

其它文本检索标引技术

有几个较为流行的文本检索标引技术，包括倒排索引（inverted indexes）和特征文件（signature file）。

倒排索引（inverted index）是一种索引结构，它包含两个哈稀索引表或两个 B+树索引表：document_table(文档表)和 term_table(词表)，其中

■ document_table 由一组文档记录组成，它包含两可字段：doc_id 和 posting_list，其中 posting_list 是出现在文档中的词（或指向词的指针）的列表，按一定的相关度排序。

■term_table 由词记录组成，每个记录包含两个字段：term_id 和 posting_list，其中 posting_list 是包含该词的文档标识的列表。

通过这种组织，可以很容易地回答类似查询“找出与给定词集相关的所有文档”，或“找出与指定文档相关的所有的词”。例如，要找出与一组词相关的所有文档，可以首先找出每个词在 term_table 中的文档标识列表，然后取其交集，结果是一组相关文档。实际中倒排索引被广泛地使用。它易于实现，但不能满足对同义词和多义词的处理。posting_list 可以非常地长，使得存储开销很大。

特征文件(signature file)是一个存储数据库中每一文档的特征记录的文件。每个特征有固定 b 位长度，用于表示词汇。如下是一种简单的编码模式。文档特征的每一位初始为 0。若某一位对应的词出现在文档中，则该位置为 1。若特征 S_1 与 S_2 的特征位一一对应，则 S_1 匹配 S_2。由于词的数量太大，所以可以把多个词对应到一位，以减少位串的长度。这种多对一映射会增加搜索开销，因为匹配查询特征的的文档，不必包含查询的关键字。文档要经过查找，分析，词根处理，和检查。可以通过一些方法加以改进，如首先经过词频分析，词根处理，和无用词的过滤，然后使用哈稀索引技术和双重编码技术将词表编码为位串表示。但是，多对一映射的问题仍然存在，这也是这一方法的主要缺点。

9．5．2 文本挖掘：基于关键字的关联和文档分类

“什么是文本数据库中的关联挖掘？可否生成文档分类模式？”以下小节将讨论这些问题。

基于关键字的关联分析

“什么是基于关键字的关联分析？”此类分析首先收集经常一起出现的关键字或词汇，然后找出其关联或相互关系。

与文本数据库中大多数分析一样，关联分析首要对文本数据进行分析，词根处理，去除无用词等预处理，然后调用关联挖掘算法。在文档数据库中，每一文档被视为一个交易，文档中的关键字组可视为交易中的交易项。即数据库可表示为

P433

文档数据库中关键字关联挖掘的问题就变成交易数据库中交易项的关联挖掘，这在第六章中已有过讨论。

注意一组经常连续出现或紧密相关的关键字可形成一个词或词组。关联挖掘有助于找出复合关联（compound associate），即领域相关的词或词组，如[Standford University]或[U.S.,总统，比尔，克林顿]，或非复合关联，如[美圆，参股（shares），交易，总额，佣金，证券]。基于这些关联的挖掘称为“词级（term level）关联挖掘”（相对应的是字级的挖掘）。词的识别和词级关联挖掘在文本文本分析中有两个优点：（1）词和词组被自动标记，无须人去标记文档；（2）挖掘算法的执行时间和无意义的结果将极大减少。

利用这种词和词组的识别，词级挖掘可以用于找出词或关键字间的关联。一些用户可能喜欢从给定关键字或词组中找出关键字或词对之间的关联，而有些用户可能希望找出一起出现的最大词集。因此，基于用户挖掘的需要，可以使用关联挖掘或最大模式挖掘算法。

文档分类分析

自动文档分类是一种重要的文本挖掘工作，由于现在存在大量的联机文档，自动对其分类组织以便于对文档的检索和分析，是至关重要的。

“如何进行自动文档分类？”一般的做法如下：首先，把一组预先分类过的文档作为训练集。然后对训练集进行分析以便得出分类模式。这种分类模式通常需要一定的测试过程，不断的细化。

之后就用这些导出的分类模式对其它联机文档加以分类。

这一处理过程与关系数据的分类相似。但还是存在本质的区别。关系数据是结构化的：每个元组定义为一组属性值对。例如，元组{sunny, warm, dry, not_windy, play_tennis}，值“sunny”对应属性 weather_outlook，“warm”对应属性 temperature，等等。分类分析判断哪一个属性值对在决定一个人是否要打网球这一事情上，具有最大影响力。文档数据库则不是结构化的，它没有属性值对的结构。与一组文档相关的关键字并不能用一组属性或维刻化。因此，通常面对关系数据的分类方法，如决策树分析，并不使用对文档数据库的分类。

对文档分类的有效方法是基于关联的分类，它基于一组关联的，经常出现的文本模式对文档加以分类。基于关联的分类方法处理过程如下：首先，通过简单的信息检索技术和关联分析技术提出关键字和词汇。其次，使用已经有的词类，如 WordNet，或基于专家知识，或使用某些关键字分类系统，可以生成关键字和词的概念层次。训练集中的文档也可以分类为类层次结构。然后，词关联挖掘方法可以用于一组发现关联词，它可以最大化地区分一类文档与另一类文档。这导致了对每一类文档，相关有一组关联规则。这些分类规则可以基于其出现频率和识别能力，加以排序，并用于对新的文档的分类。此中基于关联的文档分类方法已经证明是有效的。对 Web 文档分类，可以利用 Web 页面的链接信息，帮助文档类的识别。Web 链接分析的方法在下一节中讨论。

9．6 Web 挖掘

万维网目前是一个巨大，分布广泛，全球性的信息服务中心，它涉及新闻，广告，消费信息，金融管理，教育，政府，电子商务，和许多其他信息服务。Web 还包含了丰富和动态的超链接信息，以及 Web 页面的访问和使用信息，这为数据挖掘提供了丰富的资源。然而基于以下的分析，Web 对有效的资源和知识发现还是具有极大的挑战性。

■对有效的数据仓库和数据挖掘而言，Web 似乎太庞大了。Web 的数据量目前以兆兆字节（terabytes）计算，而且仍然在迅速地增长。许多机构和社团都在把各自大量的可访问信息置于网上。这使得几乎不可能去构造一个数据仓库来复制，存储，或集成 Web 上的所有数据。[30]

■ Web 页面的复杂性远比任何传统的文本文档复杂的多。Web 页面缺乏同一的结构，它包含了远比任何一组书籍或其它文本文档多得多的风格和内容。Web 可以看作一个巨大的数字图书馆；然而，这一图书馆中的大量文档并不根据任何有关排列次序加以组织。它没有分类索引，更没有按标题，作者，扉页，目次等的索引。对在这样一个图书馆中搜索希望得到的信息是极具挑战性的。

■ Web 是一个动态极强的信息源。Web 不仅以极快的速度增长，而且其信息还在不断地发生着更新。新闻，股票市场，公司广告，和 Web 服务中心都在不断地更新着各自的页面。链接信息和访问记录也在频繁地更新之中。

■ Web 面对的是一个广泛的形形色色的用户群体。目前因特网上连接有约五千万台工作站，其用户群仍在不断地扩展当中。各个用户可以有不同的背景，兴趣，和使用目的。大部分用户并不了解信息网络结构，不清楚搜索的高昂代价，极容易在“黑暗”的网络中迷失方向，也极容易在“跳跃式”访问中烦乱不已和在等待信息中失去耐心。

■ Web 上的信息只有很小的一部分是相关的或有用的。据说 99%的 Web 信息相对 99%的用户是无用的。虽然这看起来不是很明显，但一个人只是关心 Web 上的很小很小一部分信息确是事实，Web 所包含的其余信息对用户来说是不感兴趣的，而且会淹没所希望得到的搜索结果。

这些挑战已经推动了如何高效且有效地发现和利用因特网上资源的研究工作。

目前有许多基于索引的 Web 搜索引擎，它可以完成对 Web 的搜索，对 Web 页面的索引，和建立和存储大量的基于关键字的索引，用于定位包含某关键字的 Web 页面。利用搜索引擎，有经验的用户可以通过提供一组紧密相关的关键字和词组，快速定位到所需的文档。但是，目前基于关键字的搜索引擎存在一些问题。首先，对任一范围的话题，都可能很容易地包含成百上千的文档。这会使得搜索引擎返回的文档数过于庞大，其中很多与话题的相关性并不大，或所包含的内容质量不高。其次，很多与话题相关的文档可能并不包含相应的关键字。这被称为多义问题，如前面有关文本挖掘一节中所讨论的。例如，关键字数据挖掘可能会带出很多与采掘工业有关的 Web 页面，而可能无

[30] 最近，有一些工作在致力于存储或集成 Web 上的所有数据。例如，在http://www.archive.org/index1.html下，可访问到一个巨大的数十兆兆字节的因特网存档。

法识别有关知识发现，统计分析，或机器学习方面的论文，原因是它们不包含关键字数据挖掘。再举个例子，对关键字搜索引擎的查找，可能找不出最常见的搜索引擎，如 Yahoo!, Alta Vista, 或美国在线，如果这些引擎不在其页面上声明其为搜索引擎。这表明目前 Web 搜索引擎对 Web 资源的查找还存在缺陷。

“如果 Web 搜索引擎对 Web 资源的查找都还不够充分，何以谈得上 Web 挖掘？”Web 挖掘是一个更具挑战性课题，它实现对 Web 存取模式，Web 结构，和规则和动态的 Web 内容的查找。一般地，Web 挖掘可分为三类：Web 内容挖掘(Web content mining)，Web 结构挖掘(Web structure mining)，和 Web 使用记录的挖掘(Web usage mining)。另外，Web 结构也可以被认为是 Web 内容挖掘的一部分，这样 Web 挖掘可以简单分为两类，即 Web 内容挖掘和 Web 使用记录挖掘。

在以下小节中，我们将讨论与 Web 有关的几个重要问题：Web 链接结构的挖掘（9。6。1 节）；Web 文档自动分类（9。6。2 节）；多层次 Web 信息库的建立（9。6。3 节）；和 Weblog 挖掘（9。6。4 节）。

9．6．1 挖掘 Web 链接结构，识别权威 Web 页面

“什么是‘权威’(authoritative) Web 页面？”假设要搜索某一给定话题的 Web 页面，例如金融投资方面的页面。这时我们希望得到与之相关的 Web 页面外，还希望所检索到的页面具有高质量，或针对该话题具有权威性。

“但是搜索引擎如何能够自动找出话题的权威 Web 页面？”这里基于了一个有意思的发现，即权威性（authority）可由 Web 页面链接来反映。Web 不仅由页面组成，而且还包含了从一个页面指向另一个页面的超链接。超链接包含了大量人类潜在的语义，它有助于自动分析出权威性语义。当一个 Web 页面的作者建立指向另一个页面的指针时，这可以看作是作者对另一页面的注解。把一个页面的来自不同作者的注解收集起来，就可以用来反映该的页面的重要性，并可以很自然地用于权威 Web 页面的发现。因此，大量的 Web 链接信息提供了丰富的关于 Web 内容相关性，质量，和结构方面的信息，这对 Web 挖掘是可以利用的一个重要资源。

这一思想激发了一些有趣的权威 Web 页面挖掘的研究工作。在七十年代，信息检索的研究者提出了使用杂志论文引用情况的评估研究论文质量的方法。然而，与杂志的引用率不同，Web 链接结构具有特殊的特征。首先，不是每一个超链接都具有注解性。有些链接是为其它目的而创建的，如为了导航或为了付费广告。总体上，若大部分超链接具有注解功能，就可以用于权威判断。其次，基于商业或竞争的考虑，很少有 Web 页面会指向其竞争领域的权威页面。例如，可口可乐不会链接到其竞争对手百氏的 Web 页面。第三，权威页面很少具有特别的描述。如 Yahoo!主页面不会明确给出 “Web 搜索引擎”之类的自描述信息。

由于 Web 链接结构存在这些局限性，人们提出了另外一种重要的 Web 页面，称为 hub。一个 hub 是指一个或多个 Web 页面，它提供了指向权威页面的链接集合。Hub 页面本身可能并不突出，或着说可能没有几个链接指向它们。但是，hub 页面却提供了指向就某个公共话题而言最为突出的站点链接。此类页面可以是主页上的推荐链接列表，例如一门课程主页上的推荐参考文献站点，或商业站点上的专业装配站点。Hub 页面起到了隐含说明某话题权威页面的作用。通常，好的 hub 是指向许多好的 authority 的页面；好的 authority 是指由许多好的 hub 所指向的页。这种 hub 与 authority 之间的相互作用，可用于权威页面的挖掘和高质量 Web 结构和资源的自动发现。

“那么，如何利用 hub 页去找出权威页？”算法 HITS（Hyperlink-Induced Topic Search），是利用 hub 的搜索算法，其内容如下。

首先，HITS 由查询词得到一初始结果集，比如，由基于索引的搜索引擎得到 200 个页面。这些页面构成了**根集**（root set）。由于这些页面中的许多页面是假定与搜索内容相关的，因此它们中应包含指向最权威页面的指针。故此，根集可进一步扩展为**基本集**（base set），它包含了所有由根集中的页所指向的页，以及所有指向根集页的页。可以为基本集设定一个上限，如 1000 至 5000（页），用于指明扩展的一个尺度。

其次，是权重传播（weight-propagation）阶段。这是一递归过程，用于决定 hub 与 authority 权重的值。值得一提的是，由于具有相同 Web 域（即在 URL 中具有相同一级域名）的两个页面之间的链接，经常是起到导航的功能，因此对 authority 没有贡献，此类链接可以从权重传播分析中去除。

我们首先可以为基本集中的每一页面赋予一个非负的 authority 权重 a_p 和非负的 hub 权重 h_p，并将所有的 a 和 h 值初始为同一个常数。权重被规范处理，保证不变性，如所有权重的平方和为 1。

hub 与 authority 的权重可按如下公式计算：

P438

（ q 满足 （9.7）

（ q 满足 （9.8）

公式（9。7）反映了若一个页面由很多好的 hub 所指，则其 authority 权重会相应增加（即，权重增加为所有指向它的页面的现有 hub 权重之和）。公式（9。8）反映了若一个页面指向许多好的权威页，起 hub 权重也会相应增加（即，权重增加为该页面链接的所有页面的 authority 权重之和）。

这两个公式可以按如下的矩阵形式重写。用{1，2，…., n}表示页面，并定义邻接矩阵 A 为 n×n 矩阵。其中若页面 i 链接到页面 j，则 A（i,j）设为 1，否则设为 0。同样定义 **authority 权重向量** a=(a1,a2,….,an)，和 **hub 权重向量** h=(h1,h2,…,hn)。这样，我们有

P439

其中 A^T 是矩阵 A 的转置。对两公式进行 k 次迭代，得到

P439

根据线性代数，当规范化后，两迭代序列分别趋于特征向量 AA^T 和 A^TA。这也证明了 authority 和 hub 权重是彼此链接页面的本质特征，它们与权重的初始设置无关。

最后，HITS 算法输出一组具有较大 hub 权重的页面，和具有较大 authority 权重的页面。许多实验表明，HITS 对许多查询具有非常良好的搜索结果。

虽然基于链接的算法可以带来很好的结果，但这种方法由于忽略文本内容，也遇到一些困难。例如当 hub 页包含多个话题的内容时，HITS 有时会发生偏差。这一问题可以按如下的方法加以克服，即将公式（9.7）和（9.8）置换为相应权重的和，降低同一站点内多链接的权重，使用 anchor 文本（Web 页面中与超连接相连的文字）调整参与 authority 计算的链接的权重，将大的 hub 页面分裂为小的单元。

基于 HITS 算法的的系统包括 Clever。Google 也基于了同样的原理。这些系统由于纳入了 Web 链接和文本内容信息，查询效果明显优于基于词类索引引擎产生的结果，如 Alta Vista，和基于人工的本体论生成的结果，如 Yahoo!。

9．6．2 Web 文档的自动分类

在 Web 文档自动分类中，基于一组预先分类好的文档，可以从予定义好分类目录中为每一文档赋予一个类标签。例如，Yahoo!的分类和其相关文档可以作为训练集，用于导出 Web 文档分类模式。这一模式可以用于对新的 Web 文档加以分类。

基于关键字的文档分类方法和关联分析方法已在 9。5。2 节讨论过。这些方法也可用于 Web 文档的分类。由于超链接包含了有关页面内容的高质量信息，因此很好利用这些信息进行分类可以比基于关键字的分类方法，来的更准确和更完美。然而，由于围绕一个文档的超链接可能是“噪音”的，因此一味使用超链接中的词信息，有时甚至会降低查询的精确性。为此有人提出了一些新的方法，如将一些统计方法如马尔可夫随机场（Markov random field, MRF），结合宽松标识（relaxation labeling）方法，就是一种尝试，实验显示此方法可以极大地改善 Web 文档分类的准确性。

9．6．3 多层次 Web 信息库的构造

在第二章曾指出过，基于关系数据库可以构造数据仓库，用于提供数据的多维与层次化视图。

“那么能否构造多层次 Web 信息库（multilayered Web information base），用于提供 Web 的多维与层次化视图？”大家可能不置可否。这里尝试设计一种多层次 Web 信息库，大家可以分析一下它是否是可行和有益的。

首先，建立一种 Web 数据仓库，包含了 Web 中的每一页面的复本，这是不太现实的。因为这只能导致一个巨大，重复的 WWW。这表明多层次 Web 信息库的最底（最详细）一层必须是 Web 本身。它不可能成为单独的数据仓库。我们把这一层称为 layer-0。

其次，我们可以把 layer-1 定义为 Web 页描述层（Web page descriptor layer），包含了 Web 上页的描述信息。因此，layer-1 是 layer-0 的抽象层。它应当大大小于 layer-0，但仍然包含足够的信息，用于基于关键字或多维的搜索或挖掘。

基于 Web 页的内容不同，layer-1 可以组织为若干半结构化的类，如 document（文档），person（人），organization(组织)，advertisement(广告)，directory(目录)，sale(销售)，software(软件)，game(游戏)，stocks(股票)，library_catalog（图书馆目录），geographic_data(地理数据)，scientific_data(科学数据)，等等。例如，我们可以将类 document 如下：

●P440

其中每一项 Web 页文档的一个抽象。第一个属性，file_addr，记录文件名和 URL 网络地址。属性 doc_category 和 authoritative_rank 包含了可由 Web 链接分析和文档分类方法（见前两节的讨论）所得到的一些重要信息。许多属性包含了与文档相关的主要语义信息，如（P441）。其它属性反映格式信息，如 form，它指明文件的格式（例如，.ps, .pdf, .tex, .doc, .html, 文本，压缩，uuencoded，等）。还有几个属性直接反映与文件有关的信息，如 size_doc（文档文件的大小）和 time_stamp（最新更新时间）。属性 access_frequency 表示被访问的频率。

第三，各种更高级别（higher-layer）的 Web 目录服务可以在 layer-1 之上加以构造，用于提供针对数据库系统研究的黄页服务。这种目录可以包含几个维的层次结构，如主题分类，地理位置，发表时间等等。

“我们需要包括每一个 Web 页的信息吗？”使用 Web 页的等级（ranking） 和页或文档分类服务，我们可以在构造 layer-1 和/或更高层次信息库中有选择地保留质量高，相关性高的必要信息。

随着结构化标记语言 XML 越来越流行，被人们接受和采纳，可以预期未来将会有大量的 Web 页面用 XML 书写，并遵循一组好的 DTD（Document Type Declarations，文档类型说明）。类似 XML 的标准化语言，可以有利促进不同 Web 站点间的信息交换，和方便构造多层次 Web 信息库的信息提取。而且，更便于设计和实现基于 Web 的信息搜索和知识发现语言。

总之，基于以上讨论，构造多层次 Web 信息库应该是可能的，它可以方便因特网上的资源发现，多维分析，和数据挖掘。可以预期基于 Web 的多维分析和数据挖掘将成为因特网上信息服务的重要部分。

9．6．4 Web 使用记录的挖掘

“什么是 Web 使用记录的挖掘（Web usage mining）？”除了 Web 内容和 Web 链接结构，Web 挖掘的另一个重要任务是 Web 使用记录挖掘，它通过挖掘 Web 日志记录，来发现用户访问 Web 页面的模式。通过分析和探究 Web 日志记录中的规律，可以识别电子商务的潜在客户，增强对最终用户的因特网信息服务的质量和交付，并改进 Web 服务器系统的性能。

Web 服务器通常保存了对 Web 页面的每一次访问的（Web）日志项，或叫 Weblog 项。它包括了所请求的 URL，发出请求的 IP 地址，和时间戳。对基于 Web 的电子商务服务器，保存了大量的 Web 访问日志记录。热点的 Web 站点每天可以记录下数以百兆的 Weblog 记录。Weblog 数据库提供了有关 Web 动态的丰富信息。因此研究复杂的 Weblog 挖掘技术是十分重要的。

在开发 Web 使用记录挖掘技术中，我们可能要考虑如下问题。首先，虽然 Weblog 分析可以设想出许多激动人心的潜在应用，但重要的一点是此类应用的成功要依赖于从这一巨大原始日志数据中能够发现什么样可靠和有效的知识，有能发现多少。通常，原始的 Weblog 数据需要经过清洗，精简，和转换，以便于检索和分析有意义和有用的信息。原则上，这些预处理方法与第 3 章中讨论的类似，只不过经常需要定制的预处理方法。

其次，基于 URL，时间，IP 地址，和 Web 页面内容信息，可以在 Weblog 数据库上构造多维视图，进行多维分析 OLAP 分析，用于找出头 N 个用户，头 N 被访问页面，最频繁访问时间期，等等，这有助于发现潜在客户，市场等。

第三，在 Weblog 记录上可以进行数据挖掘，用于找出关联模式，序列模式，和 Web 访问趋势等。对 Web 访问模式挖掘，通常需要采用进一不的手段获得用户访问的附加信息，以便于做更为详细的 Weblog 分析。此类附加信息一般包括 Web 服务器缓冲中，用户浏览 Web 页面的序列等等。

通过使用这类 Weblog 文件，可以进行一些研究工作，如系统性能分析，通过 Web 缓存改进系统设计，Web 页面预取，Web 页面交换（swapping）；认识 Web 信息访问的本质；理解用户的反映和动机。例如，有些研究提出了可适应站点（adaptive site）的概念：即可以通过用户访问模式的学习改进其自身的 Web 站点。Weblog 分析还有助于建立针对个体的个性化 Web 服务。

由于 Weblog 数据提供了什么样的用户访问什么样的 Web 页面的信息，因此 Weblog 信息可以与 Web 内容和 Web 链接结构挖掘集成起来，用于 Web 页面的等级划分，Web 文档的分类，和多层次

Web 信息库的构造。

9．7 总结

●大量数据具有各种各样的复杂形式，如结构化或非结构化，超文本，和多媒体等。因此复杂数据类型的挖掘，包括对象数据，空间数据，多媒体数据，时序数据，文本数据，和 Web 数据，已经成为数据挖掘中日益重要的研究内容。

●在对象-关系和面向对象数据库中，可以进行多维分析和数据挖掘，方法包括（1）复杂对象基于类的概化，复杂对象包括集合值，列表值，和其它复杂的数据类型，类/子类层次，和类组成层次；（2）构造对象数据立方体；和（3）进行基于概化的挖掘。规划数据库可以通过基于概化的，分而治之的方法加以挖掘，用于找出在不同抽象级别上的有意义的一般模式。

●空间数据挖掘是指从大数据量的地理空间数据库中发现有意义的模式。可以构造出包含空间维和度量的空间数据立方体。可以实现空间 OLAP 用于多维空间数据分析。空间数据挖掘包括空间数据描述，分类，关联，聚类，和空间趋势和孤立点(outlier)分析。

●多媒体数据挖掘是指从多媒体数据库中发现有意义的模式。多媒体数据库存储和管理大量多媒体对象，包括音频数据，图象数据，视频数据，序列数据，和包含有文本、文本标记、链接的超文本数据。多媒体数据挖掘包括基于内容的检索和相似分析，概化和多维分析，分类和预测分析，以及多媒体数据中的关联挖掘。

●时序数据库是指由随时间变化的值或事件序列组成的数据库，如股票市场数据，商业交易序列，动态产品处理，医疗，Web 页面访问序列，等等。有关时序和序列数据挖掘的研究内容包括趋势分析，在时序分析中的相似检索，和与时间相关数据中序列模式和循环模式的挖掘。

●大量可获得信息是存储在文本或文档数据库中，它包含了丰富的文档内容，如新闻文章，技术论文，书籍，数字图书馆，电子邮件信息，和 Web 页面。文本数据挖掘因此日益成为重要的研究方向。文本挖掘超出了基于关键字和基于相似的信息检索范畴，它是利用基于关键字的关联和文档分类之类的方法从半结构化文本数据中发现知识。

●万维网作为一个巨大，广泛分布的全球信息服务中心，服务内容涉及新闻，消费信息，金融管理，教育，政府，电子商务，和许多其它服务。它还包含了丰富和动态的超链接信息，和访问及使用信息，这为数据挖掘提供了丰富的资源。Web 挖掘包括 Web 链接结构，Web 内容和 Web 访问模式的挖掘。这里讲到了用于识别权威页面的 Web 链接结构挖掘，Web 文档的自动分类，多层次 Web 信息库的建立，以及 Weblog 挖掘。

习题

9.1 异构数据库系统由多个数据库系统组成，这些数据库的定义是相互独立的，但彼此间需要一定的信息交换，能够处理局部和全局查询。试述在这种系统中如何使用基于概化的方法处理描述性挖掘查询。

9.2 对象立方体可以通过对面向对象数据库的概化，抽象为结构化数据，用于多维分析。试述如何在对象立方体中处理集合值数据。

9.3 空间关联挖掘可以至少按如下两种方式加以实现：（1）基于挖掘查询的要求，可以动态计算不同空间对象之间的空间关联关系；（2）预先计算出空间对象间的空间距离，使得关联挖掘可以基于这些预计算结果求得。试述（1）如何高效实现上述方法；（2）各方法的适用条件。

9.4 假设某城市的交通部门需要规划高速公路的建设，为此希望基于每天不同时刻收集到的交通数据进行有关高速公路交通方面的数据分析。

（a） 设计一存储高速公路交通信息的空间数据仓库，可以方便地支持人们按高速公路，按某天的某一时间，和按周末查看平时和高峰时间的交通流量，可以在发生重大交通事故时，查到出事地点。

（b） 可以从该空间数据仓库中挖掘什么样的信息用于支持城市规划人员？

（c） 该数据仓库既包含了空间数据，也包含了时态数据。设计一种挖掘技术，可以高效地从该空间-时态数据仓库挖掘有意义的模式。

9.5 多媒体中的相似检索已经成为多媒体数据检索系统开发中的主要内容。然而，许多多媒体数据挖掘方法只是基于简单的多媒体特征分析，如颜色，形状，描述，关键字，等等。

（a） 请指出将数据挖掘与基于相似的检索结合，可以给多媒体数据挖掘带来重要的进步。可以任一数据挖掘技术为例，如多维分析，分类，关联，或聚类，等。

（b） 请概述应用基于相似的检索方法增强多媒体数据中聚类质量的实现技术。

9.6 假设一供电站保存了按时间，按地区的能源消耗量，和每一地区每一用户的能源使用信息。讨论在这一时序数据库中，如何解决如下的问题：

（a） 找出星期五某一给定地区的相似的能源消耗曲线；

（b） 当能源消耗曲线急剧上升时，20 分钟内会发生什么情况？

（c） 如何找出可以区分稳定能源消耗地区与不稳定能源消耗地区的最突出特征？

9.7 假设某连锁餐厅想挖掘出与主要体育事件相关的顾客消费行为，如“每当电视播出法裔加拿大人的曲棍球比赛时，肯德鸡的销量会在比赛前一小时上升 20%”。

（a） 给出一种找出这种模式的有效方法。

（b） 大部分与时间相关的关联挖掘算法都使用了类 Apriori 算法来挖掘此类模式。6。2。4 节中介绍的基于数据库投影的 frequent pattern(FP) growth 方法，对挖掘 frequent itemset 是十分有效的。可否扩展 FP-growth 方法去找出此类与时间相关的模式？

9.8 一个电子邮件数据库是指包含了大量电子邮件（e-mail）信息的数据库。它可以被视为主要包含文本数据的半结构化数据库。讨论以下问题：

（a） 如何使一个 e-mail 数据库变成结构化的，以便支持多维检索，如按发送者，按接受者，按主题，按时间，等等检索。

（b） 从 e-mail 数据库中可以挖掘什么信息？

（c） 假设对以前的一组 e-mail 信息有一个粗略的分类，如 junk(垃圾), unimportant（不重要）, normal（一般）, 或 important（重要）。试述一数据挖掘系统如何以此为训练集来自动分类新的 e-mail 消息或反分类（unclassify）e-mail 信息。

9.9 构造针对万维网的数据仓库是十分困难的，原因是它的动态性和存储数据的海量。不过，在因特网上构造包含汇总（summarized）、局部(localized)、多维信息的数据仓库仍然是一件有趣而有用的事情。假设一因特网信息服务公司希望建立一个基于因特网的数据仓库，以帮助旅游者选择当地旅馆和餐厅。

（a） 请设计一个能满足此服务的基于 Web 的旅游数据仓库。

（b） 假设每一旅馆和/或餐厅都有一个自己的 Web 页面。讨论为使这一基于 Web 的旅游数据仓库大众化，如何查询这些页面，和用什么方法去从这些页面中抽取信息。

（c） 讨论如何实现一种挖掘方法，能够提供关联信息，如“呆在市中心希尔顿的９０％的顾客至少要在 Emperor Garden 餐厅就餐两次。”

9.10 每一个科学或工程学科都有其自身的主题索引分类标准，用于对该学科的文档加以分类。

（a） 设计一个 Web 文档分类方法，可以利用此类主题索引对一组 Web 文档实现自动分类。

（b） 讨论如何利用 Web 链接信息来改进此类分类的质量。

（c） 讨论如何利用 Web 使用信息来改进此类分类的质量。

9.11 Weblog 记录为数据挖掘提供了丰富的 Web 使用信息。

（a） 挖掘 Weblog 访问序列，可有助于预取一定 Web 页面到 Web 服务器的缓冲中，例如那些在接下来需要访问的页面。设计一种可用于挖掘此类访问模式信息的有效的实现方法。

（b） 挖掘 Weblog 访问记录，可有助于聚类用户为分离组别，以便实现个性化的市场服务。讨论如何设计一种实现用户聚类的有效方法。

文献注解

复杂数据类型挖掘已经是一个发展迅速，颇为热点的研究领域，有关数据挖掘和数据库系统的会议和杂志上涌现了大量的这方面的研究论文和辅导报告。本章讨论了几个这方面的重要主题，包括复杂数据对象的多维分析，空间数据挖掘，时序数据和其它与时间相关的数据挖掘，文本挖掘，和 Web 挖掘。

[。。。]给出了有关处理复杂数据对象的高级数据库系统的系统介绍。至于复杂数据对象的多维分析和挖掘，[。。。]提出了一种通过多维概化设计和构造对象立方体的方法，用于面向对象和对象关系数据库中的复杂数据类型挖掘。[。。。]提出了通过基于概化的数据挖掘技术构造多层次数据库的方法，用于处理语义异构问题。[。。。]提出了基于概化的方法，通过分而治之的策略，实现对规

划数据库的挖掘。

[。。。]给出了有关空间数据库的一些入门性材料。对地理空间数据挖掘，[。。。]给出有关空间数据挖掘的较全面的综述。[。。。]在地理数据挖掘和知识发现方面有不少贡献。[。。。]提出了面向属性归纳的基于概化的空间数据挖掘方法。[。。。]提出一种基于聚类结果，而不是预先定义的概念层次的描述性空间数据分析方法。[。。。]研究了有关空间数据立方体的设计和构造问题。[。。。]提出了一种高效的多边形合并（polygon amalgamation）的方法，用于联机的多维空间分析和空间数据挖掘。[。。。]提出了一种逐步求精的空间关联规则的挖掘方法。[。。。]提出了一种在空间数据库中挖掘聚集接近关系和共性的方法。[。。。]给出了空间分类和趋势分析的方法。[。。。]提出了一种空间数据分类的两步方法。空间聚类是地理空间数据挖掘近来较活跃的研究领域。有关空间聚类方法的详细的参考文献清单，可参见第八章中的文献注解。[。。。]开发了一个空间数据挖掘系统原型 GeoMiner。

有关多媒体数据库的理论与实践，已有很多的教材和综述论及过，这包括[。。。]。[。。。]介绍了 IBM QBIC(Query By Image and Video Content，基于图象和视频内容的查询)系统。[。。。]提出了一种快速算法，FastMap，用于对传统和多媒体数据集的索引，数据挖掘和可视化处理。[。。。]提出了 WALRUS，一种图象数据库的相似检索算法，算法采用了具有地区粒度的基于小波的特征标识。在 Palomar Observatory Sky Survey(POSS-II)项目中，[。。。]采用决策树的方法，对星系，星星，和其它恒星体进行分类。[。。。]开发了一个数据挖掘系统，Quakefinder，用于从遥感图象中发现地震。[。。。]提出了在大型多媒体数据库中挖掘对象和特征关联的逐步深化的方法。[。。。]开发了多媒体数据挖掘系统原型 MultiMediaMiner。

已经有人提出了对时序分析的统计方法，并在统计学中有较充分的研究，如[。。。]。[。。。]研究了序列数据库中高效相似搜索的方法。[。。。]提出了在时序数据库中子序列匹配的方法。[。。。]提出了一种在时序数据库中出现噪音，伸缩，偏移情况下的快速相似搜索方法。[。。。]提出了用于查询历史情况的语言。其它有关时序数据的相似搜索的研究工作有[。。。]。

针对序列模式挖掘，[。。。]提出了一种类 Apriori 技术和一种一般的序列模式挖掘算法，[。。。]考虑了模式中的 frequent episodes，这里的 episodes 本质上是事件的非环图，其边表明了时态性前-和-后（temporal before-and-after）的关系，它不具有时间间隔的限制。[。。。]提出了事务间的关联规则，它是蕴涵规则，其两端是具有时间间隔限制（对 episodes 中的事件和对规则的两端）的有序的 episodes。[。。。]考虑了事务间关联规则的概化。[。。。]研究了挖掘循环关联规则的方法。[。。。]提出了对 plan failures 的序列模式挖掘。[。。。]提出了局部周期挖掘的最大子模式匹配集（max-subpattern hit set）方法。[。。。]提出了一种基于约束的序列模式的挖掘方法。[。。。]设计了一种高效的序列模式挖掘方法，FreeSpan，它基于了数据库投影和分片挖掘。[。。。]给出了针对时间序列的联机挖掘方法。

文本挖掘在信息检索和文本分析中已有相当充分的研究，这方面有很多好的教材和综述性文章，如[。。。]。最近[。。。]的专著对信息检索做了信息的论述。[。。。]提出了针对文本相似分析的潜在语义标引的方法。[。。。]描述了特征文件的使用。[。。。]研究了文本数据库中关联规则的挖掘。[。。。]提出了基于关联挖掘的自动文档分类方法。

有关 Web 数据建模和 Web 查询系统已有研究工作，如[。。。]的 W3QS，[。。。]的 WebSQL，[。。。等。。]的 Lorel，[。。。]的 Weblog，[。。。]的 WebOQL，[。。。]的 NiagraCQ。[。。。]给出了 Web 数据库的较全面的综述。

[。。。]给出了挖掘 Web 链接结构来识别权威 Web 页面的方法。[。。。]提出了 HITS 算法。[。。。]提出了页面等级排列算法。[。。。]研究了 Web 页面分类算法。[。。。]提出了一种 Web 挖掘语言，WebML。[。。。]提出了构造 Web 仓库的多层次数据库方法。有关超文本和 Web 数据挖掘方面的辅导性综述参见[。。。]。

Web 使用记录挖掘（Web usage mining）已由许多企业推广和实现。[。。。]提出了基于 Weblog 的用户访问模式学习自动构造可适应 Web 站点的方法。[。。。]报道了一个研究原型系统，WebLogMiner。[。。。]给出了 Web 使用记录挖掘及其应用方面的综述。

第十章 数据挖掘的应用和发展趋势

“有哪些突出的例子能够说明数据挖掘在科学和商业领域中的应用？数据挖掘未来向何处去？”在阅读了本书的前面一些章节后，这些问题可能是大家最为关心的。在这最后一章中，我们将讨论一下数据挖掘的应用，并对购买数据挖掘软件系统应注意的问题给出一些建议。另外介绍一下数据挖掘中的其他一些主题，如视频和音频挖掘，数据挖掘的统计方法，数据挖掘的理论基础，以及通过引入数据挖掘技术支持智能查询应答等。数据挖掘的的社会影响和未来趋势也在本章讨论之中。

10．1 数据挖掘的应用

在本书的前面章节中，我们主要讨论了对关系数据，数据仓库，和复杂数据类型（包括空间数据，多媒体数据，时序数据，文本数据，和 Web 数据）的挖掘原理和方法。由于数据挖掘是一门具有广泛应用的新兴学科，数据挖掘的一般原理与针对特定应用领域需要的有效数据挖掘工具之间，还存在不小的距离。本节我们分析几个应用领域，讨论如何为这些应用定制专门的数据挖掘工具。

10．1．1 针对生物医学和 DNA 数据分析的数据挖掘

在过去的十年里，生物医学研究有了迅猛的发展，从新药物的开发和癌症治疗的突破，到通过大规模序列模式和基因功能的发现，进行人类基因的识别与研究。由于目前生物医学的大量研究都集中在 DNA 数据的分析上，这里我们重点研究此应用的情况。近期 DNA 分析的研究成果已经导致了对许多疾病和残疾的基因成因的发现，以及对疾病的诊断，预防，和治疗的新药物、新方法的发现。

基因研究中的一个重要关注点是 DNA 序列的研究，因为这种序列构成了所有活的生物体的基因代码的基础。所有的 DNA 序列由四个基本的构块（称为核苷）组成：腺嘌呤（A），胞核嘧啶（C），鸟嘌呤（G），胸腺嘧啶（T）。这四个核苷组合构成很长的序列或链，类似一个双绞旋梯。

人类有约 100，000 个基因。一个基因通常由成百个核苷按一定次序组织而成。核苷按不同的次序和序列可以形成不同的基因，几乎是不计其数。具有挑战性的问题是从中找出导致各种疾病的特定基因序列模式。由于在数据挖掘中已经有许多有意思的序列模式分析和相似检索技术，因此数据挖掘成为 DNA 分析中的强有力工具，并在以下方面对 DNA 分析起着不小的贡献：

异构、分布基因数据库的语义集成：由于广泛多样的 DNA 数据高度分布、无控地生成与使用，对这种异构和广泛分布的基因数据库的语义集成就成为一项重要任务，以便于对 DNA 数据库进行系统而协同的分析。这促进了集成式数据仓库和分布式联邦数据库的开发，用于存储和管理原始的和导出的基因数据。数据挖掘中的数据清洗和数据集成方法将有助于基因数据集成和用于基因数据分析的数据仓库的构造。

DNA 序列间相似搜索和比较：我们已经研究过时序数据挖掘中的相似搜索方法。在基因分析中一个最为重要的搜索问题是 DNA 序列中的相似搜索和比较。对分别来自带病和健康组织的基因序列，进行比较以识别两类基因间的主要差异。做法可以是首先从两类基因中检索出基因序列，然后找出并比较每一类中频繁出现的模式。通常，在带病样本中出现频度超出健康样本的序列，可以认为是导致疾病的基因因素；另一方面，在健康样本中出现频度超出带病样本的序列，可以认为是抗疾病的因素。注意，虽然基因分析需要相似搜索，但这里所需要的技术与时序数据中使用的方法截然不同。例如，数据变换的方法如伸缩，规范化，和窗口缝合等，这些是在时序数据分析中经常用到的方法，对基因数据而言是无效的，因为基因数据是非数字的，其内部的不同种类核苷间的精确交叉起着重要的功能角色。另一方面，频繁序列模式的分析在基因序列相似与非相似分析中非常重要。

关联分析：同时出现的基因序列的识别：目前，许多研究关注的是一个基因与另一个基因的比较。然而，大部分疾病不是由单一基因引起的，而是由基因组合起来共同起作用的结果。关联分析方法可用于帮助确定在目标样本中同时出现的基因种类。此类分析将有助于发现基因组和对基因间

的交叉与联系的研究。

路径分析（path analysis）：发现在疾病不同阶段的致因基因：引起一种疾病的基因可能不止一个，不过不同基因可能在疾病的不同阶段起着作用。如果能找到疾病发展的不同阶段遗传因素序列，就有可能开发针对疾病不同阶段的治疗药物，从而取得更为有效的治疗效果。在遗传研究中路径分析会起到重要的作用。

可视化工具和遗传数据分析：基因的复杂结构和序列模式通常可以由各种可视化工具以图，树，方体（cubiods），和链的形式展现。这种可视化的结构和模式方便了模式理解，知识发现，和数据交互。可视化因此在生物医学的数据挖掘中起着重要的作用。

10．1．2 针对金融数据分析的数据挖掘

大部分银行和金融机构都提供丰富多样的储蓄服务(如支票，存款，和商业及个人用户交易)，信用服务（如交易，抵押，和汽车贷款），和投资服务（如共有基金（mutual funds））。有些还提供保险服务和股票投资服务。

在银行和金融机构中产生的金融数据通常相对比较完整，可靠，和高质量，这大大方便了系统化的数据分析和数据挖掘。以下给出几种典型的应用情况。

为多维数据分析和数据挖掘设计和构造数据仓库：与许多其它应用类似，需要为银行和金融数据构造其数据仓库。多维数据分析用于分析这些数据的一般特性。例如，人们可能希望按月，按地区，按部门，以及按其它因素，查看负债和收入的变化情况，同时希望能提供最大，最小，总和，平均，和其它统计信息。数据仓库，数据立方体，多特征（multifeature）和发现驱动（discovery-driven）数据立方体，特征和比较分析，和孤立点分析（outlier analysis）等都会在金融数据分析和挖掘中发挥重要作用。

贷款偿还预测和客户信用政策分析：贷款偿付预测和客户信用政策分析对银行业务是相当重要的。有很多因素会对贷款偿还效能和客户信用等级计算产生不同程度的影响。数据挖掘的方法，如特征选择和属性相关性计算，有助于识别重要因素，剔除非相关因素。例如，与贷款偿还风险相关的因素包括贷款（load-to-value）率，贷款期限，负债率（月负债总额与月收入总额之比），偿还与收入（payment-to-income）比率，客户收入水平，受教育水平，居住地区，信用历史，等等。分析客户偿还的历史信息，可以发现，比如说，偿还收入比是主导因素，而受教育水平和负债率则不是。银行于是可以据此调整贷款发放政策，以便将贷款发放给那些以前曾被拒绝，但根据关键因素分析，其基本信息显示是相对低风险的申请。

对目标市场（targeted marketing）客户的分类与聚类：分类与聚类的方法可用于用户群体的识别和目标市场分析。例如，通过多维聚类分析，可以将具有相同储蓄和贷款偿还行为的客户分为一组。有效的聚类和协同过滤（collaborative filtering）方法（即，使用各种技术滤出信息，如邻近分类，决策树，等等）有助于识别客户组，将新客户关联到适合的客户组，以及推动目标市场。

洗黑钱和其它金融犯罪的侦破：要侦破洗黑钱和其它金融犯罪行为，重要的一点是要把多个数据库的信息（如银行交易数据库，联邦或州的犯罪历史数据库等）集成起来，只要这些数据库是与侦破工作有关的。然后可以采用多种数据分析工具来找出异常模式，如在某段时间内，通过某一组人，发生大量现金流量，等等。有用的工具包括数据可视化工具（用图形的方式按一定时间一定人群显示交易活动），链接分析工具（识别不同人和活动之间的联系），分类工具（滤掉不相关的属性，对高度相关属性排级），聚类分析工具（将不同案例分组），孤立点分析工具（探测异常资金量的转移或其它行为），序列模式分析工具（分析异常访问模式的特征）。这些工具可以识别出一些重要的活动关系和模式，有助于调查人员聚焦可疑线索，做进一步的处理。

10．1．3 零售业中的数据挖掘

零售业是数据挖掘的主要应用领域，这是因为零售业积累了大量的销售数据，顾客购买历史记录，货物进出，消费与服务记录，等等。其数据量在不断地迅速膨胀，特别是由于日益增长的 Web，或电子商务上的商业方式的方便，流行。今天，许多商店都有自己的 Web 站点，顾客可以方便地联机购买商品。一些企业，如 Amazon.com，只有联机方式，没有砖瓦构成的（物理的）商场。零售数据为数据挖掘提供了丰富的资源。

零售数据挖掘可有助于识别顾客购买行为，发现顾客购买模式和趋势，改进服务质量，取得更

好的顾客保持力和满意程度，提高货品销量比率，设计更好的货品运输与分销策略，减少企业成本。

以下给出零售业中的几个数据挖掘的例子。

基于数据挖掘的数据仓库的设计与构造：由于零售数据覆盖面广（包括销售，顾客，职员，货品运输，销量和服务），所以有许多方式设计数据仓库。所包含的细节级别可以丰富多样。由于数据仓库的主要用途是支持数据分析和数据挖掘，预先的一些数据挖掘例子的结果可作为设计和开发数据仓库结构的参考依据。这涉及要决定包括哪些维和什么级别，以及为保证高质量和有效的数据挖掘应进行哪些预处理。

销售，顾客，产品，时间，和地区的多维分析：考虑到顾客的需求，产品的销售，趋势和时尚，以及日用品的质量，价格，利润，和服务，零售业需要的是适时的信息。因此提供强有力的多维分析和可视化工具是十分重要的一件事情，这包括提供根据数据分析的需要构造复杂的数据立方体。第二章介绍的多特征数据立方体（multifeature data cube），在零售数据分析中是一种有用的数据结构，因为它方便了带有复杂条件的聚集上的分析。

促销活动的有效性分析：零售业经常通过广告，优惠券，和各种折扣和让利的方式搞促销活动，以达到促销产品，吸引顾客的目的。认真分分析促销活动的有效性，有助于提高企业利润。多维分析可满足这方面分析的要求，方法是通过比较促销期间的销售量和交易数量与促销活动前后的有关情况。此外，关联分析可以找出哪些商品可能随降价商品一同购买，特别是与促销活动前后的销售相比。

顾客保持力——顾客忠诚分析：通过顾客荣誉卡信息，可以记录下一个顾客的购买序列。顾客的忠诚和购买趋势可以按系统的方式加以分析。由同一顾客在不同时期购买的商品可以分组为序列。序列模式挖掘可用于分析顾客的消费或忠诚的变化，据此对价格和商品的花样加以调整，以便留住老客户，吸引新顾客。

购买推荐和商品参照：通过从销售记录中挖掘关联信息，可以发现购买某一品牌香水的顾客很可能购买其它一些商品。这类信息可用于形成一定的购买推荐。购买推荐可在 Web，每周传单，或收条上宣传，以便改进服务，帮助顾客选择商品，增加销售额。同样，诸如“本周热点商品”之类的信息或有吸引力的买卖也可以相关信息一同发布，以达到促销的目的。

10．1．4 电信业中的数据挖掘

电信业已经迅速地从单纯的提供地话和长话服务演变为提供综合电信服务，如语音，传真，寻呼，移动电话，图象，电子邮件，计算机和 Web 数据传输，以及其它数据通讯服务。电信，计算机网络，因特网，和各种其它方式的通讯和计算的融合是目前的大势所趋。而且随着许多国家对电信业的开放和新兴计算与通讯技术的发展，电信市场正在迅速扩张并越发竞争激烈。因此，利用数据挖掘技术来帮助理解商业行为、确定电信模式、捕捉盗用行为、更好地利用资源和提高服务质量是非常有必要的。

以下是几个利用数据挖掘改进电信服务的具体例子。

电信数据的多维分析：电信数据本身具有多维性，如呼叫时间，持续时间，呼叫者位置，被呼叫者位置，呼叫类型等。对此类数据的多维分析有助于识别和比较数据通讯情况，系统负载，资源使用，用户组行为，利润，等等。例如，分析人员希望经常查看有关呼叫源，呼叫目标，呼叫量，和每天使用模式等方面的图表。因此，将电信数据构造为数据仓库十分有用，可以经常使用 OLAP 和可视化工具进行多维分析。

盗用模式分析和异常模式识别：盗用行为每年可以耗掉电信业数百万美元。确定潜在的盗用者和他们的非典型的使用模式，检测想侵入用户账户的企图，发现需要引起注意的异常模式是非常重要的。这些模式包括：老是占线无法接入，转换和路由阻塞，从被恶意篡改过的自动拨出设备（如传真机）发出的周期性呼叫。通过多维分析，聚类分析，和孤立点分析，可以发现许多这类模式。

多维关联和序列模式分析：多维分析中关联和序列模式的发现可以用来推动电信服务的发展。例如：假设你想发现一系列电信服务的使用模式（按用户组，月或日历分组），按客户分组的呼叫记录可以表现为如下形式：

（customer_id，residense,office,time,date,service_1,service_2,...）

为了决定呼叫是否在两个特定的城市之间或特定的人群间发生，这样的一个序列模式，“如果一个洛杉矶地区的客户在和他居住地不同的另一个城市工作，它可能在每个工作日的下午五点先使用两个地区之间的长途服务，然后在接下来的时间里使用 30 分钟的蜂窝电话”，可以通过上钻和下钻检

测到。这有助于促进特定的长途销售额和蜂窝电话结合，可以用于扩展某个地区的特殊服务。

电信数据分析中可视化工具的使用：OLAP 可视化，链接可视化，关联可视化，聚类，和孤立点可视化等工具已经证明对电信数据分析是非常有用的。

10．2 数据挖掘系统产品和研究原型

尽管数据挖掘是一个新兴的领域，有很多问题需要深入研究，但是已经有了很多现成的数据挖掘系统产品和特定领域的数据挖掘应用软件。作为一个新兴的学科，数据挖掘历史相对较短，正在稳定地发展，每年市场上都会出现新的数据挖掘系统。基础相对稳定的现已存在的系统也在不断地增加新功能，新特性和可视化的工具。对数据挖掘语言的标准化工作刚刚开始。因此，本书不准备对商用的数据挖掘系统做详细的描述，而是给出一些在选择数据挖掘系统时用户需要考虑的特性，并对几个典型的数据挖掘系统做一个简单介绍。有关参考文章，Web 站点和有关数据挖掘系统方面最近的研究成果在参考目录中给出。

10．2．1 怎样选择一个数据挖掘系统

市场上有很多数据挖掘系统产品，大家可能会问：“我该选择哪一种系统呢？”一些人可能有这样的印象：数据挖掘系统象许多商用关系数据库系统一样，共享相同的定义好的操作和标准的查询语言，在一些通用功能上的表现也很类似。如果真是这样的话，系统产品的选择主要取决于系统的硬件平台、兼容性、鲁棒性、可伸缩性、价格和服务。不幸的是事实并不是这样。许多数据挖掘系统在数据挖掘的功能和方法上很少有相似性，有时甚至在完全不同的数据集上进行工作。

要选择一种适合当前任务的数据挖掘系统，重要的是要从多维角度来看它。一般来说，评价一个数据挖掘系统应该包括如下几个方面：

数据类型：市场上的大多数数据挖掘系统能可处理如下的数据，即：有一定格式的基于记录的数据或者是带有数字、分类和符号属性的类似于关系的数据。数据形式可以是 ASCII 文本的，也可以是关系数据库的数据或数据仓库数据。考察一下系统能处理哪种格式的数据是非常重要的，因为一些数据或应用可能需要特定的算法来查找模式，而现有的或通用的数据挖掘系统有可能不能满足需求，相反一些特殊的数据挖掘系统有可能能派上用场，这些特殊的系统或者挖掘文本文档、地理数据、多媒体数据、时间序列数据、DNA 序列、Weblog 记录及其它的 Web 数据，或者用于特定的应用（如金融、零售业、电信业等）。而且，许多数据挖掘公司提供定制的数据挖掘解决方案，将一些本质的数据挖掘功能和方法结合起来。

系统问题：一个数据挖掘系统有可能只在一个系统上运行，也可能同时在多个系统上运行，支持数据挖掘系统的最流行操作系统是 UNIX 和 Microsoft Windows（包括 95、98、2000 和 NT）。也有一些数据挖掘系统运行在 OS/2、Macintosh 和 Linux 上。大的面向工业的数据挖掘系统理想地应该支持 Client/Server 结构，Client 一般是个人机，运行 Microsoft Windows，Server 是一系列强大的并行计算机，运行在 UNIX 上。让数据挖掘系统提供基于 Web 的接口是最近的趋势，这样的接口允许输入和输出 XML 数据。

数据源：这是指数据挖掘系统操作的特定的数据格式。一些系统只能操作 ASCII 文本，另外一些可以操作关系数据，访问多个关系数据源。一个数据挖掘系统支持 ODBC 连结和 OLEDB 是非常重要的，这保证了它能与数据库进行开放连结，也就是具有访问关系数据（包括 DB2，Informix，Microsoft SQL Server，Microsoft Access，Microsofr Execl，Oracle，Sybase 等）的能力。和数据仓库一块工作的数据挖掘系统必须遵循 OLAP 标准的 OLEDB 规范，这样才能保证系统不仅能够访问由 Microsoft SQL Server 7.0 提供的数据仓库,也能访问支持这个标准的其它数据仓库产品.

数据挖掘的功能和方法: 数据挖掘功能是数据挖掘系统的核心，一些数据挖掘系统只提供一种数据挖掘功能,例如分类。而有些数据挖掘系统能够支持多种数据挖掘功能，例如：描述，发现驱动的 OLAP 分析，关联，分类，预测，聚类，孤立点分析，相似性查找，序列模式分析，可视化数据挖掘等。对于一个给定的数据挖掘功能如分类，一些系统可能只支持一种方法，有些可能支持多种方法（例如决策树，贝叶斯网，神经网，遗传算法，基于案例的推理等）。支持多种数据挖掘功能同时每一种功能又支持多种方法的数据挖掘系统，能提供给用户很大的灵活性和很强的分析能力，许多问题要求用户尝试不同的数据挖掘功能或者把几种功能集成起来使用，不同的数据集使用不同的方法，非常有效。当然，由于系统更加灵活，用户可能需要进行培训或者得有经验，因此这些系

统应该给初级用户提供方便，使其能访问最通用的功能和方法，或将最通用的功能和方法作为缺省的设置。

数据挖掘系统和数据库或数据仓库系统的结合：一个数据挖掘系统应该和数据库或数据仓库系统结合起来，以各种组件形式无缝地集成到一个信息处理环境中。其结合方式有四种形式：无耦合的，松耦合的，半松耦合的，和紧耦合。一些只操作 ASCII 文本数据文件的数据挖掘系统不和数据库或数据仓库系统结合，这样的系统在使用大的数据集或者存储在关系数据库中的数据时会有困难。在和数据库或数据仓库松散结合的数据挖掘系统中，数据首先被数据库或数据仓库返回到缓冲区或主内存，然后利用数据挖掘功能进行分析，这样的系统伸缩性不好，执行某些数据挖掘查询的时候不是很有效。和数据库或数据仓库半松散结合的数据挖掘系统只对少数几个数据挖掘原操作（例如排序，索引，聚集，直方图分析，多路联接，一些统计值的预计算等）提供了有效的实现。最理想的是，数据挖掘系统应该和数据库或数据仓库在以下意义上进行紧密结合，即：通过把数据挖掘查询优化成循环的数据挖掘和获取过程，将二者结合起来。数据挖掘和基于 OLAP 的数据仓库紧密结合也是非常必要的，这样数据挖掘和 OLAP 操作就能够集成起来提供 OLAP 挖掘功能。

可伸缩性：数据挖掘有两种可伸缩性问题：行（数据库大小）伸缩和列（维）伸缩。如果一个数据挖掘系统行数扩大了 10 倍，而执行同样的数据挖掘查询的时间最多也不超过其原来时间 10 倍的话，则说这个系统是行可伸缩的；如果数据挖掘查询执行时间和列数呈线性增长关系，则说这个系统是列（属性或维）可伸缩的，由于多维性的原因，使一个系统成为列可伸缩的比让其成为行可伸缩的更具有挑战性。

可视化工具：“一幅图胜过一千句话”，在数据挖掘中是非常真实的。数据挖掘的可视化分为数据可视化，挖掘结果可视化，挖掘过程可视化，和可视化数据挖掘。10.3.1 节将会对其进行详细的讨论。可视化工具的种类，质量和灵活性严重地影响了数据挖掘系统的使用，解释和吸引力。

数据挖掘查询语言和图形用户接口：数据挖掘是一个探测的过程。一个易使用高质量的图形用户接口，对于促进用户指导，进行高交互的数据挖掘非常重要，许多数据挖掘系统提供了友好的用户挖掘界面。在关系数据库系统中，许多图形用户接口是在 SQL（它作为一个标准，具有良好的基于数据的查询语言）的基础上搭建的，但是，大多数数据挖掘系统和它不一样，它们不共享任何底层的数据挖掘查询语言，由于缺少标准的数据挖掘语言，要使数据挖掘产品标准化和使不同的数据挖掘系统之间进行互操作是很难的。在第 4 章介绍了在定义和标准化数据挖掘查询语言方面最近所做的工作。Microsoft’s OLE DB for DM 就是这样的一种语言，我们将在附录 A 中进行描述。

10．2．2 商用数据挖掘系统的例子

前面提到，由于数据挖掘市场还处于起步阶段，而且发展很快，在本书中，我们不想详细的描述任何一个特殊的商用数据挖掘系统，为了帮助读者对目前的数据挖掘产品能做什么事情有一个大致的了解，我们仅对几个典型的数据挖掘产品做一个简单的介绍。

许多数据挖掘系统只提供某一特殊的数据挖掘功能，例如分类，或只提供一个数据挖掘功能的一个方法，如决策树分类法。有些数据挖掘系统能提供多个数据挖掘功能。这里我们介绍几个提供了多种数据挖掘功能和利用了多种知识挖掘技术的系统，

- **Intelligent Miner** 这是 IBM 公司的数据挖掘产品，它提供了很多数据挖掘算法，包括：关联，分类，回归，预测模型，偏离检测，序列模式分析和聚类。它也提供一个应用工具集，包括：神经网络算法，统计方法，数据准备模型和数据可视化工具。Intelligent Miner 的特色有两点：一是它的数据挖掘算法可伸缩，二是它与 IBM DB/2 关系数据库系统紧密地结合在一起。
- **Enterprise Miner** 是 SAS 公司开发的产品，提供多种数据挖掘算法，包括：回归，分类和统计分析包。它的特色是具有多种统计分析工具，这得益于 SAS 公司在统计分析市场多年的经验和历史。
- **MineSet** 是由 SGI（Silicon Graphics Inc.）公司开发的，它也提供了多种数据挖掘算法，包括：关联和分类，高级统计和可视化工具。特色是它具有的强大的图形工具，包括：规则可视化工具，树可视化工具，地图可视化工具，多维数据分散可视化工具，它们用于实现数据和数据挖掘结果的可视化功能。
- **Clementine** 是由 ISL(Integral Solutions Ltd.)公司开发的,它为终端用户和开发者提供了一个集成的数据挖掘开发环境,系统集成了多种数据挖掘算法,如：规则归纳，神经网络，分类和可视化工具。特色是它具有面向对象的扩展的模块接口，该接口使用户算法和工具可以加到

Clementine 的可视化编程环境中。Clementine 已经被 SPSS 公司收购。

- **DBMiner** 是由 DBMiner Technology 公司开发的，它提供多种数据挖掘方法包括：发现驱动的 OLAP 分析，关联，分类，聚类。DBMiner 的特色是它的基于数据立方体的联机分析挖掘，它包含多种有效的频繁模式挖掘功能和集成的可视化分类方法，附录 B 对该系统做了更加详细的介绍。

还有很多其它的商用数据挖掘产品系统和研究原型，其发展也很快，有些读者可能对当前的数据仓库和数据挖掘产品比较感兴趣。

10．3 数据挖掘的其他主题

数据挖掘范围很广，有很多数据挖掘的方法，本书不可能覆盖所有的数据挖掘主题，这一部分我们主要讨论几个在本书前面章节中没有涉及到的比较有趣的主题。

10．3．1 视频和音频数据挖掘

可视化数据挖掘用数据或知识可视化技术从大的数据集中发现隐含的和有用的知识。人们的视觉系统是由眼睛和人脑控制的，后者可看作一个强有力且高度并行的处理和推理引擎，它带有一个大的知识库。可视化数据挖掘把这些强大的组件有效地组合起来，使它成为一个吸引人的有效的工具，用来对数据的属性，模式，簇，孤立点进行综合分析。

可视化数据挖掘可看作是由数据可视化和数据挖掘两个学科融合而成的。它和计算机图形，多媒体系统，人机接口，模式识别，高性能处理都紧密相关。总之，数据可视化和数据挖掘可以从以下方面进行融合：

- 数据可视化 数据库和数据仓库中的数据可看作具有不同的粒度或不同的抽象级别，也可以看作是有不同属性和维组合起来的。数据能用多种可视化方式进行描述，比如：盒状图，三维立方体，数据分布图表，曲线，平面，连结图，等等。图 10.1 和图 10.2 显示了 StatSoft 中的多维空间数据分布。可视化显示能把数据库中数据特性的总体印象提供给用户。
- 数据挖掘结果可视化 数据挖掘结果可视化指将数据挖掘后得到的知识和结果用可视化的形式描述出来。这些形式包括分散划分（scatter plots）和盒状图（通过描述性的数据挖掘），以及决策树，关联规则，簇，孤立点，一般规则，等等。如图 10.3 中显示的 SAS Enterprise Miner 的分散划分结果。图 10.4 显示的是 MiniSet 3.0 中的一个画面,它用一个和一些直方图关联的平面来描述从数据库中挖掘出来的一些关联规则。图 10.5 是 MiniSet 3.0 的决策树。图 10.6 是 IBM Intelligent Miner 提供的簇以及与其相关的属性.
- 数据挖掘过程可视化 这种可视化用可视化形式描述各种挖掘过程,从中用户可以看出数据是从哪个数据库或数据仓库中抽取出来的,怎样抽取的以及怎样清洗,集成,预处理和挖掘的.而且,可以看出数据挖掘选用的方法,结果存储的地方及显示方式.图 10.7 描述了 Clementine 数据挖掘系统的一个可视化的数据挖掘过程.
- 交互式的可视化数据挖掘 交互式的可视化数据挖掘在数据挖掘过程中使用了可视化工具,它用来帮助用户做出明智的数据挖掘决策.例如,一系列属性的数据分布可以用彩色扇区或列(取决于整个空间是使用一个圆形描述还是使用列的集合描述)来表示,这种表示方式可以帮助用户决定哪个扇区作为分类首先被选中,哪个地方是最好扇区分割点。图 10.8 显示的是一个这样的例子,它是 Munich 大学开发的 PBC(perceptinb-based classification)系统的输出界面.

图 10.1 StatSoft 中展示多变量组合的盒状图

图 10.2 StatSoft 中多维数据分布分析

图 10.3 SAS Enterprise Mine 中数据挖掘结果的可视化

图 10.4 MiniSet 3.0 中的关联规则可视化

图 10.5 MiniSet 3.0 中的决策树可视化

图 10.6 IBM Intelligent Miner 中簇分组的可视化

图 10.7 Clementine 中数据挖掘的可视化

图 10.8 Perception-based classification(PBC): 一种交互的可视化挖掘方法

音频数据挖掘用音频信号来显示数据模式或数据挖掘结果的特征.尽管可视化数据挖掘用图形显示能揭露一些有趣的模式,但它要求用户专注于观察模式,确定其中有趣的或新的特征.这有时是很烦人的,如果模式能转换成声音和音乐,这样我们就可以通过听基调,旋律,曲调和音调而不是看图片来确定任何有趣的或不同寻常的东西.在很多情况下,这种方式可能比较轻松,因此,用音频数据挖掘代替可视化数据挖掘是一个有趣的选择.

10.3.2 科学和统计数据挖掘

本书中描述的数据挖掘技术主要是面向数据库的,用于处理大量的多维和各种复杂类型的数据.然而还有很多用于统计数据尤其是数值数据分析的技术,这些技术已经被扩展应用到科学(如心理学,医学,电子工程,或制造业的实验数据)以及经济或社会科学数据中。其中一些技术，如主要成分分析（principle component analysis）,回归，和聚类,已在本书讲过.对数据分析中的主要统计方法的透彻的讨论超出了本书的范围,但是,为了完整性起见,下面我们还是提了一些方法.这些技术的参考文献在参考目录中给出.

- 回归 一般来说,这些方法用来预测从一个或多个取测器来的反应变量的值,它们是数值类型的,有很多种回归方法,如:线形回归,多回归,加权回归,多项式回归,无参数回归,强回归(当错误不满足通常的条件或者数据包含重要的外层是强回归方法非常有用)
- 广义线形模型（generalizzed linear model） 这些模型和它们的广义模型(通用的附加模型),允许一个分类响应变量(或它的一些变种)和一系列预测器变量相关, 这和使用线性回归的模型中的数值响应变量类似.
- 回归树（regression tree） 这可用于分类和预测,构造成的树是二叉树,回归树和决策树在测试都是在节点上做这方面有点类似,它们主要的区别在叶子层,决策树是通过大众选举产生的类标号作为叶子,回归树是通过计算目标属性的平均值作为预测值.
- 方差分析（analysis of variance） 这些技术为用一个数值响应变量和一个或多个分类变量描述的两个或多个人分析实验数据.通常,一个 ANOVA(变量的单因子分析)问题通过 k 个人或对待方式的比较,来决定是否至少有两种方式是不同的.也存在更复杂的 ANOVA 问题.
- 混合效应模型（mixed-effect model） 这些模型用来分析分组数据,也就是那些用一个或多个组变量分类的数据,它们通过一个或多个因素来描述一个响应变量和一些共变量之间的关系.应用的公共领域包括多层数据,重复值数据,块设计数据和纵向数据,
- 因素分析（factor analysis） 这种方法用来决定哪些变量一块产生了一个给定因子,比如,对许多精神病学数据,不可能值测量某个特别的因子(例如智能),然而,用于测量其它的数量(比如学生考试成绩)是可能的,这里没有设计依赖变量.
- 判别式分析（discriminant analysis） 这种技术用来预测分类响应变量,不象通用的线形模型,假定独立变量遵循多元的通常的分布,这个过程企图决定几个判别式函数(独立变量的线性组合),用来区别由响应变量定义的组, 判别式分析在社会科学中经常使用.
- 时间序列 有很多统计技术用来分析时间序列数据,例如自动回归方法,单变量的 ARIMA 模型，长记忆（long memory）的时间序列模型.
- 幸存分析（survival analysis） 有好几种统计技术用于生存分析,起初用于预测一个病人经过治疗后能或至少 t 这么长时间,生存分析的方法也用于制造设备,来估计工业设备的生命周期.流行的方法包括 Laplan-Meier 幸存估计法,Cox 比例危险回归模型,以及它们的扩展.
- 质量控制（quality control） 各种统计法可以用来准备质量控制的图表,例如 Shewhart 图表和 cusum 图表.(都用于显示组合统计)这些统计包括:平均值,标准差(standard deviation),区间,计数,移动平均,移动标准差,和移动区间(moving range).

10．3．3 数据挖掘的理论基础

有关数据挖掘的理论基础研究还没有成熟。坚实系统的理论基础对于数据挖掘非常重要，因为它给数据挖掘技术的开发、评价和实践提供一个一致的框架。数据挖掘的理论基础有很多，比如包括以下内容：

- 数据归约（data reduction） 按照这一理论，数据挖掘的基础是减少数据的描述。在大型数据库里，数据归约能换来快速近似查询的准确性。数据归约技术主要包括奇异值分解（在主要组件分析背后的驱动元素），小波，回归，日志线形模型（log-linear model），直方图(histogram)，簇，取样和索引树构造。
- 数据压缩（data compression） 根据这一理论，数据挖掘的基础是对给定的数据进行压缩，它一般是通过按位、关联规则，决策树，簇等进行编码实现的。根据最小描述长度原理(minimum description length principle)认为，从一个数据集合中推导出的最好的理论是这样的理论，即它本身的长度和用它作为预测器（predicator）进行编码的长度都最小。编码典型的是按位编码。
- 模式发现（pattern discovery） 这个理论基础是由于在数据库中发现模式，比如关联规则，分类模型，序列模式，等等。它涉及机器学习，神经网络，关联挖掘，序列模式挖掘，聚类，和其它的子领域。
- 概率理论（probability theory） 它基于统计理论。依据这一理论，数据挖掘的基础是发现随机变量的联合的可能的分布，例如，贝叶斯置信网络（Bayesian belief network）和层次贝叶斯模型(hierarchical Bayesian models)。
- 微观经济观点（microeconomic view） 它把数据挖掘看作发现模式的任务，通过数据挖掘来发现那些对企业决策过程（如指定市场策略，产品计划等）有用的并在一定程度上有趣的模式。这个观点认为如果模式能发生作用的话则认为它是有趣的。企业在碰到优化问题的时候最大限度的使用这个对象。在此数据挖掘变成一个非线性的优化问题。
- 归纳数据库（inductive databases） 在这个模式中，数据库模式看作是由存储在数据库中的模式和数据组成的，数据挖掘的问题变成了对数据库进行归纳的问题，它的任务是查询数据库中的数据和理论（即模式）。这个观点在数据库系统的许多研究者当中非常流行。

上述理论不是互相排斥的，例如，模式发现可以看作是数据归约和数据压缩的一种形式，一个理想的理论框架应该能够对典型的数据挖掘任务（如关联，分类和聚类）进行建模，有一个概率特性，能够处理不同形式的数据，并且对数据挖掘的反复和交互的本性加以考虑。建立一个能满足这些要求的定义很好的数据挖掘框架是我们进一步努力的目标。

10．3．4 数据挖掘和智能查询应答

在我们的数据挖掘过程的处理框架中，其处理是由查询初始化的，即由查询指定和任务相关的数据，要求发掘的知识种类，关联限制，有趣的阈值等。然而，在很多情况下，用户可能并不精确地知道要挖掘什么东西或者数据库有什么限制，因此不能给出精确的查询。智能查询应答（intelligent query answering）在这种情况下能帮助分析用户的目的，用智能的方式回答查询请求。

下面讲述数据挖掘和智能查询应答结合的一个通用的框架。在数据库系统中，可能存在两种类型的查询，数据查询和知识查询(knowledge query)。数据查询用来发现存储在数据库系统中的具体数据，它与数据库系统的一个基本的检索语句对应。知识查询用来发现规则、模式和数据库中的其它知识，它对应于对数据库知识的查询，包括演绎规则、完整性约束，概化规则，频繁模式以及其它的规则等。例如，“找出在 2000 年 5 月购买尿布的所有顾客的 ID 号”属于数据查询，而“描述这些顾客的通用特征和他们还可能要购买什么”属于知识查询，其查询对象并没有明显地存储在数据库中的知识，通常要由一个数据挖掘的过程导出。

查询应答机制可以根据它们反应方式的不同分为如下两类：直接查询应答（direct query answering）和智能(或协同)查询应答。直接查询应答是指通过精确地返回所要的东西来回答查询，而智能查询应答包括两个阶段，先分析查询目的，然后返回通用的类似的相关的信息。返回与查询

相关的信息但是并不是明显要求的东西提供了对相同查询的智能回答。

Example 10.1 假设一个网上在线商店中心维护了几个数据库,这几个数据库可能包括在线知识库,在线事务历史库,和 web 知识库。

数据查询是执行许多在线服务的例程，例如这样的查询“列出所有在卖的自行车”，或者“找出 Jack Waternan 在 2000 年四月购买的所有的东西”。对这些查询的直接回答是列出有特定属性的项目列表，而智能回答提供给用户的是用于辅助决策的附加信息。这里是智能查询应答和数据挖掘技术相结合来提高商店服务的几个例子。

- 通过提供综合信息来回答查询　当客户查询当前正在卖的自行车列表时，提供附加的综合信息，比如：关于自行车的最好交易，去年卖出的每一种自行车的数量，不同种类自行车吸引人的新特性等，这些综合信息可以用数据仓库和数据挖掘技术得到。
- 通过关联分析来得出附加项目　当一个客户想购买某种特殊牌子的自行车时，可以提供给用户附加的关联信息，如“想购买这种自行车的人可能要购买下列运动设备”，或者“你将考虑购买这种自行车的维修服务吗？”，这样就可以推动公司的其它产品销售额。
- 通过序列模式挖掘来促进产品销售　当客户在线购买一台个人电脑时，系统可能根据以前挖掘出来的序列模式建议他考虑同时购买其它的一些东西，比如：“购买这种个人电脑的人在三个月之内很可能要再买某种特殊的打印机或 CD-ROM”或者送给用户一个短期优惠券.

从这几个例子可以看出,使用数据挖掘方法的智能查询应答能够给电子商务应用提供更有趣的服务.这有可能形成数据挖掘重要的应用,需要进一步的探索

10．4 数据挖掘的社会影响

随着社会的快速计算机化,数据挖掘的社会影响不可低估。数据挖掘是宣传出来的还是真正存在的？数据挖掘成为一种被接受的主流企业或个人应用技术会碰到那些障碍？保护数据隐秘和安全还需要做些什么？下面对每个问题做出回答。

10．4．1 数据挖掘是宣传出来的还是持久的稳定增长的商业？

数据挖掘最近变得很流行，很多人都投身到数据挖掘的研究、开发和商业中，并宣称它们的软件系统是数据挖掘产品。观察一下就会发现，“数据挖掘是宣传出来的还是真正存在的？它怎样被人们作为一种技术很好的接受？”

坦白地说，数据挖掘从二十世纪八十年代出现以来关于它的宣传有很多，尤其是许多人希望数据挖掘能成为一种从数据中挖掘知识的工具，使它能帮助企业经理作决策，促进商业竞争，或者做其它很多有趣的事情。

数据挖掘是一种技术，和其他技术一样，数据挖掘也需要时间和精力来研、开发和逐步成熟，最终被人们接受。整个生命周期应包含下列几个阶段（图 10。9）：

- 创新者（innovator）：　研究者开始认识到需要找到解决某个问题的方法时，新的技术就开始出现了。
- 早期接受者（early adopter）：　当关于这项技术提出的方法越来越多的时候，人们对它的兴趣就相应增长了。
- 停滞（chasm）：　一种技术被作为主流技术广泛接受之前必须碰到的障碍或挑战
- 早期多数接受者（early majority）：　这种技术成熟并被广泛接受和应用。
- 后期多数接受者（late majority）：　这种技术被广泛接受，但由于初始的问题，人们对它的兴趣减小，它或者变得不重要，或者被其他需求取代。
- 落后（laggards）：　因为过时，这种技术开始消失。

创新者　早期接受者　停滞　早期多数接受者　后期多数接受者　落后

图 10。9 技术采纳（technology adopter）的生命周期

“那么，数据挖掘正处于哪个阶段？”最近有些讨论认为数据挖掘正处于停滞阶段。为了让数据挖掘成为一种广泛接受的技术，本书中作为挑战提到的很多地方（如有效性和可伸缩性，增加用户交互，和背景知识及可视化技术的结合，数据挖掘语言标准的发展，发现有趣模式的有效方法，提高复杂数据类型的可控制性，web 查询等等）都需要做进一步的研究和开发。

数据挖掘要走过停滞期，我们需要关注数据挖掘和现存商业技术的集成。目前已经有很多通用的数据挖掘系统，但是，他们中的很多都是给那些非常熟悉数据挖掘和数据分析技术的专家设计的，他们需要懂关联规则，分类和聚类等技术。这就使这些系统很难被企业经理或普通百姓使用。而且，这些系统都趋向于提供适用于各种商业应用的横向解决方案（horizontal solution），而不是针对某个特定商业应用的解决方案。由于有效的数据挖掘要求商业逻辑与数据挖掘功能的平滑的集成，所以我们不能期望通用的数据挖掘系统在商业智能方面能够取得多么大的成功，就象与领域无关的关系数据库系统在商业事务和查询处理上取得的成功一样。

许多数据挖掘研究者和开发者相信：数据挖掘比较有前途的方向是创建能够提供纵向解决方案(vertical solution)的数据挖掘系统，也就是，把特殊领域的商业逻辑和数据挖掘系统集成起来。由于越来越多的公司从建立在 Web 上的电子商店（e-store）(也称为 Web 商店（Web store）)收集大量的数据，网上商业或电子商务成为数据挖掘很有前途的应用。因此，我们应该仔细琢磨怎样给电子商务应用提供特殊领域的数据挖掘解决方案。

目前，许多定制的系统需要具有面向市场竞争管理（通常叫电子市场（e-marketing））的功能。理想情况下，这样的系统能同时提供客户数据分析(把 OLAP 和挖掘技术嵌入到友好的用户界面中)，客户个性分析（profiling）(或一对一片段（one-to-one segment）)，竞争出局，和竞争分析。

这些系统日益增加使用数据挖掘来进行客户关系管理（CRM），以期在大规模的市场中帮助公司给他们的客户提供更特殊的个人化的服务。通过研究 Web 商店的浏览和购买模式（比如通过分析点击流（clickstreams），也就是用户通过鼠标点击提供的信息），公司能够得到更多的关于某个用户或用户组的信息。这些信息将使公司和客户同时受益。例如，如果有非常准确的客户模型，公司可以更好的理解用户的需求。满足这些需求将会在很多方面取得更大的成功，如相关产品的连带销售（cross-selling）、提高销售额(up selling)、一对一促销(one-to-one promotion)、产品吸引力(production affinity)、一揽子购买(larger basket)、客户保持（customer retention）等。如果所做的定制广告和促销正好满足客户个性需求的话，客户就很少会对发到他信箱里的垃圾邮件感到腻烦了。所有这些活动能为公司节约很多花销。客户将会喜欢你通知他购买他真正感兴趣的东西，从而节约个人时间，获得满意的服务。公司除了在网上商店散发广告外，将来还可以在数字电视和在线图书以及报纸上提供广告。

这些广告是通过客户个性信息和统计信息为某些特定的用户或用户组专门设计的。

明确数据挖掘只是集成解决方案的一项内容是很重要的，其它还有数据清洗和数据集成，OLAP，用户安全，库存和订单管理，产品管理等等。

10．4．2 数据挖掘只是经理的事还是每个人的事？

数据挖掘在帮助公司经理理解市场和商业上面作用很大，但是，“数据挖掘只是经理的事还是每个人的事？”随着越来越多的数据可以从网上或者你自己的磁盘上得到，在日常工作或生活中，利用数据挖掘来理解你访问的数据并从中受益是可能的。而且，随着时间的推移，会出现更多的数据挖掘系统，它们功能更强，用户界面更加友好并且更加多才多艺。因此，每个人都具有使用数据挖掘的需求，并且具有使用他们的手段是可能的，换句话说，数据挖掘不可能一直只被由经理和商业分析者组成的传统知识分子使用，每个人都将可以得到它。

“我在家里用数据挖掘能做些什么呢？”，数据挖掘能具有很多个人用处。例如：你可能想挖掘你们家的医学史，确定出和遗传有关的医学条件的模式，比如癌症和染色体变异，这些知识能帮助你决策你的寿命和健康状况；将来，你可能挖掘和你打过交道的公司的记录并且评价他们的服务，在此基础上选择最好的公司进行合作；你可以用基于内容的文本挖掘来查找你的 E-mail 消息，或者自动地创建分类来管理你收到的消息；你可以通过挖掘股票或公司的业绩来辅助你进行投资；其它的例子包括通过挖掘网上商店来找出最好的交易项目或最好的休假方式。这样，当数据挖掘走出低谷，变得更加普通，有更多的个人计算机和网上数据，数据挖掘将被普通大众所接受，并最终成为

每个人手中的工具。

“那么，在使用数据挖掘之前我必须理解数据挖掘系统和数据挖掘算法的内容么？”，就象电视、计算机、办公软件一样，我们希望用一种用户友好的数据挖掘工具，而不需太多的培训。而且将会有更多的智能软件隐含地把数据挖掘作为他们的功能部件，例如：智能网上搜索引擎，适应用户的网上服务，智能数据库系统，协同查询应答（cooperative query answering）系统，e-mail 管理器，日历管理器，售票系统，等等，他们可以把数据挖掘模块作为他们内部的模块，用户根本感觉不到它的存在。数据挖掘的这种隐含的应用叫做不可见的数据挖掘(invisible data mining)。期望将来不可见数据挖掘能成为普通大众执行有效数据挖掘的重要的手段。

10．4．3 数据挖掘对隐私或数据安全构成威胁么？

随着越来越多的信息以电子形式或从网上得到，并且有越来越多的数据挖掘工具开发出来并投入使用，我们可能想知道，“数据挖掘对隐私或数据安全构成威胁么？”数据挖掘和其它任何一种技术一样，它的应用有好的一面也有坏的一面。因为数据挖掘揭示不容易发现的模式或各种知识，如果不正确使用的话它可能对隐私和信息安全构成威胁。

有些消费者为了使公司的服务更好地满足他们的需求，不介意给公司提供个人信息，例如，购物者如果能得到打折回报的话，他们将很乐意在地区超市的荣誉卡上签字。

如果你停下来想一下，记录了多少关于你的信息，这些信息都说了些什么？每次在你使用信誉卡、赊帐卡（debit card）、超市荣誉卡、宣传卡(frequent flyer card)，或申请这些卡的时候，当你在网上冲浪、回答网上新闻组、订阅杂志、租影碟、参加俱乐部、或考试登记表、填写新生儿信息、付药方费用的时候，或者看病时提供你的医疗卡的时候，关于你的个人信息就会被公司收集到。很明显，收集信息很容易，并不局限于通过零售活动来进行，它可以反映出用户的爱好，财力，医疗，和保险数据。下次做上面类似的事情的时候，可以仔细想一想，你可能有被人监视的感觉。

如 10。4。1 节所描述的，个人数据的收集证明对企业和消费者有利，但也有被误用的问题。如果这些数据用作其它的目的，例如：可帮助保险公司根据你购买的食物来确定你的脂肪消费水平？超市可以用荣誉卡来指证一个跌跌撞撞的购物者为酗酒者（基于他购买的酒的数量）。这些例子只是用来说明客户不经意泄漏的数据可能反过来对他本人构成侵害。

考虑上述问题的时候，你可能想知道：

- “我什么时候给公司提供过我自己的信息，这些数据会被用于了我所不希望用到的地方吗？”
- “这些数据被卖给别的公司了吗？”
- “我能发现记录的关于我的信息是什么吗？”
- “我怎么能知道哪个公司有关于我的信息？”
- “我有权利和方法拒绝公司使用我的个人数据吗？”
- “有什么手段可以修改我的个人数据中的错误？如果我想删除，完善，增补或更新数据怎么办？”
- “关于我的信息可以“匿名化”，或者可以跟踪处理吗？”
- “怎样保证数据的安全”
- “公司如何对收集到的数据负责，如果丢失或误用怎么办呢？”

这些问题没有简单的答案。有关的国际性准则，著名的**公平信息实践**（fair information practices），就是专注数据隐私保护，它涵盖了数据收集，使用，质量，开放，个体参与（individual participation），责任等方面内容，它包含下列原则：

- 目的说明和使用限制　收集数据的时候必须指定数据的使用目的，不能超出此目的范围使用数据。数据挖掘是一种典型的使用收集到的数据另做它用的行为。有人提出过这样的建议，即对允许用于数据挖掘的数据附加一个“放弃”的申明，但因为意图过于暴露而未被广泛接受。由于数据挖掘具有的暴露本质（exploratory nature），不可能知道什么模式该发掘，什么不该挖掘；因此如何使用数据挖掘没有什么确定性。
- 开放性：人们有权利知道关于他们的什么样的信息被收集了，由谁来访问数据，以及数据怎样使用。

“那么，考虑这些问题的可能的解决方案是什么？”公司应该提供给用户多种选择，允许用户指定他们个人数据的使用限制，比如：（1）消费者的信息不允许用于数据挖掘；（2）消费者数据能

用于数据挖掘，但可以识别用户的信息或者可以导致识别用户的信息被泄漏的信息应该删掉；（3）数据只能用于内部数据挖掘；（4）数据可用于内部或外部数据挖掘；公司应该给用户积极的承诺，允许用户在他们的数据用于第二目的时进行选择，最好用户可以用免费的号码或者进入公司站点进行选择，可以对他们个人的数据进行访问。

“数据安全性怎样？”数据库系统最初曾遭到反对，因为在大型在线数据存储系统中，很多个人的数据面临着安全的威胁，许多数据安全增强技术（data security-enhancing techniques）因此而得以发展。尽管“黑客入侵”时有发生，但鉴于数据库管理系统带来的实惠，人们对数据的安全性比较放心，这样的数据安全增强技术同样可以用于数据挖掘中的匿名信息和隐私保护，这些技术包括盲签名（建立在公共密钥加密基础上），生物加密（人的肖像和指纹用于加密个人数据），匿名数据库（anonymous databases）（允许不同的数据库联合，但是只有那些需要访问数据库的人才可以访问数据库，个人信息被加密存储在不同的地方）。

数据挖掘可能对人们的隐私和数据安全构成威胁，然而，就像我们所看到的一样，为防止收集的数据误用已经提出了很多解决方案。而且，数据库系统中的数据安全增强技术也可以用在数据挖掘中，来保证收集到的数据或挖掘出来的数据的安全。虽然有些当今的数据挖掘技术有可能走不出低谷，但是鉴于对这种技术的强大的需求，数据挖掘肯定会成功的。随着公司和消费者的不断的共同努力，找到更多的保护数据隐私和安全的解决方案，数据挖掘一定能给我们带来更多的利益，可以节约我们的时间和金钱，并发现新的知识。

10．5 数据挖掘的发展趋势

鉴于数据，数据挖掘任务和数据挖掘方法的多样性，给数据挖掘提出了许多挑战性的课题。数据挖掘语言的设计，高效而有效的数据挖掘方法和系统的开发，交互和集成的数据挖掘环境的建立，以及应用数据挖掘技术解决大型应用问题，都是目前数据挖掘研究人员，系统和应用开发人员所面临的主要问题。本节描述一些数据挖掘的发展趋势，它反映了面对这些挑战的应对策略。

应用的扩展：早期的数据挖掘应用主要集中在帮助企业提升竞争能力。随着数据挖掘的日益普及，数据挖掘也日益扩展其应用范围，如生物医学，金融分析，和电信等领域。此外，随着电子商务和电子市场逐渐成为零售业的主流因素，数据挖掘也在不断扩展其在商业领域的应用面。通用数据挖掘系统在处理特定应用问题时有其局限性，因此目前的一种趋势是开发针对特定应用的数据挖掘系统。

可伸缩的数据挖掘方法：与传统的数据分析方法相比，数据挖掘必须能够有效地处理大量数据，而且，尽可能是交互式的。由于数据量是在不断地激增，因此针对单独的和集成的数据挖掘功能的可伸缩算法显的十分重要。一个重要的方向是所谓基于约束的挖掘（constraint-based mining），它是致力于在增加用户交互的同时如何改进挖掘处理的总体效率。它提供了额外的控制方法，允许用户说明和使用约束，引导数据挖掘系统对感兴趣模式的搜索。

数据挖掘与数据库系统，数据仓库系统，和 Web 数据库系统的集成：数据库系统，数据仓库系统，和 WWW 已经成为信息处理系统的主流。保证数据挖掘作为基本的数据分析模块能够顺利地集成到此类信息处理环境中，是十分重要的。如在 4。4 节所述，数据挖掘系统的理想体系结构是与数据库和数据仓库系统的紧耦合方式。事务管理，查询处理，联机分析处理，和联机分析挖掘应集成在一个统一框架中。这将保证数据的可获得性，数据挖掘的可移植性，可伸缩性，高性能，以及对多维数据分析和扩展的集成信息处理环境。

数据挖掘语言的标准化：标准的数据挖掘语言或其它方面的标准化工作将有助于数据挖掘的系统化开发，改进多个数据挖掘系统和功能间的互操作，促进数据挖掘系统在企业和社会中的教育和使用。近期在这方面的工作包括 Microsoft’s OLE DB for Data Mining（附录 A 提供了这方面的介绍）。其它工作见 4。2。7 的讨论。

可视化数据挖掘：可视化数据挖掘是从大量数据中发现知识的有效途径。系统研究和开发可视化数据挖掘技术将有助于推进数据挖掘作为数据分析的基本工具。

复杂数据类型挖掘的新方法：如第九章所述，复杂数据类型挖掘是数据挖掘中一项重要的前沿研究课题。虽然在地理空间挖掘，多媒体挖掘，时序挖掘，序列挖掘，以及文本挖掘方面取得一些进展，当它们与实际应用的需要仍存在很大的距离。对此需要进一步的研究，尤其是把针对上述数据类型的现存数据分析技术与数据挖掘方法集成起来的研究。

Web 挖掘：Web 挖掘的有关问题在 9。6 节讨论过。由于 Web 上存在大量信息，并且 Web 在当

今社会扮演越来越重要的角色，有关 Web 内容挖掘，Weblog 挖掘，和因特网上的数据挖掘服务，将成为数据挖掘中一个最为重要和繁荣的子领域。

数据挖掘中的隐私保护与信息安全：随着数据挖掘工具和电信与计算机网络的日益普及，数据挖掘要面对的一个重要问题是隐私保护和信息安全。需要进一步开发有关方法，以便在适当的信息访问和挖掘过程中确保隐私保护与信息安全。

10．6 总结

•针对特定领域的应用人们开发了许多专用的数据挖掘工具，这包括生物医学，DNA 分析，金融，零售业，和电信。这些实践将数据分析技术与特定领域知识结合在一起，提供了满足特定任务的数据挖掘解决方案。

•在过去 10 年中，开发了许多数据挖掘系统和产品。选择一个满足自己需要的数据挖掘产品，重要的一点是要从多个角度考察数据挖掘系统的各种特征。这包括数据类型，系统问题，数据源，数据挖掘的功能和方法，数据挖掘系统与数据库或数据仓库的紧耦合，可伸缩性，可视化工具，和图形用户界面。

•可视化挖掘集成可数据挖掘和数据可视化技术，用于从大量数据中发现隐含的和有用的信息。可视化数据挖掘的形式包括数据可视化，数据挖掘结果的可视化，和数据挖掘过程可视化。音频数据挖掘使用音频信号来指明数据挖掘结果的数据模式和特征。

•针对数据分析已经提出了几种完善的统计方法，如回归，广义线形模型，回归树，方差分析，混合效应模型，因素分析，判别式分析，时序分析，幸存分析，和质量控制。覆盖所有的统计数据分析方法超出本书范畴，感兴趣的读者可参考文献注解中引用的统计文献，可作为统计分析工具的基础。

•一些研究人员已在致力于建立数据挖掘的理论基础。这方面已经提出了一些有意思的成果，包括数据归约，模式发现，概率理论，数据压缩，微观经济，和归纳数据库。

•智能查询应答采用数据挖掘技术来分析用户查询的意图，提供与查询相关的概化和关联信息。这扩展了查询处理系统的能力和可用性。

•一种新技术如数据挖掘要得到认可，需经过一个生命周期，这中间通常包含一个沟坎，它表示了这种技术在成为主流技术之前必须面对的挑战。

•数据挖掘所带来的一种社会影响是有关隐私和信息安全的问题。Opt-out 策略是一种有关数据隐私保护的方法，它允许用户说明使用个人数据的限制条件。数据安全增强技术可以出于安全和隐私的考虑，将信息匿名化。

•数据挖掘发展趋势包括了需进一步研究的新应用的扩展，和处理复杂数据类型的新方法，算法的可伸缩性，基于约束的挖掘，和可视化方法，数据挖掘与数据仓库和数据库系统的集成，数据挖掘语言的标准化，以及数据隐私保护与安全。

习题

10.1 给出一个未在本章论及的数据挖掘应用的例子。讨论在此应用中如何使用各种不同的数据挖掘方法。

10.2 假设要在市场上购买一个数据挖掘系统。

（a） 考虑数据挖掘系统与数据库和/或数据仓库系统耦合方式，试述无耦合，松耦合，半紧耦合，紧耦合之间的区别。

（b） 行可伸缩性和列可伸缩性之间的区别是什么？

（c） 当选择一个数据挖掘系统时，在以上列出的诸多特征中，哪些是你要关心的？

10.3 考察一个现存的商品化数据挖掘系统。从多个不同角度来看，分析这一系统的主要特征，包括可处理的数据类型，系统体系结构，数据源，数据挖掘功能，数据挖掘方法，与数据库或数据仓库系统的耦合度，可伸缩性，可视化工具，和图形用户界面。能否对该系统提出一些改进意见，并概述其实现方法？

10.4 提出几种对音频数据挖掘的实现方法。可否将音频数据挖掘与可视化数据挖掘结合起来，使得数据挖掘生动而强大？可否开发一些视频数据挖掘方法？给出一些例子和解决方案，使得集成的音频-可视化挖掘有效。

10.5 通用计算机加上与领域独立的关系数据库系统在过去的几十年中，已形成为一个巨大的市场。然而很多人认为，通用的数据挖掘系统不会在数据挖掘市场成为主流。你的看法如何？对数据挖掘而言，我们应当致力于开发独立于领域的数据挖掘系统，还是应当开发特定领域的数据挖掘系统？请说出理由。

10.6 为什么说理论基础的建立对数据挖掘是十分重要的？列出并描述现已提出的数据挖掘的主要理论基础。评论一下每一种理论是如何满足（或不满足）数据挖掘的理想理论框架的要求。

10.7 直接查询响应与智能查询响应间的区别是什么？假设一用户要查询某渡假区的旅馆的价格，地址，和等级。举例来说明用直接查询响应与智能查询响应处理此查询的情况。

10.8 假设当地银行有一个数据挖掘系统。该银行已在研究你的信用卡的使用模式。注意到你在家庭装修店有多笔交易，银行决定与你联系，提供有关家居改善方面的特别贷款信息。

（a） 讨论一下这如何与你的隐私权相冲突。

（b） 给出另外一个使你感到数据挖掘侵犯你的隐私权的情况。

（c） 可否举出一些数据挖掘对社会有帮助的例子？你能想出一些对社会有害但可以使用的例子吗？

10.9 基于你现在已有的对数据挖掘系统和应用的知识，你认为数据挖掘会成为一个巨大的市场吗？数据挖掘研究与开发的瓶颈是什么？你认为目前数据挖掘的方法会赢得巨大系统应用市场份额吗？如果不是，你能提出一些建议吗？

10.10 基于你的研究，提出一个本章没有讨论到的数据挖掘新的课题。

文献注解

有很多讨论数据库应用方面的书。有关数据挖掘在生物信息中应用，包括生物医学和 DNA 数据分析，[。。。]很好的总结了这方面的有关方法和算法。对数据挖掘在金融数据分析和金融建模方面的应用，可参见[。。。]。有关零售业数据挖掘和客户关系管理，见[。。。]等人的书。有关电信业方面的数据挖掘工作见[。。。]的书。[。。。]报道了有关在数据仓库/OLAP 框架下的可伸缩的电信数据分析方面的工作。[。。。]讨论了科学知识发现中人机协同的问题。

许多数据挖掘的书籍中都包含了对各种数据挖掘系统和产品的介绍。[。。。]给出了有关数据挖掘和知识发现软件工具的综述。有关专用数据挖掘系统和产品的详细信息可从提供这些产品的公司网页，用户手册，以及数据挖掘和数据仓库方面的杂志和期刊中得到。例如，本章中介绍的一些数据挖掘系统的 URL 是[。。。]。由于数据挖掘系统及其功能发展迅速，我们无意在本书提供任何这方面的综述。因此若那一个数据挖掘系统或工具未被提及，我们表示歉意。

有关可视化数据挖掘，[。。。]的书是关于数据与信息可视化显示的畅销书。[。。。]总结了可视化数据技术。由[。。。]开发的 VisDB 系统用于数据库的挖掘，用到了多维可视化方法。Keim 在他的几个会议辅导材料中（[。。。]）总结了数据可视化和可视化数据挖掘方法。[。。。]提出了一种基于理解（perception-based）的分类方法，PBC，它是一种交互的可视化分类。

关于数据分析的统计技术已有好几本书籍介绍过，包括由[。。。]编著的 Intellligent Data Analysis(智能数据分析)；[。。。]编著的第四版[。。。]；[。。。]编著的[。。。]；[。。。]编著的[。。。]；[。。。]编著的第三版[。。。]；[。。。]编著的[。。。]；[。。。]编著的[。。。]；[。。。]编著的[。。。]。

有关数据挖掘的理论基础方面的问题见诸许多研究论文。Mannlia 在[Man97]中给出了数据挖掘基础理论研究方面的综述。有关数据挖掘的数据回归分析在[。。。]的报告。。。。中给出了总结。数据挖掘的数据压缩问题参见最小描述长度原理，如[。。。]。数据挖掘的模式发现理论在很多的机器学习和数据挖掘的研究中可以见到，因为模式发现的方法包括了决策树归纳，神经网络分类，关联挖掘，序列模式挖掘，聚类，等等。数据挖掘的概率论可以参看统计方面的文献，如第 7 章介绍的贝叶斯网络和层次贝叶斯模型。[。。。]提出了一种数据挖掘的微观经济学理论，把数据挖掘看成为一种优化问题。[。。。]提出了把数据挖掘作为归纳数据查询的观点。

很多学者研究了智能查询应答方法，包括[。。。。]。[。。。]研究了基于知识发现技术的智能查询响应。

由[。。。]编著的。。。讨论了电子商务和客户关系管理，对数据挖掘的未来提出了有意思的预见。[。。。]讨论了技术采纳生命周期问题，包括跨越鸿沟（crossing the chasm）的问题。Agrawal 在 KDD’99 的特邀报告中中论及了数据挖掘与技术采纳生命周期的问题。数据挖掘有关隐私和数据安全的问题有过几篇论文讨论，包括[。。。]（提出了一种允许个人信息买卖的规范的全国信息市场的概念），[。。。]

（讨论了公平信息实践和 opt-out 选择问题），[。。。]。[。。。]讨论了保持隐私的数据挖掘方法。

附录 A Microsoft's OLE DB for Data Mining 简介

本附录对 Microsoft's OLE DB for Data Mining(OLE DB for DM)规范[31]做一个非正式的简单介绍。OLE DB for DM 是朝着将数据挖掘语言标准化迈进的重要一步，其目标是成为工业标准。OLE DB for DM 可以使数据挖掘（DM）客户应用（或叫数据挖掘消费者（data mining consumer））使用由广泛的数据挖掘软件包（或称为数据挖掘提供者（data mining provider））提供的服务。

OLE DB for DM 描述了数据挖掘过程中的抽象概念。作为 OLE DB 的扩展，它引入了一个新的虚拟对象，称为数据挖掘模型(Data Mining Model, DMM)，并定义了对 DMM 进行操作的语句，这些语句在形式以及功能上类似于 SQL 语言，主要有以下三种语句：

1、 创建数据挖掘模型对象：用 CREATE 语句创建 DMM 对象，该语句类似 SQL 中的 CREATE TABLE 语句。CREATE 语句定义了 DMM 中的列（例如，在挖掘过程中需要分析的属性）以及数据挖掘算法，稍后 DM 提供方使用该算法对模型进行训练。数据挖掘算法包括决策树、聚类（称为分段）和回归(为了预报)。CREATE 语句并没有定义 DMM 的内容（即要学习的图形结构），在下面的语句执行前，DMM 被认为是“空”的。

2、 向 DMM 中装入训练数据并对之进行训练：用 INSERT 语句装入训练数据，来生成模型，类似于 SQL 中的 INSERT INTO 语句。INSERT 语句使得 DM 提供方用 CREATE 语句中指定的算法对新装入的训练数据进行处理。经过处理后的结果模型（或抽象）替代原有的训练数据存入 DMM。其结果称为 DMM 内容。

3、 使用数据挖掘模型：用 SELECT 语句查询 DMM 的内容，用于作出预报或浏览由模型得出的统计结果。

在关系数据库中，被挖掘的数据被表示为一组表。属于某个单一实体的数据被叫做案例(case)，相关的一系列案例被叫做案例集(case set)。像 Microsoft Data Access Components(MDAC)产品中的 Data Shaping Service 那样，OLE DB for DM 提供了嵌套表（或表值列）。利用嵌套表，对给定的实体可以有记录集，而不限于单一的记录。例如，对于顾客实体，可以有 Customer ID（顾客标识）、Gender（性别）、Age（年龄）、Item Purchases（购买商品）等属性，而 Item Purchase 由一个嵌套表（Item Name（商品名），Item Quantity（商品数量），Item Type（商品类型））来描述，表示该顾客所购买的一系列商品，如表 A.1 所示。一个案例，可以有多个嵌套表，而每个嵌套表可以有不同数量的行。案例的主要的行被叫做案例行（case row），嵌套表中的行被叫做嵌套行(nested row)。

表 A。1 描述“customer(顾客)”实体的具有嵌套表的样本案例

P486

CD 机	家庭娱乐
	家庭娱乐
汽车报警器	安全

对以上的主要操作我们给出如下的例子。

A.1 创建 DMM 对象

例 A.1 创建一个用于分类的 DMM：以下的语句对用于年龄预测的 DMM 对象指定了列（或属性）以及用于其后训练的 DM 算法。

该语句包含以下的信息：Customer ID 被指定为关键字，意味着它可以唯一的标识一个顾客的案例行。Age 属性是模型所要预测的属性，它是一个连续型的属性，但要把它离散化。DISCRETIZED() 表示使用默认的方法进行离散化。也可以使用 DISCRETIZED(*method,n)*，*method* 是提供方提供的某

[31] 本附录提供的信息是基于 OLE DB for Data Mining, Draft Specification, 9.0 版, 微软公司, 2000 年 2 月. 本附录未提到的其它信息可参见该文档或其最新版本.

种离散化方法，*n* 是将年龄范围值划分为若干桶或间隔的推荐个数。Item Purchase 是一个嵌套表，包含列 Item Name（表 ItemPurchase 的关键字）、Item Quantity 和 Item Type。某些 DM 提供方可以利用连续型属性的概率分布知识。Item Quantity 属性具有正态分布。其他的分布模型包括 **UNIFORM、LOGNORMAL、BINOMIAL、MULTINOMIAL、POISSON** 等。在定义中，*Item Type RELATED TO Item Name* 的意思是 Item Name 按照 Item Type 来分类。例如，“TV”可分类为“家庭娱乐”。还有其他一些属性类型没有出现在上例中，像 **ORDERED、CYCLICA、SEQUENCE_TIME、PROBABILITY、VARIANCE、STDEV 和 SUPPORT**。USING 子句定义了通过 INSERT 语句装入数据时，提供方应使用决策树算法构造模型。该子句也可附加一个提供方特殊对（provider-specific pair），用以说明算法所使用的参数设置。

例 A.2 创建一个用于关联规则的 DMM：以下的语句创建了一个用于关联规则挖掘的 DMM 对象：

Minimum_size = 3 意味着仅对经常一起出售的商品，并且商品的个数至少为 3 个的关联规则进行挖掘。

A.2 向模型中装入训练数据并对模型进行训练

下面的语句给定了用于构造 Age Prediction 模型的训练数据。DM 算法（在定义该模型时指定）被启动，对输入的数据进行分析，并生成模型。

生成 DMM 的方式类似于生成一个普通的关系表。因为 DMM 并不使用数据源中的第一列，所以在 INTO 子句中出现了 SKIP 关键字。注意，SHAPE 命令用于创建嵌套表 Item Purchase。

A.3 模型的使用

经过训练的模型具有“真值表”的形式，它对 DMM 每列（属性）的所有值的可能组合都有对应的一行。对离散属性，要考虑到属性的所有离散值（或桶）。（使用每个桶的中间值）。对连续型属性，要考虑最小值、最大值以及平均值。每一种属性类型都包含了“missing（空缺）”标签。通过对这张表的浏览，可以作出预测或查询学习到的统计值。本小节给出了使用 DMM 的几个例子，包括预测、查询或浏览全面的内容。

下面的 OLE DB for DM 语句说明了 SELECT 命令用于预测的情况。

该语句采用了 PREDICTION JOIN 对 DMM 中的所有可能的案例与指定的案例集（年龄属性的值未知）进行连接。SELECT 语句对连接的结果进行操作，返回每个客户的预测年龄。注意，对一个给定的测试案例，PREDICTION JOIN 可能发现一组案例满足 ON 子句中的条件，如果发生这种情形，这些案例就“倒塌”聚合为一个聚集案例，它包含了对 Age 的最佳预测（或为模型中所有可预测列的最佳预测）。SELECT 语句也可以检索测试案例的已知 Age 值，用于检验模型的准确性。

DMM 的“真值表”含有属性值的各种可能的组合，可以从中查询各种各样的值和统计结果。例如；如果在 Age Prediction 模型中包含像头发颜色这样的属性，则可以用下面的语句查找到所有不同的头发颜色：

SELECT DISTINCT Hair Color FROM [Age Prediction]

类似的，所有可以购买的商品清单可以由下面的语句获得：

SELECT DISTINCT [Item Purchases].[Item Name] FROM [Age Prediction]

注意，在嵌套表的里，可以用点操作符（“.”）表示嵌套表的一个属性。

OLE DB for DM 提供了大量的函数用于统计性的描述预测结果。例如，一个预测值的可能性可以用函数 PredictProbability()来刻画，如同下面的例子，它返回一个表，包含每个客户的预测年龄以及相应的概率：

SELECT [Customer ID],Predict(Age),PredictProbability([Age]),…

可以类似的用函数 Cluster()和 ClusterProbability()去刻画每个簇(Cluster)和其相应的概率，如下例：

SELECT [Customer ID],[Gender],Cluster() as C,ClusterProbability() as CP,…

上例中 C 是一个簇的标识符，标识某个案例最可能属于哪个簇，CP 标识相应的概率。其他的以列名作为参数的函数还有 PredictSupport()它返回出现该列值的案例的个数，PredictVariance(),PredictStdev(),PredictProbabilityVariance() 和 PredictProbabilityStdev()。函数 RangeMid(),RangeMin(),RangeMax()分别返回对 DISCRETIEZD 属性的被预测桶的中间值，最小值，最大值。

函数 PredictHistogram 返回要预测或聚类的列的直方图，该图用嵌套表的形式显示了该列各种可能的值以及对应的统计值。例如，预测一个像性别这样的离散型属性，选用直方图显示每个案例的性别属性各值的支持度以及概率。该信息可用以下的语句可以实现：

其他的返回值包括$Variance,$Stdev,$ProbabilityVariance 和$ProbabilityStdev。如果以 Cluster()作为 PredictHistogram()的参数，则直方图中显示每个案例的所有可能的簇的标识符和对应的支持度、概率等。另外，OLE DB for DM 提供了查看每一个案例嵌套表的前或后 N 行的函数，这样的函数在每个案例的嵌套表都有大量的行时非常有用。

DMM 表除了包含有关预测涉及的属性值的所有可能组合外，DMM 的内容也可包括一组结点（比如，决策树）、规则、公式、或分布。这依赖于使用的 DM 算法如何。可以从 DMM 中抽取 XML 描述，以字符串的形式描述了 DMM 的内容。要解释这样的字符串，客户应用程序需要专门的技术。对以有向图（如决策树）表示的内容还提供了导航操作。通过挖掘而发现的规则以 PMML(Predictive Model Markup Language，预测模型标记语言)形式表示，可以通过查询观察其内容。

附录 B　DBMiner 简介

DBMiner 是一个数据挖掘系统。它起源于 Intelligent Database Systems Research Laboratory，Simon Fraser University,British Columbia,Canada,由 DBMiner Technology Inc.做了进一步的开发而形成的产品。它是在数据挖掘和数据库中的知识发现领域多年研究成果的结晶。本附录对 DBMiner 做一简短的介绍。

DBMiner 是一个联机分析挖掘系统，用于在大型关系数据库和数据仓库中交互的挖掘多层次的知识。其独特之处在于紧密集成了联机分析(OLAP)和多种数据挖掘功能，包括特征、关联、分类、预测、和聚类[Han98,HCC98,HF96]。这种集成开创了大有前途的数据挖掘方法学--联机分析挖掘（OLAM）：系统提供了对数据进行多角度观察，交互挖掘的环境，用户可以动态的选择数据挖掘和联机分析功能，可以对挖掘的结果进行 OLAP 操作（如钻取、切块/切片、旋转），也可以对 OLAP 的结果进行挖掘，即在多个抽象层对数据的不同的部分进行挖掘。

通过实现了一系列先进的数据挖掘技术，系统易于对多维数据库进行基于查询的交互式数据挖掘。这些技术包括：基于 OLAP 的多维统计分析，高效的频繁模式（frequent-pattern）挖掘算法，逐步加深的挖掘精确知识，可视化的数据挖掘，元规则导引的挖掘，数据和知识的可视化。DBMiner 实现了与关系数据库和数据仓库的平滑集成，提供了一个对用户友好的、交互的、高性能的数据挖掘环境。

B.1 系统结构

DBMiner 的系统结构遵循了图 2.24 建议的联机分析挖掘结构。它从关系数据库和（或）数据仓库中抽取数据，通过集成和转换装入多维数据库（数据的一部分或全部被合并到数据立方体(data cube)），然后，根据用户的处理请求进行联机分析和联机分析挖掘。

结构的核心模块是 OLAM 引擎，它以类似于 OLAP 引擎进行联机分析的方式在多维数据库中进行联机分析挖掘。DBMiner 的 OLAM 引擎能完成多种数据挖掘任务，包括概念描述、相关、分类、预测、聚类和时间序列分析。

更重要的是，系统集成了 OLAM 和 OLAP 引擎，它们能通过图形化的用户接口 API 接受用户的联机查询(或命令)，通过 MDDB_API 对多维数据库进行分析挖掘。在这种结构中 OLAP 和 OLAM 引擎是交互作用的，因为 OLAM 引擎可以对 OLAP 的结果进行挖掘，OLAP 引擎可以对挖掘结果进行分析。元数据字典存储了数据库模式、数据仓库模式和概念层次信息，它用于指导对多维数据库的存取以及执行维相关的 OLAP 操作，如钻取和切片。多维数据库的构造可以通过存取数据库、过滤数据仓库和（或）集成多种类似资源，对数据库和数据仓库的存取可以用数据库 API 来实现（目前 Microsoft SQL Server 7.0 OLAP Manager 支持这种 API）

DBMiner 的图形用户界面见图 B.1

图 B。1 DBMiner 的图形用户界面

B.2 输入和输出

DBMiner 从 SQL Server OLAP 的数据立方体中取数据，立方体中的数据来自一个或多个关系表，来自数据仓库和或其他形式的数据如电子表格。

根据不同的数据挖掘任务和不同的用户需求，系统可用多种形式表示获得的知识。数据汇总（summarization）和特征的输出利用 Microsoft 2000 以交叉表，综合规则，条形图，饼图，曲线以及其他的图形化工具表示。相关用相关规则表，相关计划和相关规则图表示。分类用可视化的决策树和决策表表示。簇用地图来表示（对于二维分析而言），每个簇用不同的颜色标识出它们的轮廓。

系统提供了观察概念层次和数据立方体内容的工具。概念层次用类似于目录/子目录结构的树来表示。数据立方体的内容用三维的形式表示，立方体的每个单元的大小和颜色表示了在一个三维间隔中所选的测试值的汇总数据。二维表可被看作二维的 boxplot,每个 boxplot 表示了相应间隔的数据

离差（dispersion）视图(包含中值，第一个四分点(first quartile),第三个四分点，须状孤立点(whiskers outliers))

系统的一个重要特征是具有对输出的知识进行诸如钻取，切块以及转换等操作的灵活性。例如，在对一个多维和层次的组合进行相关规则的挖掘后，可以对任一维进行钻取，以便在新的数据集中得出相关规则。

B.3 系统支持的数据挖掘任务

DBMiner 系统支持以下的数据挖掘任务：

- OLAP 分析器。这个功能是通过钻取，切块，切片和其他的 OLAP 操作，从不同的角度，多个抽象层次展现数据立方体中的内容。其输出可以用多种多样的可视化或图形的形式表示。此外，借助数据离差(dispersion)分析得到最大值，最小值，标准差以及其他分布情况可以作为 OLAP 数据的注解。OLAP 可以对综合数据感兴趣的部分进行钻取、切块以便做进一步的分析。图 B.2 以三维的形式表示了一组汇总数据。
- 关联。该功能从多维数据库中挖掘一系列关联规则。这样获得的规则可用于市场分析，关联分析等。用户可以指定元模式(metapatterns)以限制对规则的搜索。例如指定元模式：major(S:student,X) ∧ P (S,Y) => grade (S,Y,Z) ,其中，major 和 grade 是关系 student 的属性上的谓词，S 是一个变量，表示学生，P 是一个谓词可实例化为 student 上的一个属性，X,Y,Z 取允许的值。也可以沿着任一维在多个抽象层次上进行规则的挖掘。图 B.3 用关联球图表示关联规则，球表示项目(item)，箭头表示规蕴涵。
- 分类。该功能对一组训练数据（即一组已经确定分类的对象）进行分析，根据数据的特性，对每一个分类构造一个模型，再根据测试数据对模型进行调整。用决策树或决策表来表示模型，并利用模型对其他数据分类，以便更好的理解数据库中的数据。图 B.4 用决策树的形式表示了一个分类结果。
- 聚类。 该功能将一组选定的数据对象，分成若干的簇，使之簇内的数据相似度高，而不同簇中的数据相似度小。高维聚类也可以在多维数据库中完成。图 B.5 是一个数据聚类结果，在图中，二维聚类的每一个聚类用不同的颜色和形状表示。
- 预测。该功能对一组选定对象的丢失或未知数据的值或值的分布进行预测。这涉及到选择一组与感兴趣的属性相关的属性（借助于某些统计分析），一组与选定对象类似的数据，然后作出值分布的预测。例如，一个雇员的可能的工资可以根据公司中与他相似雇员的工资分布而作出预测。
- 时间序列分析。这个模块包括若干个分析功能，像相似性分析，周期分析，序列模式分析，趋势和背离分析。

图 B.2 以三维 立方体形式表示的汇总数据

图 B.3 用一组球和箭头表示的关联规则

图 B.4 用决策树的形式表示的分类

图 B.5 用一组不同的形状标记表示聚类的结果

DBMiner 2.0 版包括 OLAP 分析器，分类，相关和聚类模块，预测和时间序列分析将在以后的版本中实现。

B.4 对任务和方法选择的支持

DBMiner 支持任务和方法选择，通过一个基于窗口的图形用户界面，用户使用挖掘向导选择所需的任务，与挖掘结果进行交互在其他的维及层次上进行挖掘。根据用户的输入，系统产生一个挖掘查询供用户检查，该查询是用一个像 SQL 语言的 DMQL 语言书写的，如果需要的话，在提交执行之前，用户还可以修改查询语句。

B.5 对 KDD 处理过程的支持

由于 DBMiner 是与数据仓库一起工作的，若有必要的话，某些知识发现的先期处理工作可以由底层的数据仓库系统完成，这些工作包括数据清洗，数据集成，数据综合。数据的选择由 DBMiner 作为挖掘查询的组成部分来完成。

在 DBMiner 中，对挖掘出的知识进行后期处理的大部分工作被集成到数据挖掘过程，这是因为数据挖掘查询不但指定了与任务相关的数据和挖掘任务，而且也指定了兴趣测量值（例如像支持度，信度，噪声等挖掘阀值）和期望的规则模式。数据挖掘和模式评价的集成减少了搜索空间，使用户将精力集中到挖掘过程。

B.6 主要应用

DBMiner 作为一个通用的联机分析挖掘系统，可用于在关系数据库和数据仓库中的联机分析和数据挖掘。该系统已经应用于从中等规模到大规模的关系数据库，具有快速的响应时间。

作为 DBMiner 的扩展，已经研制了若干专业的数据挖掘系统原型，如 GeoMiner,MultiMediaMiner，WeblogMiner。

B.7 现状

虽然 DBMiner 已经从一个研究系统原型转化为一个产品，但是它在新技术方面的创新与进步仍与大学研究实验室紧密相连。

DBMiner 需要的最小硬件配置是奔腾 550，64MB RAM ，运行 Windows/NT。DBMiner 可以直接连接到 Microsoft SQL Server 7.0 或者通过 Microsoft SQL Sever OLAP Manager 实现访问多种关系数据库系统 。

DBMiner 2.0 可以从http://db.cs.sfu.ca/DBMiner 或 http://www.dbminer.com 免费下载，有 90 天的试用期。有关单用户、用户组、教育用户的许可证可以从http://www.dbminer.com获得。