据的结构发现上，将描述同一类对象的共有结构按照相似度聚集到一起，然后从类中发现满足用户兴趣度的结构。与同类算法比较，structure_discovery 为增量式算法，同时，聚类技术的使用提高了结构发现效率。

分布式数据开采系统框架描述了一个灵活的、具有交互性、扩展性的数据开采系统。

关键词：数据开采；聚类分析；关联规则；分类数据；高维数据；半结构化数据；结构发现

ABSTRACT

Data Mining is to discover the implicit, previously unknown, and potentially useful knowledge (patterns or relations) from large amount of data sets by using modelization. Data mining research has drawn on a number of other fields such as database system, artificial intelligent and statistics etc. Clustering analysis has been studied in statistics and recognized as an important area of data mining research. As a data mining task, clustering analysis identifies clusters, according to some measurement, in a large, multidimensional data set. A good clustering method obtains a partition of the objects into clusters such that the objects in a cluster are more similar to each other than to objectes in different clusters. Clustering analysis in data mining deploys many traditional methods. All these methods have not been considered large volume data sets. However, to efficiently obtain knowledge from large amount of data sets is the top-leading problem in data mining area. In addition, traditional clustering analysis has maily focusd on numeric data rather than other types of data that exists in real world. Therefore, clustering analysis in data mining aims at improving efficiency of algorithm and ability of processing variant types of data such as document, categorical data and high-dimensional data.

Following are several efficient clustering algorithms dealing with different types of data sets for real world application.

Existing algorithms for clustering categorical data require several passes over the databases, and obviously the role of I/O overhead is significant for very large databases. CCDCS (Clustering Categorical Data using Clustering Summary) is an algorithm for clustering categorical data. It improves the efficiency by using compress technology to reduce the size of original data sets, and it also presents a new similarity criterion for compressed categorical data. CCDCS only needs to scan the database one time. The experimental results show that CCDCS performs better than other algorithms for clustering categorical data. The compress technology does not affect too much the quality of clustering result。

An algorithm CHID (Clustering HIgh-dimensional Data) is proposed for high-dimensional data sets. The bottleneck of distance-based methods in clustering high-dimensional data sets is calculating the distance between data points. Instead of distance calculation, CHID searches the dense units in *n*-dimenaion space and subspace

from both bottom-up and top-down directions in the meantime, then it clusters these dense units by using bitwise AND. The search strategy reduces search space to improve efficiency and the only use of bitwise AND and bit-shift machine instructions in clustering make the algorithm more efficiency.

Two algorithms are proposed to apply clustering analysis to other fields. One is MARC. It integrates clustering into association rules discovery to reduce the size of data sets. It also uses *CS* (Clustering Summary) transformation to alleviate the loss of information brought by the compression. The other is structure_discovery. It finds the common structure of a class of objects in hierarchical, semi-structured data by using clustering. Then the algorithm learns the structure that satisfies users interest.

At last, a framework of a distributed data mining system describes a flexible interactive and scalable data mining system.

Keywords: data mining; clustering analysis; association rules; categorical data; high-dimensional data; semi-structured data; structure discovery

1 综述

1.1 引言

一切新事物的产生都是由需求驱动的，让计算机能够自动智能地分析数据库中的大量数据以获取信息是推动数据开采技术产生并发展的强大动力。例如，股票经纪人需要从日积月累起来的大量股票行情变化的历史记录中发现其变化规律以供预测未来趋势之用；超级市场的经理人员希望能从过去几年的销售记录中分析出顾客的消费习惯和行为，以便及时变换营销策略；地质学家想通过分析地球资源卫星发回的大量数据和照片来发现有开采价值的矿物资源，等等[1]。有一个例子可以说明数据开采技术是如何对商业营销产生积极影响的：假设甲是移动电话服务供应商 A 的客户，但是他最近决定中断和 A 的合约，原因是另一个移动电话服务供应商 B（A 的竞争者），向他免费提供一种新型号的移动电话，这种新型号的移动电话比 A 向他提供的要先进，这对供应商 A 来说当然不是什么好消息，而且如果供应商 A 事先知道供应商 B 的营销策略会对甲有效，他会事先采取相应措施，如决定增加对甲的投资，只需多加 100 元升级手机（而甲的每月话费为 350 元）就可以挽留住一位有价值的客户。那么如何能够事先知道客户甲会中断与 A 的合约呢？答案是：在供应商 A 的客户数据库中使用数据开采技术，建立一个预测模型显示这些容易更换服务商的风险人群。通过这个例子可以看到，数据开采能为决策者提供重要的、极有价值的信息或知识，从而产生不可估量的经济效益[2]。

数据开采能够帮助人们从大型数据库中提取有用的信息和知识。数据开采技术的意义在于所操作的对象是包含海量数据的大型数据库，对于小规模的数据库，人们不需要新的技术用以发现有用信息。例如，一个传统的小杂货店，它的顾客数量大约为一两百人，店主可以叫得出他们每个人的姓名，甚至他们的职业和购买习惯也了然于胸，所以不需要借助任何分析工具，直接利用人的大脑就可以进行数据分析及得出预测模型；而今天的大型超市和连锁店拥有数以万计的顾客，每天产生上百万条交易记录，仅依靠人的大脑记忆及分析这些数据显然是不可能的，有关顾客的所有数据都累积存储到数据库中，当数据量累积到一定程度时，原有的数据库工具已经无法满足用户的需求，用户不仅需要一般的查询和报表工具，更需要的是那些能够帮助他们从浩瀚的数据海洋中提取出高质量信息（预测性）的工具，数据开采的出现和发展正符合了这一潮流。

1.2 数据开采

1.2.1 定义及过程

有关数据开采 DM（Data Mining）或数据库中的知识发现 KDD（Knowledge Discovery in Database）有多种定义，其中比较著名的如下：

1. 数据开采或数据库中的知识发现（KDD）指从大量数据中提取隐含的、事先未知的、潜在有用的信息的非平凡过程。——William J Frawley, Gregory Piatetsky-Shapiro and Christopher J Matheus[3]
2. 数据开采指从大型数据库中搜寻那些隐含的模式和数据关系，如患者数据和医疗诊断之间的关系，这些关系或模式蕴含了数据库中一组对象之间的特定关系，揭示出一些有用的信息。——Marcel Holshemier & Arno Siebes (1994)
3. 数据开采指利用多种技术从大量数据中识别并提取信息或决策知识以应用到决策支持、预测估计等领域。——Clementine User Guide, a data mining toolkit

这些定义从不同的角度说明了数据开采的任务特征。① 用于数据开采的数据集合规模巨大；② 数据开采是一个多步骤的处理过程；③ 数据开采综合了多种技术，可发现不同类型的知识；④ 开采出的结果直观易用。

数据开采还有一些其它名称，如：数据挖掘、数据考古、知识提取、数据捕捞、数据分析等[4]。

数据开采利用不同技术通过建立模型揭示隐藏在大量数据中的知识（模式和关系），但是，要进行复杂的智能数据开采，并不是简单地把一些数据代入模型进行计算，然后就可获得有意义的结果。它需要一个严谨的系统过程，一般包括确定问题、准备数据、数据开采、使用与监测四个步骤。

1. 确定问题：为了充分利用数据开采，客户必须明确表述出目标是什么，不同的目标需要不同的模型。例如某客户期望其用户能够对发送的邮寄宣传资料产生积极回应，对于两个不同的目标——“提高用户对于宣传资料的回应率”和“提高用户的回应质量”——将产生完全不同的数据开采模型。一个好的目标应该明确要解决的企业问题，并有可度量的结果，这些结果能够引发明智的行动，为企业提供有效的帮助。这样的目标还应该包括决

策者可能做出的各种错误预测所造成的损失以及正确预测所带来的效益。

2. 数据准备：有效的数据准备工作应该包括数据收集、数据预处理、数据选择和数据转换等步骤，有时还需要反复地重复这些步骤。

   (1) 数据收集：这一步确定所需要开采的数据源。如果所需要的某些数据从来都没有收集过，这时就需要一个数据收集阶段，这一阶段可能需要从公共数据库或专用数据库获取外部数据（例如人口普查或气象数据）。

   (2) 数据预处理：这一步是一个数据清洗过程。当数据来自多个数据源时，有必要把它们合并成一个单一的数据库，为此，可能需要着手处理不一致的数据定义、不同的数据编码、同一个数据中不一致的项目值等问题。即使数据是来自单一的数据源，也需要仔细检验是否存在影响数据完整性的问题。除此之外，还需要考虑数据的来源、物理数据类型、度量单位，以及以什么样的频度、在什么时间和什么条件下收集数据等问题。从根本上说，必须确保所有的数据以同样的方式度量同一个事物。另外，在这一步要建立将要实施数据开采的数据库。

   (3) 数据选择：数据收集到位以后，就需要为数据开采任务选择恰当的数据。原则上，一些数据开采算法会自动忽略无关的变量，并适当地计算相关变量，但实际上，一般不会只依赖某一种工具来进行数据开采。

   (4) 数据转换：选择好数据以后，有时还需要做一些数据转换的工作，例如，可能需要用债务一收入比而不是把债务和收入都作为独立变量来预测信用风险。

3. 数据开采：这里的数据开采指的是使用某种或某几种数据开采方法和技术从准备好的数据中提取模式的过程。首先要确定模型，选择和使用开采算法开采出模式，然后将模式表达成容易理解和使用的知识。对知识的评估可能导致该过程重复执行。

4. 使用与检测：知识在使用后，必须严格考察模型的工作情况。因为随着时间的推移，数据开采所涉及的因素会有不同程度的发展和变化，这就需要经常重新测试、甚至可能是彻底重构模型。

1.2.2 相关领域

数据开采综合利用了其它学科或领域中某些成熟的方法和技术，涉及到以下

几个领域：机器学习[5]、统计学、联机分析处理[6]-[8](OLAP——On_Line Analytical Processing)等。

1.2.2.1 机器学习

机器学习是一个自学习过程，学习在这里的含义是通过观察环境状态的改变来构造规则，机器学习包括范例学习、强制学习和监督学习等。学习算法以数据集及其附属信息为输入并输出学习结果，如概念。机器学习通过分析这些样本及其结果学习如何重复该过程并且能够举一反三。通常，机器学习系统使用训练集——一个完整的有限集合——来完成学习，训练集中包含有观察陈述和样本数据，它们由一种机器能够理解的形式来表达。由于这样的训练集是一个有限集合，所以机器学习系统学习到的概念不可能是完全的。

数据开采与机器学习中有关范例学习部分有许多重叠部分，但还是存在差异：① 数据开采关心易于理解的知识的获取，而机器学习则关心方法性能的提升；② 数据开采面对的是海量的、现实世界中的数据库，而机器学习大多使用较小的数据集合，所以对于数据开采，效率问题至关重要。可以将数据开采看作是机器学习中某些技术在数据库系统上的应用，但与机器学习相比，数据开采需要注意以下两点：① 学习算法的效率要更高，因为真实数据库中的数据通常容量相当大而且存在噪音数据，而机器学习使用的是近乎理想的实验室数据。用于数据开采的数据库通常并不专为开采任务设计，那些能够简化学习过程的数据属性并不一定存在于数据库中，而且，真实数据库中的数据一定会存在不完整、不一致等错误，所以数据开采算法必须善于处理这些情况；② 更好的数据表达能力和知识表达能力。数据表达如关系数据库中的记录给出了一个问题域中的实例；知识表达如规则系统中的规则，可用来解决问题领域中的用户问题，以及关系组合中的语义信息。

1.2.2.2 统计学

统计学是一门成熟的学科，具有坚实的理论基础。大多数基于经典理论和分析方法的统计技术在数据量适度的数据集合上工作得相当好，但是随着计算机性能的提升以及成本的降低，再加上对分析大容量数据集合的需求迫使新的统计技术出现，数据开采因此成为传统统计方法的扩展。另外，由传统统计分析得出的结果难以理解，统计分析的过程也相当专业化，而数据开采能够将专家知识和分析技术很好地结合起来。

1.2.2.3 OLAP

数据开采和 OLAP 都属于分析型工具，但两者之间有着明显的区别。数据开采能自动地发现隐藏在数据中的模式，它与其他分析型工具的最大不同在于：它的分析过程是自动的。使用数据开采工具的用户不必提出明确的问题，只需让它去开采隐藏的模式并预测未来的趋势，这样更有利于发现未知的事实。OLAP 是一种自上而下、不断深入的分析工具：用户提出问题或假设，OLAP 负责从上至下深入地提取出关于该问题的详细信息，并以可视化的方式呈现给用户。与数据开采相比，OLAP 更多地依靠用户输入问题和假设，但用户先入为主的局限性可能会限制问题和假设的范围，从而影响最终的结论。因此，OLAP 更需要对用户需求有全面而深入的了解。一个典型的 OLAP 问题可能会是这样：去年哪个城市的用户购买了更多的汽车，是上海还是广州？而一个典型的数据开采问题可能会是这样：预测上海和广州的汽车销售情况。

数据开采可以发现 OLAP 所不能发现的更为复杂而细致的信息，但某些 OLAP 厂商声称他们的产品也具有数据开采的功能。实际上，如果减弱数据开采的定义，OLAP 也能做数据开采，但两者最关键的区别在于信息开采过程是否是自动的。

1.2.3 数据开采技术

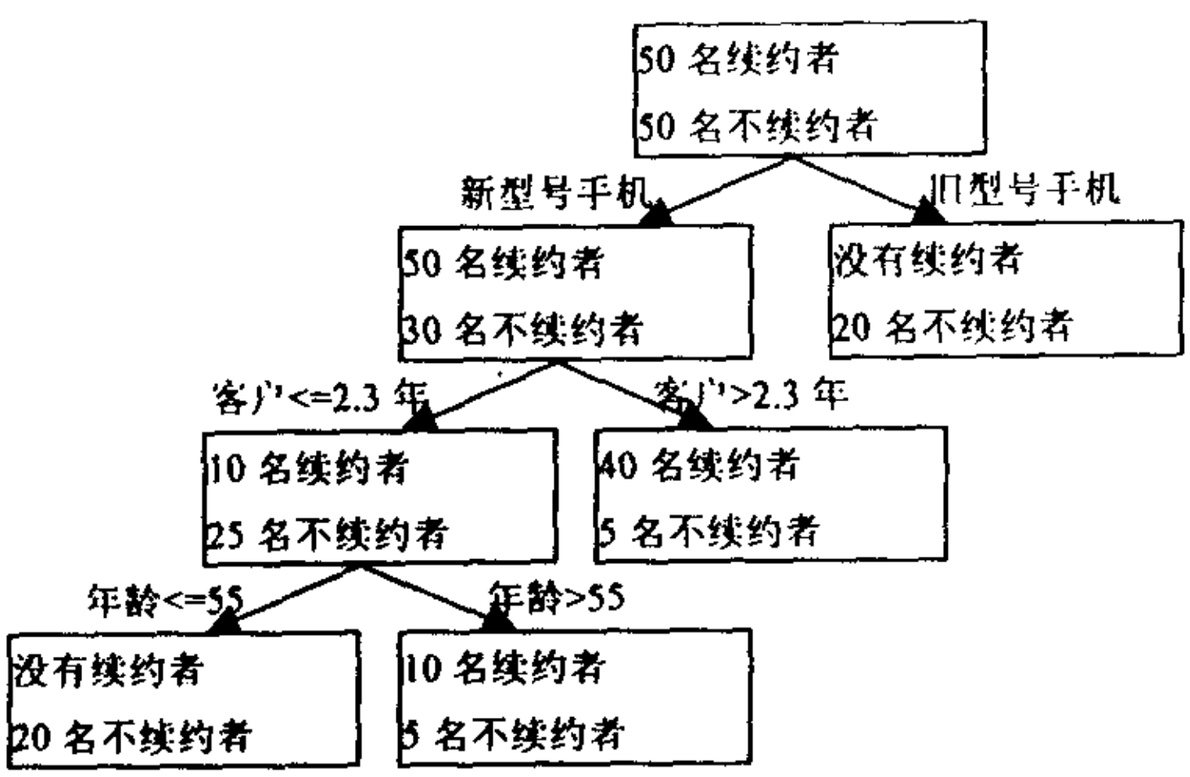

图 1.1 决策树用于客户分类

数据开采的技术基础是人工智能 AI（Artificial Intelligence），它采用了 AI 中部分已经成熟的技术，包括决策树、神经网络、遗传算法、规则归纳等。

1.2.3.1 决策树

决策树是一个预测模型，呈树形，决策树中的每一个分支都是一个分类问句，树中的叶子则是整个数据集合上的一个分类。举例：给某移动电话服务商的用户分类，查找哪些用户不易与当前服务商中断合约，图 1.1 即为解决该问题的决策树，可以得出结论“拥有新型手机、与该服务商签约时间大于 2.3 年的用户不易中断合约”。

决策树的构造过程大致如图 1.1 所示，从树的根节点开始提问，问题的选择很重要，判断问题是否合适的标准是要使得树中的叶子尽可能包含相似数据，这里相似指有关预测目标相似，如图 1.1 举的用户分类例子，决策树构造好之后，最好的情况是叶子节点包含的数据要么都是续约者要么都是不续约者。国际上最早的、也最有影响的决策树算法是 1970 年出 John R. Quinlan 提出的 ID3[9]-[10]，ID3 判断分支问题及分裂值的选取是否合适采用的标准如下：比较分裂前后系统对预测结果作出正确判断所需的信息量，如果该信息量在分裂后减少了，则认为问题及分裂值的选取合适。

1.2.3.2 神经网络

简单地说，神经网络研究人类大脑是如何组织以及如何学习的。组成神经网络的两个主要结构如下：

1. 节点（node）——对应人类大脑中的神经元。
2. 链接（link）——对应人类大脑中连接神经元的轴突、树突和神经键。

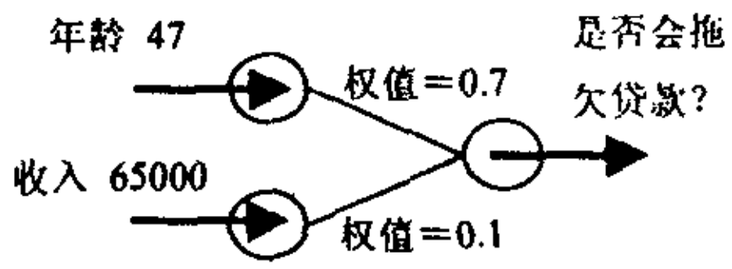

图 1.2 神经网络用于贷款拖欠预测

图 1.2 给出了一个简单的神经网络，圆圈代表节点，连接圆圈的直线代表链接，该神经网络根据年龄和收入预测顾客是否会拖欠银行贷款。神经网络算法首先获取两个输入节点的值：年龄=47、收入=65000，将它们规格化后分别等于 0.47 和 0.65；接着，根据模型计算输出节点的值：（年龄×年龄权值）+（收入×收入权值）=（0.47×0.7）+（0.65×0.1）=0.39。在本例中，输出值为 1 表示信用很差，输出值为 0 代表顾客有着很好的信用，绝对不会欠款，0.39 与 0 接近，

所以认为该顾客不会拖欠贷款。

神经网络模型的创建是一个监督学习过程，给定一个训练集，比较由神经网络算法得出的结果和训练集给定的标准结果，如果存在误差，则通过修改链接的权值来修改神经网络的行为，直到结果正确为止。

1.2.3.3 最近邻居

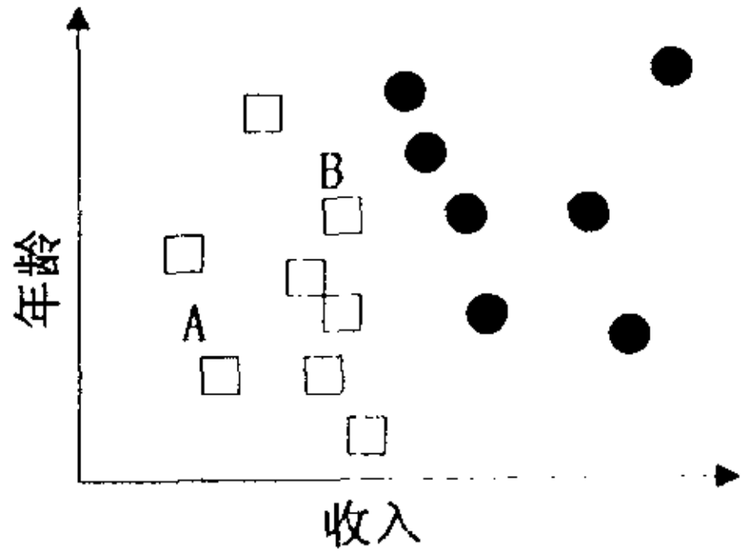

图 1.3 最近邻居用于风险预测

最近邻居方法是一个监督学习方法，它利用特征空间为未知类别的记录在样本数据集合中找出与之最相似的记录所属的类，并将这条新记录分配到该类中。数据点之间的距离大小被用来判断记录是否相似，最常用的距离公式是曼哈顿距离（Manhattan distance）和欧几里得距离（euclidean distance）。

图 1.3 给出了一个利用最近邻居方法预测某客户是否会拖欠贷款的例子，图中给出了一个基于年龄和收入的二维空间，样本数据库中的记录根据收入值和年龄值映射到这个二维空间中，形成数据点，其中实心圆代表不会拖欠贷款，空心矩形代表会拖欠贷款，A、B 是两个待分类的数据点，距离记录 A 最近的数据点是一个空心矩形，并且 A 周围的邻居都是空心矩形，那么自然 A 被分配到会拖欠贷款的类别中；记录 B 的情形没有 A 那么清晰，B 周围有空心矩形也存在实心圆，但是距离 B 最近的是空心矩形，所以，记录 B 也要分配到会拖欠贷款的类别中。

1.2.3.4 遗传算法

遗传算法指所有模拟生物进化过程的系统，更确切地说，遗传算法是规定并控制有机群体如何形成、如何评估、如何变更的算法。遗传算法的难点是如何将现实世界问题的解决方法转换成模拟遗传物质的计算机表达形式。一个最简单的例子能够说明遗传算法是如何工作的，将遗传算法用于解决如下问题：邮寄给用

户的产品宣传信件中放置多少优惠券能够产生最大效益？邮件数量与效益关系如下：

1. 优惠券放置过多会增加邮件重量，提高邮件成本；
2. 用户不会使用邮件中未提及的优惠券；
3. 优惠券种类过多增加用户使用难度，达不到预期效果。

该问题可以转换为一个简单的遗传算法，每一个模拟有机体包含一个基因，每个基因代表该有机体对于正确优惠券数量的猜测，遗传算法最初随机创建一个有机群体，然后通过模拟进化过程，在这个过程中修改基因、删除表现很差的有机体、复制表现优异的有机体（允许复制过程中对有机体的基因进行微小改动），直至产生最好结果。

以下描述了遗传算法从一代演变到下一代的过程：

1. 定义需要解决的问题并将其转换为遗传算法形式，如有机体、染色体和组成染色体的基因等。
2. 初始化有机群体，这个有机群体随机产生。
3. 使用预先定义的评估函数计算群体中单个有机体的适应值，评价单个有机体的适应性。
4. 复制那些具有最好适应值的有机体的遗传物质，同时删除适应值较小的有机体。
5. 让新形成的有机群体产生突变和有性繁殖。
6. 再次使用评估函数评价单个有机体的适应性。
7. 如果满足下列任何一个条件，则停止算法。
   (1) 得到能够满足需求的解决方案。
   (2) 系统演化已经到了第 n 代（n 是预先定义的阈值）。
   (3) 系统停滞不前，不再产生更好的有机体。
8. 重复步骤 4 至 7。

遗传算法在数据开采中的应用通常基于其它数据开采技术，如神经网络和最近邻居分类。遗传算法可以利用交叉验证作为评估函数找出数据开采中的最优预测模型。

1.2.3.5 规则归纳

规则用来表示数据库中项目之间的关系，它的一般形式为“if…then…”，例如“如果购买了纸盘子，则购买了塑料叉子”。规则可以很简单，前提只包含一个元素，也可以很复杂，前提包含由连接词连接的多个元素。规则的两个重要特性可以帮助用户评价一条规则是否有用：

1. 准确率——前提为真时，结论也为真的概率。如果一条规则的准确率很高，则认为该规则相当可靠。
2. 覆盖率——数据库中符合该规则的记录数。高覆盖率意味着这条规则经常被使用，错误的可能性很小，但它不能用来体现数据库特征。

规则归纳生成的规则与决策树生成的规则非常相似，但也存在一个主要的差异，决策树穷举了训练数据库中所包含的所有规则，并且这些规则具有互斥性，而规则归纳则不保证穷举所有规则，这些规则不具互斥性。这一差异是由产生规则的方式不同造成的，规则归纳自底向上搜索可能的用户感兴趣的模式，而决策树是自顶向下的穷尽搜索。

规则归纳算法的一般过程如下：

1. 数据预处理，主要是属性值的离散化，将连续值分成几个有限的区间值，如收入属性的值可离散化为以下几个区间(200, 800)、(800, 1500)、(1500, 3000]等。
2. 生成初始规则，初始规则的生成通常选取每条记录的一个属性值与其它属性值配对形成 if…then 规则，初始规则满足一个约束条件。
3. 新增一个约束条件，从初始规则中找出满足该约束条件的规则作为候选规则。
4. 重复步骤 3，直至满足中止条件。
5. 根据准确率、覆盖率等组织规则，并呈现给用户。

1.2.4 数据开采模型

数据开采通常分成两个大的类别：描述型（descriptive）的数据开采和预测型（predictive）的数据开采，前者给出对数据集合简洁的、总结性的描述，而后者则通过创建一个或多个模型，试图从当前数据集合推导出未知数据集合的行为。数据开采系统通常包含多个数据开采模型以完成多种数据开采任务，这些数据开

采模型中，分类[11]-[20]（classification）、回归[21]（regression）和时间序列（time series）模型主要用于预测，而聚类[22]-[28]（clustering）、关联分析[29]-[37]（association analysis）和序列发现[38]-[42]（sequence discovery）主要用于描述数据库行为。

1. 分类：分类模型识别并描述每个数据对象的特征，并根据这些特征分类未知数据对象。由分类模型得到的模式既可用于理解现存的数据又可预测未知实例的行为，如解决如下问题："谁最可能选择另一家长途电话服务供应商？"以及"谁是外科手术的最佳候选人？"。数据开采通过分析训练数据集（数据集合中的数据对象已经打上了分类标签）并根据类别特征构造预测模型。训练数据来自历史数据，如已经选择了另外的长途电话服务供应商的客户或做过该类外科手术的人员，要生成训练集首先从完整的数据库中挑选一部分数据作为样本，然后通过现实世界测试这些样本，最终根据测试结果为每个样本数据对象打上类别标签形成训练数据集，对训练数据集的分析结果是产生分类器，如决策树或一组分类规则。机器学习、统计学、神经网络和粗集等领域都有分类方法的研究，分类分析应用于客户分类、商业建模和信用分析。

2. 回归：回归分析利用现有的数据值预测将来的可能值，最简单的回归分析是线性回归，但在大多数实际问题中，数据值并不是简单的线性映射，如销售量、股票价格等。它们难以准确预测的原因是受太多因素的交互影响，因此需要更为复杂的分析方法。相同的数据开采技术既可用于回归又可用于分类，如决策树技术 CART（Classification And Regression Trees）和神经网络技术。

3. 时间序列：时间序列同回归相似，它利用现有的数据值序列预测将来的可能值，时间序列模型需要考虑数据的时间属性，尤其是时间周期的层次（包括时间周期的各种定义，如 5 天制或 7 天制工作周）、季节性以及其它一些特殊的因素如过去对将来的影响程度。时间序列分析可以帮助用户根据以往的股票价格预测其走势、公司的经济状况和竞争对手的业绩等。

4. 聚类：聚类模型的任务是将数据集分成不同的组，一个组就是一个聚类，一个聚类中的数据对象彼此相似而处于不同聚类中的数据对象相似度则很低。判断数据对象相似度的方法称作相似度判别标准，对于数值数据，通常采用距离函数作为相似度判别标准，目前也提出了针对非数值数据的相似度判别标准。好的聚类方法产生的高质量聚类能够确保类与类之间的相似度非常低而类中的相似度非常高。与分类不同，聚类是一个非监督学

习过程，输入集是一组没有分类标签的记录，也就是说事先得到的数据对象没有进行任何分类。数据开采领域有关聚类的研究主要集中于方法的可扩展性和正确性，以及算法的效率等方面。

5. 关联分析：关联分析的目的是为了发现隐藏在数据间的相互关系。关联分析就是给定一组项目（Item）和一个记录集合，通过分析记录集合，推导出项目之间的相关性，这种相关性通常用规则来表达，形式为：X→Y，表示数据库中包含项目 X 的记录同时也包含项目 Y。进行关联分析时，需要用户给出两个参数，最小置信度（Minimum Confidence）和最小支持度（Minimum Support），最小置信度用来滤掉可能性过小的规则，最小支持度表示规则发生的概率，即可信度。关联分析广泛应用于零售交易数据分析，用来获取哪些商品较可能同时被用户购买，关联分析还可应用于医疗保险业等。

6. 序列发现：序列发现和关联分析相似，其目的也是为了发现数据之间的联系，但序列发现的侧重点在于分析数据间的前后关系，在进行序列发现时，同样也需要由用户指定最小置信度和最小支持度。运用序列发现分析零售交易数据可以发现客户潜在的购物模式，如客户在购买微波炉前最常购买何种商品；序列发现可以根据需要考虑前后时间间隔，如客户购买了锤子，在其后三周又购买钉子的概率为 18%，序列发现还可应用于股市分析、医疗保险业、Web 网站的访问模式发现等。

其它重要的数据模型还有数据汇总和偏差分析等，数据汇总对数据进行浓缩，给出紧凑描述，偏差分析用来发现实际数据与预期数据之间的不一致。不同的数据开采模型有着不同的适用范围，一个真正的数据开采系统通常是综合利用、协同使用这些模型的。

1.2.5 研究与应用

1.2.5.1 学术研究

KDD 一词是在 1989 年 8 月于美国底特律市召开的第一届 KDD 国际学术会议上正式形成的。国际 KDD 学术会议起初每两年召开一次，1993 年后每年召开一次。在几次国际 KDD 学术会议上讨论的问题有：① 定性知识和定量知识的发现；② 数据汇总；③ 知识发现方法；④ 数据依赖关系的发现和分析；⑤ 发现过程中知识的应用；⑥ 交互式的知识发现系统；⑦ 知识发现的应用。

1995 年，在加拿大召开了第一届知识发现和数据开采国际学术会议。由于数据库中的数据被形象地喻为矿床，因此数据开采一词很快流传开来。1995 年以来，国外在数据开采知识发现方面的论文非常多，已形成了热门研究方向。1997 年，亚太地区举行了第一届知识发现和数据开采国际会议（PAKDD：Pacific-Asia Conference on Knowledge Discovery and Data Mining），在欧洲也举办有类似的国际会议 PKDD（European Conference on Principles of Data Mining and Knowledge Discovery）。1998 年，ACM（Association for Computing Machinery）正式成立了有关 KDD 的特别兴趣小组 SIGKDD（Special Interest Group on Knowledge Discovery in Data and Data Mining）。此外，数据库、人工智能、情报检索等领域的国际学术团体也将 KDD 和数据开采作为研究讨论的热点问题。

目前，有关数据开采讨论的问题主要集中在因特网应用[43]-[58]和科学研究应用方面。在因特网应用方面包括数据开采在电子商务中的应用、Web 开采、Web 语义学、XML 与数据开采；科学研究主要集中于生物信息学中的数据开采应用。除了传统的研究方向——如数据和知识表达[59]-[67]、元数据、数据缩减和维数缩减、预处理[68]-[71]和后处理技术[72]-[75]、开采语言[76]-[78]——大家最新关注的问题还包括已发现知识的管理和精炼、文本开采[79]-[86]用于知识管理、数据开采中的安全和隐私保护等。

随着国外知识发现的兴起，我国也很快跟上了国际步伐。《计算机世界》报技术专题版于 1995 年 3 月发表了由中国电子设备系统工程公司研究所李德毅教授组织的 KDD 专题；于 1995 年 4 月发表了由中国科学院史忠植研究员组织的“机器学习、神经网络”专题；于 1995 年 12 月发表了由国防科技大学陈文伟教授组织的“机器发现和机器学习”专题，于 1999 年在北京召开的第三届 PAKDD 国际学术会议，都对我国开展知识发现的研究起到了一定的推动作用。近几年，国内各计算机学术刊物也纷纷刊登有关知识发现和数据开采的论文，所涉及的研究领域集中于学习算法的研究和有关数据开采的理论研究。

1.2.5.2 应用领域

数据开采在各行各业都具有广阔的应用前景，如下：

1. 零售/营销：分析顾客的购买行为和习惯；顾客关系管理 CRM（Customer Relationship Management）研究消费者之间的特征关系；分析顾客对促销手段的回应；商品价格优化以及商品销售预测等。
2. 银行/电信：欺诈行为侦测：对于银行业是信用卡欺诈，对于电信则是恶

性欠费；发现在何种情况哪些顾客会更换服务提供商。另外，银行业还需要对借贷企业实施破产预测等。

3. 股票期货：从股票或期货交易的历史数据中得到交易的规则或规律，用于分析股票价格的未来走势、为某笔交易提供实时建议、进行投资分析、建立期货交易模型等。

4. 保险/医疗保健：需求分析——为顾客提供最适合的医疗保健程序；预测什么样的顾客将会购买新险种；识别高风险顾客的行为模式；欺诈甄别。

5. 生物/制药：DNA 蛋白序列分析；分析疾病的治疗方法，识别疗效好的药品治疗案例等。

6. 行政机构的知识管理：集成机构中存在的多种异构数据源，包括电子邮件、文字文档、图片、幻灯片、电子表格等，并提供易于使用的可视化界面供查找、开采，如人力资源部可以通过分析应聘者的申请表格和简历挑选最合适的职位人选。

7. Web 分析：发现存在于 Web 日志数据中的模式，用于分析网站浏览者的行为，可以为用户提供一对一的个性化服务，如提供个性化主页面、对用户的某个点击动作产生个性化回应等；对于电子商务网站还可以分析购买者行为，为不同的消费群体提供个性化产品信息；Web 文本分析，包括文本总结、文本分类及解释、搜索有用的经过提取的信息；Web 广告效益分析；Web 站点信息流量的实时监控等。

就我国现状来说，保险业、金融业和信息部门有能力有条件也有必要引入数据开采工具，一些大的零售商店，可以考虑将销售细节数据保存下来用以分析，从而向顾客提供更好的服务，而对于信息技术尚未被广泛采用的企业单位应先做好数据库的建设和应用工作，只有在低层次的应用需求得到满足时，才能真正满足较高层次的应用需求。

1.2.5.3 商业产品

目前，国内计算机应用水平比较低，不少单位数据库的规模还比较小，更多的单位还没有建立完善的数据库系统，数据库及数据仓库是信息系统及决策支持系统的基础设施，没有完善的数据库或数据仓库，再好的开采工具也发挥不了作用，所以，目前国内还没有出现完整的数据开采产品。

国外的数据开采产品多种多样，有包含多种数据开采模型的成套产品，据 IDC

的统计，IBM 公司的 IntelligentMiner 系列（http://www.ibm.com/software/data/iminer）目前是数据发掘领域最先进的产品。IntelligentMiner 通过典型数据集自动生成、关联发现、序列规律发现、概念性分类和可视化呈现等技术，自动实现数据选择、数据转换、数据开采和结果呈现这一整套数据开采操作，IntelligentMiner 由数据开采（for data）和文本开采（for text）这两部分组成；统计分析软件 SAS（http://www.sas.com）为前端用户提供了友好的图形用户界面展示整个分析过程，包括取样、探测、修改、建模和评估；其他数据开采系列产品还包括 SPSS（http://www.spss.com/clementine）、支持并行服务器的高性能数据开采工具——ORACLE 的 Darwin（http://www.oracle.com/datawarehouse/products/datamining）以及 Microsoft SQL Server 2000（http://www.microsoft.com/SQL/evaluation/features/ datamine.asp），它支持决策树、聚类和第三方算法；专门用于关联发现的 Magnum Opus（http://www.rulequest.com/MagnumOpus-info.html）是快速发现关联规则的工具，有 Windows、Linux 和 Solaris 版本；专门用来聚类的产品有采用可视化系统聚类分析的 ClustanGraphics3（http://www.clustan.com）和用于高维数据集合的 CViz Cluster Visualization（http://www.alphaworks.ibm.com/tech/cviz）；序列发现产品有 MineIT EasyMiner（http://www.mineit.com），它支持多种序列模式的发现，还有属于 SPSS Clementine 的 CaprI；专门用于 Web 搜索和内容挖掘的数据开采工具有 Miner3D for web（http://miner3d.com），它提供了 Web 搜索和信息发现的可视化；最后还有专门针对文本数据的文本分析产品 Text Analyst（http://www.megaputer.com）、TextQuest（http://www.textquest.de）等。

1.3 聚类分析

聚类分析是数理统计中研究“物以类聚”的一种方法。分类学是人类认识世界的基础科学，在古老的分类学中，人们主要靠经验和专业知识，很少利用数学，随着生产技术和科学的发展，分类越来越细，以致有时光凭经验和专业知识还不能进行确切分类，于是数学这个有用的工具逐渐被引进到分类学中，形成了数值分类学。近几十年来，数理统计的多元分析方法有了迅速的发展，多元分析的技术自然被引进到分类学中，于是从数值分类学中逐渐分离出聚类分析这个新的分支。[87]

作为一项数据开采任务，聚类分析的目的是将大型、多维数据集合中相似的数据对象分配到同一个类中。统计学中的聚类方法将研究集中于数值数据，而且不考虑所处理的数据集合规模大小，因此，数据开采中的聚类分析是统计学中传统聚类方法针对现实问题的应用扩展。

的统计，IBM 公司的 IntelligentMiner 系列（http://www.ibm.com/software/data/iminer）目前是数据发掘领域最先进的产品。IntelligentMiner 通过典型数据集自动生成、关联发现、序列规律发现、概念性分类和可视化呈现等技术，自动实现数据选择、数据转换、数据开采和结果呈现这一整套数据开采操作，IntelligentMiner 由数据开采（for data）和文本开采（for text）这两部分组成；统计分析软件 SAS（http://www.sas.com）为前端用户提供了友好的图形用户界面展示整个分析过程，包括取样、探测、修改、建模和评估；其他数据开采系列产品还包括 SPSS（http://www.spss.com/clementine）、支持并行服务器的高性能数据开采工具——ORACLE 的 Darwin（http://www.oracle.com/datawarehouse/products/datamining）以及 Microsoft SQL Server 2000（http://www.microsoft.com/SQL/evaluation/features/ datamine.asp），它支持决策树、聚类和第三方算法；专门用于关联发现的 Magnum Opus（http://www.rulequest.com/MagnumOpus-info.html）是快速发现关联规则的工具，有 Windows、Linux 和 Solaris 版本；专门用来聚类的产品有采用可视化系统聚类分析的 ClustanGraphics3（http://www.clustan.com）和用于高维数据集合的 CViz Cluster Visualization（http://www.alphaworks.ibm.com/tech/cviz）；序列发现产品有 MineIT EasyMiner（http://www.mineit.com），它支持多种序列模式的发现，还有属于 SPSS Clementine 的 CaprI；专门用于 Web 搜索和内容挖掘的数据开采工具有 Miner3D for web（http://miner3d.com），它提供了 Web 搜索和信息发现的可视化；最后还有专门针对文本数据的文本分析产品 Text Analyst（http://www.megaputer.com）、TextQuest（http://www.textquest.de）等。

1.3 聚类分析

聚类分析是数理统计中研究“物以类聚”的一种方法。分类学是人类认识世界的基础科学，在古老的分类学中，人们主要靠经验和专业知识，很少利用数学，随着生产技术和科学的发展，分类越来越细，以致有时光凭经验和专业知识还不能进行确切分类，于是数学这个有用的工具逐渐被引进到分类学中，形成了数值分类学。近几十年来，数理统计的多元分析方法有了迅速的发展，多元分析的技术自然被引进到分类学中，于是从数值分类学中逐渐分离出聚类分析这个新的分支。[87]

作为一项数据开采任务，聚类分析的目的是将大型、多维数据集合中相似的数据对象分配到同一个类中。统计学中的聚类方法将研究集中于数值数据，而且不考虑所处理的数据集合规模大小，因此，数据开采中的聚类分析是统计学中传统聚类方法针对现实问题的应用扩展。

1.3.1 聚类分析法

为了将一组对象进行分类，就需要研究对象之间的关系，一种方法是用相似系数，性质越接近的对象，它们的相似系数越接近于 1（或-1），而彼此无关的对象之间的相似系数则越接近于 0，比较相似的对象归为一类，不相似的对象属于不同的类。另一种方法是将每一个对象看作 n 维空间中的一个点，并在空间定义距离，距离较近的点归为一类，距离较远的点应属于不同的类。对象之间的相似系数和距离有各式各样的定义，而这些定义与对象属性的类型关系紧密，对象属性按照其度量尺度分类：

1. 数字属性：用连续的量来表示，如长度、重量、压力等。
2. 分类属性：用离散的量来表示，分类属性可以是无序的，称作名义（nominal）属性，如产品名称或城市，也可以是有序的，如评价酒分成好、中、次三等。

不同类型的属性在定义距离和相似系数时有很大的差异，研究得较多的是数字属性的数据。用 d_{ij} 表示对象 i 和对象 j 的距离，A_{ik} 代表对象 i 的第 k 个属性的值，$k = 1, 2, \ldots, n$，最常见最直观的距离公式是

$$d_{ij}(1) = \sum_{k=1}^{n} \left| A_{ik} - A_{jk} \right| \tag{1.1}$$

$$d_{ij}(2) = \sqrt{\sum_{k=1}^{n} \left(A_{ik} - A_{jk} \right)^2} \tag{1.2}$$

前者叫做绝对值距离，后者叫做欧氏距离，这两个距离可以统一成

$$d_{ij}(q) = \sqrt[q]{\sum_{k=1}^{n} \left| A_{ik} - A_{jk} \right|^q} \tag{1.3}$$

公式(1.3)叫做明考斯基（Minkowski）距离，当 $q = 1$ 和 2 时就是距离(1.1)和(1.2)。

以上几种距离均适合于数字属性的数据，如果是分类属性的数据也有一些定义距离的方法，对于对象 i 和对象 j，如果 A_{ik} 等于 A_{jk}，则称作属性匹配，否则称作属性不匹配，m_1 代表对象 i 和对象 j 之间的属性匹配数，m_2 代表不匹配数，对象 i 和 j 的距离如下：

$$d_{ij} = \frac{m_2}{m_1 + m_2} \tag{1.4}$$

表 1.1 列联表

x_i \ x_j	r_1	r_2	…	r_q	$\sum$
t_1	n_{11}	n_{12}	…	n_{1q}	$n_{1\cdot}$
t_2	n_{21}	n_{22}	…	n_{2q}	$n_{2\cdot}$
…	…	…	…	…	…
t_p	n_{p1}	n_{p2}	…	n_{pq}	$n_{p\cdot}$
$\sum$	$n_{\cdot 1}$	$n_{\cdot 2}$	…	$n_{\cdot q}$	$n_{\cdot\cdot}$

在聚类分析中不仅需要将对象分类，也需要将属性分类，在属性之间也可以定义距离，更常用的是相似系数，用 c_{ij} 表示属性 x_i 和 x_j 的相似系数，$|c_{ij}|$越接近于 1，x_i 和 x_j 的关系越密切，对于数字属性，常用的相似系数有① 夹角余弦：受相似形的启发得来，形状相似而长度不是主要矛盾时，需要定义一种相似系数表达两个图形的紧密联系，夹角余弦恰好能够满足这一要求。夹角余弦在图像识别中很有用。② 相关系数：这是将数据标准化（使均值为 0）后的夹角余弦。除了以上两个外，还有指数相似系数和非参数方法等。考虑分类属性的相似系数，设属性 x_i 和属性 x_j 的取值分别是 $t_1, \ldots, t_p$ 和 $r_1, \ldots, r_q$，用 n_{kl} 表示 x_i 取 t_k，x_j 取 r_l 的对象数，形成列联表，形式如表 1.1，用 e_{ij} 表示当对象相互独立时的期望观察数，易见它的估计值是 $e_{ij}^{k} = \dfrac{n_{i\cdot} n_{\cdot j}}{n_{\cdot\cdot}}$。反映 x_i 和 x_j 的关系常用到

$$x^2 = \sum_{i=1}^{p}\sum_{j=1}^{q}(n_{ij} - e_{ij})^2 / e_{ij} \tag{1.5}$$

建立在 x^2 基础上的相似系数有列联系数和连关系数，公式请参阅文献[87]。如果 x_i 和 x_j 的取值只有两个，不失一般性可设它们的取值是 0 和 1，列联表简化成表 1.2，当 x_i 取 0，x_j 也取 0 时或两者都取 1 时称为匹配的，否则称为不匹配的，有关这种情况的相似系数很多，主要几种如下：

表 1.2 布尔属性列联表

x_i \ x_j	0	1	$\sum$
0	a	b	a+b
1	c	d	c+d
$\sum$	a+c	b+d	a+b+c+d

点相关系数：

$$c_{ij}=\frac{ad-bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \tag{1.6}$$

四分相关系数：

$$c_{ij}=\sin\left[\frac{(a+d)-(b+c)}{a+b+c+d}90°\right] \tag{1.7}$$

夹角余弦，利用数字属性的夹角余弦得到

$$c_{ij}=\left[\frac{a}{a+b}\frac{a}{a+c}\right]^{1/2} \tag{1.8}$$

1.3.1.1 系统聚类法

系统聚类法（Hierarchical Clustering Method）是目前国内外使用最多的一种方法，有关它的研究极为丰富。这种方法的基本思想是：先将 n 个对象各自看成一类，然后规定对象之间的距离和类与类之间的距离。开始，因每个对象自成一类，类与类之间的距离和对象之间的距离是相等的，选择距离最小的一对合并成一个新类，计算新类与其它类的距离，再将距离最近的两类合并，这样每次减少一类，直至所有对象成为一类为止。

用 d_{ij} 表示对象 i 和对象 j 的距离，$C_1, C_2, \ldots, C_n$ 表示类，用 D_{pq} 表示类 C_p 与类 C_q 之间的距离，系统聚类法的步骤如下：

1. 规定对象之间的距离，计算对象两两距离，开始每个对象自成一类，这时 $D_{pq}=d_{pq}$，将两两距离构成对称阵，记作 $D_{(0)}$，形式如表 1.3。
2. 选择 $D_{(0)}$的最小元素，设为 D_{pq}，将 C_p 和 C_q 合并成一个新类，记为 C_r，$C_r=\{C_p, C_q\}$。
3. 计算新类 C_r 与其他类的距离，更新 $D_{(0)}$得到矩阵 $D_{(1)}$，矩阵的更新利用递推公式完成。
4. 对 $D_{(1)}$重复步骤 2，然后得到 $D_{(2)}$，如此下去直到所有元素成为一类为止。

如果某一步 $D_{(k)}$中最小元素不止一个，则对应这些最小元素的类可以同时合并。

表 1.3 对称阵 $D_{(0)}$

	C_1	C_2	$\cdots C_{i-1} \cdots$	C_{n-1}
C_2	D_{12}			
C_3	D_{13}	D_{23}		
⋮	⋮	⋮	⋮	
C_i	D_{1i}	D_{2i}	$\cdots D_{(i-1)i} \cdots$	
⋮	⋮	⋮	⋮	
C_n	D_{1n}	D_{2n}	$\cdots D_{(i-1)n} \cdots$	$D_{(n-1)n}$

类与类之间的距离有许多定义的方法，例如定义类与类之间的距离为两类中距离最近的对象之间的距离，或者定义类与类之间的距离为两类中心之间的距离等等，不同的定义就产生了不同的系统聚类方法。以下公式中，D_{pq} 为类间距离，D_{rk} 为计算新类和其它类的距离使用的递推公式。

1. 最短距离法：类与类之间的距离定义为两类中距离最近对象之间的距离。

$$D_{pq} = \min_{i \in C_p, j \in C_q} d_{ij} \tag{1.9}$$

$$D_{rk} = \min_{i \in C_r, j \in C_k} d_{ij} = \min\left\{ \min_{i \in C_p, j \in C_k} d_{ij}, \min_{i \in C_q, j \in C_k} d_{ij} \right\} = \min\{D_{pk}, D_{qk}\} \tag{1.10}$$

2. 最长距离法和中间距离法：

(1) 最长距离法定义类与类之间的距离为两类之间最远对象的距离。

$$D_{pq} = \max_{i \in C_p, j \in C_q} d_{ij} \tag{1.11}$$

$$D_{rk} = \max\{D_{pk}, D_{qk}\} \tag{1.12}$$

(2) 如果类与类之间的距离既不采用两类之间最近的距离，也不采用两类之间最远的距离，而是采用介于两者之间的距离，这时称为中间距离法，（1.13）给出了它的递推公式

$$D_{kr}^2 = \frac{1}{2} D_{kp}^2 + \frac{1}{2} D_{kq}^2 - \frac{1}{4} D_{pq}^2 \tag{1.13}$$

3. 重心法：从物理的观点来看，一个类用它的重心（该类对象的均值）做代表比较合理，类与类之间的距离就用重心之间的距离来代表，设 C_p 和 C_q

的重心分别是 X_p 和 X_q，则 C_p 和 C_q 之间的距离和递推公式：

$$D_{pq} = d_{\bar{x}_p \bar{x}_q} \tag{1.14}$$

$$D_{kr}^2 = \frac{n_p}{n_r} D_{kp}^2 + \frac{n_q}{n_r} D_{kq}^2 - \frac{n_p}{n_r} \frac{n_q}{n_r} D_{pq}^2 \tag{1.15}$$

以上这几种主要的系统聚类法，合并类的原则和步骤是完全一样的，所不同的是类与类之间的距离有不同的定义，从而得到不同的递推公式，在 1969 年，维希特（Wishart）发现它们的递推公式可以统一起来，统一的形式是

$$D_{kr}^2 = \alpha_p D_{kp}^2 + \alpha_q D_{kq}^2 + \beta D_{pq}^2 + \gamma \left| D_{kp}^2 - D_{kq}^2 \right| \tag{1.16}$$

其中系数 $\alpha_p, \alpha_q, \beta, \gamma$ 对于不同的方法有不同的取值。递推公式的统一使系统聚类的主要几种的共性完全统一起来了，对于编制计算机程序提供了极大的方便。

1.3.1.2 动态聚类法

用系统聚类法聚类，对象一旦划分到某个类以后就不变了，这要求分类的方法比较准确。此外系统聚类法要存入距离阵 $D_{(0)}$ 或者 $D^2_{(0)}$，当对象较多即 n 值较大时占用内存太多，计算方法中的迭代思想给我们以启发，能否先给一个粗糙的初始分类，然后按照某种标准进行修改，直至分类比较合理为止，采用这种思想产生的聚类法叫做动态聚类法。为了得到初始分类，有时设法选择一些凝聚点，让对象按照某种原则向凝聚点凝聚，动态聚类法大体可用图 1.4 表示：

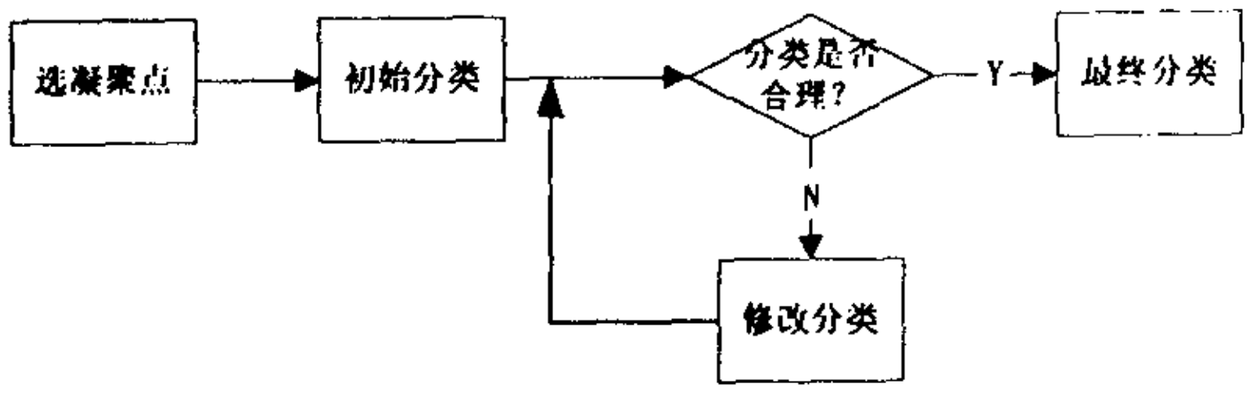

图 1.4 动态聚类过程

图 1.4 的每一部分均有很多种方法，这些方法按框图进行组合就会得到各种动态聚类法。

1. 选择凝聚点：所谓凝聚点就是一批有代表的点，是欲形成类的中心，选择凝聚点有如下方法：

   (1) 凭经验。如果对问题已有相当了解，对象集合如何分类，分成几类是心中有数的，这时将为每一类选择一个有代表性的对象作为凝聚点。

(2) 随机选取 k 个对象分别作为 k 个类的凝聚点。

(3) 将数据人为地分成 k 类，计算每一类的均值，将这些均值做为凝聚点。

(4) 密度法。事先定义两个正数 d_1 和 d_2，其中 $d_2 > d_1$，以每个对象映射到 n 维空间的数据点为球心，以 d_1 为半径，落在这个球内的对象数（不包括球心的对象）就叫做这个点的密度，因此密度是 d_1 和对象数据点的函数。首先选择具有最大密度的对象点作为第一凝聚点，然后选择次大密度的数据点，它和第一凝聚点的距离如果小于 d_2，该点取消；如大于 d_2，则该点作为第二凝聚点。用这个方法按照对象的密度由大到小选下去，第一次和已选的任一凝聚点的距离不小于 d_2 的对象作为新的凝聚点。常取 $d_2 = d_1$。

2. 初始分类：有多种方法可以根据凝聚点进行初始分类，另外初始分类不一定非用凝聚点不可。

(1) 定义对象间的距离，每个对象按最近凝聚点归类。

(2) 选择了一批凝聚点，每个凝聚点自成一类，对象逐个读入，每读入一个对象将它们分配到最近凝聚点的一类，并计算该类的重心，以这个重心代替原来的凝聚点，再读入下一个对象，按同样规则处理。

(3) 给定一个正数 d，类 C_1 包含一个对象 x_1，即 $C_1 = \{x_1\}$，如对象 x_2 与 x_1 的距离 $d_{21} < d$，则将 x_2 归入 C_1，否则 $C_2 = \{x_2\}$。当某一步轮到 x_l 输入时，如已形成了 k 个类：$C_1, C_2, \ldots, C_k$，每个类第一次进去的对象记作 $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$（显然 $i_1 = 1$）。如果 $\min\limits_{1 \le j \le k} d_{li_j} \le d$，则将 x_l 归入达到极小值的那一类，如果 $\min\limits_{1 \le j \le k} d_{li_j} > d$，则令 $C_{k+1} = \{x_l\}$。

3. 修改分类：有了初始分类，需要制定原则判断这个分类是否合理，根据不同的原则有不同的分类修改法。

(1) 按批修改法，聚类步骤如下：

① 选择一批凝聚点，并定义对象之间的距离。

② 将所有对象按最近凝聚点归类。

③ 计算每一类的重心，将重心作为新的凝聚点，如果所有的新凝聚点与上一次的老凝聚点重合，过程终止，否则执行步骤②。

(2) 逐个修改法，按批修改法是等对象全部归类后才改变凝聚点，另一种自然的想法是每读进一个对象就将它分类，同时改变凝聚点，这就产

生了逐个修改法。逐个修改法最早由 MacQueen 在 1967 年提出[88]，最简单的逐个修改法步骤如下：

① 人为地定出分类数目 k，取前 k 个对象作为凝聚点。

② 逐个读入剩余的 n-k 个对象，每读入一个将它归入最近的凝聚点的那一类，随即计算该类重心，将重心代替原凝聚点。

③ 将这 n 个对象再从头至尾读入一遍，每读入一个对象，将它归入最近的凝聚点那一类，重新计算该类重心，以重心代替原凝聚点，如果 n 个对象读入后所得到的分类和原来一样，则停止分类，否则重复步骤③。

这个算法计算简单，分类迅速，占用计算机内存小，但由于人为定义了 k，有时定得不合适会影响分类效果，改进的办法是在修改分类的过程中类的数目可以根据情况有所变化，太近的类可以合并，太远的对象可以分离出来产生新类，改进后步骤如下：

1. 指定三个参数 k, c 和 R。
2. 取前 k 个样品作为凝聚点，计算这 k 个凝聚点的两两距离，如最小的距离小于 c，则将相应的两个凝聚点合并，用这两点的重心作为新凝聚点。重复这一步骤，直至所有凝聚点之间的距离均大于等于 c 为止。
3. 逐个读入余下的 $n-k$ 个对象，每读入一个对象，计算该对象与所有凝聚点的距离，如最小的距离大于 R，则该对象作为新的凝聚点；如最小的距离小于等于 R，则该对象归入最靠近它的凝聚点的那一类，随即重新计算这一类的重心，以重心作为新的凝聚点。然后重新验证凝聚点之间的距离，如有小于 c 的用上一步的办法将相应类合并，直至所有的凝聚点之间的距离均大于等于 c。
4. 将 n 个对象从头至尾逐个读入，用步骤 3）的办法归类，但当读入一个对象后，如分类和原来一样，则重心不必计算；如分类与原来不同，所涉及到的两类重心都要重新计算。如果得到的新分类和上次相同，则聚类结束，否则重复步骤 4）。

1.3.1.3 分解法

系统聚类是将类由多到少合并，另一种聚类的思想是类由少到多分裂，开始将全体对象看作一类，然后分成两类、三类、……直至所有的对象各自成为一类。

一分为二：这是将某一类分解成两个子类，然后对其子类又一分为二的方法。假设类 C 中有 n 个对象，它的两个子类 C_1 和 C_2 各有 n_1 和 n_2 个对象，两类重心为 $\bar{x}_1$ 和 $\bar{x}_2$，C 的重心为 $\bar{x}$。类 C、C_1 和 C_2 的离差平方和为

$$S = \sum_{x_i \in C} (x_i - \bar{x})'(x_i - \bar{x}) \tag{1.17}$$

$$S_j = \sum_{x_i \in C_j} (x_i - \bar{x}_j)'(x_i - \bar{x}_j) \qquad j = 1、2 \tag{1.18}$$

如果类分解得合理，应使 $S_1 + S_2$ 尽可能的小，或使 $S - S_1 - S_2$ 尽可能的大。

$$S - S_1 - S_2 = \frac{n_1 n_2}{n} (\bar{x}_1 - \bar{x}_2)'(\bar{x}_1 - \bar{x}_2) \tag{1.19}$$

或者

$$S - S_1 - S_2 = \frac{n n_1}{n_2} (\bar{x}_1 - \bar{x})'(\bar{x}_1 - \bar{x}) \tag{1.20}$$

用 E 表示 $S - S_1 - S_2$ 并作为目标函数，选择某种分法使 E 达到极大。n 个对象分成两类，一切可能的分法有 $2^n - 1$ 种，当 n 较大时要比较这么多分类是不可行的，于是只好放弃求精确最优解，代之以局部最优解：一开始所有对象均在 C_1 中，首先找到一个对象，将它分配到 C_2 中使 E 达到极大，然后再找第二个对象，将它分配到 C_2 中使 E 达到极大，如此分下去，某个对象一旦分配到 C_2 就不变了。令 $E(1), E(2), \ldots, E(k)$ 为 C_2 中包含有一个对象、两个对象、……、k 个对象的 E 值，那么一定存在 k^* 使得 $E(k^*) = \max_{1 \le k \le n} E(k)$，于是将前 k^* 次进入 C_2 的对象分为一类，其余 $n - k^*$ 个对象为另一类。

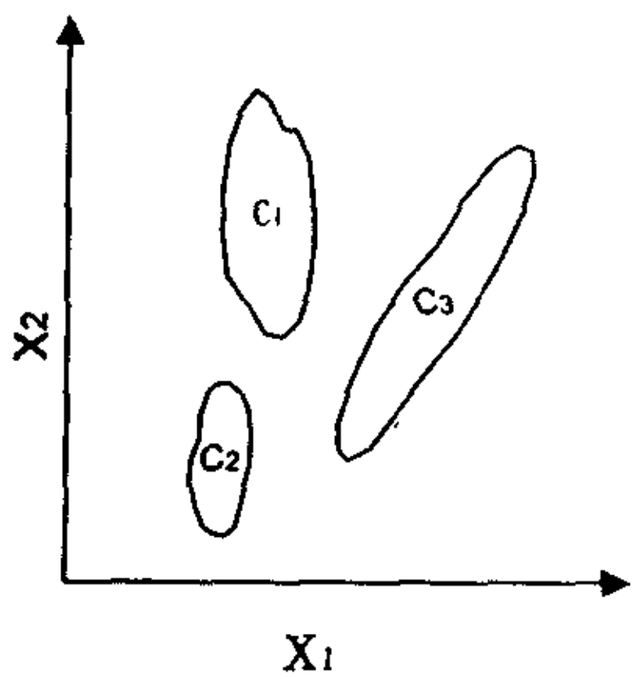

图 1.5 可选择属性的分解法

可选择对象属性的分解法：假设对象有 m 个属性，许多聚类方法的整个分类

过程中 m 个属性都在起作用，但有时这么做不一定合适，当类不太大时，起主要作用的属性一般不会太多，而使过多无用的属性参与分类只会起干扰作用，并无其他好处。如图 1.5 中有三个类，如果第一次分解为两类$\{C_1, C_2\}$和 C_3，这时如将$\{C_1, C_2\}$再分解，属性 x_1 已不起作用，可以将其删除，基于这个思想产生了可选择对象属性的分解法。

用 x_{ij} 表示第 i 个对象的第 j 个属性，每个属性的标准差记为 $S_1, S_2, \ldots, S_m$，将 $T_1, T_2, \ldots, T_m$ 作为事先规定的阈值，如果某一类对象组成的一类 C，该类对象的每个属性的标准差记为 $S_1^G, S_2^G, \ldots, S_m^G$，如对某个 i，S_i^G 小于等于 T_i，表明属性 x_i 在 C 中变化很小，因而可以去掉。由于采用的是分解法，类越分越小，当某个属性 x_i 在某一次分类中去掉后，在其所属的所有子类中均要去掉，所以属性只会越来越少，属性筛选后，每一次分类都用一分为二法。

1.3.2 聚类算法

目前，聚类算法主要分为两种：分区和层次。分区法就是将数据集合分成多个区，每个区就是一个类，分区算法大都采用统计学中的动态聚类方法，是 K-means 或 K-medoid 方法[89]的扩展，这一类的算法有 CLARANS[27]、Focused CLARANS[23]、BIRCH[28][90]等，另外分区法还有一种根据数据空间中数据点分布的密度划分类的方法，即在数据空间中存在数据点密集的区域和数据点稀疏的区域，将那些被稀疏区域隔断的密集区域作为类输出，这类算法有 OPTICS[91]、DBSCAN[92]。层次法分两大部分[26][93]，聚合层次法和分解层次法，分别对应统计学中的系统聚类法和分解法，聚合层次法就是先将每个数据对象看作一个类，然后根据最近距离不断合并直到得到一个类为止；分解层次法的过程正好与此相反。

除了数值数据外，现实世界中还存在许多其它类型的数据，用距离方法聚类这些数据得到的结果会不准确，因此，聚类算法的研究人员根据具体应用提出了不同算法，如用于处理分类数据的 ROCK[94]、K-modes[95]、CACTUS[96]和基于动态系统的算法[97]。处理数值分类混合数据集合的 k-prototypes[98]，还有利用大集处理交易数据的算法[99]，分析基因数据的聚类算法[100]，处理高维数据的 CLIQUE[101]、BUBBLE、BUBBLE-FM[102]，处理文档集合的聚类算法 Word-IC、Phrase-IC[103]、STC[104]、ARHP 和 PCA[105]、WAKNN [106]等。

此外，还有基于超图的聚类算法[107]、模糊聚类算法[108]-[112]；将聚类技术应用到其它数据开采领域的算法，如 ARCS 算法[113]用来聚类关联规则，关联规则超图上的聚类算法[114]以及用于数据分段的算法[115]等。

1.3.3 面临的挑战

从数据开采应用的角度，对聚类算法要求如下：

1. 处理不同类型的数据：数据开采应用于已有的大型数据库或者数据仓库，而这些数据集合又是为不同应用而设计的，数据类型多种多样。聚类算法不仅要考虑有效处理关系型数据（主要包括数字属性、分类属性以及混合属性的数据），还要考虑其他复杂型数据（如图形、图像、文本、声音）。
2. 效率高、具可扩展性：聚类算法的执行时间必须是可预测的，存在组合爆炸问题或者具有多项式复杂度的算法都不能用于具体的应用。聚类算法还应当具有可扩展性，执行时间不应因数据规模和数据维数的线性增高而呈级数级增长。另外，聚类算法应当具有稳定性，数据输入顺序对聚类结果不能产生显著影响。
3. 对结果的解译能力：聚类分析属于描述型的数据开采工具，对类的描述应当简单、易于理解；解译能力对高维数据集合尤为重要，因为用可视化技术表达高维空间比较困难。
4. 隐私保护和数据安全：这是各种数据开采模型都面临的问题。数据开采需要从不同角度和不同的抽象层分析观察数据，有时会侵犯需要保护的数据或信息，这时需要采取措施侦测和阻止不合法的数据入侵行为。

1.4 论文组织

本文对数据开采领域中的聚类算法进行了深入而全面的研究，论文第一章综述数据开采及聚类分析；第二章详细讨论了数据开采领域中具有代表性的聚类算法；第三章给出了一个新的聚类分类数据的算法 CCDCS（Clustering Categorical Data using Clustering Summary），该算法能够高效率地处理大型分类数据集合；在第四章提出的算法 CHID（Clustering HIgh-dimensional Data）能够从高维空间及其子空间发现密集数据点所形成的类；第五章描述了两个混合算法，其中 MARC（Mining Association Rules using Clustering）算法将聚类技术应用到关联规则发现领域以提高开采效率，Structure_Discovery 算法利用聚类发现半结构化、层次型数据中的对象的共有结构；第六章描述了一个具有可扩展性的分布式数据开采应用系统；第七章给出了全文总结并提出了未来工作的研究方向。

1.3.3 面临的挑战

从数据开采应用的角度，对聚类算法要求如下：

1. 处理不同类型的数据：数据开采应用于已有的大型数据库或者数据仓库，而这些数据集合又是为不同应用而设计的，数据类型多种多样。聚类算法不仅要考虑有效处理关系型数据（主要包括数字属性、分类属性以及混合属性的数据），还要考虑其他复杂型数据（如图形、图像、文本、声音）。
2. 效率高、具可扩展性：聚类算法的执行时间必须是可预测的，存在组合爆炸问题或者具有多项式复杂度的算法都不能用于具体的应用。聚类算法还应当具有可扩展性，执行时间不应因数据规模和数据维数的线性增高而呈级数级增长。另外，聚类算法应当具有稳定性，数据输入顺序对聚类结果不能产生显著影响。
3. 对结果的解译能力：聚类分析属于描述型的数据开采工具，对类的描述应当简单、易于理解；解译能力对高维数据集合尤为重要，因为用可视化技术表达高维空间比较困难。
4. 隐私保护和数据安全：这是各种数据开采模型都面临的问题。数据开采需要从不同角度和不同的抽象层分析观察数据，有时会侵犯需要保护的数据或信息，这时需要采取措施侦测和阻止不合法的数据入侵行为。

1.4 论文组织

本文对数据开采领域中的聚类算法进行了深入而全面的研究，论文第一章综述数据开采及聚类分析；第二章详细讨论了数据开采领域中具有代表性的聚类算法；第三章给出了一个新的聚类分类数据的算法 CCDCS（Clustering Categorical Data using Clustering Summary），该算法能够高效率地处理大型分类数据集合；在第四章提出的算法 CHID（Clustering HIgh-dimensional Data）能够从高维空间及其子空间发现密集数据点所形成的类；第五章描述了两个混合算法，其中 MARC（Mining Association Rules using Clustering）算法将聚类技术应用到关联规则发现领域以提高开采效率，Structure_Discovery 算法利用聚类发现半结构化、层次型数据中的对象的共有结构；第六章描述了一个具有可扩展性的分布式数据开采应用系统；第七章给出了全文总结并提出了未来工作的研究方向。

1.5 本章小结

本章为数据开采技术以及聚类分析方法的综述，首先叙述了数据开采的定义、数据开采的过程以及相关领域，然后给出了几种基本的数据开采技术和知识模型，还描述了数据开采领域的国内外现状；接着叙述了什么是聚类分析、数理统计中基本的聚类分析方法，介绍了数据开采中的聚类算法以及聚类算法面临的问题和挑战；最后，简要介绍了本文的研究目标和主要内容。

2 聚类算法

本章将详细介绍数据开采领域中具有代表性的聚类算法，CLARANS（Clustering Large Applications based upon RANdomized Search）、BIRCH（Balanced Iterative Reducing and Clustering）和 DBSCAN（Density Based Spatial Clustering of Applications with Noise）都是经典的数字数据聚类算法，其中 DBSCAN 是基于密度的分区算法；ROCK（Robust Clustering using linKs）算法专为分类属性数据的聚类问题提出，STC（Suffix Tree Clustering）则用于文档聚类。

2.1 CLARANS

Raymond T. Ng 和 Jiawei Han 基于随机搜索以及统计学中的两个聚类算法 PAM（Partitioning Around Medoids）和 CLARA（Clustering LARge Application）[89]，给出了一个适合于大型应用的聚类算法 CLARANS（Clustering Large Applications based upon RANdomized Search）[27]。

PAM 算法属于动态聚类法，它的任务是从 n 个对象中发现 k 个类。PAM 首先任意选择 k 个对象作为凝聚点，称作 medoid，它位于类的中央；接着算法反复交换 medoid 对象和 non-medoid 对象，并利用代价函数计算这种交换对聚类质量的影响，从而选出更好的 medoid，代价函数得到的值小于零时，表示让 non-medoid 对象成为 medoid 对象能够提高聚类质量；一旦交换不能提高聚类质量，算法中止，然后将 non-medoid 对象按最近距离的 medoid 聚类。每一轮聚类质量计算的时间复杂度为 $O(k(n-k)^2)$，当 n 和 k 取值较大时，算法效率非常低。CLARA 算法对此作了改进以适应大型数据集合。CLARA 算法完成与 PAM 相同的任务，但它使用了取样技术，它首先从完整的数据集合中取出一部分作为样品数据，然后在其上使用 PAM 算法找出 medoid 对象，为得到更为准确的结果，CLARA 进行多次取样，将每次在样本数据上得到的 medoid 对象用于整个数据集合计算聚类质量，找出其中质量最好的聚类。对于包含 1000 个对象的数据集合，CLARA 每次抽取的样本数量为 $40 + 2k$，所以使用 PAM 聚类时每一轮的计算时间复杂度为 $O(k(40+k)^2 + k(n-k))$，这表明 CLARA 能够比 PAM 处理更大的数据集，但当抽取的样本并不包括最好的 medoid 时，聚类结果就不会是最好的。

CLARANS 算法集成了 PAM 和 CLARA 的优点。给定 n 个对象，CLARANS 首先构造图 $G_{n,k}$（n 为数据集合的对象数量，k 为类的数量），然后通过图搜索从中发现 k 个 medoid。图 $G_{n,k}$ 由一组以弧线连接的节点构成，每个节点由 k 个对象

2 聚类算法

本章将详细介绍数据开采领域中具有代表性的聚类算法，CLARANS（Clustering Large Applications based upon RANdomized Search）、BIRCH（Balanced Iterative Reducing and Clustering）和 DBSCAN（Density Based Spatial Clustering of Applications with Noise）都是经典的数字数据聚类算法，其中 DBSCAN 是基于密度的分区算法；ROCK（Robust Clustering using linKs）算法专为分类属性数据的聚类问题提出，STC（Suffix Tree Clustering）则用于文档聚类。

2.1 CLARANS

Raymond T. Ng 和 Jiawei Han 基于随机搜索以及统计学中的两个聚类算法 PAM（Partitioning Around Medoids）和 CLARA（Clustering LARge Application）[89]，给出了一个适合于大型应用的聚类算法 CLARANS（Clustering Large Applications based upon RANdomized Search）[27]。

PAM 算法属于动态聚类法，它的任务是从 n 个对象中发现 k 个类。PAM 首先任意选择 k 个对象作为凝聚点，称作 medoid，它位于类的中央；接着算法反复交换 medoid 对象和 non-medoid 对象，并利用代价函数计算这种交换对聚类质量的影响，从而选出更好的 medoid，代价函数得到的值小于零时，表示让 non-medoid 对象成为 medoid 对象能够提高聚类质量；一旦交换不能提高聚类质量，算法中止，然后将 non-medoid 对象按最近距离的 medoid 聚类。每一轮聚类质量计算的时间复杂度为 $O(k(n-k)^2)$，当 n 和 k 取值较大时，算法效率非常低。CLARA 算法对此作了改进以适应大型数据集合。CLARA 算法完成与 PAM 相同的任务，但它使用了取样技术，它首先从完整的数据集合中取出一部分作为样品数据，然后在其上使用 PAM 算法找出 medoid 对象，为得到更为准确的结果，CLARA 进行多次取样，将每次在样本数据上得到的 medoid 对象用于整个数据集合计算聚类质量，找出其中质量最好的聚类。对于包含 1000 个对象的数据集合，CLARA 每次抽取的样本数量为 $40+2k$，所以使用 PAM 聚类时每一轮的计算时间复杂度为 $O(k(40+k)^2+k(n-k))$，这表明 CLARA 能够比 PAM 处理更大的数据集，但当抽取的样本并不包括最好的 medoid 时，聚类结果就不会是最好的。

CLARANS 算法集成了 PAM 和 CLARA 的优点。给定 n 个对象，CLARANS 首先构造图 $G_{n,k}$（n 为数据集合的对象数量，k 为类的数量），然后通过图搜索从中发现 k 个 medoid。图 $G_{n,k}$ 由一组以弧线连接的节点构成，每个节点由 k 个对象

集合表示$\{O_{m1}, \ldots, O_{mk}\}$，这 k 个对象代表选中的 k 个 medoid；以弧线连接的两个节点为邻居节点，给定两个节点 $S_1 = \{O_{m1}, \ldots, O_{mk}\}$, $S_2 = \{O_{w1}, \ldots, O_{wk}\}$，当且仅当 $|S_1 \cap S_2| = k - 1$ 时，S_1 和 S_2 互为邻居节点，容易得出每个节点有 $k(n - k)$个邻居。由于相邻两个节点只相差一个 medoid，可以利用代价函数比较两个节点的聚类质量。

图 $G_{n,k}$ 中每个节点对应一种聚类结果，因此每个节点都对应着一个代价值代表聚类质量，这个代价值为每个对象和其所在类的 medoid 之间的平均距离，记作 Cost。PAM 算法可以看作是在图中搜索 Cost 值最小的节点，在每一轮中检验当前节点的所有邻居，利用代价函数找出 Cost 值较小的邻居代替当前节点，直到发现 Cost 值最小的节点，每一轮需要检验 $k(n - k)$个节点，算法效率不高；类似地，CLARA 算法可以看作是在 $G_{n,k}$ 的子图中搜索 Cost 值最小的节点，CLARA 存在的问题是，聚类结果的质量依赖所选择的子图，如果具有最小 Cost 值的节点不被包括在子图中，就得不到最好结果。

CLARANS 同 CLARA 类似，都采用了随机取样，但不同的是，CLARA 利用随机取样在整个数据集合内划定搜索范围，CLARANS 则是对当前节点的邻居随机取样，它不会像 PAM 算法一样检验当前节点的所有邻居，只检验随机抽取的邻居样本。CLARANS 算法如下：

CLARANS（*NumLocal*，*MaxNeighbor*）

1. $i = 1$；*MinCost* = 足够大的数；
2. 在图 $G_{n,k}$ 中任意选择一个节点作为当前节点 *CurrentNode*；
3. $j = 1$；
4. 随机选择 *CurrentNode* 的一个邻居节点 *S*，然后利用代价函数比较 *CurrentNode* 和 *S* 的聚类质量；
5. 如果 *S* 的聚类质量较高，则 *CurrentNode* = *S*，然后执行步骤 3；
6. 否则，$j = j + 1$，如果 $j <= MaxNeighbor$，则执行步骤 4；
7. 如果 $j > MaxNeighbor$，则比较 *CurrentNode* 的 *Cost* 值和 *MinCost* 值的大小，如果前者小，则 *MinCost* = *CurrentNode* 的 *Cost* 值，并且最佳节点 *BestNode* = *CurrentNode*；
8. $i = i + 1$，如果 $i > NumLocal$，则算法中止，输出 *BestNode*，否则，执行步骤 2。

参数 *MaxNeighbor* 代表对当前节点，最多检验 *MaxNeighbor* 个邻居，*NumLocal* 代表最多随机选取 *NumLocal* 个节点作为当前节点。

然而，CLARANS 也有其局限性，Ester 等人提出了 *R**树和聚焦技术对其进行改进[92]。首先，CLARANS 假设对象集合都可以存储在内存中，当数据库较大，

该假设不能成立，Ester 等人引入了一种有效的空间存取方法 *R**树来解决这个问题；其次，CLARANS 算法在计算聚类结果的质量（即 *Cost*，每个对象和其所在类的 medoid 之间的平均距离）时耗时太长，Ester 等人提出聚焦技术减少这一步骤的开销，聚焦技术包括两个方面，一方面减少计算 medoid 时用到的对象，称作代表对象的聚焦，*R**树叶子节点存放的都是邻居节点，在计算时只考虑位于叶子节点最中心位置的那些邻居节点，这种抽取样本邻居的方法能够保证得到较好聚类结果，缺点是可能会忽略那些更好的 medoid。聚焦技术的另一个方面是相关类聚焦和类聚焦，使用 *R**结构只检查能够提高聚类质量的对象。Ester 等人的报告表明，聚焦技术能够大大提高聚类效率，而对聚类结果准确性影响较小。

2.2 BIRCH

Tian Zhang 等人提出的聚类算法 BIRCH（Balanced Iterative Reducing and Clustering）[28]通过逐个读入对象构造 CF（Clustering Feature）树完成聚类过程，因此是一个增量方法，BIRCH 另一个特点是可以通过控制 CF 树的大小使存储需求符合实际内存大小，适合大型数据库。先介绍 CF 和 CF 树的概念：

CF 概念基于向量空间，给定一个包含 N 个 d 维对象的类$\{\vec{X}_i\}$，$i = 1, 2, \ldots, N$，该类的 CF 定义为一个三元组：$CF = (N, \overrightarrow{LS}, SS)$，其中 N 代表类中对象的个数、$\overrightarrow{LS}$ 代表这 N 个对象的线性和，即 $\sum_{i=1}^{N}\vec{X}_i$，SS 代表平方和，即 $\sum_{i=1}^{N}\vec{X}_i^2$。令 R 为类的半径，即类中每个对象和质心对象距离的均值。假设 $CF_1 = (N_1, \overrightarrow{LS}_1, SS_1)$；$CF_2 = (N_2, \overrightarrow{LS}_2, SS_2)$为两个不相交子类的 CF，将这两个子类合并后得到一个新的类，该类的 CF 为：

$$CF_1 + CF_2 = (N_1 + N_2, \overrightarrow{LS}_1 + \overrightarrow{LS}_2, SS_1 + SS_2)$$

CF 包含了足够用于聚类计算的信息，同时由于它仅存储子类的汇总信息使得算法效率得到提高。

CF 树是高度平衡树，带有两个参数：分支因子 B 或 L 以及一个阈值 T，每个非叶子节点最多包含 B 个项目，形式为$[CF_i, child_i]$，i = 1, 2, …, B，$child_i$ 是指向第 i 个子节点的指针，CF_i 是第 i 个子节点所代表子类的 CF；每个叶子节点最多包含 L 个项目，每个项目是其子类的一个 CF，另外，每个叶子节点有两个指针分别指向左右两边的叶子节点，这些指针将 CF 树中所有叶子节点链接起来；CF 树中每个节点都可以看作由其 B 个或 L 个项目所代表的子类组成。所有叶子节点子类的半径必须小于阈值 T。CF 树的大小可以通过调整 T 得到控制，T 值越大树越小。CF 树是在不断插入对象的过程中动态建立起来的，每读入一个对象，将它

该假设不能成立，Ester 等人引入了一种有效的空间存取方法 *R**树来解决这个问题；其次，CLARANS 算法在计算聚类结果的质量（即 *Cost*，每个对象和其所在类的 medoid 之间的平均距离）时耗时太长，Ester 等人提出聚焦技术减少这一步骤的开销，聚焦技术包括两个方面，一方面减少计算 medoid 时用到的对象，称作代表对象的聚焦，*R**树叶子节点存放的都是邻居节点，在计算时只考虑位于叶子节点最中心位置的那些邻居节点，这种抽取样本邻居的方法能够保证得到较好聚类结果，缺点是可能会忽略那些更好的 medoid。聚焦技术的另一个方面是相关类聚焦和类聚焦，使用 *R**结构只检查能够提高聚类质量的对象。Ester 等人的报告表明，聚焦技术能够大大提高聚类效率，而对聚类结果准确性影响较小。

2.2 BIRCH

Tian Zhang 等人提出的聚类算法 BIRCH（Balanced Iterative Reducing and Clustering）[28]通过逐个读入对象构造 CF（Clustering Feature）树完成聚类过程，因此是一个增量方法，BIRCH 另一个特点是可以通过控制 CF 树的大小使存储需求符合实际内存大小，适合大型数据库。先介绍 CF 和 CF 树的概念：

CF 概念基于向量空间，给定一个包含 N 个 d 维对象的类$\{\vec{X}_i\}$，$i = 1, 2, \ldots, N$，该类的 CF 定义为一个三元组：$CF = (N, \overrightarrow{LS}, SS)$，其中 N 代表类中对象的个数、$\overrightarrow{LS}$ 代表这 N 个对象的线性和，即 $\sum_{i=1}^{N}\vec{X}_i$，SS 代表平方和，即 $\sum_{i=1}^{N}\vec{X}_i^2$。令 R 为类的半径，即类中每个对象和质心对象距离的均值。假设 $CF_1 = (N_1, \overrightarrow{LS}_1, SS_1)$；$CF_2 = (N_2, \overrightarrow{LS}_2, SS_2)$为两个不相交子类的 CF，将这两个子类合并后得到一个新的类，该类的 CF 为：

$$CF_1 + CF_2 = (N_1 + N_2, \overrightarrow{LS}_1 + \overrightarrow{LS}_2, SS_1 + SS_2)$$

CF 包含了足够用于聚类计算的信息，同时由于它仅存储子类的汇总信息使得算法效率得到提高。

CF 树是高度平衡树，带有两个参数：分支因子 B 或 L 以及一个阈值 T，每个非叶子节点最多包含 B 个项目，形式为$[CF_i, child_i]$，i = 1, 2, …, B，$child_i$ 是指向第 i 个子节点的指针，CF_i 是第 i 个子节点所代表子类的 CF；每个叶子节点最多包含 L 个项目，每个项目是其子类的一个 CF，另外，每个叶子节点有两个指针分别指向左右两边的叶子节点，这些指针将 CF 树中所有叶子节点链接起来；CF 树中每个节点都可以看作由其 B 个或 L 个项目所代表的子类组成。所有叶子节点子类的半径必须小于阈值 T。CF 树的大小可以通过调整 T 得到控制，T 值越大树越小。CF 树是在不断插入对象的过程中动态建立起来的，每读入一个对象，将它

看作一个子类，将该子类的 CF，记作 Ent，插入到树中，过程如下：

1. 为 CF 找到合适的叶子节点：从树的根节点开始，选择一个距离最小的子节点作为下一个目的地，直至到达树的叶子节点。

2. 修改叶子节点：到达叶子节点后，找到距离最小的项目 L_i，检测 Ent 和 L_i 是否能够合并，即合并后子类的半径是否还能够满足阈值 T，如果能够合并，则将 Ent 加入到 L_i，如果不能，则根据 Ent 为该叶子节点创建新的项目，这时需要检验该叶子节点是否还有空间容纳得下新的项目，如果不能，则进行节点分裂操作，首先选择距离最远的对象对分别作为凝聚点，然后根据与凝聚点的距离远近分配余下的对象。

3. 修改路径：Ent 插入到叶子节点后，跟着修改路径上各非叶子节点的项目值。如果 Ent 插入没有引起节点分裂，则加入 Ent 到各节点相应的项目中；否则，分裂节点的父节点需要增加新项目，这种增加又可能会导致父节点的分裂，重复此过程可能会导致根节点分裂，这时 CF 树将增加一层。

4. 合并精炼：节点分裂操作会导致聚类结果受数据读入顺序的影响，算法采用合并精炼减少这种影响：假设插入 Ent 导致某叶节点分裂，并且由它引起的一连串分裂在非叶子节点 N_j 停止，这时算法检查 N_j 中距离最近的项目对，如果它们不是引起分裂的项目，则合并它们，合并后占用的空间若大于一个页面，需要重新分裂，重分裂的过程与节点分裂不同，先将其中一个凝聚点代表的子类节点装满，余下的分配到另一个子类中。这么做有利于充分利用存储空间并且延缓下一轮分裂的发生。

根据实际内存的要求，有时需要增大阈值 T，这时要重建 CF 树，然而这种重建并不是逐个读入对象重新构造，而是在现有的 CF 树上作修改。假设 CF 树 t_i 的阈值为 T_i，高度为 h，节点数为 S_i，利用 t_i 所有的叶节点项目重新构造 CF 树 t_{i+1}，其阈值为 T_{i+1}，$T_{i+1} > T_i$，很明显，t_{i+1} 的节点数不会大于 S_i，重建过程如下：

1. CF 树的每一个叶子节点都对应着一条自根节点起始的路径，按照从左至右的顺序给路径排序，并复制 t_i 的第一条路径到 t_{i+1}。

2. 复制下一条路径 p_i 到 t_{i+1}，接着将 p_i 叶子节点中的项目逐个插入到 t_{i+1} 中，插入过程同插入算法相似，只是不处理分裂过程，即当创建新项目会引起节点分裂时终止插入过程，若项目没有插入到 p_i，则从 p_i 的叶子节点中删除该项目。

3. 处理完叶子节点中的所有项目后，删除 p_i 中的空节点；

4. 重复步骤 2 直至处理完 t_i 中所有路径。

2.3 DBSCAN

DBSCAN（Density Based Spatial Clustering of Applications with Noise）[92]提出了基于密度的聚类概念，它的基本思想是：对于类中的每一个对象 O，必须满足以 O 为圆心，在半径为 *Eps* 的球体内存在不少于 *MinPts* 个邻居对象。给定对象集合 D、*Eps* 和 *MinPts* 以及两个对象 p 和 q：

定义 1：directly density-reachable，如果满足 $p \in N_{Eps}(q)$（$N_{Eps}(q)$是 D 的子集，存在于对象 q 的 *Eps* 半径内所有邻居对象的集合），并且 $|N_{Eps}(q)| \geq MinPts$，则称 p 是对象 q 的 directly density-reachable 对象。

定义 2：density-reachable，如果存在一系列对象 $p_1, p_2, \ldots, p_n$，其中 $p_1 = q, p_n = p$, $p_i \in D$，并且 p_{i+1} 是 p_i 的 directly density-reachable 对象，则称 p 是对象 q 的 density-reachable 对象，记作 $p >_D q$。

density-reachable 是可传递不对称关系，只有当对象都满足 $|N_{Eps}(o)| \geq MinPts$ 时，才会有对称关系，对于处在同一个类边缘地带的两个对象，有可能都不是对方的 density-reachable 对象，因为它们极有可能不满足条件 $|N_{Eps}(o)| \geq MinPts$，所以需要提出 density-connected 概念。

定义 3：density-connected，如果存在对象 $o \in D$，使得 p 和 q 都是 o 的 density-reachable 对象，则称 p 和 q 之间存在 density-connected 关系，这是一个对称关系。

定义 4：类，类 C 是 D 的非空子集，满足下列条件：① $\forall p, q \in D$，如果 $p \in C$ 并且 $p >_D q$，那么同时 $q \in C$；② $\forall p, q \in C$，p 与 q 存在 density-connected 关系。

定义 5：噪声，$C_1, C_2, \ldots, C_k$ 为 D 中的聚类，将噪声定义为不属于其中任何一类的对象，即 $\{p \in D \mid \forall i: p \notin C_i\}$。

一个类中的对象可以分作两类：核心（core）对象和非核心（non-core）对象，核心对象指满足条件 $|N_{Eps}(o)| \geq MinPts$ 的对象，不满足该条件的对象就是非核心对象，非核心对象又可以分为边缘（border）对象和噪声对象，边缘对象是非核心对象，同时它应当是核心对象的 density-reachable 对象，噪声对象既不是核心对象也不是其他对象的 density-reachable 对象。

DBSCAN 算法就是要根据上述定义快速发现数据集合中的类和噪声。它首先

4. 重复步骤 2 直至处理完 t_i 中所有路径。

2.3 DBSCAN

DBSCAN（Density Based Spatial Clustering of Applications with Noise）[92]提出了基于密度的聚类概念，它的基本思想是：对于类中的每一个对象 O，必须满足以 O 为圆心，在半径为 *Eps* 的球体内存在不少于 *MinPts* 个邻居对象。给定对象集合 D、*Eps* 和 *MinPts* 以及两个对象 p 和 q：

定义 1：directly density-reachable，如果满足 $p \in N_{Eps}(q)$（$N_{Eps}(q)$是 D 的子集，存在于对象 q 的 *Eps* 半径内所有邻居对象的集合），并且 $|N_{Eps}(q)| \geq MinPts$，则称 p 是对象 q 的 directly density-reachable 对象。

定义 2：density-reachable，如果存在一系列对象 $p_1, p_2, \ldots, p_n$，其中 $p_1 = q, p_n = p$, $p_i \in D$，并且 p_{i+1} 是 p_i 的 directly density-reachable 对象，则称 p 是对象 q 的 density-reachable 对象，记作 $p >_D q$。

density-reachable 是可传递不对称关系，只有当对象都满足 $|N_{Eps}(o)| \geq MinPts$ 时，才会有对称关系，对于处在同一个类边缘地带的两个对象，有可能都不是对方的 density-reachable 对象，因为它们极有可能不满足条件 $|N_{Eps}(o)| \geq MinPts$，所以需要提出 density-connected 概念。

定义 3：density-connected，如果存在对象 $o \in D$，使得 p 和 q 都是 o 的 density-reachable 对象，则称 p 和 q 之间存在 density-connected 关系，这是一个对称关系。

定义 4：类，类 C 是 D 的非空子集，满足下列条件：① $\forall p, q \in D$，如果 $p \in C$ 并且 $p >_D q$，那么同时 $q \in C$；② $\forall p, q \in C$，p 与 q 存在 density-connected 关系。

定义 5：噪声，$C_1, C_2, \ldots, C_k$ 为 D 中的聚类，将噪声定义为不属于其中任何一类的对象，即 $\{p \in D \mid \forall i: p \notin C_i\}$。

一个类中的对象可以分作两类：核心（core）对象和非核心（non-core）对象，核心对象指满足条件 $|N_{Eps}(o)| \geq MinPts$ 的对象，不满足该条件的对象就是非核心对象，非核心对象又可以分为边缘（border）对象和噪声对象，边缘对象是非核心对象，同时它应当是核心对象的 density-reachable 对象，噪声对象既不是核心对象也不是其他对象的 density-reachable 对象。

DBSCAN 算法就是要根据上述定义快速发现数据集合中的类和噪声。它首先

从 D 中任意选取一个对象 p；然后找到 p 的所有 density-reachable 对象，如果 p 是核心对象，则该步骤将产生一个类；如果 p 是一个边缘对象，则它没有 density-reachable 对象，将 p 定为噪声；接着算法从 D 中选取下一个对象重复该过程。算法如下：

DBSCAN（D，Eps，$MinPts$）

1. 读取 D 中任意一个未分类的对象 o；
2. 检索出属于 $N_{Eps}(o)$的所有对象；
3. 如果$|N_{Eps}(o)| < MinPts$（即 o 为非核心对象），则将 o 标记为噪声并执行步骤 1；
4. 否则（即 o 为核心对象），给 $N_{Eps}(o)$中的所有对象打上一个新的类标签 newid，然后将这些对象压入堆栈 seeds 中；
5. 让 CurrentObject = seeds.top，然后检索属于 N_{Eps}(CurrentObject)的所有对象，如果$|N_{Eps}(\text{CurrentObject})| \geq MinPts$，则剔出已经打上标记（要么已经分类，要么为噪声对象）的对象，将余下的未分类对象打上类标签 newid，然后压入堆栈；
6. seeds.pop，判断 seeds 是否为空，是则执行步骤 1，否则执行步骤 5；

Ester 等人又发现在已经完成聚类的数据集合 D 中删除或新增一个对象只对其邻居对象产生影响，因此 Ester 等人又提出了一种基于 DBSCAN 的增量聚类算法[116]。

2.4 ROCK

ROCK（Robust Clustering using linKs）是由 Sudipto Guha 等人提出的一种具有鲁棒性的聚类分类属性数据的算法[94]，由于距离度量方法不适合分类属性的数据，ROCK 算法使用了连接（links）概念衡量对象之间的相似性。在现实世界的数据库中，分类属性的数据存在某些属性为空值的情况，为解决这个问题，ROCK 将分类属性的数据转换为交易数据，一条交易是一组项目的集合，如交易 T = {肥皂, 牙刷, 毛巾, 面包}；分类属性的值域是一个有限集合，所以可以将属性 A 及其值$\{v_1, v_2, \ldots, v_n\}$转换为项目 $A.v_1, A.v_2, \ldots, A.v_n$，按照这种方式处理数据对象的其它分类属性值以完成转换。

邻居：给定对象 p_i和 p_j，如果 p_i和 p_j的相似度 $\text{sim}(p_i, p_j) \geq \theta$，则 p_i和 p_j互为邻居。θ 为用户事先定义的阈值；$\text{sim}(p_i, p_j)$为衡量对象相似度的函数，对于数字属性的数据，它可以是距离函数，在 ROCK 中，使用 Jaccard 系数：$\text{sim}(p_i, p_j) = |p_i \cap p_j| / |p_i \cup p_j|$。

连接：$\text{link}(p_i, p_j)$定义为对象 p_i和 p_j所共有邻居对象的个数，从定义得出，连

从 D 中任意选取一个对象 p；然后找到 p 的所有 density-reachable 对象，如果 p 是核心对象，则该步骤将产生一个类；如果 p 是一个边缘对象，则它没有 density-reachable 对象，将 p 定为噪声；接着算法从 D 中选取下一个对象重复该过程。算法如下：

DBSCAN（D，Eps，$MinPts$）

1. 读取 D 中任意一个未分类的对象 o；
2. 检索出属于 $N_{Eps}(o)$的所有对象；
3. 如果$|N_{Eps}(o)| < MinPts$（即 o 为非核心对象），则将 o 标记为噪声并执行步骤 1；
4. 否则（即 o 为核心对象），给 $N_{Eps}(o)$中的所有对象打上一个新的类标签 newid，然后将这些对象压入堆栈 seeds 中；
5. 让 CurrentObject = seeds.top，然后检索属于 N_{Eps}(CurrentObject)的所有对象，如果$|N_{Eps}(\text{CurrentObject})| \geq MinPts$，则剔出已经打上标记（要么已经分类，要么为噪声对象）的对象，将余下的未分类对象打上类标签 newid，然后压入堆栈；
6. seeds.pop，判断 seeds 是否为空，是则执行步骤 1，否则执行步骤 5；

Ester 等人又发现在已经完成聚类的数据集合 D 中删除或新增一个对象只对其邻居对象产生影响，因此 Ester 等人又提出了一种基于 DBSCAN 的增量聚类算法[116]。

2.4 ROCK

ROCK（Robust Clustering using linKs）是由 Sudipto Guha 等人提出的一种具有鲁棒性的聚类分类属性数据的算法[94]，由于距离度量方法不适合分类属性的数据，ROCK 算法使用了连接（links）概念衡量对象之间的相似性。在现实世界的数据库中，分类属性的数据存在某些属性为空值的情况，为解决这个问题，ROCK 将分类属性的数据转换为交易数据，一条交易是一组项目的集合，如交易 T = {肥皂, 牙刷, 毛巾, 面包}；分类属性的值域是一个有限集合，所以可以将属性 A 及其值$\{v_1, v_2, \ldots, v_n\}$转换为项目 $A.v_1, A.v_2, \ldots, A.v_n$，按照这种方式处理数据对象的其它分类属性值以完成转换。

邻居：给定对象 p_i和 p_j，如果 p_i和 p_j的相似度 $\text{sim}(p_i, p_j) \geq \theta$，则 p_i和 p_j互为邻居。θ 为用户事先定义的阈值；$\text{sim}(p_i, p_j)$为衡量对象相似度的函数，对于数字属性的数据，它可以是距离函数，在 ROCK 中，使用 Jaccard 系数：$\text{sim}(p_i, p_j) = |p_i \cap p_j| / |p_i \cup p_j|$。

连接：$\text{link}(p_i, p_j)$定义为对象 p_i和 p_j所共有邻居对象的个数，从定义得出，连

接值越大，则 p_i 和 p_j 越相似，越有可能属于同一个类。ROCK 算法需要使用连接函数来判定对象是否应当被合并到一个类中，与以往使用的判别标准 sim 函数相比，基于连接的方法获取有关数据对象分布的全局信息，所以这种方法具有鲁棒性。

标准函数：聚类算法中的标准函数用来表示类的质量高低，如果将标准函数定义为类中对象对连接值的和，那么固然也能够保证有着较大连接值的对象对被分配到同一个类中，却有可能出现最终集合中所有对象都被分配到一个类中，为避免发生这种状况，算法找到类 C_i 的连接期望值 $n_i^{1+2f(\theta)}$，然后用它去除类 C_i 的连接值之和，这样当连接值很小的对象对被分配到同一个类的时候，会导致该类连接期望值比实际值要大，因此，标准函数值就会减小，显示出该类质量不高，提示用户考虑重新分配。ROCK 标准函数如下：

$$Q = n_i * \sum_{p_q, p_r \in C_i} \frac{link(p_q, p_r)}{n_i^{1+2f(\theta)}} \tag{2.1}$$

基于同样的原因，公式(2.2)给出了两个类之间的相似性函数 $g(C_i, C_j)$取代连接值，其中 $link[C_i, C_j] = \sum_{p_q \in C_i, p_r \in C_j} link(p_q, p_r)$，该函数用于判别 C_i 和 C_j 是否应当合并为一个类。

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}} \tag{2.2}$$

ROCK 算法首先随机从集合中抽取样本数据，然后利用连接函数和系统方法在样本数据上聚类，最终根据样本数据给其余数据分类。算法如下，参数 S 为样本数据，k 为要得到的分类数目：

```
ROCK（S, k）
1. 搜索 S 中每个对象 s_i 的邻居并存放到 NeighborList[i]中；
2. 初始化二维数组 link[i, j]；
3. for i = 1 to |S| do
4.     N = NeighborList[i];
5.     for j = 1 to |N - 1| do
6.         for l = j + 1 to |N| do
7.             link[N[j], N[l]] := link[N[j], N[l]] + 1；
8. 将 S 中每个对象看作单独的类，计算 S 中每个类 i 与其他类 j 的 link[i,
   j]，如果不为 0，则计算 g[i,j]并按照其降序将 j 存放到 q[i]中；
9. 对 S 中的每个类 j，根据 g[j, max(q[j])]对 j 排序，并按照降序将 j 存
   放在 Q 中；
```

10. while size(Q) > k do {
11. u = Q 中排列在第一位的类；
12. v = $q[u]$中排列在第一位的类；
13. 从 Q 中删除 v；
14. 将 u 和 v 合并成 w；
15. 对 $q[u] \cup q[v]$中的每个类 x，从 $q[x]$中删除与 u 和 v 有关的元素；
16. 计算 $g[x, w]$，按照其值大小分别在 $q[x]$、$q[w]$中插入 x、w；
17. 更新 Q；
18. 插入 w 到 Q 中；
19. 删除 $q[u]$和 $q[v]$；
20. }

为计算 S 中所有对象对的连接值，构造一个 $n \times n$（$n = |S|$）的邻接矩阵 A，$A[i, j]$等于 1 表示对象 i 和对象 j 为邻居，等于 0 则不为邻居，步骤 2 至 7 则根据这个矩阵计算 i 和 j 的连接值；步骤 10 至 20 为一个 while 循环，在每一轮选取 $g(i,j)$值为最大的类合并，直到剩下 k 个类，停止合并，中止算法；线性表 q 中的每一个元素 $q[i]$也是一个线性表，计算 S 中类 i 和 S 中其余类 j 的 $g(i, j)$，并按照 $g(i, j)$值从大到小存放 j，线性表 Q 则找到 q 中每个元素 $q[i]$的第一个值 max($q[i]$)（即最大值），获取 $g(i, \max(q[i]))$，并按照该值由大到小存放 i；线性表 Q 和 q 的使用是为了快速找到 $g(i,j)$值最大的对象对，以便合并，所以每轮合并后，需要更新这两个表。

2.5 STC

STC（Suffix Tree Clustering）[104]是文档聚类算法，文档聚类在情报检索（Information Retrieve）领域得到广泛的研究[117]，随着 Web 技术的发展和 Web 文档的不断增多积累，文档聚类被应用到 Web 开采领域。文档聚类一般的做法是将文档集合看作 n 维空间$\{d_1, d_2, \ldots, d_n\}$，其中 $d_j(j = 1, 2, \ldots, n)$是文档集合包含的所有单词，每一个文档可以看作是这个 n 维空间的一个数据点，然后再对这些数据点进行聚类。与一般做法不同，STC 算法将文档处理成短语，然后找出那些共同拥有短语较多的文档，将它们分配到同一个类中，STC 算法大致分为三个步骤：① 文档清洗；② 确定基本分类；③ 合并基本分类。

文档清洗步骤使用词干获取算法删除单词的前缀、后缀，并将词的双数形式变为单数形式，在句子和句子间作上标记，剔除句子中那些非单词符号，如标点符号、数字等。

确定基本分类：该过程可以看作是为文档集合中所有短语构造倒排索引，算

10. while size(Q) > k do {
11. u = Q 中排列在第一位的类；
12. v = $q[u]$中排列在第一位的类；
13. 从 Q 中删除 v；
14. 将 u 和 v 合并成 w；
15. 对 $q[u] \cup q[v]$中的每个类 x，从 $q[x]$中删除与 u 和 v 有关的元素；
16. 计算 $g[x, w]$，按照其值大小分别在 $q[x]$、$q[w]$中插入 x、w；
17. 更新 Q；
18. 插入 w 到 Q 中；
19. 删除 $q[u]$和 $q[v]$；
20. }

为计算 S 中所有对象对的连接值，构造一个 $n \times n$（$n = |S|$）的邻接矩阵 A，$A[i, j]$等于 1 表示对象 i 和对象 j 为邻居，等于 0 则不为邻居，步骤 2 至 7 则根据这个矩阵计算 i 和 j 的连接值；步骤 10 至 20 为一个 while 循环，在每一轮选取 $g(i,j)$值为最大的类合并，直到剩下 k 个类，停止合并，中止算法；线性表 q 中的每一个元素 $q[i]$也是一个线性表，计算 S 中类 i 和 S 中其余类 j 的 $g(i, j)$，并按照 $g(i, j)$值从大到小存放 j，线性表 Q 则找到 q 中每个元素 $q[i]$的第一个值 max($q[i]$)（即最大值），获取 $g(i, \max(q[i]))$，并按照该值由大到小存放 i；线性表 Q 和 q 的使用是为了快速找到 $g(i,j)$值最大的对象对，以便合并，所以每轮合并后，需要更新这两个表。

2.5 STC

STC（Suffix Tree Clustering）[104]是文档聚类算法，文档聚类在情报检索（Information Retrieve）领域得到广泛的研究[117]，随着 Web 技术的发展和 Web 文档的不断增多积累，文档聚类被应用到 Web 开采领域。文档聚类一般的做法是将文档集合看作 n 维空间$\{d_1, d_2, \ldots, d_n\}$，其中 $d_j(j = 1, 2, \ldots, n)$是文档集合包含的所有单词，每一个文档可以看作是这个 n 维空间的一个数据点，然后再对这些数据点进行聚类。与一般做法不同，STC 算法将文档处理成短语，然后找出那些共同拥有短语较多的文档，将它们分配到同一个类中，STC 算法大致分为三个步骤：① 文档清洗；② 确定基本分类；③ 合并基本分类。

文档清洗步骤使用词干获取算法删除单词的前缀、后缀，并将词的双数形式变为单数形式，在句子和句子间作上标记，剔除句子中那些非单词符号，如标点符号、数字等。

确定基本分类：该过程可以看作是为文档集合中所有短语构造倒排索引，算

法选择使用后缀树（Suffix Tree）完成倒排索引的构造，利用后缀树有两个好处，一个是整个构造过程耗费的时间与文档集合成线性比，具有可扩展性；另一个是新读入文档不影响已经构造好的部分，是一个增量方法。关于字符串 S 的后缀树有如下描述：

1. 后缀树是具有根节点的有向树。
2. 每个非叶子节点至少有两个子节点。
3. 每一条边都有标记，为 S 的一个非空子串；每个节点的标记也是一个字符串，该字符串为根节点到该节点的路径上所有边的标记组合而成的字符串。
4. 从同一个节点出来的边，没有两条是以相同的单词打头的，因此后缀树是一种压缩数据结构。
5. 对于字符串 S 的每个后缀，都存在一个后缀节点，标记为 s。

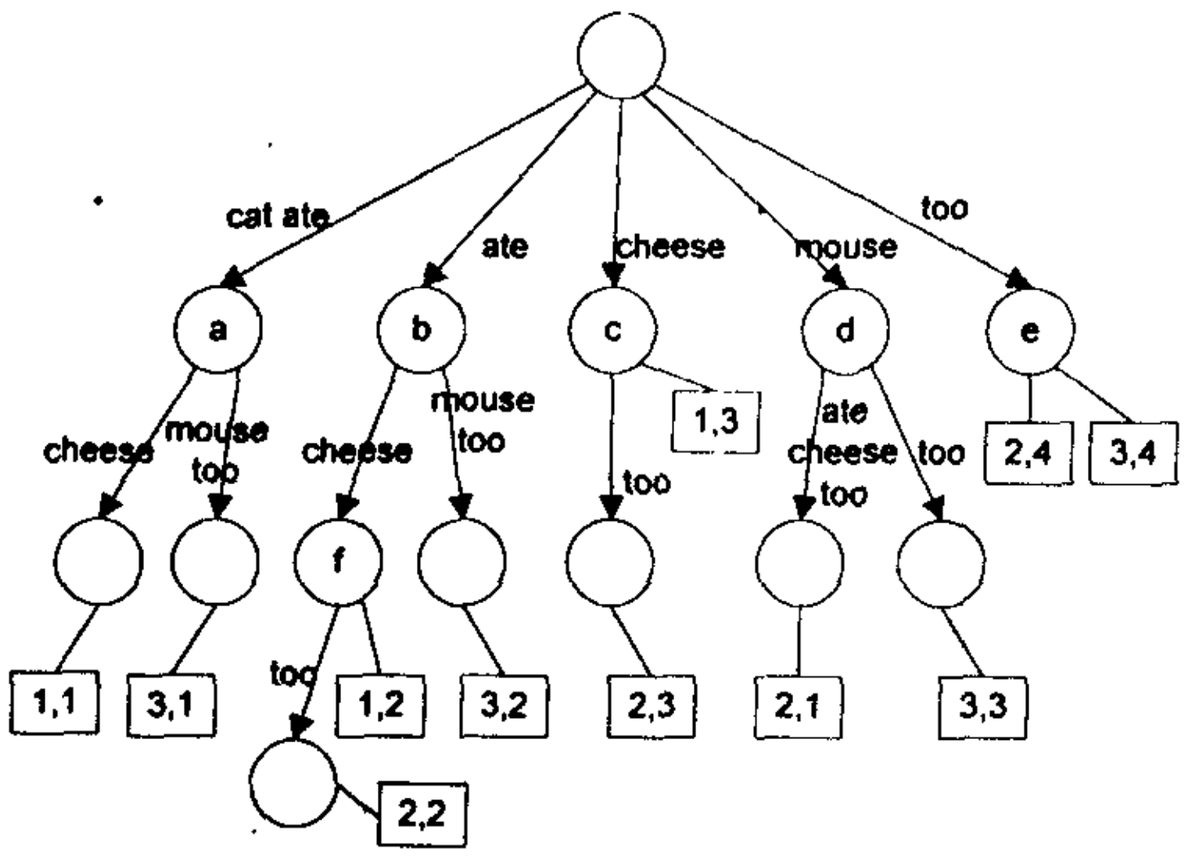

图 2.1 后缀树

字符串集合的后缀树则为包含所有字符串后缀的压缩树。用一个例子来说明后缀树的构造，给定一个字符串集合，包含三条记录，分别是 1：“cat ate cheese”、2：“mouse ate cheese too”、3：“cat ate mouse too”，图 2.1 给出了该字符串集合的后缀树，树中的节点用圆圈表示，每个后缀节点附带有一个或多个方框，方框内有两个数字，第一个数字代表后缀来自哪个字符串，为字符串标识；后一个数字说明了后缀节点的标记是字符串中的哪个后缀。后缀树中的每一个节点代表一个

短语以及包含这个短语的所有文档，节点标记就是这个短语，与该节点在一条路径上的所有后缀节点就可以组成包含这个短语的所有文档集合，因此，每个节点就是一个基本类，表 1 给出了与图 2.1 对应的基本类及其短语。给定一个基本类 B，其短语为 P，用函数 $s(B)$为基本类 B 评分：$s(B) = |B| \cdot f(|P|)$，$|B|$为类 B 包含的文档数，$|P|$为短语的长度，函数 f 用来控制短语的长度值，当短语中某个单词在文档中出现率过高或过低，又或者该单词为用户事先列出的停止字，则计算短语长度时，该单词忽略不计，这样，当短语长度在 2 至 6 个单词的范围内时，$s(B)$线性增长，短语过长时，$s(B)$则趋向为一个常数。

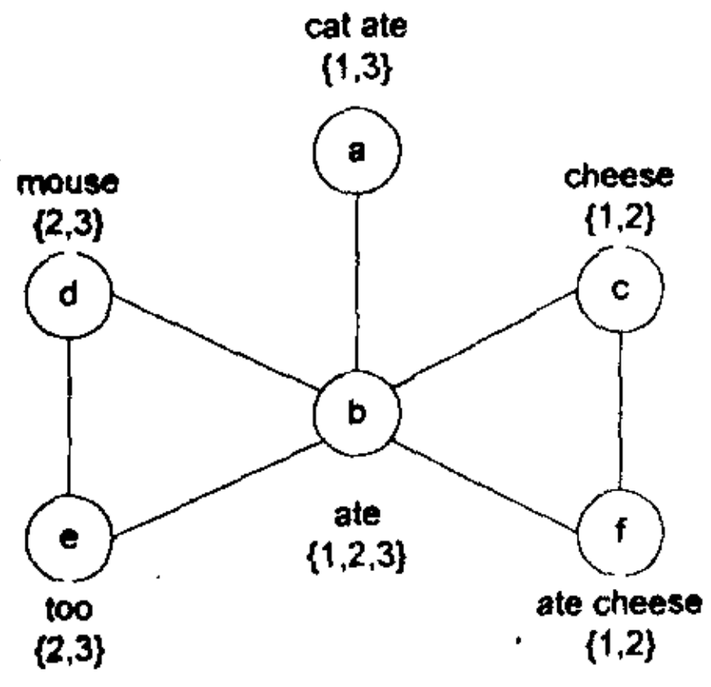

图 2.2 合并基本类后得到聚类结果

表 2.1 与图 2.1 对应的基本分类

节点	短语	文档
a	cat ate	1, 3
b	ate	1, 2, 3
c	cheese	1, 2
d	mouse	2, 3
e	too	2, 3
f	ate cheese	1, 2

基本分类是在共享单个短语的基础上得到的，然而，文档可以共享多个短语，那么必定存在相似的文档甚至同一个文档被分配到不同的基本分类中，这种重叠必然导致聚类质量不高，因此，算法的第三步要合并那些高度重叠的基本分类。

考虑两个基本分类 B_m 和 B_n，如果$|B_m \cap B_n| / |B_m| > T$ 并且$|B_m \cap B_n| / |B_n| > T$，则 B_m 和 B_n 的相似度为 1，否则为 0，T 为事先给定的阈值，类定义为那些相似度为 1 的基本分类的集合。图 2.2 给出了表 2.1 中六个基本分类的合并情况，用节点代表基本分类，相似度为 1 的节点之间用无向边连接，从图 2.2 可以看到，这六个基本分类被连接到一起，所以聚类结果为一个类。如果单词 ate 是停止字，则基本类 b 的 $s(b) = 0$，需要删除节点 b，这时的聚类结果就为三个类，分别是{a}、{d, e}、{c, f}。

2.6 本章小结

本章详细讲述了几种不同类型的聚类算法，CLARANS 算法基于统计学中 K-means 方法，提出采用动态取样的方法提高在处理大型应用时的效率；BIRCH 算法是一种增量式算法，它的主要贡献在于：能够根据内存决定需要存储的数据量，因此，特别适合大容量数据集合；DBSCAN 算法根据数据空间中数据点的分布密度发现聚类；以上三个算法用来处理数字数据，而 ROCK 算法则用来处理分类属性的数据；STC 算法用来聚类文档集合。以上算法都是数据开采领域中具有代表性的聚类算法。

考虑两个基本分类 B_m 和 B_n，如果$|B_m \cap B_n| / |B_m| > T$ 并且$|B_m \cap B_n| / |B_n| > T$，则 B_m 和 B_n 的相似度为 1，否则为 0，T 为事先给定的阈值，类定义为那些相似度为 1 的基本分类的集合。图 2.2 给出了表 2.1 中六个基本分类的合并情况，用节点代表基本分类，相似度为 1 的节点之间用无向边连接，从图 2.2 可以看到，这六个基本分类被连接到一起，所以聚类结果为一个类。如果单词 ate 是停止字，则基本类 b 的 $s(b) = 0$，需要删除节点 b，这时的聚类结果就为三个类，分别是{a}、{d, e}、{c, f}。

2.6 本章小结

本章详细讲述了几种不同类型的聚类算法，CLARANS 算法基于统计学中 K-means 方法，提出采用动态取样的方法提高在处理大型应用时的效率；BIRCH 算法是一种增量式算法，它的主要贡献在于：能够根据内存决定需要存储的数据量，因此，特别适合大容量数据集合；DBSCAN 算法根据数据空间中数据点的分布密度发现聚类；以上三个算法用来处理数字数据，而 ROCK 算法则用来处理分类属性的数据；STC 算法用来聚类文档集合。以上算法都是数据开采领域中具有代表性的聚类算法。

3 聚类分类数据

对于现实世界中存在的大量分类数据，基于空间距离的聚类算法得到的结果都不理想，而近年来，国外许多学者提出的许多针对分类属性数据的聚类算法[94]-[97]需要对数据库进行多遍扫描，对于海量数据集合，算法的效率和扩展性受到限制。因此，本文给出了一个新的分类属性数据的聚类算法 CCDCS（Clustering Categorical Data using Clustering Summary），该算法只需对数据库扫描一遍即可得到理想的聚类结果。

3.1 问题的提出

数据开采中的聚类算法在提高效率方面做了大量研究，这些算法都利用空间距离来衡量两个对象的相似性，空间距离较小的对象分到同一个类，而空间距离较大的对象被分到不同的类中。基于距离尺度的算法对数字属性的数据非常有效，例如对“存款额”进行聚类时，可以说 100.21 比 10.22 更相似于 100.22，但是在现实世界中，除了数字属性的数据外还存在大量分类数据，分类数据是那些具有分类属性的数据，分类属性的值域是一个离散的有限集合，如性别、籍贯。布尔属性可看作是分类属性的特例，属性值要么为“真”（1）要么为“假”（0）。分类属性数据聚类的应用非常广泛，例如：购物模式分类、疾病成因的发现、文档分类、基于内容的图像检索、个性化主页的生成、构造类似于 Yahoo!* 那样的分层信息结构等[99]。对于具有分类属性的数据，基于空间距离的聚类算法得到的结果都不理想，它们在聚类分类数据时，首先需要将分类属性转换成二进制属性，然后用处理数字数据的方法对它们聚类。以超级市场中的交易数据库为例，一条记录就是某个客户一次购买的所有物品，如客户在某次交易中购买了“牛奶”、“面包”和“黄油”，则对应的记录为{牛奶, 面包, 黄油}，其中“牛奶”等物品被称作项目。算法首先找出所有出现在交易库中的项目的集合（也可以根据具体问题确定所需的项目集合），然后将项目集合中的每个项目看作一个属性，对于记录中包含的项目，其对应属性值为 1，相应地，记录不包含的项目，其对应的属性值为 0。再次以交易数据库为例，假设该数据库中包含的项目集合为：{牛奶, 饼干, 黄油, 啤酒, 面包}，则上例中的记录转换后形式为：{1, 0, 1, 0, 1}。例 3.1 将说明距离算法作用于这样的数据集合的局限性。

* WWW 大型门户网站，链接地址为 www.yahoo.com，为互联网用户提供分类检索等综合服务。

3 聚类分类数据

对于现实世界中存在的大量分类数据，基于空间距离的聚类算法得到的结果都不理想，而近年来，国外许多学者提出的许多针对分类属性数据的聚类算法[94]-[97]需要对数据库进行多遍扫描，对于海量数据集合，算法的效率和扩展性受到限制。因此，本文给出了一个新的分类属性数据的聚类算法 CCDCS（Clustering Categorical Data using Clustering Summary），该算法只需对数据库扫描一遍即可得到理想的聚类结果。

3.1 问题的提出

数据开采中的聚类算法在提高效率方面做了大量研究，这些算法都利用空间距离来衡量两个对象的相似性，空间距离较小的对象分到同一个类，而空间距离较大的对象被分到不同的类中。基于距离尺度的算法对数字属性的数据非常有效，例如对“存款额”进行聚类时，可以说 100.21 比 10.22 更相似于 100.22，但是在现实世界中，除了数字属性的数据外还存在大量分类数据，分类数据是那些具有分类属性的数据，分类属性的值域是一个离散的有限集合，如性别、籍贯。布尔属性可看作是分类属性的特例，属性值要么为“真”（1）要么为“假”（0）。分类属性数据聚类的应用非常广泛，例如：购物模式分类、疾病成因的发现、文档分类、基于内容的图像检索、个性化主页的生成、构造类似于 Yahoo!*那样的分层信息结构等[99]。对于具有分类属性的数据，基于空间距离的聚类算法得到的结果都不理想，它们在聚类分类数据时，首先需要将分类属性转换成二进制属性，然后用处理数字数据的方法对它们聚类。以超级市场中的交易数据库为例，一条记录就是某个客户一次购买的所有物品，如客户在某次交易中购买了“牛奶”、“面包”和“黄油”，则对应的记录为{牛奶, 面包, 黄油}，其中“牛奶”等物品被称作项目。算法首先找出所有出现在交易库中的项目的集合（也可以根据具体问题确定所需的项目集合），然后将项目集合中的每个项目看作一个属性，对于记录中包含的项目，其对应属性值为 1，相应地，记录不包含的项目，其对应的属性值为 0。再次以交易数据库为例，假设该数据库中包含的项目集合为：{牛奶, 饼干, 黄油, 啤酒, 面包}，则上例中的记录转换后形式为：{1, 0, 1, 0, 1}。例 3.1 将说明距离算法作用于这样的数据集合的局限性。

* WWW 大型门户网站，链接地址为 www.yahoo.com，为互联网用户提供分类检索等综合服务。

例 3.1 给定的交易集合 D 包含三条交易：T_1 = {a, b, c, d, e}、T_2 = {a, b}和 T_3 = {f}。按字母顺序得到集合 D 的项目集合 I = { a, b, c, d, e , f}，因此，转换后的三条记录分别为：T_1 = {1, 1, 1, 1, 1, 0}、T_2 = {1, 1, 0, 0, 0, 0}和 T_3 = {0, 0, 0, 0, 0, 1}，根据欧氏距离（见公式 1.2）得到 $d_{T_1T_2}=\sqrt{3}$、$d_{T_1T_3}=\sqrt{6}$、$d_{T_2T_3}=\sqrt{3}$，按照基于距离的算法，距离越近的对象越相似，因此算法很有可能将交易 T_2 和 T_3 分配到同一个类中；而通过直接观察可发现，T_2 和 T_3 之间却没有任何共同的项目，因而使用基于距离的算法，将可能把属于不同分类的对象聚集到同一个类中。

近年来，许多研究人员提出了针对分类属性数据的聚类算法[94]-[97]，然而这些算法将研究重点集中在找出适合分类数据的标准函数和对象相似性判别方法，在效率方面还不能适应大型数据集合的需求。如 ROCK 算法[94]的时间复杂度为 $O(n^2+nm_mm_a+n^2\log n)$，n 为读入的对象数量，m_a 和 m_m 分别为一个对象的平均和最大邻居数量，此外，该算法还需要对数据库进行两次磁盘扫描，对于大型数据集合，I/O 所耗费的时间是不能容忍的，所以 ROCK 选择在一个随机选取的样本数据集合上聚类，这样得到的聚类结果又不太理想；Ke Wang 等人提出的利用频繁项目聚类算法[99]也需要对数据库扫描两次；k-prototype 算法[98]的时间复杂度为 $O((t+1)kn)$，其中 n 为数据集合中的对象数量，k 为最终需要的分类数量，t 为算法的迭代次数，算法的中止条件为在一次迭代中没有任何对象的移动能够导致标准函数的值变得更小，所以在大型数据集合中 t 值是不确定的。

为解决上述问题，本文提出一种新的判别分类属性数据相似度的标准，并提出了聚类汇总 CS（Clustering Summary）的概念，CS 是一个类中所有对象的压缩汇总，CS 有助于压缩原来的数据集合从而减少聚类时间。为了有效地进行聚类，数据结构采用基于 CS 的聚类树 CT（Clustering Tree）。随后，本文给出了一种新的聚类分类属性数据的算法 CCDCS，只需对数据库扫描一遍即可得到理想的聚类结果，因此该算法适合海量数据集合。

3.2 分类数据

本章论及的分类数据指用来描述具有分类属性的对象的那些数据[118]。下面给出分类属性、分类值域和分类对象的定义。

给定空间 Ω，令 $A_1, A_2, \ldots, A_m$ 为描述 Ω 的 m 个属性，$\mathrm{DOM}(A_1), \mathrm{DOM}(A_2), \ldots, \mathrm{DOM}(A_m)$分别为这些属性的值域。如果值域 $\mathrm{DOM}(A_j)$是一个离散的有限集合，则被称作是分类值域，A_j 被称作分类属性；如果 $A_1, A_2, \ldots, A_m$ 都是分类属性，则 Ω 被称作分类空间。本文定义的分类值域中的值都是原子值，对于存在于现实世界

例 3.1 给定的交易集合 D 包含三条交易：T_1 = {a, b, c, d, e}、T_2 = {a, b}和 T_3 = {f}。按字母顺序得到集合 D 的项目集合 I = { a, b, c, d, e , f}，因此，转换后的三条记录分别为：T_1 = {1, 1, 1, 1, 1, 0}、T_2 = {1, 1, 0, 0, 0, 0}和 T_3 = {0, 0, 0, 0, 0, 1}，根据欧氏距离（见公式 1.2）得到 $d_{T_1T_2}=\sqrt{3}$、$d_{T_1T_3}=\sqrt{6}$、$d_{T_2T_3}=\sqrt{3}$，按照基于距离的算法，距离越近的对象越相似，因此算法很有可能将交易 T_2 和 T_3 分配到同一个类中；而通过直接观察可发现，T_2 和 T_3 之间却没有任何共同的项目，因而使用基于距离的算法，将可能把属于不同分类的对象聚集到同一个类中。

近年来，许多研究人员提出了针对分类属性数据的聚类算法[94]-[97]，然而这些算法将研究重点集中在找出适合分类数据的标准函数和对象相似性判别方法，在效率方面还不能适应大型数据集合的需求。如 ROCK 算法[94]的时间复杂度为 $O(n^2+nm_mm_a+n^2\log n)$，n 为读入的对象数量，m_a 和 m_m 分别为一个对象的平均和最大邻居数量，此外，该算法还需要对数据库进行两次磁盘扫描，对于大型数据集合，I/O 所耗费的时间是不能容忍的，所以 ROCK 选择在一个随机选取的样本数据集合上聚类，这样得到的聚类结果又不太理想；Ke Wang 等人提出的利用频繁项目聚类算法[99]也需要对数据库扫描两次；k-prototype 算法[98]的时间复杂度为 $O((t+1)kn)$，其中 n 为数据集合中的对象数量，k 为最终需要的分类数量，t 为算法的迭代次数，算法的中止条件为在一次迭代中没有任何对象的移动能够导致标准函数的值变得更小，所以在大型数据集合中 t 值是不确定的。

为解决上述问题，本文提出一种新的判别分类属性数据相似度的标准，并提出了聚类汇总 CS（Clustering Summary）的概念，CS 是一个类中所有对象的压缩汇总，CS 有助于压缩原来的数据集合从而减少聚类时间。为了有效地进行聚类，数据结构采用基于 CS 的聚类树 CT（Clustering Tree）。随后，本文给出了一种新的聚类分类属性数据的算法 CCDCS，只需对数据库扫描一遍即可得到理想的聚类结果，因此该算法适合海量数据集合。

3.2 分类数据

本章论及的分类数据指用来描述具有分类属性的对象的那些数据[118]。下面给出分类属性、分类值域和分类对象的定义。

给定空间 Ω，令 $A_1, A_2, \ldots, A_m$ 为描述 Ω 的 m 个属性，$\mathrm{DOM}(A_1), \mathrm{DOM}(A_2), \ldots, \mathrm{DOM}(A_m)$分别为这些属性的值域。如果值域 $\mathrm{DOM}(A_j)$是一个离散的有限集合，则被称作是分类值域，A_j 被称作分类属性；如果 $A_1, A_2, \ldots, A_m$ 都是分类属性，则 Ω 被称作分类空间。本文定义的分类值域中的值都是原子值，对于存在于现实世界

数据库中的组合值以及值域中的概念包含关系（如“交通工具”和“汽车”同为一个值域中的两个值，但“汽车”同时也是“交通工具”的一种），可以在聚类开采之前通过数据清洗将它们简化。

给定分类对象 $X\in\Omega$，$X=[A_1=x_1]\wedge[A_2=x_2]\wedge\ldots\wedge[A_m=x_m]$，其中 $x_j\in DOM(A_j)$，$1\leqslant j\leqslant m$，为简便起见，本文将对象 $X\in\Omega$ 用向量 $(x_1, x_2, \ldots, x_m)$ 表达，如果属性 A_j 的值不存在，则 $A_j=\varepsilon$。

也可以用集合形式表达分类对象，令 $D=\{X_1, X_2, \ldots, X_n\}$ 为 n 个分类对象的集合，则 $X_i=\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$，如果属性 A_j 的值不存在，则集合中不出现 $x_{i,j}$，容易得到 $|X_i|\leqslant m$。如果存在 $x_{i,j}=x_{k,j}$，$1\leqslant j\leqslant m$，则 $X_i=X_k$。在现实数据库中可能存在不同属性的值域有重叠的部分，即 $x_{i,u}=x_{i,v}$，$1\leqslant u\leqslant v\leqslant m$，可以在数据清洗过程中在属性值前加上属性名称作为前缀以区分不同属性的相同值。

3.3 CCDCS

给定一个分类空间 Ω，算法的目的是把 Ω 中的对象划分成若干个类，使得属于同一类的对象之间的相似性尽可能大，而属于不同类的对象间的相似性尽可能小，同时类与类之间的相似性尽可能小。在描述算法之前，先给出聚类汇总 CS、聚类树 CT 的概念以及对象之间、类之间相似性的定义。为方便表述，除非特别说明，分类对象用集合方式表达。

3.3.1 聚类汇总 CS

CCDCS 利用聚类汇总来压缩原始数据，从而达到提高算法效率的目的。一个类 C 由如下三元组 (n, I, S) 来表示。其中 n 为类 C 中的对象数量，$I=\{i_1, i_2, \cdots, i_u\}$ 是 C 内所有属性值的集合，$S=\{s_1, s_2, \cdots, s_u\}$，其中 s_j 为 i_j 在类 C 中的数量，$i_j\in I$，$1\leqslant j\leqslant u$。集合 S 按升序排列，即 $s_1\leqslant s_2\leqslant\cdots\leqslant s_u$，这同时也暗示集合 I 的元素按其在 C 中的数量按升序排列。三元组 (n, I, S) 被称作类 C 的聚类汇总 CS，CS 的三个成员分别记作 $CS.n$、$CS.I$ 和 $CS.S$；对于 $CS.I$ 的任一元素 $i_j\in CS.I$，则记作 $CS.I.i_j$，对于 $s_j\in CS.S$，则记作 $CS.S.s_j$，其中 $1\leqslant j\leqslant u$。

给定一个包含 n 个对象 $\{X_1, X_2, \ldots, X_n\}$ 的类 C，其中 $X_i=\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$，$i=1, 2, \ldots, n$，则类 C 的聚类汇总 $CS(n, I, S)$ 各成员可通过下列公式得到。

$$CS.n=n$$

$$CS.I=X_1\cup X_2\cup\ldots\cup X_n$$

数据库中的组合值以及值域中的概念包含关系（如“交通工具”和“汽车”同为一个值域中的两个值，但“汽车”同时也是“交通工具”的一种），可以在聚类开采之前通过数据清洗将它们简化。

给定分类对象 $X\in\Omega$，$X=[A_1=x_1]\wedge[A_2=x_2]\wedge\ldots\wedge[A_m=x_m]$，其中 $x_j\in DOM(A_j)$，$1\leqslant j\leqslant m$，为简便起见，本文将对象 $X\in\Omega$ 用向量 $(x_1, x_2, \ldots, x_m)$ 表达，如果属性 A_j 的值不存在，则 $A_j=\varepsilon$。

也可以用集合形式表达分类对象，令 $D=\{X_1, X_2, \ldots, X_n\}$ 为 n 个分类对象的集合，则 $X_i=\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$，如果属性 A_j 的值不存在，则集合中不出现 $x_{i,j}$，容易得到 $|X_i|\leqslant m$。如果存在 $x_{i,j}=x_{k,j}$，$1\leqslant j\leqslant m$，则 $X_i=X_k$。在现实数据库中可能存在不同属性的值域有重叠的部分，即 $x_{i,u}=x_{i,v}$，$1\leqslant u\leqslant v\leqslant m$，可以在数据清洗过程中在属性值前加上属性名称作为前缀以区分不同属性的相同值。

3.3 CCDCS

给定一个分类空间 Ω，算法的目的是把 Ω 中的对象划分成若干个类，使得属于同一类的对象之间的相似性尽可能大，而属于不同类的对象间的相似性尽可能小，同时类与类之间的相似性尽可能小。在描述算法之前，先给出聚类汇总 CS、聚类树 CT 的概念以及对象之间、类之间相似性的定义。为方便表述，除非特别说明，分类对象用集合方式表达。

3.3.1 聚类汇总 CS

CCDCS 利用聚类汇总来压缩原始数据，从而达到提高算法效率的目的。一个类 C 由如下三元组 (n, I, S) 来表示。其中 n 为类 C 中的对象数量，$I=\{i_1, i_2, \cdots, i_u\}$ 是 C 内所有属性值的集合，$S=\{s_1, s_2, \cdots, s_u\}$，其中 s_j 为 i_j 在类 C 中的数量，$i_j\in I$，$1\leqslant j\leqslant u$。集合 S 按升序排列，即 $s_1\leqslant s_2\leqslant\cdots\leqslant s_u$，这同时也暗示集合 I 的元素按其在 C 中的数量按升序排列。三元组 (n, I, S) 被称作类 C 的聚类汇总 CS，CS 的三个成员分别记作 $CS.n$、$CS.I$ 和 $CS.S$；对于 $CS.I$ 的任一元素 $i_j\in CS.I$，则记作 $CS.I.i_j$，对于 $s_j\in CS.S$，则记作 $CS.S.s_j$，其中 $1\leqslant j\leqslant u$。

给定一个包含 n 个对象 $\{X_1, X_2, \ldots, X_n\}$ 的类 C，其中 $X_i=\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$，$i=1, 2, \ldots, n$，则类 C 的聚类汇总 $CS(n, I, S)$ 各成员可通过下列公式得到。

$$CS.n=n$$

$$CS.I=X_1\cup X_2\cup\ldots\cup X_n$$

$$CS.S.s_j = \sum_{k=1}^{n} |\{i_j\} \cap X_k|,\ i_j \in CS.I,\ j = 1, 2, \ldots, u$$

例 3.2　$CS.S$ 按照 $CS.S.s_j$ 升序排列，$CS.I$ 根据 $CS.S$ 中的元素排列与之对应的属性值。假设类 C 有 3 个对象：X_1 = {a, b, c}，X_2 = {b, c, d, e}，X_3 = {a, d, f}；按照上述公式，得到 $CS.n = 3$、$CS.I$ = {e, f , a, b, c, d}、$CS.S$ = {1, 1, 2, 2, 2, 2}；得到类 C 的聚类汇总形式如下 CS = (3, {e, f , a, b, c, d}, {1, 1, 2, 2, 2, 2})。

给定一个属性值 i 和类 C 的聚类汇总 CS（n, I, S），定义属性值 i 在类 C 中的数量如下：

$$CS.\text{num}(i) = \begin{cases} 0 & i \notin CS.I \\ s_j & i \in CS.I \text{且} i = i_j, \text{其中} 1 \le j \le u \end{cases} \tag{3.1}$$

沿用例 3.2 的类 C，属性值 a 在类 C 中的数量为 CS.num(a) = 2，而对于属性值 g，CS.num(g) = 0。类似地，定义属性值 i 在对象 $X = \{x_1, x_2, \ldots, x_m\}$ 中的数量如下：

$$X.\text{num}(i) = \begin{cases} 0 & i \notin X \\ 1 & i \in X \end{cases} \tag{3.2}$$

例如，给定属性值 a、d 和 X_1 = {a, b, c}，则 X_1.num(a) = 1，X_1.num(d) = 0。对于两个聚类汇总 CS_1（n_1, I_1, S_1）和 CS_2（n_2, I_2, S_2），它们的加法定义为：

$$CS_3 = CS_1 \oplus CS_2 \tag{3.3}$$

CS_3 具有形式（n_3, I_3, S_3），其中 $n_3 = n_1 + n_2$；$I_3 = I_1 \cup I_2$；$CS_3.S_3.s_j = CS_1.\text{num}(i_j) + CS_2.\text{num}(i_j)$，其中 $i_j \in CS_3.I_3$，$j = \{1, 2, \ldots, |I_3|\}$，$CS_3.S_3$ 需要按照 $CS_3.S_3.s_j$ 的值从小到大排序，$CS_3.I_3$ 也要根据 $CS_3.S_3$ 重新排列其属性值的顺序。令 CS_1 = (3, {e, f , a, b, c, d}, {1, 1, 2, 2, 2, 2})，CS_2 = (2, {a, e, f}, {1, 1, 2})，按照上述方法合并 CS_1 和 CS_2 后得到 CS_3 = (5, {e, f, b, c, d, a}, {2, 2, 2, 2, 2, 3})。

当类 C 中仅包含一个对象 X 时，则 $CS.n = 1, CS.I = X, CS.S.s_j = 1, j = \{1, 2, \cdots, |X|\}$。如果公式（3.3）中的 CS_1 或 CS_2 所代表的类仅包含一个对象，则公式（3.3）表示如何将对象合并到类中。

从聚类汇总的概念可以看到 CS 具有如下优点：① 存储的信息比原来的类要少得多，因而具有数据压缩功能。② 存储了计算相似度（见下一小节）所需的足够的信息，因而不会因为数据压缩而影响到聚类结果的正确性。

3.3.2 相似度定义

有两种方法计算两个类之间的相似性：① 计算两个类的质心（即最能代表类的对象）之间的相似度作为两个类的相似度。② 计算两个类中所有对象对的相似度，取其平均值代表两个类的相似度。第一种方法计算量小，然而不适合分类属性数据的计算，其误差太大。第二种方法准确性高，但计算量太大，对分别包含 m 和 n 个对象的两个类，必须要计算 $m*n$ 次相似度，才能得到它们的相似度，对海量数据集合来讲，必将陷入组合爆炸的困境中。为了克服这两种计算方法的不足，本文提出了聚类汇总的概念，以聚类汇总的相似度作为类的相似度，由于聚类汇总包含了一个类中足够的信息，因而弥补了第一种计算方法误差太大的缺点，同时由于不必考察类中的每个对象，因而算法的时间复杂度要远远小于第二种计算方法，此外由于聚类汇总压缩了原来的数据集合，因而算法的空间复杂度也将大大减小。从实验结果可以看出 CCDCS 算法效率要明显高于同类算法，同时聚类结果的准确度也不比其它算法逊色。

集合论中的 Jaccard Coefficient 公式常被用来衡量两个集合的相似度，给定两个分类对象 X_1 和 X_2，它们之间的相似度为$|X_1 \cap X_2| \ / \ |X_1 \cup X_2|$。然而由于 Jaccard Coefficient 公式仅适合用来计算两个点（对象）之间的相似度，无法直接计算类之间的相似度，当然也无法计算聚类汇总之间的相似度，因此本文给出了基于 Jaccard Coefficient 的聚类汇总相似度的公式。

3.3.2.1 对象相似度

首先考虑分类对象 X_1 和 X_2 之间的相似度，很明显，两个对象相同的属性值越多，则两者越相似，相反则两者就越不相似。若两个对象的所有属性值完全相同，即 $X_1 = X_2$，则两者最相似，若两个对象没有一个属性值是相同的，则两者最不相似。本文不仅考虑分类对象之间相同的属性值，同时也考虑它们之间的差别。在 X_1 中且同时也被 X_2 包含的属性值数量为$|X_1 \cap X_2|$，在 X_1 中但不在 X_2 中的项目数量为$|X_1| - |X_1 \cap X_2|$；同理，在 X_2 中同时也被 X_1 包含的属性值数量为$|X_2 \cap X_1|$，在 X_2 中但不在 X_1 中的属性值数量为$|X_2| - |X_1 \cap X_2|$。定义分类对象 X_1 和 X_2 之间的相似度为：

$$\begin{aligned} sim(X_1, X_2) &= [|X_1 \cap X_2| + |X_2 \cap X_1| - (|X_1| - |X_1 \cap X_2|) - (|X_2| - |X_1 \cap X_2|)] \\ &\quad / (|X_1| + |X_2|) \qquad (3.4) \\ &= (4|X_1 \cap X_2| - |X_1| - |X_2|) / (|X_1| + |X_2|) \end{aligned}$$

公式（3.4）的分子为两个对象所包含的共有属性值的数量减去它们包含的不相同的属性值数量，分母部分保证了相似度的值域被限制在[-1，1]之间，把公式（3.4）进行[0, 1]规格化后为：

$$\mathrm{sim}(X_1, X_2) = 2|X_1 \cap X_2| / (|X_1|+|X_2|) \tag{3.5}$$

从公式（3.5）可以得出两个对象的相似度与它们共同的属性值数量有关，它们共有的属性值越多则相似性越大，当两个对象所包含的属性值完全相同时，相似度最大，为 1，当两个对象没有任何共同的属性值时，它们的相似度最小，为 0。

例 3.3　表 3.1 记录了两种蘑菇的外观特征及生长环境，记蘑菇 1 为 X_1 = {暗红, 树林, 少, 小}，蘑菇 2 为 X_2 = {黄褐, 树林, 小}，则根据公式（3.5）得到分类对象 X_1 和 X_2 之间的相似度 $\mathrm{sim}(X_1, X_2) = 2\times2 / (4+3) = 0.57$。

表 3.1 蘑菇数据库

	颜色	生长环境	皱褶	体型	花纹
蘑菇 1	暗红	树林	少	小	
蘑菇 2	黄褐	树林		小	

3.3.2.2　类相似度

给定类 C 的聚类汇总 $CS(n, I, S)$，$CS.I = \{i_1, i_2, \ldots, i_u\}$，$CS.S = \{s_1, s_2, \ldots, s_u\}$，假设 $CS.S$ 中不同元素的个数为 N，$1 \leqslant N \leqslant u$，把 CS 转换成一个具有 N 个元素的集合 $P = \{p_1, p_2, \ldots, p_N\}$，$p_j$ 为一个二元组 (m_j, I_j)，$1 \leqslant j \leqslant N$，集合 m_j 为一个数字，$I_j = \{I_{j1}, I_{j2}, \ldots, I_{jx}\}$。令 $CS.S^{dist}$ 为 $CS.S$ 中不同元素的集合，如图 3.1 给出的 $CS.S$ = {4, 5, 5, 7}，它的 $CS.S^{dist}$ = {4, 5, 7}，将 $CS.S^{dist}$ 中的元素记作 $CS.S^{dist}.s_j$，$1 \leqslant j \leqslant N$，令集合 $\Gamma_1 = CS.I$，$S'_1 = CS.S$，通过递推公式得到 Γ_j 和 S'_j，m_j 和 I_j 定义如下，图 3.1 举例说明了如何将 CS 转换为集合 P。

$$m_j = CS.S^{dist}.s_j - CS.S^{dist}.s_{j-1} \quad (CS.S^{dist}.s_{j-1} = 0)$$

$$I_j = \Gamma_j$$

$$S'_{j+1} = \{\, \alpha \mid \alpha = \beta - m_j, \beta \in S'_j \text{ 且 } \beta \neq 0\}$$

$$\Gamma_{j+1} = \{\Gamma_j\} - \{i_i \mid i_i \in \Gamma_j, \text{ 且 } s_i = 0, s_i \in S'_{j+1}\}$$

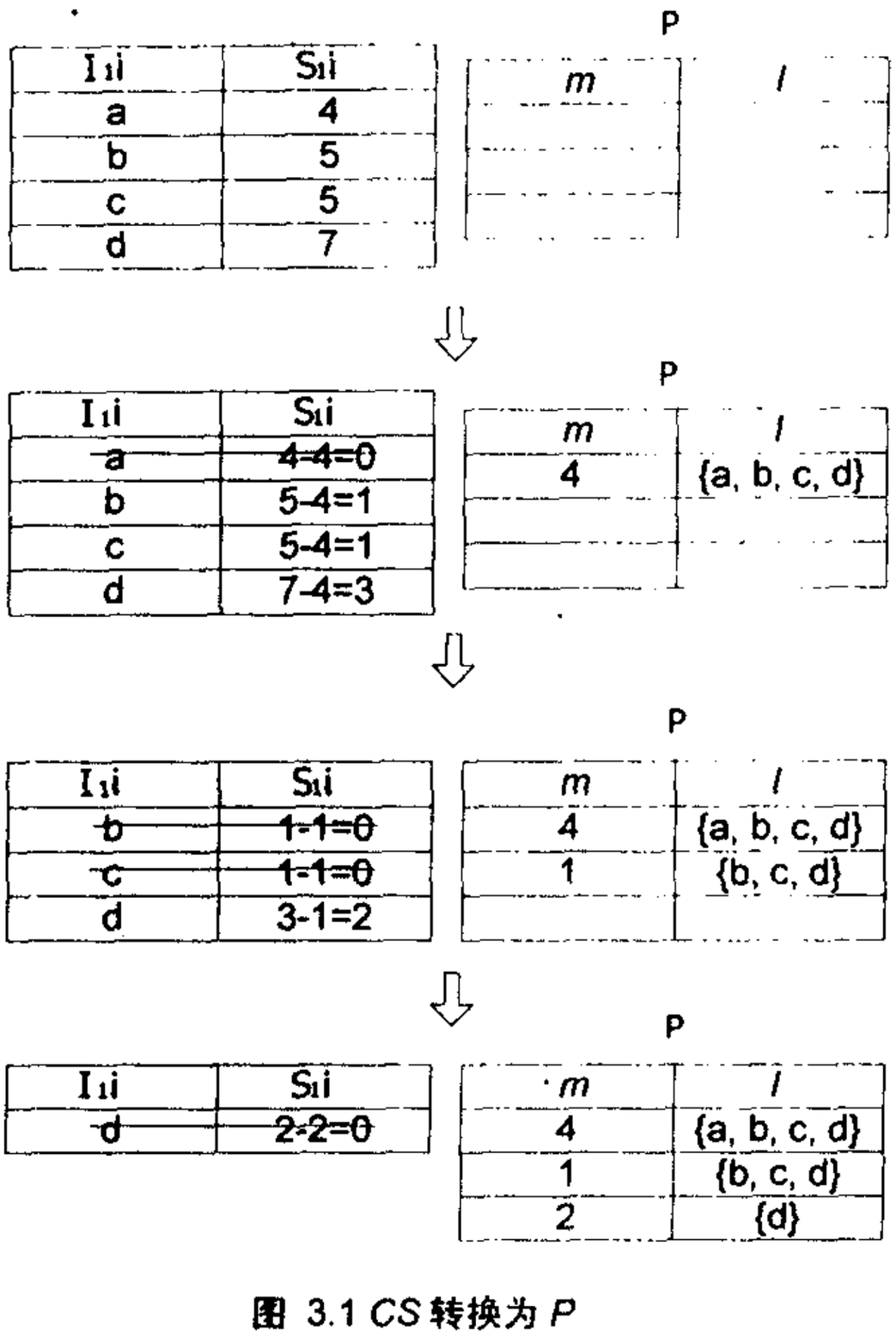

图 3.1 *CS* 转换为 *P*

给定两个类 C_1 和 C_2，它们的聚类汇总分别是 CS_1 和 CS_2，它们的相似度定义如下：

$$\mathrm{sim}(C_1, C_2) = \frac{\sum_{j=1}^{|P_1|}\sum_{k=1}^{|P_2|}\frac{(P_1.p_j.m_j + P_2.p_k.m_k)\,|\,P_1.p_j.I_j \cap P_2.p_k.I_k\,|}{P_1.p_j.m_j\,|\,P_1.p_j.I_j\,| + P_2.p_k.m_k\,|\,P_2.p_k.I_k\,|}}{|\,P_1\,||\,P_2\,|} \tag{3.6}$$

由公式(3.6)引申出类 C 和对象 X 的相似度为：

$$\mathrm{sim}(C, X) = \frac{1}{|\,P\,|}\sum_{j=1}^{|P|}\frac{(P.p_j.m_j + 1)\,|\,P.p_j.I_j \cap X\,|}{P.p_j.m_j\,|\,P.p_j.I_j\,| + |\,X\,|} \tag{3.7}$$

给定两个类 C_1 和 C_2 以及一个对象 X，如果 $\mathrm{sim}(X, C_1) \geqslant \mathrm{sim}(X, C_2)$，则认为 C_1 与 X 更相似。下面给出一个完整的例子：

例 3.4　让 CS_1 = {7, {a, b, c, d}, {4, 5, 5, 7}}，CS_2 = {5, {a, c, d, e}, {2, 4, 5,

5}}，X = {b, c, d}，得到 P_1 = {(4, {a, b, c, d}), (1, {b, c, d}), (2, {d})}，P_2 = {(2, {a, c, d, e}), (2, {c, d, e}), (1, {d, e})}，根据公式(3.6)得到类 C_1 和 C_2 的相似度为 sim(C_1, C_2) = 0.54，sim(C_1, X) = 0.80，sim(C_2, X) = 0.54。

3.3.2.3 聚类树 CT

为了有效地聚类，算法采用聚类树 CT（Clustering Tree，类似 B⁺树*）来进行数据组织。一棵 m 阶的 B⁺树，或为空树，或满足下列特性：

1. 树中每个节点至多有 m 棵子树；
2. 若根节点不是叶子节点，则至少有两棵子树；
3. 除根节点之外的所有非叶子节点至少有$\lceil m/2 \rceil$棵子树；
4. 有 n 棵子树的节点中含有 n 个关键字；
5. 所有叶子节点中包含了全部关键字的信息，及指向含这些关键字记录的指针，且叶子节点本身依关键字大小自小而大顺序链接；
6. 所有的非终端节点可以看成是索引部分，节点中仅含有其子树中的最大（或最小）关键字，表示如下（$K_1, A_1, K_2, A_2, \ldots, K_n, A_n$），其中：$K_i (i = 1, 2, \ldots, n)$为关键字，满足 $K_i < K_{i+1}$ $(i = 1; 2, \ldots, n)$；$A_i (i = 1, 2, \ldots, n)$为指向子树根节点的指针，指针 A_{i-1} 所指子树中所有节点的关键字均小于 K_i。

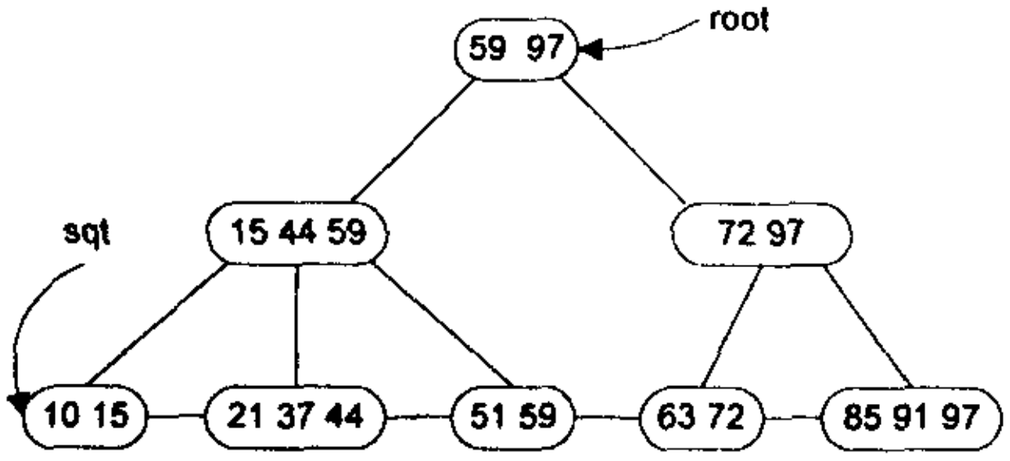

图 3.2 三阶的 B⁺树

图 3.2 给出了一棵三阶的 B⁺树。聚类树 CT 是高度平衡树，与 B⁺树类似，聚类树 CT 每个非叶子节点最多包含 b 个子节点，非叶子节点具有如下形式：(CS, $pParent$, $pChild_i$)，其中 CS 是该节点的所有子节点形成的聚类汇总，$pParent$ 指向该节点的父节点，$pChild_i$ 指向其第 i 个子节点，$i = \{1, 2, \cdots, k\}$, $1 \leqslant k \leqslant b$。CT 树

---

* B⁺树是数据库和文件系统中常用的一种数据存储结构。

的每个叶子节点是一个聚类汇总 *CS*，该聚类汇总为叶节点包含的所有交易所形成的聚类汇总，与非叶子节点相同，叶子节点也有一个指向父节点的指针 *pParent*，由于不存在子节点，所以没有指向子节点的指针 $pChild_i$，但是叶子节点包含一个指向其右节点的指针 *pNext*，这个指针将 CT 中所有叶子节点链接起来。构造聚类树 CT 需要用户事先给定一个阈值——最小相似度 *minsim*（minimum similarity），它用来判断对象 X 能否合并到叶节点 C 中，只有当 $\mathrm{sim}(C, X) \geq minsim$ 时，才允许对象 X 加入到叶节点 C 中。另外，所有的叶节点都属于聚类树的同一层，即最底层。图 3.3 给出了一棵 $b = 3$ 的聚类树。

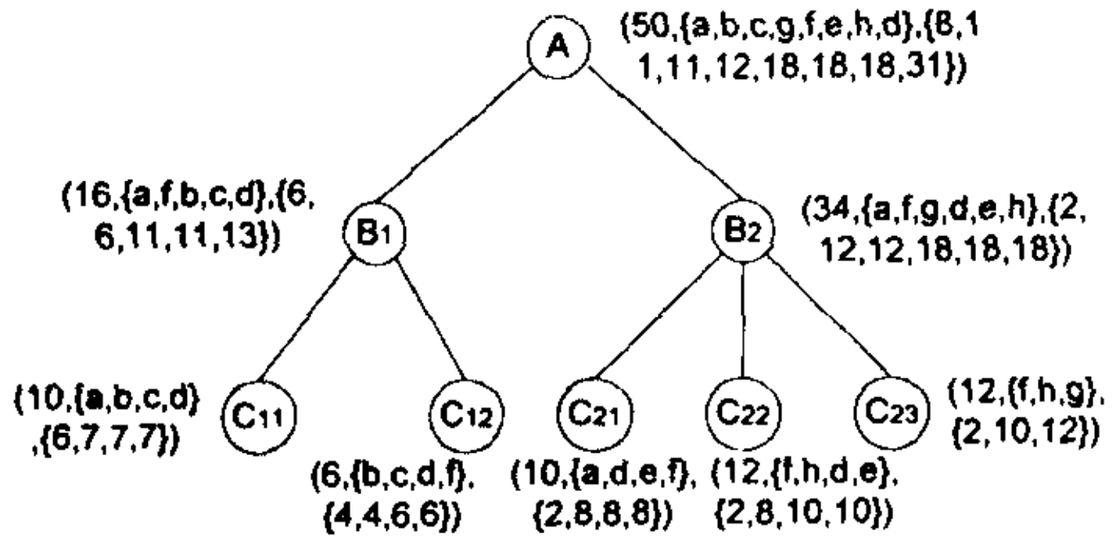

图 3.3 三阶聚类树

给定一个节点 C，其聚类汇总为 CS，C 的 b 个子节点为 $C_1, C_2, \ldots, C_b$，假定这些子节点的聚类汇总分别为 $CS_1, CS_2, \ldots, CS_b$，则 CS 和子节点的聚类汇总之间的关系如下：

$$CS.n = \sum_{i=1}^{b} CS_i.n \tag{3.8}$$

$$CS.I = \bigcup_{i=1}^{b} CS_i.I \tag{3.9}$$

$$CS.S = \{s \mid s = \sum_{i=1}^{b} CS_i.num(x_j)，\text{其中 } x_j \in CS.I, j = 1, 2, \cdots, |CS.I|\} \tag{3.10}$$

聚类树从空节点开始动态生成，每次从数据库中读入一条新的对象 X，就以聚类树的根节点为当前节点，按照公式（3.7）计算 X 与当前节点的每个子节点的相似度，选择与 X 最相似的子节点作为当前节点，不断重复上述过程直到到达叶子节点。在此过程中，X 每到达一个非叶子节点 C，都将修改该非叶子节点的 CS，方法如下：

$$CS.n = CS.n + 1 \tag{3.11}$$

$$CS.I = CS.I \cup X \tag{3.12}$$

$$CS.S = \{s \mid s = CS.\text{num}(x_j) + X.\text{num}(x_j),\ 其中\ x_j \in CS.I \cup X,\ j = 1, 2, \ldots, |CS.I \cup X|\} \tag{3.13}$$

例如：给定一个对象 X = {a, b, c}和一个非叶子节点的聚类汇总 CS = (4, {a, b, d, e}, {2, 2, 4, 4})，修改后 CS 变为（5, {c, d, e, a, b}, {1, 2, 2, 5, 5}）。

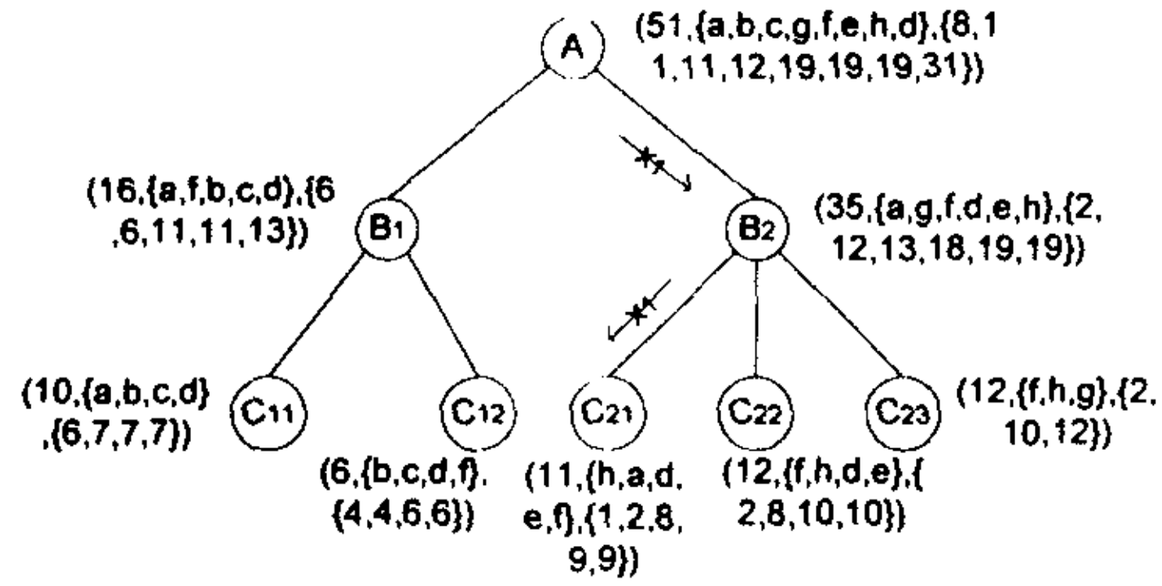

图 3.4 聚类树的对象插入

当 X 到达一个叶子节点 C，如果 $\text{sim}(C, X) \geqslant minsim$，则合并 X 到 C 中，合并的方法也是修改聚类汇总，其方法与修改非叶子节点的聚类汇总相同。如果 X 与 C 的相似度小于最小相似度，则要在当前叶子节点 C 的父节点 C_p 下为对象 X 创建一个新的叶节点 C_{new}。创建后，如果 C_p 的子节点的数量小于 b，则操作结束，否则 C_p 当前的子节点的数量为 b，则需要拆分 C_p，拆分过程如下：计算 C_p 所有子节点对的相似度，选择最不相似的两个子节点 C_i 和 C_j 作为凝聚点，然后分别计算剩余的所有子节点 C_k（其中 $k = 1, 2, \ldots, b\text{-}2$）与它们的相似度，如果 $\text{sim}(C_i, C_k) \geqslant \text{sim}(C_j, C_k)$，则把 C_k 并入到 C_i 中，否则把 C_k 并入到 C_j 中，最后得到的两个节点就为 C_p 拆分后的结果，如果 C_p 父节点的当前子节点的数量小于 b，则操作结

束，否则再拆分 C_p 的父节点，直到没有节点可以拆分，则对象 X 插入到聚类树 CT 的过程结束。图 3.4 演示了在图 3.3 所示的聚类树中依次插入对象 x_1 = {e, f, h} 和 x_2 = {c, f, h} 的结果，令 b = 3、*minsim* = 40%，x_1 首先到达根节点 A，修改根节点 A 的 CS，然后分别计算 x_1 和 B_1、B_2 的相似度，得到 $\text{sim}(x_1, B_1)$ = 7%、$\text{sim}(x_1, B_2)$ = 38.3%，所以 x_1 的下一个目标节点是 B_2，修改 B_2 的 CS，然后计算 $\text{sim}(x_1, C_{21})$ = 60%、$\text{sim}(x_1, C_{22})$ = 43.6%、$\text{sim}(x_1, C_{23})$ = 38%，x_1 选择 C_{21} 作为最终到达的叶子节点，由于 $\text{sim}(x_1, C_{21}) > minsim$，$x_1$ 加入到 C_{21} 中；与 x_1 类似，对象 x_2 选择 B_2 作为第二个到达的节点，它与 C_{21}、C_{22} 和 C_{23} 的相似度分别为 38%、29%和 38%，没有一个叶子节点与 x_2 的相似度大于 40%，为 x_2 创建新节点作为 B_2 的子节点，而 B_2 已经有了三个节点，于是分裂 B_2 得到如图 3.4 所示的结果。

3.3.3 聚类算法

给定一个具有 M 个对象的数据集合 $D = \{X_1, X_2, \cdots, X_M\}$，聚类算法 CCDCS 基本框架描述如下：

```
CCDCS（D, minsim, b）
1.  i = 1，CT = {};
2.  while i <= M do {
3.      read Xi from D;
4.      BuildCT(ref CT, Xi, minsim, b);
5.      i = i+1;
6.  }
7.  over = false;
8.  while not over do {
9.      over = true;
10.     CS = CT 树中第一个叶子节点;
11.     while CS != NULL do {
12.         CSt = CS->next;
13.         CSj = MostSimilar(CS, CT);  //找到 CT 中与 CS 最相似的叶子节
    点
14.         if CS != CSj then {
15.             插入 CS 到 CT 中的新位置;
16.             删除原来的 CS;
17.             over = false;    }
18.         CS = CSt;
19.     }
20. }
```

CCDCS 算法分为两个阶段，第一个阶段（1－6 行）为构造聚类树 CT，每次从数据库中读取一条记录（对象），然后利用函数 BuildCT()动态地将它插入到 CT 中，函数 BuildCT 利用两个参数：CT 的阶数 b 和相似度阈值 *minsim* 来控制树的大小，b 的取值一般在 2~10 之间，*minsim* 的取值非常重要，不恰当的取值将会导致聚类的质量太差或者内存耗尽。函数 BuildCT()如下：

BuildCT(ref CT, X_i, *minsim*, b)
1. *CurrentNode* = CT.root;
2. while *CurrentNode* is not LeafNode do {
3. 根据公式(3.11)—(3.13)修改 *CurrentNode* 的聚类汇总 *CS*;
4. 从 *CurrentNode* 的子节点中找到 *ChildNode* 使得 sim(X_i,*ChildNode*)最大;
5. *similarity* = sim(X_i, *ChildNode*);
6. *CurrentNode* = *ChildNode*;
7. }
8. if *similarity*≥*minsim* then
9. 将 X_i 合并到 *CurrentNode* 中，即修改 *CurrentNode* 的 *CS*;
10. else {
11. *PNode* = *CurrentNode.pParent*，*Children* = *PNode* 的子节点数;
12. *NewNode* = CreateNode(X_i)，*NewNode.pParent*->*Pnode*;
13. while *Children* = b do {
14. *PNode* 拆分为两个节点 $PNode_1$, $PNode_2$;
15. *PNode* = *PNode*1.*pParent*，*Children* = *PNode* 的子节点数;
16. if *PNode* = Null then *CS.root* = Merge($PNode_1$, $PNode_2$)合并节点;
17. }
18. }

计算对象 X_i 和节点 *CS* 的相似度的算法 sim(X_i, *Node*)如下：

sim(X_i, *Node*)
1. *CS* = *Node* 的聚类汇总，*P* = {}，*CS.Sdist* = Distinct(*CS.S*);
2. *I'* = *CS.I*，*S'* = *CS.S*;
3. for i = 1 to |*CS.Sdist*| {
4. m_i = *CS.Sdist*(i) - *CS.Sdist*(i-1);
5. I_j = *I'*;
6. *S'*中的每个元素减去 m_i;
7. 记录 *S'*中值为 0 的元素位置到 *Pos*[]，删除 *S'*中值为 0 的元素;
8. 根据 *Pos*[]记录的位置删除 *I'*中对应的元素;
9. 清空 *Pos*[];
10. }

一旦聚类树 CT 生成，剩余的聚类操作将变得非常快，原因如下：CT 的叶子节点保存的是压缩后的数据集合的汇总，而不是单个的对象，因而其数据量比原始数据集合大大减少，除了非常巨大的海量数据集合外，算法将尽量在内存中处理聚类过程，不再进行耗时的 I/O 操作，即使对海量数据集合来讲，如果必须需要进行 I/O 操作，其读取量也比原来的数据量大大减小。

算法 7－20 行给出了聚类结果的获取，即算法的第二阶段。CT 树的每个叶子节点有指针 *pNext* 用来指向下一个叶子节点，该指针把所有的叶子节点链接起来。聚类结果的获取过程如下：从第一个叶节点开始，顺序读取每个叶节点，每读到一个叶节点 CS_i，则从 CT 的根节点开始，选择与之最相似的子节点作为下一个到达的节点，直到到达叶子节点 CS_j，函数 MostSimilar(*CS*, CT)完成这一过程，如果 CS_i 和 CS_j 相同，则表示 CS_i 的聚类结果正确，否则表明 CS_i 的聚类结果存在偏差，需要把 CS_i 插入到新找到的位置，并把原来的 CS_i 从 CT 中删除。如果遍历完所有的叶节点后，没有移动任何叶节点，则聚类结束，否则重复上述过程，直到没有任何叶节点移动为止。如果对聚类精度要求不是十分苛刻，用户可以设置一个移动阈值 t，当移动量小于该阈值时即可终止聚类过程。

3.3.3.1 精炼过程

如果用户想得到更为精确的聚类结果，可以选择执行聚类结果的精炼过程，这个过程与 CCDCS 算法的第二阶段相似，不同之处在于聚类结果获取阶段是为 CS_i 重新找到位置，而精炼过程则需要为每个对象 X_i 重新找到位置；由于聚类树 CT 只保存了聚类汇总 *CS* 的信息，所以无法判断对象新找到的位置（即类）是否和它原本的类相同，这就需要在构造聚类树时，每插入一个新的对象，都需要在外部存储设备上为这个对象打上标记——分类 ID；聚类精炼过程从数据库中逐个读入已经打上分类标记的对象，然后按照插入对象的方法，从 CT 的根节点起通过选择最相似节点找到一条路径到达叶子节点，即一个分类，然后判断该分类的 ID 号与外部存储设备上该对象的分类标记是否一致，如果不一致，则需要将对象移动到新的叶子节点中；在所有对象读入后，看这一轮是否有对象移动，如果有，则继续重复这一过程，否则停止算法。

$$CS.n = CS.n - 1 \tag{3.14}$$

$$CS.I = CS.I - \{x_j \mid x_j \in X, \text{ 且 } CS.\text{num}(x_j) = 1\} \tag{3.15}$$

$$CS.S = \{s \mid s = CS.\text{num}(x_j) - X.\text{num}(x_j), \text{ 其中 } x_j \in CS.I, j = 1, 2, \ldots, |CS.I|, s \neq 0\} \tag{3.16}$$

对象的移动操作需要修改从根节点到新旧两个叶子节点两条路径上所有节点的 *CS*，对新路径上所有节点的修改方式同构造聚类树时插入新对象完全相同，包括到达叶子节点后，判断是否需要新增节点和分裂节点的操作也完全一样；对于旧路径上的所有节点，需要从中删除该对象，假设某节点的聚类汇总为 *CS*，需要从中删除对象 *X*，节点的 *CS* 使用公式(3.14)——公式(3.16)修改：

3.4 理论分析

首先分析 CCDCS 算法的第一阶段——构造聚类树的时间复杂度。在含有 M 个结点的 *b* 阶聚类树上插入新的数据对象时，从根结点到目的结点的路径上需要访问的结点数为 $1 + \log_b M$，新对象每到达一个结点，都要和该结点的所有子结点进行相似度计算，子结点的数量不超过 *b* 个；相似度的计算耗时与 p 成正比，p 是集合 P 的平均长度，有关集合 P 的描述见 3.3.2.1。因此从数据集合读入所有的数据对象用以构造聚类树的时间复杂度为 $O(p*N*b(1 + \log_b M))$，其中参数 N 为数据库包含的对象个数；参数 p 代表相似度计算耗时；*b* 为聚类树的阶数；M 为聚类树的结点数，当最小相似度 minsim 设置为 1 时，M 等于 N，这是 M 的最大值。

在 CCDCS 算法的第二阶段——获取聚类结果——需要为聚类树的叶子结点重新搜索目的类以纠正第一阶段聚类产生的偏差，重新搜索过程需要访问 $1 + \log_b M$ 个结点，并与每个结点的所有子结点进行相似度计算，与第一阶段不同的是，这里需要计算类相似度而不是对象和类的相似度，而类相似度的计算和 p^2 成正比。那么，第二阶段的时间复杂度为 $O(t* p^2*M*b(1 + \log_b M))$，其中 p^2 表示相似度计算耗时；M 为聚类树中叶子结点的最大可能数量；$b(1 + \log_b M)$为相似度计算的次数；参数 t 表示这一偏差纠正过程需要重复的次数，t 值可以由用户事先给定。

CCDCS 算法的时间复杂度为 $O(p*N*b(1 + \log_b M) + t* p^2*M*b(1 + \log_b M)) = O(b(1 + \log_b M)(pN + t\ p^2M)) \leqslant O(b*N*p(1 + \log_b M)(1 + t*p)) = O(C*N*\ p^2\ (1 + \log_b M))$，其中 C 为常量。

在 I/O 耗时方面，在第一阶段就可以将所有数据读入内存，在第二阶段无需进行 I/O 操作。

CCDCS 的空间复杂度为 S(M)，M 的数量取决于用户对算法参数最小相似度 minsim 的取值，minsim 的值域为[0, 1]，则 M 的值域为[1, N]，通常，最小相似度的取值不会太大，因此 M 远小于 N。

对象的移动操作需要修改从根节点到新旧两个叶子节点两条路径上所有节点的 *CS*，对新路径上所有节点的修改方式同构造聚类树时插入新对象完全相同，包括到达叶子节点后，判断是否需要新增节点和分裂节点的操作也完全一样；对于旧路径上的所有节点，需要从中删除该对象，假设某节点的聚类汇总为 *CS*，需要从中删除对象 *X*，节点的 *CS* 使用公式(3.14)——公式(3.16)修改：

3.4 理论分析

首先分析 CCDCS 算法的第一阶段——构造聚类树的时间复杂度。在含有 M 个结点的 *b* 阶聚类树上插入新的数据对象时，从根结点到目的结点的路径上需要访问的结点数为 $1 + \log_b M$，新对象每到达一个结点，都要和该结点的所有子结点进行相似度计算，子结点的数量不超过 *b* 个；相似度的计算耗时与 p 成正比，p 是集合 P 的平均长度，有关集合 P 的描述见 3.3.2.1。因此从数据集合读入所有的数据对象用以构造聚类树的时间复杂度为 $O(p*N*b(1 + \log_b M))$，其中参数 N 为数据库包含的对象个数；参数 p 代表相似度计算耗时；*b* 为聚类树的阶数；M 为聚类树的结点数，当最小相似度 minsim 设置为 1 时，M 等于 N，这是 M 的最大值。

在 CCDCS 算法的第二阶段——获取聚类结果——需要为聚类树的叶子结点重新搜索目的类以纠正第一阶段聚类产生的偏差，重新搜索过程需要访问 $1 + \log_b M$ 个结点，并与每个结点的所有子结点进行相似度计算，与第一阶段不同的是，这里需要计算类相似度而不是对象和类的相似度，而类相似度的计算和 p^2 成正比。那么，第二阶段的时间复杂度为 $O(t* p^2*M*b(1 + \log_b M))$，其中 p^2 表示相似度计算耗时；M 为聚类树中叶子结点的最大可能数量；$b(1 + \log_b M)$为相似度计算的次数；参数 t 表示这一偏差纠正过程需要重复的次数，t 值可以由用户事先给定。

CCDCS 算法的时间复杂度为 $O(p*N*b(1 + \log_b M) + t* p^2*M*b(1 + \log_b M)) = O(b(1 + \log_b M)(pN + t\ p^2M)) \leqslant O(b*N*p(1 + \log_b M)(1 + t*p)) = O(C*N*\ p^2\ (1 + \log_b M))$，其中 C 为常量。

在 I/O 耗时方面，在第一阶段就可以将所有数据读入内存，在第二阶段无需进行 I/O 操作。

CCDCS 的空间复杂度为 S(M)，M 的数量取决于用户对算法参数最小相似度 minsim 的取值，minsim 的值域为[0, 1]，则 M 的值域为[1, N]，通常，最小相似度的取值不会太大，因此 M 远小于 N。

表 3.2 给出了算法 CCDCS 与算法 CLARANS 和 LargeItems 的复杂度比较结果。时间复杂度中，算法 CLARANS 和 LargeItems 中的参数 n 指类中包含对象的平均个数，这两个算法的类相似度计算耗时与 n^2 成正比，而 CCDCS 中与相似度计算耗时有关的是 p^2，由于 p 是类中数量不等的项目数，因此 $p < n$。空间复杂度中，一般最小相似度取值不会太大，所以 M<<N，因此 CCDCS 的空间复杂度要优于算法 CLARANS 和 LargeItems。从复杂度比较得出，算法 CCDCS 优于其它两个算法。

表 3.2 复杂度比较

算法	时间复杂度	空间复杂度
CLARANS	$O(N^2*n^2)$	S(N)
LargeItems	$O((t+1)N^2*n^2)$	S(N)
CCDCS	$O(C*N*(1+\log_b M)*p^2)$	S(M)

3.5 实验分析

本节将讨论 CCDCS 算法同其它聚类算法在聚类结果和运行时间上的比较，实验分别在蘑菇数据集合和合成数据集合上进行。

蘑菇数据集合（Mushroom Data Sets）从 UCI Machine Learning Repository（http://www.ics.uci.edu/mlearn/MLRepository.html）获得，在使用之前，需要将数据库中的每条记录映射为上文一直使用的分类对象集合表达形式；蘑菇数据库中的每条记录包含了一个品种的外观属性（如颜色、气味、外形、生长环境等），所有的属性都属于分类属性，如外形属性的值为钟形、扁平、凸形和锥形，并且每个品种都预先打上了“可食用”或者“有毒”的标签，蘑菇数据库总共有 8,124 条记录，其中 4,208 条为可食用，3,916 为有毒蘑菇。为方便表述，蘑菇数据库记做 D_m。

合成数据的生成由表 3.3 中给出的一系列参数控制，生成方法参见文献[92]。表 3.3 中 m 为描述数据集合 D 的 $A_1, A_2, \ldots, A_m$ 个属性，$|D|$为记录数，数据集合 D 出 K 个类组成；C 为标签数量，例如蘑菇数据库中的标签数为 2（分别为“可食用”和“有毒”）；假定属性 A_i 的值域长度为$|\mathrm{DOM}(A_i)|$，$\overline{V}$ 就是所有属性值域长度的平均值，即$(|\mathrm{DOM}(A_1)| + |\mathrm{DOM}(A_2)| + \ldots + |\mathrm{DOM}(A_m)|) / m$；假定数据集合中对象 x_i 的长度为$|x_i|$，其中 $i = 1, 2, \ldots, |D|$，$\overline{L}$ 为数据集合中对象的平均长度，即$(|x_1| + |x_2|$

表 3.2 给出了算法 CCDCS 与算法 CLARANS 和 LargeItems 的复杂度比较结果。时间复杂度中，算法 CLARANS 和 LargeItems 中的参数 n 指类中包含对象的平均个数，这两个算法的类相似度计算耗时与 n^2 成正比，而 CCDCS 中与相似度计算耗时有关的是 p^2，由于 p 是类中数量不等的项目数，因此 $p < n$。空间复杂度中，一般最小相似度取值不会太大，所以 M<<N，因此 CCDCS 的空间复杂度要优于算法 CLARANS 和 LargeItems。从复杂度比较得出，算法 CCDCS 优于其它两个算法。

表 3.2 复杂度比较

算法	时间复杂度	空间复杂度
CLARANS	$O(N^2*n^2)$	S(N)
LargeItems	$O((t+1)N^2*n^2)$	S(N)
CCDCS	$O(C*N*(1+\log_b M)*p^2)$	S(M)

3.5 实验分析

本节将讨论 CCDCS 算法同其它聚类算法在聚类结果和运行时间上的比较，实验分别在蘑菇数据集合和合成数据集合上进行。

蘑菇数据集合（Mushroom Data Sets）从 UCI Machine Learning Repository（http://www.ics.uci.edu/mlearn/MLRepository.html）获得，在使用之前，需要将数据库中的每条记录映射为上文一直使用的分类对象集合表达形式；蘑菇数据库中的每条记录包含了一个品种的外观属性（如颜色、气味、外形、生长环境等），所有的属性都属于分类属性，如外形属性的值为钟形、扁平、凸形和锥形，并且每个品种都预先打上了“可食用”或者“有毒”的标签，蘑菇数据库总共有 8,124 条记录，其中 4,208 条为可食用，3,916 为有毒蘑菇。为方便表述，蘑菇数据库记做 D_m。

合成数据的生成由表 3.3 中给出的一系列参数控制，生成方法参见文献[92]。表 3.3 中 m 为描述数据集合 D 的 $A_1, A_2, \ldots, A_m$ 个属性，$|D|$为记录数，数据集合 D 出 K 个类组成；C 为标签数量，例如蘑菇数据库中的标签数为 2（分别为“可食用”和“有毒”）；假定属性 A_i 的值域长度为$|\mathrm{DOM}(A_i)|$，$\overline{V}$ 就是所有属性值域长度的平均值，即$(|\mathrm{DOM}(A_1)| + |\mathrm{DOM}(A_2)| + \ldots + |\mathrm{DOM}(A_m)|) / m$；假定数据集合中对象 x_i 的长度为$|x_i|$，其中 $i = 1, 2, \ldots, |D|$，$\overline{L}$ 为数据集合中对象的平均长度，即$(|x_1| + |x_2|$

$+ \ldots + |x_{|D|}|) / |D|$。在实验中根据需要调整这些参数值得到不同的数据集合，在后面实验中用到的合成数据，如果不特别指出，其参数都按照表 3.3 取值。同蘑菇数据库 D_m 类似，合成数据库记做 D，不同参数的合成数据集合使用下标以示区分，如 D_1。

表 3.3 合成数据生成参数

参数	值
m	30
K	200
$\overline{V}$	6
$\overline{L}$	16
$\|D\|$	50,000
C	4

选择 BIRCH、CLARANS、ROCK 和 LargeItems 算法与 CCDCS 进行比较，将这五个算法分别记做算法 BI、C、R、L 和 CC，算法 BI 和 C 是传统的数字数据聚类算法，而后三个都是为分类数据而提出的聚类算法。实验设计成三大部分，第一部分测试 CCDCS 的聚类结果，固定 CCDCS 算法的两个参数 B 和 *minsim*，取蘑菇数据库和另外两个合成数据库，在其上分别用以上五个算法聚类，比较它们的结果，由于算法 BI 和 C 只适合数字数据，算法 R、L 和 CC 得到的结果应当比它们要好；实验的第二部分比较 CCDCS 与其它算法的效率，选用三个不同参数的合成数据集合，比较五个算法的运行时间，由于算法 BI 也采用了压缩数据集合的技术，所以 CCDCS 的运行时间应当和算法 BI 相差不多，但要比另外三个效率高。第三部分用来测试不同的参数(包括算法参数和数据集合参数)对于 CCDCS 运行时间的影响。在下面的实验中，如果不加以特别说明，则算法 BI 的参数页面大小 P 取 1024、算法 C 的两个参数最大邻居数 *maxneighbor* 和聚类数 k 分别取 1000 和 20、算法 R 的参数 k 取 20、 θ 取 0.8；算法 L 的最小支持度为 50%。所有的实验都在处理器为 450 MHz 的 Intel Pentium III 机器上进行，内存 128MB，操作系统为 Windows NT4.0。

3.5.1 聚类结果

表 3.4 给出了五个算法在蘑菇数据集合上得到的结果，在这个实验中，CCDCS 算法的参数 B 和 *minsim* 分别取 4 和 70%。观察表 3.4 给出的结果，算法 C 和算法 BI 得到的聚类结果并不理想，聚类结果中可以食用的蘑菇和有毒的蘑菇之间并没有明显的界限，由于这两个算法不适用于分类属性的数据，因此 CCDCS

得到的结果比算法 BI 和 C 要好；算法 R、L 和 CCDCS 在类的“纯度”上都表现很好，得到的单个类中，总是具有某个标签的对象数量占绝对多数。算法 R 中最大的两个类(7 和 16)包含的对象覆盖了整个数据集合的 43%，与它相似，CCDCS 最大的两个类（9 和 14）包含的对象覆盖率为 45%，算法 L 得到两个最大类的覆盖率为 84%，这是因为它采取了分步变换最小支持度的方法，先设定一个最小支持度 20%得到类 1—4 和一个大类 *X*，然后在 *X* 上设置最小支持度 50%得到类 5—8 和大类 *Y*，最后在 *Y* 上设置最小支持度得到类 9—14，这么做得到的结果在结构上显得更为简洁，但条件是必须事先了解整个数据集合的数据分布状况，而通常情况下，数据分布状况是事先未知的。

表 3.4 聚类结果比较——蘑菇数据库

算法 BI

类	有毒	食用	类	有毒	食用
1	583	697	**11**	63	56
2	173	21	**12**	278	190
3	76	45	**13**	127	64
4	131	109	**14**	176	171
5	269	377	**15**	74	154
6	421	378	**16**	92	136
7	245	388	**17**	257	253
8	12	228	**18**	335	153
9	336	328	**19**	37	56
10	173	243	**20**	58	161

算法 C

类	有毒	食用	类	有毒	食用
1	672	653	**11**	121	104
2	43	357	**12**	132	43
3	75	125	**13**	116	157
4	212	193	**14**	321	285
5	35	12	**15**	81	153
6	790	388	**16**	348	467
7	357	445	**17**	43	290
8	54	97	**18**	174	63
9	47	34	**19**	54	42
10	154	156	**20**	87	144

算法 L

类	有毒	食用	类	有毒	食用
1	94	0	8	0	287
2	13	0	9	61	3388
3	6	0	10	372	77
4	682	26	11	9	0
5	2631	30	12	19	10
6	121	37	13	21	0
7	69	61	14	110	0

算法 R

类	有毒	食用	类	有毒	食用
1	96	0	11	48	0
2	0	256	12	48	0
3	704	0	13	0	288
4	96	0	14	192	0
5	768	0	15	32	72
6	0	192	16	0	1728
7	1728	0	17	288	0
8	0	32	18	0	8
9	0	1296	19	192	0
10	0	8	20	16	0
			21	0	36

算法 CC

类	有毒	食用	类	有毒	食用
1	87	0	11	58	0
2	26	2	12	0	21
3	671	0	13	12	680
4	280	0	14	1895	0
5	37	258	15	0	721
6	79	0	16	316	0
7	174	0	17	0	237
8	92	0	18	0	84
9	0	1751	19	0	162
10	481	0			

根据表 3.3 所设置的参数得到合成数据集合 D，由于 D 的类数量 K 设置为 200，用表格方式给出并不直观，所以图 3.5 利用误差率 e 比较几个算法的聚类结果。假定一个类有如下形式，$C=\{l_1, l_2, \ldots, l_n\}, n>1$，其中 n 为标签数量，l_i 为类 C 中具有标签 i 的对象数量，$1\leqslant i\leqslant n$，假定 C 中 l_j 最大，$1\leqslant j\leqslant n$，则认为 j 是最能代表类 C 的属性，好的聚类结果应当使得 $l_i=0$（$1\leqslant i\leqslant n$，$i\neq j$）同时 l_j 为最大，因此，可以用误差率 e 表示某个类的聚类质量，e 定义如下：$e=(\sum_{i=1}^{n} l_i - l_j)/\sum_{i=1}^{n} l_i$。

如果 e 越小，则说明 C 质量越好，e 越大则 C 的质量越差，当 e 等于 0，则说明 C 的质量最好，e 最大为 $(n\text{-}1)/n$，本例中由于 $n=4$，因此，它的值域为[0, 0.75]。图 3.5 中的横轴为聚类得到的类的 ID 号，对于每一个算法得到的类，首先需要根据类的误差率从小到大排序，然后再为它们顺序分配 ID 号，所以 ID 号靠前的类其误差率都比 ID 号靠后的类要小。从图中可以看出，算法 CC、R 和 L 的误差率稳定在一个较小的值，聚类质量较高，而算法 BI 和 C 得到的结果中误差率较大。

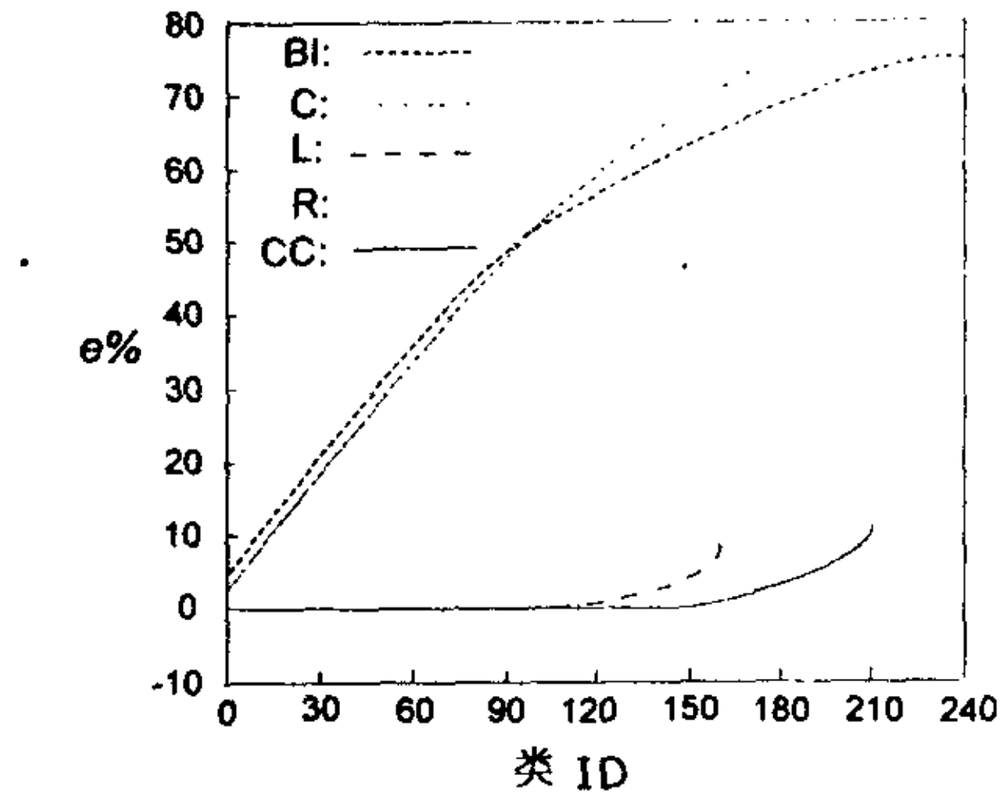

图 3.5 合成数据聚类结果

3.5.2 运行时间

图 3.6 至图 3.8 分别给出了在数据集合参数$|D|$、m 和 k 值变化时，五个算法运行时间的比较。从图中可以看到，由于采用了数据压缩技术，CCDCS 要比算法 C、L 和算法 R 效率要高，同时与同样采用了压缩技术的算法 BI 比较，在数据集合相同的条件下，CCDCS 要慢一些，这是因为算法 B 处理的是数字类型的数据，而 CCDCS 处理的原始数据是分类数据的集合，在运算处理上要更耗费时间。

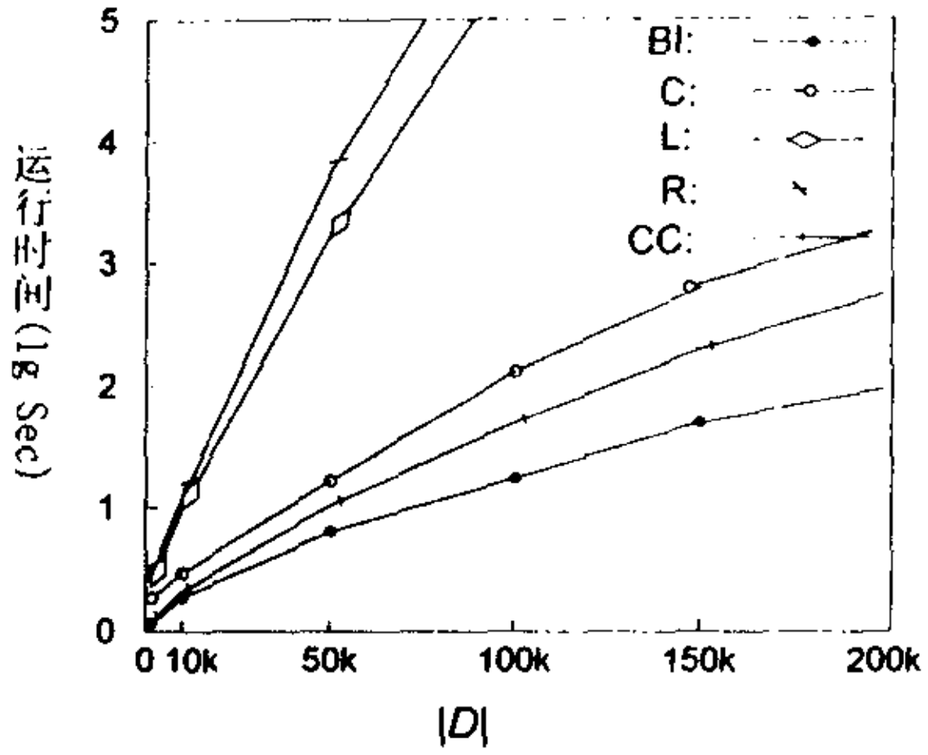

图 3.6 |D|改变时算法时间的变化

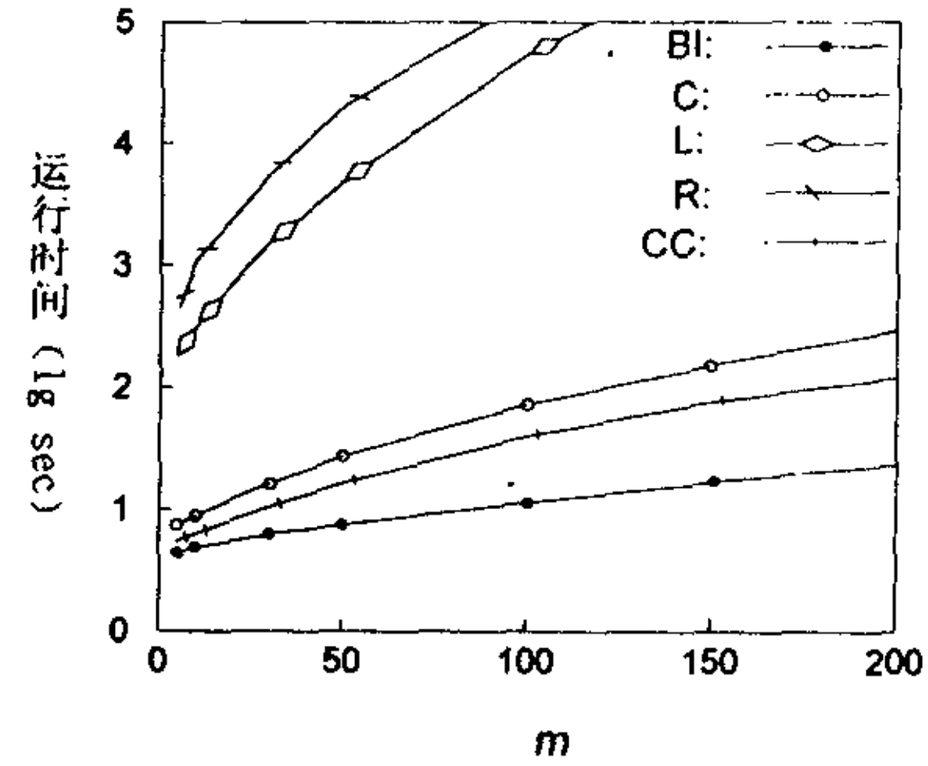

图 3.7 m 改变时算法时间的变化

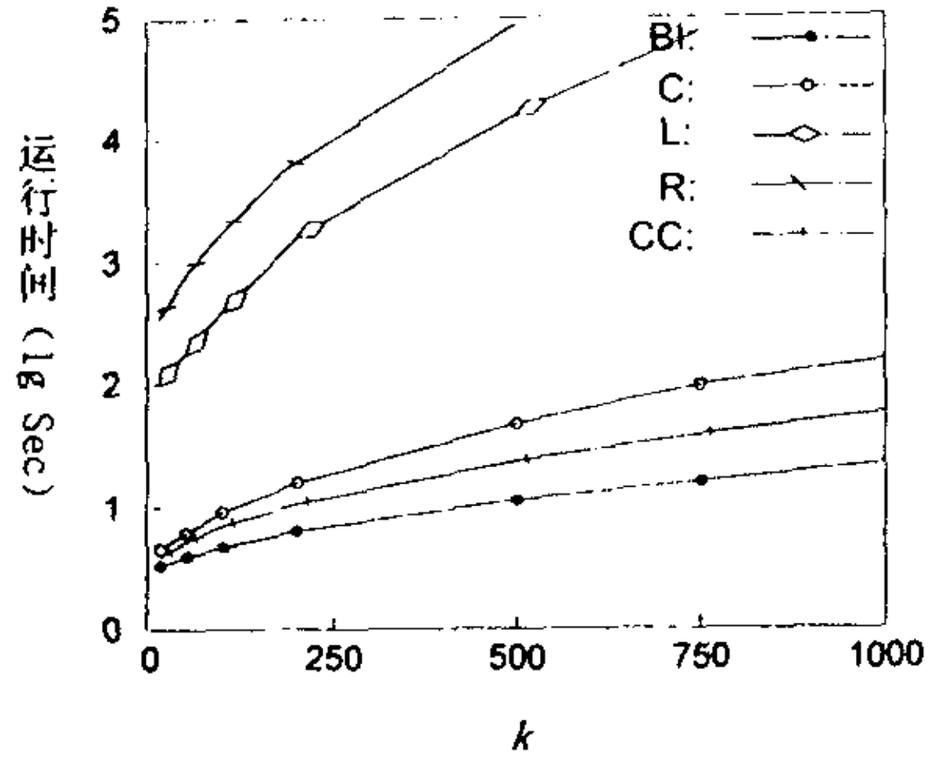

图 3.8 k 改变时算法时间的变化

3.5.3 参数影响

给出了最小相似度变化时 CCDCS 在由表 3.3 生成的合成数据集合上的运行时间，当 $minsim = 0$ 时，算法所需要做的只是从数据库中读取数据对象，所以运行时间相当快，随着最小相似度的增大，聚类树的叶子节点增多，新对象插入到聚类树的计算增多，算法的耗时也会增加。

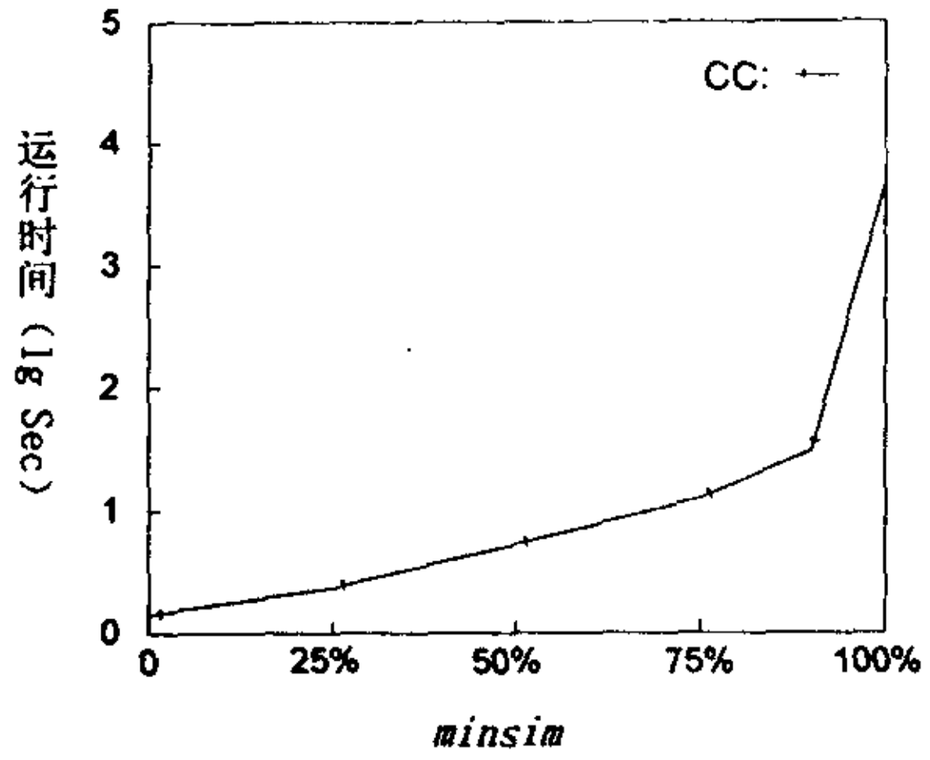

图 3.9 *minsim* 对 CCDCS 算法时间的影响

理论上，$\overline{V}$ 值的增加会导致数据集合中类的数量增加，即 K 值增大，而 K 值的增大会使得算法时间增多（图 3.8），但是从图 3.10 得到，$\overline{V}$ 值的变化对算法的运行时间影响不大，这是因为在这个实验中采取的数据集合的 K 值是固定不

变的，$\overline{V}$ 对 K 的数量不会构成影响，所以随着 $\overline{V}$ 值的改变算法时间变化不大。

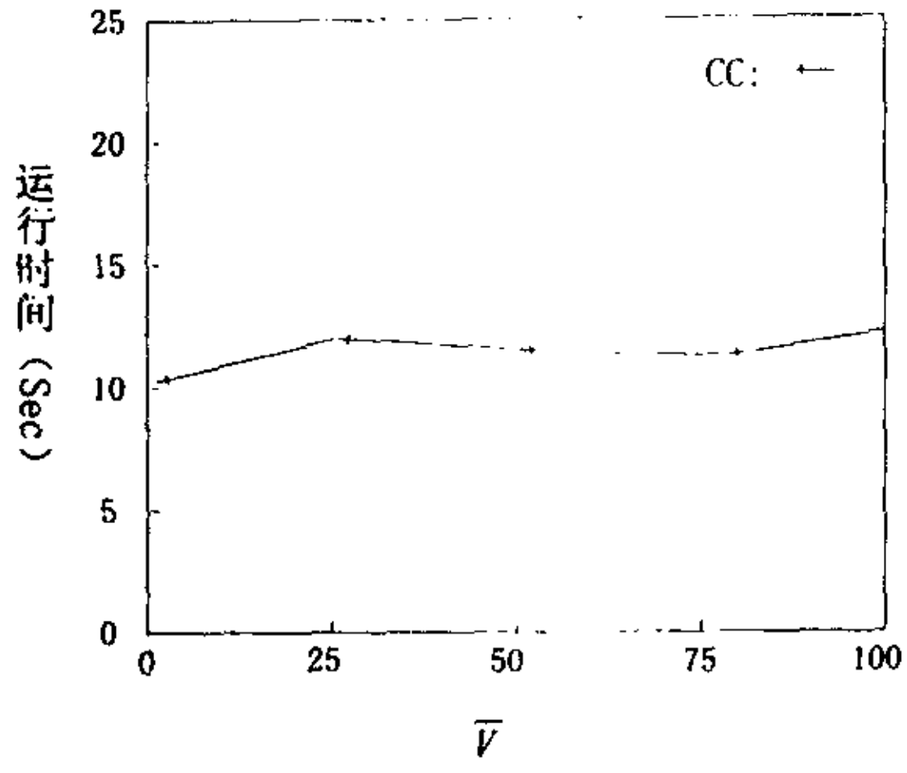

图 3.10 $\overline{V}$ 对 CCDCS 算法时间的影响

3.6 本章小结

数据开采面对的是现实世界中的问题，传统的聚类分析多用来解决数字属性数据的分类，而对于其中大量存在着的分类属性的数据，研究相对较少。现有的针对分类数据的算法需要多遍扫描数据库，对于数据开采经常处理的大容量数据，多遍 I/O 操作是一项沉重的系统开销，算法的效率会非常低。

针对这些问题，本文提出了针对分类属性数据的聚类算法 CCDCS，它利用数据压缩技术将原始数据用聚类汇总 CS 的形式表达，然后将 CS 存储到数据结构聚类树 CT 中，这样只需扫描一遍数据库就可以将数据全部存储到系统内存中，无需再次执行 I/O 操作；由于聚类汇总包含了足够的信息用于计算对象之间的相似度，因此，数据压缩不会对聚类结果造成影响；另外，聚类树的构造采用了估价函数搜索策略使得聚类过程效率得到提高。从与其它算法在聚类结果和运行时间的比较来看，CCDCS 比其它针对分类数据的算法效率要高，而聚类结果的质量和它们相差无几。

变的，$\bar{V}$ 对 K 的数量不会构成影响，所以随着 $\bar{V}$ 值的改变算法时间变化不大。

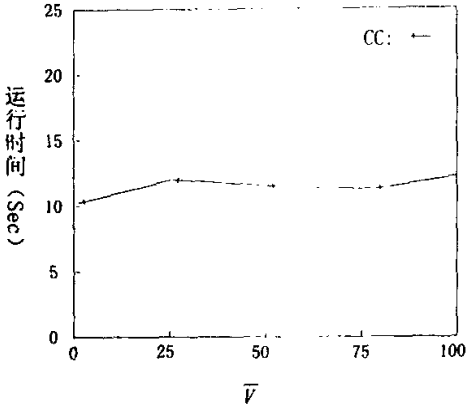

图 3.10 $\bar{V}$ 对 CCDCS 算法时间的影响

3.6 本章小结

数据开采面对的是现实世界中的问题，传统的聚类分析多用来解决数字属性数据的分类，而对于其中大量存在着的分类属性的数据，研究相对较少。现有的针对分类数据的算法需要多遍扫描数据库，对于数据开采经常处理的大容量数据，多遍 I/O 操作是一项沉重的系统开销，算法的效率会非常低。

针对这些问题，本文提出了针对分类属性数据的聚类算法 CCDCS，它利用数据压缩技术将原始数据用聚类汇总 CS 的形式表达，然后将 CS 存储到数据结构聚类树 CT 中，这样只需扫描一遍数据库就可以将数据全部存储到系统内存中，无需再次执行 I/O 操作；由于聚类汇总包含了足够的信息用于计算对象之间的相似度，因此，数据压缩不会对聚类结果造成影响；另外，聚类树的构造采用了估价函数搜索策略使得聚类过程效率得到提高。从与其它算法在聚类结果和运行时间的比较来看，CCDCS 比其它针对分类数据的算法效率要高，而聚类结果的质量和它们相差无几。

4 聚类高维数据

高维数据指那些属性数量特别大的数据对象，由这些数据对象组成的高维数据空间维数较高，数据点分布稀疏、密度平均，因此，从中发现数据聚类比较困难，但同时，在高维数据空间的子空间，即维数较低的空间，对象分布又较密集，可以根据对象分布聚类数据；另一方面，使用距离公式聚类高维数据，维数的增加使得对象对之间的距离计算时间开销增大，导致算法效率降低。一些聚类算法采用降维方法[119]解决这些问题，例如文档聚类，每个文档对象的属性集合是文档集合中所有的关键字，降低维数的方法有两种，一种是局部降维，删除每个文档对象中出现较少的关键字，这使得每个文档对象存在于不同的子空间中；另一种方法是全局降维，在聚类之前就删除掉那些“不重要”的属性。这两种方式都可以大大降低对象间距离的计算量，从而提高算法效率，但是降低维数的方法不能够自动发现子空间上的数据聚类。本章提出的算法 CHID（Clustering HIgh-dimensional Data）是我们在文献[120]提出算法的扩展，能够有效地处理高维数据，找出 k 维空间及其子空间的聚类，同时该算法具有扩展性，能够高效率开采大型数据集合。

4.1 问题描述

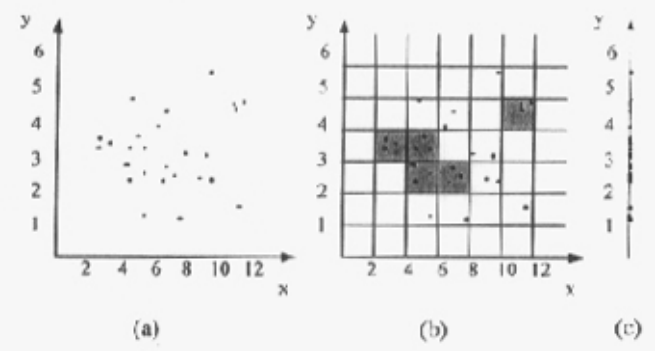

图 4.1 二维空间的聚类

将数据对象看作高维数据空间中的数据点，根据数据点在高维空间的自然分布，区分密度小和密度大的区域，将那些聚集在一起的密度较大的区域作为聚类结果。每个对象可以表示成向量（$i_1, i_2, \ldots, i_k$），为 k 维数据空间的一个点，算法的目的是找出 k 维空间及其子空间的点密集区域，这些密集区域就是我们需要的

4 聚类高维数据

高维数据指那些属性数量特别大的数据对象，由这些数据对象组成的高维数据空间维数较高，数据点分布稀疏、密度平均，因此，从中发现数据聚类比较困难，但同时，在高维数据空间的子空间，即维数较低的空间，对象分布又较密集，可以根据对象分布聚类数据；另一方面，使用距离公式聚类高维数据，维数的增加使得对象对之间的距离计算时间开销增大，导致算法效率降低。一些聚类算法采用降维方法[119]解决这些问题，例如文档聚类，每个文档对象的属性集合是文档集合中所有的关键字，降低维数的方法有两种，一种是局部降维，删除每个文档对象中出现较少的关键字，这使得每个文档对象存在于不同的子空间中；另一种方法是全局降维，在聚类之前就删除掉那些“不重要”的属性。这两种方式都可以大大降低对象间距离的计算量，从而提高算法效率，但是降低维数的方法不能够自动发现子空间上的数据聚类。本章提出的算法 CHID（Clustering HIgh-dimensional Data）是我们在文献[120]提出算法的扩展，能够有效地处理高维数据，找出 k 维空间及其子空间的聚类，同时该算法具有扩展性，能够高效率开采大型数据集合。

4.1 问题描述

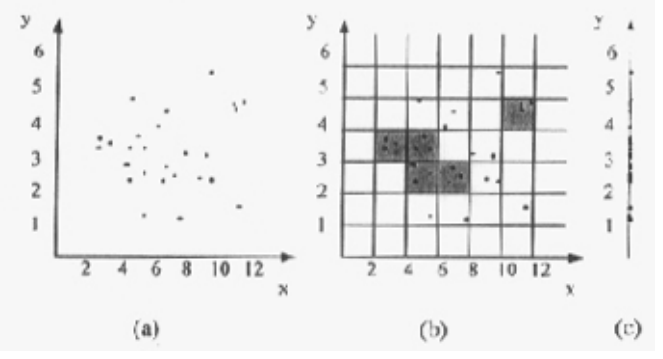

图 4.1 二维空间的聚类

将数据对象看作高维数据空间中的数据点，根据数据点在高维空间的自然分布，区分密度小和密度大的区域，将那些聚集在一起的密度较大的区域作为聚类结果。每个对象可以表示成向量（$i_1, i_2, \ldots, i_k$），为 k 维数据空间的一个点，算法的目的是找出 k 维空间及其子空间的点密集区域，这些密集区域就是我们需要的

数据聚类。

图 4.1 (a) 是一个二维空间的数据分布，其上的聚类过程如下：首先将属性 x 和 y 的坐标轴等分，在该数据空间上得到面积相同的单元格，统计每个单元格内包含的数据点数量，将它们与事先给定的阈值 T 比较，数据点数量大于或等于 T 的单元格称作密集单元格，相互连接的密集单元格构成一个类，这里相互连接可以是直接连接（在某一维邻接）也可以是间接连接（中间间隔有其它密集单元格）。图 4.1 (b) 展示了 $T=3$ 时的结果，一个阴影区域为一个类，图中显示聚类结果为两个数据类。图 4.1 (c) 是数据在属性 y 上的聚类结果，很明显，该子空间的聚类结果为一个类 1—5。

有关数据类的描述如下：已知包含 n 个数据点的集合 $P = \{p_1, p_2, \ldots, p_n\}$ 在 k 维数据空间 $S = \{1\times2\times\ldots\times k\}$ 中，其中 $p_i = (d_1, d_2, \ldots, d_k)$；将 S 沿各坐标轴等分，得到互不重叠的单位区域 ur（unit region）= $\{ur_1, ur_2, \ldots, ur_m\}$，$ur_i$ 记作($[a_{i1}, L_{i1}, R_{i1}], \ldots, [a_{ik}, L_{ik}, R_{ik}]$)，$a_{ij}$ 代表 ur_i 第 j 维的属性名称，$[L_{ij}, R_{ij}]$代表 ur_i 第 j 维上的间隔；如果对于某个 ur_i 中所有的$[L_{ij}, R_{ij}]$，数据点 p_i 满足 $L_{ij} \leqslant d_j < R_{ij}$，那么 p_i 被包含于 ur_i 中，一个单位区域包含的数据点数量记作 P_num(ur_i)；如果 P_num(ur_i)$\geqslant T$，则称 ur_i 为密集单位区域 dur（dense unit region）；如果 $ur_i = ([a_{i1}, L_{i1}, R_{i1}], \ldots, [a_{ik}, L_{ik}, R_{ik}])$中有且只有一个$[L_{ij}, R_{ij}]$与 $ur_n = ([a_{n1}, L_{n1}, R_{n1}], \ldots, [a_{nk}, L_{nk}, R_{nk}])$中的$[L_{nj}, R_{nj}]$存在如下关系：$[L_{ij}] = [R_{nj}]$或$[L_{nj}] = [R_{ij}]$，并且其他 k-1 个$[L_{ij}, R_{ij}] = [L_{nj}, R_{nj}]$，则称单位区域 ur_i 和 ur_j 两两相邻，一组互相邻接的单位区域称作相互连接，很明显，聚类结果就是一组相互连接的 dur。

如上文所述，最简单的聚类方法就是计算 k 维空间及其子空间的数据点分布，找到所有的 dur，然后将相互连接的 dur 划分为类即可。但对于高维海量数据集合，其时间复杂度相当高，因此，CHID 在计算数据点分布时利用了 Apriori 算法提出的两个性质[29]减少搜索空间，使算法具有收敛性，定理 1 和定理 2 就是根据 Apriori 算法的两个性质提出：

定理 1：如果 S 是 k 维空间的一个类中的数据点集合，那么将 S 映射到 k-1 维空间得到 S'，则 S'将是 k-1 维空间某个类的子集。

定理 2：如果 S 是 k-1 维空间的数据点集合，但 S 不属于任何类，那么如果将 S 扩展到 k 维空间得到 S'，则 S'也不可能属于任何类。

为方便描述，将 k 维空间中的 S 映射到 k-1 维空间得到的 S'称作 S 的子集，S 称作 S'的超集。

4.2 相关工作

基于数据分布密度的聚类算法有 DBSCAN[92]和 CLIQUE[101]，DBSCAN 通过计算数据点的邻居数量决定这些数据点是否构成高密度区域，但是该算法不能有效处理高维数据；而 CLIQUE 算法则能够发现最高维空间及其子空间存在的类，该算法分为三个步骤：① 子空间搜索；② 发现类；③ 描述类。① CLIQUE 算法采用自底向上法，首先扫描数据库，找出 1 维空间中的密集单位区域，然后根据（k-1）维的密集单位区域生成 k 维空间密集单位区域的候选集，该候选集是 k 维空间密集单位区域集合的超集，有关候选集的生成方式详情见文献[30]。得到 k 维 *dur* 的候选集 C_k 后，逐个查看 C_k 中的 *dur* 在（k-1）维上的映射是否包含于 C_{k-1}，对于那些在（k-1）维上的映射不被 C_{k-1} 包含的 *dur*，根据定理 1 从 C_k 中删除以减少下一轮生成候选集的计算量。同时 CLIQUE 采取基于 MDL（Minimal Description Length）的剪枝策略删除某些“兴趣度不大”的子空间，给定子空间 $S_1, S_2, \ldots, S_n$，计算每个 S_j（$j = 1, 2, \ldots, n$）中所有密集单位区域包含的数据点数量，记作 xS_j，算法删除具有较小 xS_j 值的子空间，该方法能够提高算法效率，但同时也会造成类的丢失。② 子空间搜索的目的是发现 k 维空间及其子空间中的 *dur*，将这些 *dur* 组成的集合记作 D，类发现的目的就是要将 D 中互相连接的 *dur* 聚集在一起，形成 q 个类 $D^1, D^2, \ldots, D^q$。CLIQUE 算法采用深度优先搜索算法完成类发现，从 D 中任选一个 *dur* 作为当前 *dur*，为它分配一个类 ID，然后分别在不同的维上寻找与当前 *dur* 相邻的 *ur*，判断该 *ur* 是否为 *dur*，如果是，则为它们分配同一个类 ID，并将该 *dur* 作为当前 *dur*，重复以上过程；如果不是，则从 D 中任选未访问过的 *dur*，重复以上过程直到所有 *dur* 都打上类标签。③ 用简洁易懂的形式描述类。

与 CLIQUE 算法相似，CHID 算法也用来发现 k 维空间及其子空间中的类，与 CLIQUE 不同的是，CHID 采用了更为有效的搜索策略和聚类方法使算法能高效地处理高维大型数据集合。

4.3 算法 CHID

4.3.1 生成频繁项目集

发现频繁项目集是数据开采领域中的重要课题，广泛应用于关联规则发现、聚类等应用领域，有关频繁项目集的定义见 5.1.1。在聚类高维数据时，我们首先生成维数不同的频繁项目集，也就是生成不同子空间中的 dur。

4.2 相关工作

基于数据分布密度的聚类算法有 DBSCAN[92]和 CLIQUE[101]，DBSCAN 通过计算数据点的邻居数量决定这些数据点是否构成高密度区域，但是该算法不能有效处理高维数据；而 CLIQUE 算法则能够发现最高维空间及其子空间存在的类，该算法分为三个步骤：① 子空间搜索；② 发现类；③ 描述类。① CLIQUE 算法采用自底向上法，首先扫描数据库，找出 1 维空间中的密集单位区域，然后根据（k-1）维的密集单位区域生成 k 维空间密集单位区域的候选集，该候选集是 k 维空间密集单位区域集合的超集，有关候选集的生成方式详情见文献[30]。得到 k 维 *dur* 的候选集 C_k 后，逐个查看 C_k 中的 *dur* 在（k-1）维上的映射是否包含于 C_{k-1}，对于那些在（k-1）维上的映射不被 C_{k-1} 包含的 *dur*，根据定理 1 从 C_k 中删除以减少下一轮生成候选集的计算量。同时 CLIQUE 采取基于 MDL(Minimal Description Length）的剪枝策略删除某些“兴趣度不大”的子空间，给定子空间 $S_1, S_2, \ldots, S_n$，计算每个 S_j（$j = 1, 2, \ldots, n$）中所有密集单位区域包含的数据点数量，记作 xS_j，算法删除具有较小 xS_j 值的子空间，该方法能够提高算法效率，但同时也会造成类的丢失。② 子空间搜索的目的是发现 k 维空间及其子空间中的 *dur*，将这些 *dur* 组成的集合记作 D，类发现的目的就是要将 D 中互相连接的 *dur* 聚集在一起，形成 q 个类 $D^1, D^2, \ldots, D^q$。CLIQUE 算法采用深度优先搜索算法完成类发现，从 D 中任选一个 *dur* 作为当前 *dur*，为它分配一个类 ID，然后分别在不同的维上寻找与当前 *dur* 相邻的 *ur*，判断该 *ur* 是否为 *dur*，如果是，则为它们分配同一个类 ID，并将该 *dur* 作为当前 *dur*，重复以上过程；如果不是，则从 D 中任选未访问过的 *dur*，重复以上过程直到所有 *dur* 都打上类标签。③ 用简洁易懂的形式描述类。

与 CLIQUE 算法相似，CHID 算法也用来发现 k 维空间及其子空间中的类，与 CLIQUE 不同的是，CHID 采用了更为有效的搜索策略和聚类方法使算法能高效地处理高维大型数据集合。

4.3 算法 CHID

4.3.1 生成频繁项目集

发现频繁项目集是数据开采领域中的重要课题，广泛应用于关联规则发现、聚类等应用领域，有关频繁项目集的定义见 5.1.1。在聚类高维数据时，我们首先生成维数不同的频繁项目集，也就是生成不同子空间中的 dur。

4.3.1.1 搜索策略以及剪枝方法

一般发现频繁项目集的算法仅利用定理 1 来剪枝候选项目集，这决定了它们必须使用单向的自底向上搜索策略，而利用定理 2 可以对数据空间进行自顶向下的搜索，为了尽快发现频繁项目集，算法 CHID 采用了自底向上和自顶向下相结合的双向搜索策略，利用两个方向搜索到的信息对数据进行剪枝。

在算法 CHID 中，设置了两个数组 *RC*（Reverse Candidate Sets）和 *C*(Candidate Sets)分别保存自顶向下方向和自底向上方向生成的候选项目集。在进行第 k 遍搜索时，C_k 中每个项目集的维数为 k，而 RC_k 中每个项目集的维数为 n-k，其中 n 为最大的维数。

假设已经生成了 k 维和 n-k 维上的频繁项目候选集，分别为 C_k 和 C_{n-k}，算法比较这两个候选集中所包含的 *ur* 数量，如果$|C_{n-k}| \geq |C_k|$，则先逐个判断 C_k 中哪些 *ur* 是密集的，哪些是非密集的，删除 C_k 中的非 *dur*，同时记录下这些非 *dur* 到集合 S_k 中；接着逐个判断 C_{n-k} 中的 *ur*，如果某个 *ur* 是集合 S_k 中某个元素的超集，则不用计算该 *ur* 中包含数据点的个数以判断它是否为 *dur*，因为根据定理 2 该 *ur* 也应当是非 *dur*，判断完毕后，记录下 C_{n-k} 中的 *dur* 到集合 F 中，这样在对以后生成的频繁项目候选集（k+1 维至 n–k-1 维）剪枝时，根据定理 1，F 中每个 *dur* 的子集都是 *dur*，这样也不用计算候选集中这些 *ur* 包含数据点的个数。如果$|C_k| > |C_{n-k}|$，则先判断 C_{n-k} 中的 *ur*，再判断 C_k 中的 *ur*，其余步骤按照上述对应过程执行。

4.3.1.2 候选频繁项目集的生成

自底向上从 k 维项目集生成 k+1 维项目集的方法为：

Gen_bu(D_{k-1})

1. INSERT INTO C_k
2. SELECT $u_1.[L_1, R_1]$, $u_1.[L_2, R_2]$, …, $u_1.[L_{k-1}, R_{k-1}]$, $u_2.[L_{k-1}, R_{k-1}]$
3. FROM D_{k-1} u_1, D_{k-1} u_2
4. WHERE $u_1.a_1 = u_2.a_1$, $u_1.L_1 = u_2.L_1$, $u_1.R_1 = u_2.R_1$,
5. $u_1.a_2 = u_2.a_2$, $u_1.L_2 = u_2.L_2$, $u_1.R_2 = u_2.R_2$, …,
6. $u_1.a_{k-2} = u_2.a_{k-2}$, $u_1.L_{k-2} = u_2.L_{k-2}$, $u_1.R_{k-2} = u_2.R_{k-2}$,
7. $u_1.a_{k-1} < u_2.a_{k-1}$;
8. return C_k;

其中 C_k 是要生成的 k 维空间的 *dur* 候选集，D_{k-1} 是（k-1）维空间的 *dur* 集合，上面算法中 *dur* 的属性 a_i 按照字典序进行大小比较。

自顶向下从 *n-k* 维项目集生成 *n-k*-1 维项目集方法为：

Gen_ub(RC_k)

1. for $i = 1$ to $n-k$
2. INSERT INTO RC_{k-1}
3. SELECT $u_1.[L_1, R_1]$, $u_1.[L_2, R_2]$, …, $u_1.[L_{i-1}, R_{i-1}]$, $u_1.[L_{i+1}, R_{i+1}]$, …, $u_1.[L_{k-1}, R_{k-1}]$, $u_2.[L_{n-k}, R_{n-k}]$
4. FROM RC_k;
5. return RC_k;

其中 RC_{k-1} 是 *n–k*-1 维空间的 *dur* 候选集，RC_k 是 *n-k* 维空间的非密集 *ur* 的集合。

4.3.2 聚类

频繁项目集，即 *dur* 生成后，下一步就是要把这些密集单位区域聚类，将这个密集单位区域的集合记作 *D*，根据 4.1 问题描述中的类定义，CHID 采用“位与”的方法进行聚类。算法原理如下（以 2 维为例）：在 2 维空间中将 *dur* 填上 1，非 *dur* 填上 0，然后以任意一维为基础，将 2 维空间看成 *n* 行长度相等的字符串（串中只包括 0 和 1），从第 1 行开始，将其它所有行分别与第一行进行逐位相“与”，若发现第 *i* 行相与后得到的结果同 *i*-1 行相与后得到的结果不同，则输出第 *i*-1 行相与后的结果；如发现某行相“与”后的结果每一位都为 0，则中止这一轮计算，然后对其它 *n*-1 行重复这一过程，对于第 *j*（$j = 2, 3, .., n$）行，从 *j*+1 行开始与第 *j* 行相“与”，所有行执行完毕后，把所有的输出结果相比较得到最大的矩形区域，这些最大矩形区域就是聚类结果。

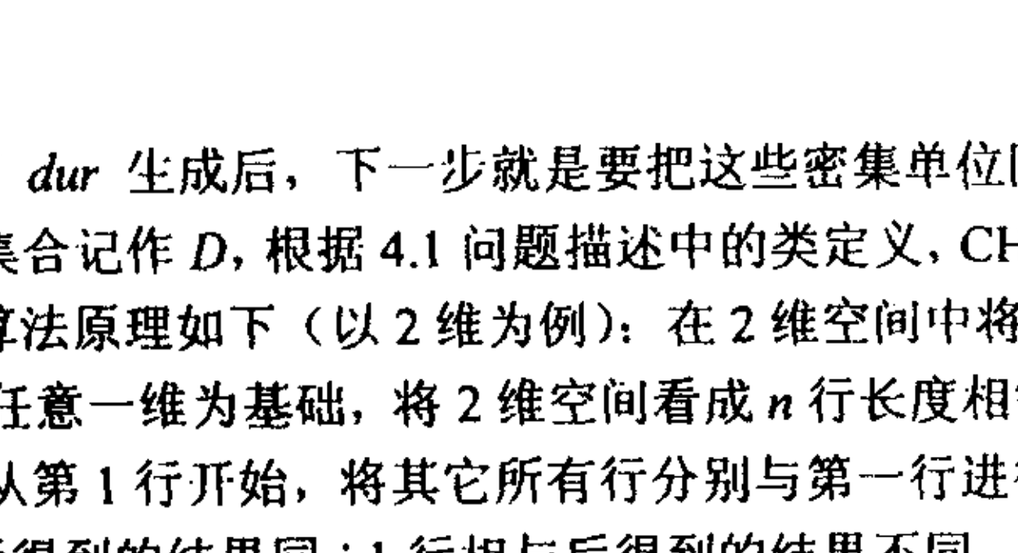

图 4.2 聚类过程

图 4.2 给出了 2 维空间的聚类过程，对于(a)所示的格子，从第一行开始与各行逐位相“与”，得到结果为(b)，接着用第 2 行和第 3 行进行同样的操作，得到(c)和(d)，最终得到聚类结果(e)，值得注意的是，最终聚类的结果可能不唯一。

对于 n 维空间，以任意一维 k 为基础，令 k 维上的等分单元数量为$|k|$，那么这个空间可以用 $S_1, S_2, \ldots, S_m$（$m = 1, 2, \ldots, |k|$）表示，其中 $S[i]$（$1 \leqslant i \leqslant m$）为 0 和 1 组成的字符串，串中每一位对应 n 维空间中一个单位区域，如果该单位区域为 *dur*，则为 1，否则为 0；该单位区域的向量表示为：$(i, a_1, a_2, \ldots, a_{n-1})$，$a_j$ 是第 j 维上某个单元的序号；对于任意两个字符串 $S[i]$和 $S[j]$，其中每一位的单位区域向量，除了 i 不等于 j 外，其余都应该相同，原因是要保证沿着第 k 维将单位区域相与。由于 $S_1, S_2, \ldots, S_m$ 也可以看成 m 行长度相等的由 0 和 1 组成的字符串，因此，可以按照 2 维空间的方法聚类。

4.3.3 算法描述

CHID（*D*）

1. DUR_1 = 1 维密集单位区域，$UR_n = \{ur_1, ur_2, \ldots, ur_m\}$，$k = 1$，$S = \{\}$；
2. repeat
3. $DUR_{k+1} = \{\}$；
4. DUR_{k+1} = Gen_bu(DUR_k)；
5. UR_{n-k} = Gen_ub(DUR_{n-k+1})；
6. Prunning(ref DUR_{k+1}, ref UR_{n-k})；
7. Mark(*S*)；
8. $k = k+1$；
9. until ($n-k = k$) or (DUR_{k+1} = NULL) or (UR_{n-k} 中全部都是 *dur*)；
10. *S*'' = NULL；
11. for i = 1 to $|S|$ {
12. *S*' = NULL；
13. for *row* = i+1 to $|S|$
14. *S*'[*row*] = And(*S*[i], *S*[*row*])；
15. if *S*'[*row*] <> *S*'[*row*-1] then
16. *S*'' = *S*'' ∪ *S*'[*row*-1]；
17. if *S*'[*row*]为全'0' then continue；
18. }
19. return Merge(*S*'')；

CHID 算法的参数 D 为包含数据点的 n 维空间，D 已沿各坐标轴等分，得到 m 个单位区域，步骤 1 获取 1 维 *dur* 存放到 DUR_1，将 m 个单位区域存放到 UR_n 中，S 的数据结构在上一节描述，是用来聚类的数组，其中记录了 n 维空间中每个单元区域的密集状态；步骤 2-9 分别从自顶向下和自底向上两个方向搜索 *dur*，其中函数 Prunning()用来对 *dur* 候选集剪枝，剪枝过程在 4.3.1.1 中描述；函数 Mark 则根据搜索结果逐步生成 S 供聚类使用。步骤 10－19 则是聚类过程，将每次输出

的类存放在 S'' 然后利用函数 Merge 从中发现最大区域作为聚类结果输出。

4.4 实验结果

本文将在高维合成数据集合上比较算法 CLIQUE 和 CHID，实验分别从算法执行效率和准确性两方面验证 CHID 的性能，表 4.1 给出了合成数据的参数取值范围，合成方法参见[92]。

表 4.2 给出了两个算法在不同合成数据集合上的运行时间比较，表中第一列为四个合成数据集合，括号中数字为数据集合参数 n 的取值，其余参数按照表 4.1 取值。实验结果表明 CHID 效率较高，同时，在这 5 个数据集合上，CLIQUE 和 CHID 发现的类的数量一样多，和合成数据集合给定的参数相同。

表 4.1 合成数据集合参数取值范围

$\|D\|$	数据库中的记录数：100,000
n	数据空间的维数：10——50
Cn	类的最大维数：5
Dom(A)	属性的平均值域：50
C	类的数量：5

表 4.2 运行时间比较

	CHID	CLIQUE
D_1(10)	170 sec	360 sec
D_2(20)	1200 sec	1700 sec
D_3(30)	1800 sec	2100 sec
D_4(50)	3900 sec	4800 sec

4.5 本章小结

本章提出的算法 CHID 能够有效地处理高维数据，它首先将 n 维空间划分为

的类存放在 S'' 然后利用函数 Merge 从中发现最大区域作为聚类结果输出。

4.4 实验结果

本文将在高维合成数据集合上比较算法 CLIQUE 和 CHID，实验分别从算法执行效率和准确性两方面验证 CHID 的性能，表 4.1 给出了合成数据的参数取值范围，合成方法参见[92]。

表 4.2 给出了两个算法在不同合成数据集合上的运行时间比较，表中第一列为四个合成数据集合，括号中数字为数据集合参数 n 的取值，其余参数按照表 4.1 取值。实验结果表明 CHID 效率较高，同时，在这 5 个数据集合上，CLIQUE 和 CHID 发现的类的数量一样多，和合成数据集合给定的参数相同。

表 4.1 合成数据集合参数取值范围

$\|D\|$	数据库中的记录数：100,000
n	数据空间的维数：10——50
Cn	类的最大维数：5
Dom(A)	属性的平均值域：50
C	类的数量：5

表 4.2 运行时间比较

	CHID	CLIQUE
D_1(10)	170 sec	360 sec
D_2(20)	1200 sec	1700 sec
D_3(30)	1800 sec	2100 sec
D_4(50)	3900 sec	4800 sec

4.5 本章小结

本章提出的算法 CHID 能够有效地处理高维数据，它首先将 n 维空间划分为

的类存放在 S'' 然后利用函数 Merge 从中发现最大区域作为聚类结果输出。

4.4 实验结果

本文将在高维合成数据集合上比较算法 CLIQUE 和 CHID，实验分别从算法执行效率和准确性两方面验证 CHID 的性能，表 4.1 给出了合成数据的参数取值范围，合成方法参见[92]。

表 4.2 给出了两个算法在不同合成数据集合上的运行时间比较，表中第一列为四个合成数据集合，括号中数字为数据集合参数 n 的取值，其余参数按照表 4.1 取值。实验结果表明 CHID 效率较高，同时，在这 5 个数据集合上，CLIQUE 和 CHID 发现的类的数量一样多，和合成数据集合给定的参数相同。

表 4.1 合成数据集合参数取值范围

$\|D\|$	数据库中的记录数：100,000
n	数据空间的维数：10——50
Cn	类的最大维数：5
Dom(A)	属性的平均值域：50
C	类的数量：5

表 4.2 运行时间比较

	CHID	CLIQUE
D_1(10)	170 sec	360 sec
D_2(20)	1200 sec	1700 sec
D_3(30)	1800 sec	2100 sec
D_4(50)	3900 sec	4800 sec

4.5 本章小结

本章提出的算法 CHID 能够有效地处理高维数据，它首先将 n 维空间划分为

大小相同的单元区域，然后采用双向搜索策略在 n 维空间及其子空间上自动发现数据点密集的单元区域，将密集的单元区域标记为 1，其余的为 0，最后采用逐位相与的方法为这些密集单元区域聚类。CHID 用到的双向搜索能够减少搜索空间，同时聚类过程只用到逐位与和位移两种机器指令，这些都保证了算法的高效率。

5 聚类算法的应用

一个为实际应用而设计的数据开采系统通常会综合利用、协同使用多种开采模型，例如 ARCS 算法[113]为满足聚类关联规则的需求，将聚类开采模型用于已被发现的关联规则上，向用户提供结构更为简洁的知识表达方式；ARHP 算法[105]首先利用关联规则开采模型发现文档集合中的频繁项目，然后使用聚类算法在这些频繁项目构成的超图上对文档分类；其它综合了多种开采模型的分析方法还有基于关联规则超图的聚类[114]、以及论文第四章给出的高维数据的子空间聚类[101]等。本章将详细讨论聚类算法在关联规则发现和层次型半结构化数据模式发现上的应用。

5.1 关联规则发现

关联规则发现也是数据开采的一个非常重要的子领域，最初由 Rakesh Agrawal 等人在 1993 年提出[29]用来开采数据间隐含的联系。给定一个交易数据库，数据库中的每一条记录都是一条交易，所谓交易就是由一组项目构成的集合，关联规则用来表达交易中项目之间的联系，例如："购买啤酒的用户中有 30%也购买了尿布，而在所有的顾客中有 2%同时购买了这两样东西"，在这条规则中，30%称作规则的置信度（*confidence*）、2%称作规则的支持度（*support*），关联规则开采的目的就是要找出数据集合中满足最小支持度（*minsup*）和最小置信度（*minconf*）的所有规则，这里 *minsup* 和 *minconf* 是用户预先定义的阈值。

大容量的数据集合能够保证开采出质量较高的关联规则，但是算法效率与数据集合的规模成反比，数据集合规模越大，则算法效率越低。大多数关联规则算法都需要多次扫描数据库，而频繁的 I/O 操作对于大容量数据集在时间耗费上是不能容忍的，为解决以上问题，本文提出算法 MARC[121](Mining Association Rules using Clustering)，它将聚类技术运用到关联规则发现领域，该算法只需对数据库扫描一遍，MARC 算法首先对交易数据库聚类，将类似的交易聚集到同一个类中，然后在聚类结果上而不是在原始数据集上发现关联规则，这样大大减少了关联规则开采算法需要处理的数据量，从而提高开采效率。采用这种方法，一个需要解决的问题是，如何能够保证聚类所耗费的时间不影响整个算法的效率，本文第三章提出的 CCDCS 算法是一种高效的针对分类属性数据的方法，交易数据库中的项目都属于分类属性的数据，因此 MARC 算法的聚类部分采用基于 CCDCS 的算法，该部分根据关联规则的特性简化了交易聚类之间的相似度等计算，提高了聚

5.1.4.6 数据输入顺序影响

这项实验用来测试数据输入顺序不同对 MARC 算法准确性的影响。取四个交易量不等的合成数据集合作为实验数据，分别记作 D_1、D_2、D_3 和 D_4，然后在每个 D_i（$i=1,2,3,4$）上，使用不同的数据输入顺序运行 MARC 算法五次，最后计算每次得到关联规则的误差率评估结果是否稳定，如果误差率稳定在一个较小值，则认为数据输入顺序对算法准确性的影响不大。四个实验数据集合根据表 5.8 的参数设置生成（参数|D|除外），D_1、D_2、D_3 和 D_4 的交易量|D|分别设为 1k、10k、100k 和 200k。

表 5.9 不同数据输入顺序得到的 e_R(MARC)

顺序号	D_1	D_2	D_3	D_4
1	0.006	0.008	0.009	0.008
2	0.014	0.009	0.010	0.009
3	0.012	0.010	0.009	0.008
4	0.011	0.008	0.009	0.009
5	0.008	0.007	0.008	0.008

表 5.9 分别列出了四个数据集合上采用不同的数据输入顺序时，MARC 算法得到的关联规则的误差率，实验结果显示数据集合越大，数据输入顺序对于得到的关联规则的稳定性影响越小。

5.2 半结构化数据的结构发现

Web 数据的不断增长和异构数据集成的应用，导致了大量半结构化数据的产生，这些数据的特点是其结构隐含、不规则或不完整[44][126]。如一个有关商品信息的 web 页面集合，虽然每一个页面描述的商品不同，但是它们都包含相似的信息（货号、单价、颜色、外型、规格、单位、产地、性能），这一信息框架隐含在数据中，需要经过分析工具（如文本分类器等）得到，由于没有严格的结构限制，有的页面会缺少某些信息，而其它一些页面可能多出几条，而且每条信息的表达方式也可能不尽相同，如产品性能：有的用表格形式表达，而有的则用一段文字描述。从关系数据库的角度来看，数据结构不规整的原因是缺少预定义的、固定的、独立于数据的模式框架，半结构化数据正是这样，作为一种自描述数据，数据中虽然存在模式，但是模式与数据间的界限模糊，导致新数据的加入没有预先

定义的模式约束，所以这种由数据描述的模式会随着数据的不断增多而扩展。

在 WWW 环境中，大多数 web 查询使用情报检索技术，计算文档内容与查询关键字的相似度，并返回相似度较大的文档集合[127][128]。这种方法得到的结果数量很多，但其中真正为用户所需的信息却很少，因此人们期望 web 数据与数据库数据一样，能够利用数据结构对其进行有效地检索[129]。对于某些由数据库的数据集成而形成的半结构化数据，可以采用强制施加模式的办法使其具有结构，因为这些数据本身就具有一定的规则结构，但是对于结构松散的 web 数据，这样的方法会产生大量空值而降低查询效率，同时还会丢失大量信息。Nestorov 等人提出表示对象 RO（Representative Objects）的概念[130]用来描述层次型的半结构化数据，其中 FRO（Full RO）描述全局数据的结构、*k*-RO 用来描述局部结构、而能够提供最有效率的模式查询的对象是最小 FROs，为获取最小 FROs，该文采用的方法是将 OEM（Object-Exchange Model）图转换为非确定有穷自动机 NFA（Nondeterministic finite automaton）的状态转移图，然后将之确定化及最小化得到最小 FRO；最小 FRO 生成的模式有效地表达了数据对象间的关系，但是，Svetlozar Nestorov 等人的目的只是从单个的 OEM 图中抽取模式，无法发现 OEM 模型中存在的多个对象的共有结构，为此，Ke Wang 等人提出根据兴趣度（最小支持度和最小置信度）发现 OEM 模型中满足条件的子对象的典型结构（typical structure）[131][132]，然而，对于大容量的半结构化数据，发现算法的效率较低，并且每次兴趣度阈值的调整以及新数据的加入都需要在完整数据源上重新执行算法。

与 Ke Wang 等人一样，本文的目的也是从 OEM 模型表达的半结构化层次数据中发现满足用户兴趣度的共有结构，不同的是，为提高效率，结构发现利用了聚类技术将具有相似结构的对象聚到同一个类。聚类技术的使用基于两点理由：① 结构发现是针对那些包含相同信息的对象而言，对于描述不同事物的对象，发现它们的共有结构是无意义的，如不会去寻找描述电影的对象和描述足球联赛的对象之间的共有结构，所以没有必要在整个数据集合上进行结构发现；② 根据半结构化数据的特征，描述同一类事物的对象结构虽然不完全相同，但也会非常近似。所以按照结构相似聚类，能够保证一个类中的所有对象都是描述同一种事物的，也就是说一个类包含了描述某一类事物的所有结构。合并这些结构，将得到一个用来表达描述某一类事物的完全结构，然后用户可以根据自己的兴趣度需求，选择是否继续在这个完全结构上发现一个更为简明的结构或者子结构。

本文提出的半结构化层次型数据的结构发现方法基于增量式的聚类算法，在

聚类得到的 FS 上发现满足兴趣度的结构速度会提高很多，因为 FS 的数据量大大小于原来的数据集合；保存聚类结果，这样当新数据加入时只需对聚类结果做少量修改，不需从头执行算法。实验也证实了与 Ke Wang 等人提出的方法相比，聚类技术的引用大大提高了结构发现效率，同时不影响结果质量。

5.2.1 OEM 模型

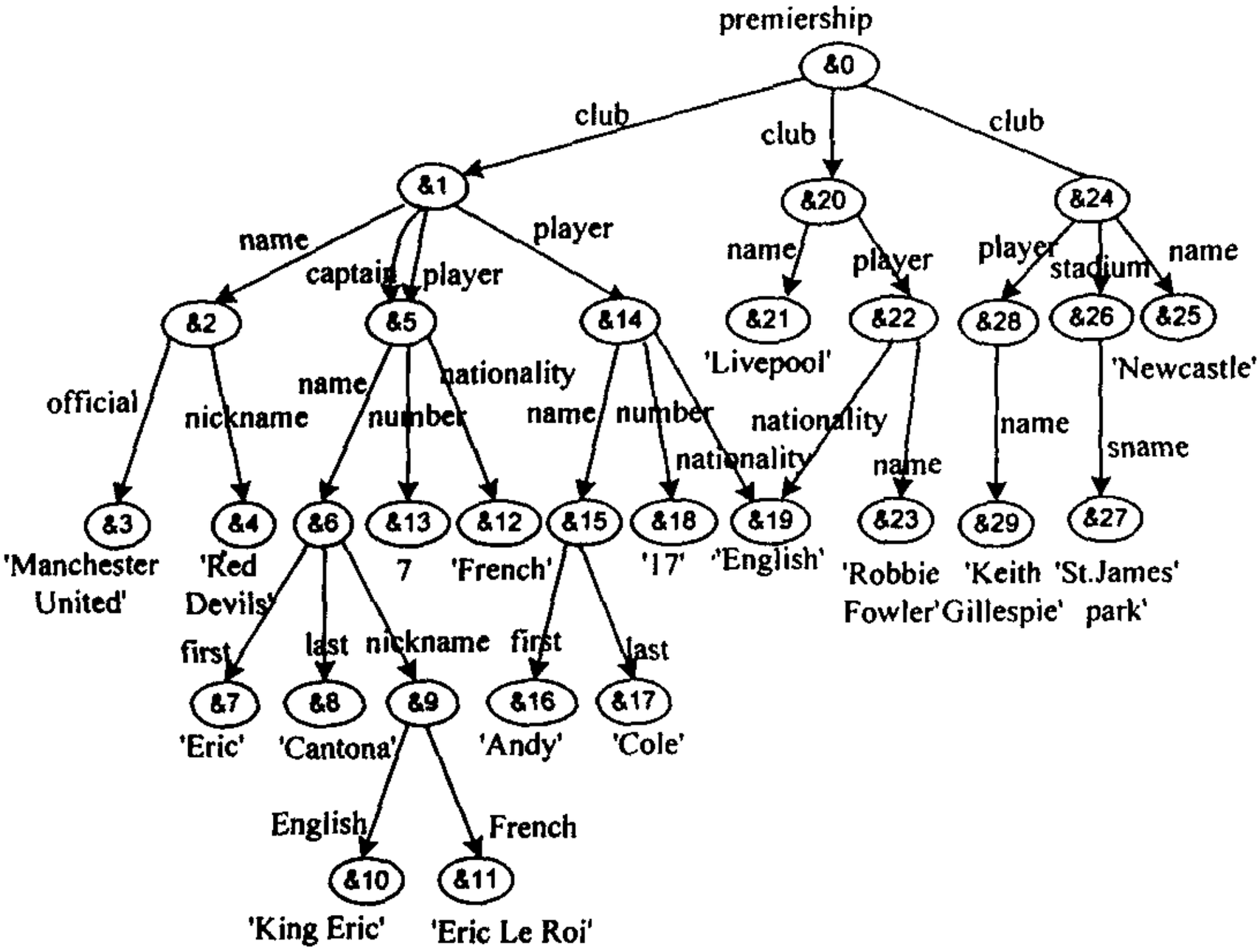

图 5.7 OEM 模型

OEM 是自描述对象模型，专为表达半结构化数据而设计。它最初的目的是为异构数据源间的数据交换提供高度灵活的转换工具[133]。不同应用中的 OEM 模型大多在原模型的基础上作了一些小的改动，本文采用的模型与文献[77]使用的相同。

OEM 模型可以看作一个图，数据对象用节点表示，对象间的层次关系则由边上的标签（label）体现。每一个对象包含对象标识（identifier）和对象值（value），identifier 是对象的唯一标识，value 则有两种表示形式：如果以对象 O 为顶点的

出度 OD(O) = 0，即对象 O 没有子对象，则 O 为原子对象，value(O)为原子值，其值的类型为字符串或整型等原子类型；如果 OD(O) = $n>0$，则 value(O) = {<l_1,id_1>, ...,<l_n,id_n>}，其中 l_i ($1\leqslant i\leqslant n$) 是从顶点 O 引出的第 i 条边上的 label，id_i 是第 i 条边连接对象的 identifier，该对象是 O 的子对象。我们借用文献[130]中的 OEM 模型来图例说明本节的概念，图 5.7 忽略了原图中的环，因为本文提出的算法不考虑环的问题。

例 5.1 对象 &2，OD(&2) = 2；identifier(&2) = 2；value(&2) = {<official,3>,<nickname,4>}；对象&3，OD(&3) = 0；identifier(&3) = 3；value(&3) = 'Manchester United'。

简单路径表达式 pe（simple path expression）是以点为间隔的 label 序列，记作 $pe = l_1.l_2...l_n$，n 为 pe 的长度；在本文中，l_i 和 l_j 为不同层的 label（$i, j = 1, 2, ..., n$ 且 $i \neq j$），如果有 $i+1 = j$，则一定有对象 O 为 l_i 所在边的终端点并且同时为 l_j 所在边的起始点。$pe = l_1.l_2...l_n$，$pe' = l_i.l_{i+1}...l_{i+k}$，$k$ 为 pe' 的长度，如果 $i\geqslant 1$ 并且 $i+k\leqslant n$，我们称 pe' 是 pe 的有序子集，pe 有序包含 pe'。给出简单路径表达式集合 $\{pe_1、pe_2、...、pe_n\}$，对于某个 pe_i，如果在集合中不存在任何 pe_j（$i, j = 1, 2, ..., n$ 且 $i\neq j$）有序包含 pe_i，则称 pe_i 是该集合中的最长 pe，记作 mpe（maximal pe）。

例 5.2 pe = club.player.name 为一条长度为 3 的简单路径表达式；{pe_1 = club.name.official，pe_2 = club.name}；则 pe_1 为该集合中的 mpe。

数据路径 dp（data path）是以逗号为间隔的以 label、identifier 交替出现的序列，它是简单路径表达式的实例，记作 $dp = O_0, l_1, O_1, l_2, ..., l_n, O_n$，$i = 1, ..., n$，$<l_i,O_i>\in$ value(O_{i-1})，n 为数据路径 dp 的长度。图 5.7 中有多条数据路径是例 5.2 的实例，见例 5.3：

例 5.3 dp_1 = 0, club, 1, player, 5, name, 6；dp_2 = 0, club, 1, player, 14, name, 15；

dp_3 = 0, club, 20, player, 22, name, 23；dp_4 = 0, club, 24, player, 28, name, 29。

5.2.2 层次对象的相似度

本文用于发现结构的聚类采用了与第三章类似的方法，采用的数据结构也与聚类树 CT 相似，不同的是，结构包含了层次信息，算法采用对象结构树 STO（Structure Tree of Objects）表达这种层次结构，因此 CT 树中的每个节点不再是三元组（N, I, S）的表达方式（有关详细内容见第三章），而用一棵对象树 STO 表

达，类相似度也要随之重新定义。虽然数据结构内部发生了以上改变，但基于该数据结构的聚类方法没有发生变化，所以本章仍将用于聚类的数据结构称为聚类树 CT。本小节首先描述 CT 的节点表达 STO，然后给出计算层次对象相似度的方法。

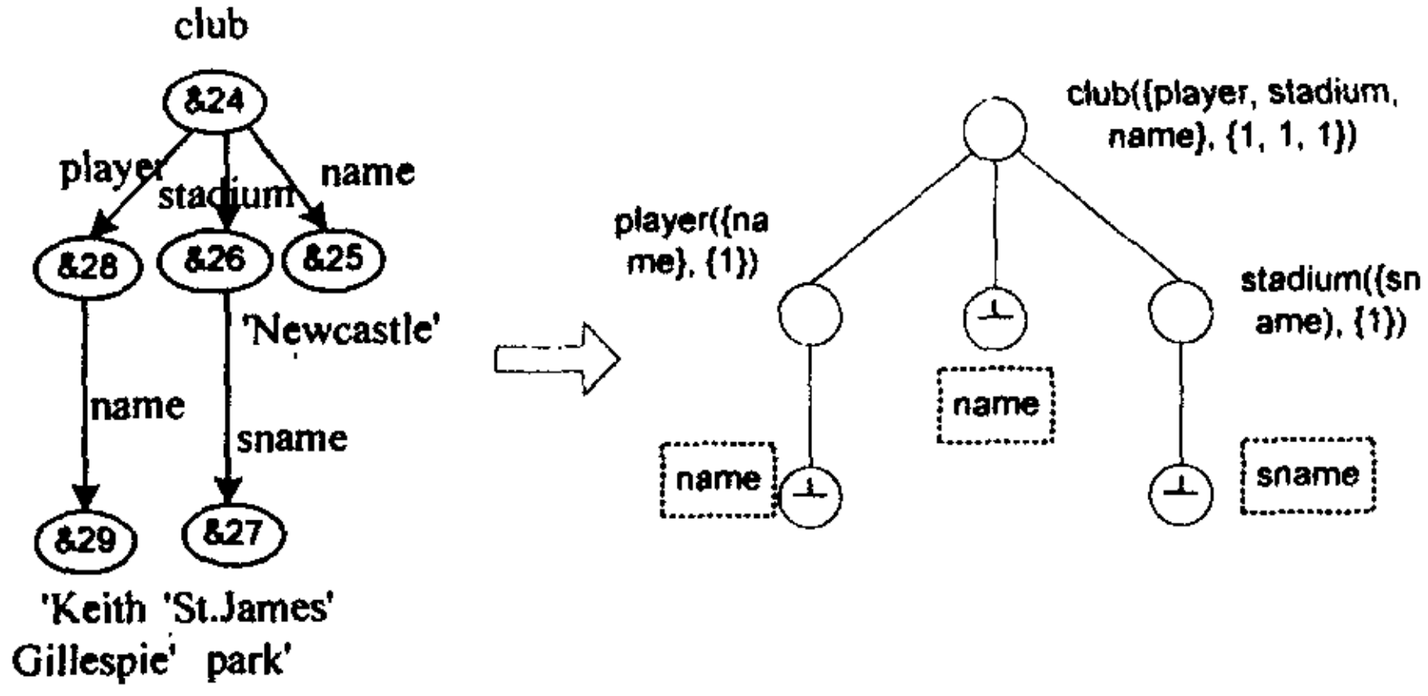

图 5.8 用 OEM 图描述 STO 树

先简单说明 CT 的结构，聚类树 CT 中的非叶子节点具有如下形式：(STO, *pParent*, $pChild_i$)，其中 *pParent* 指向该节点的父节点，$pChild_i$ 指向其第 i 个子节点，$i = \{1, 2, \ldots, k\}$，k 为子节点数目，STO 为对象结构树；CT 树的叶子节点为聚类结果，每个叶子节点是一个类，用 STO 表示，与非叶子节点相同，叶子节点有一个指向父节点的指针 *pParent*，由于不存在子节点，所以没有指向子节点的指针 $pChild_i$，但是叶子节点包含一个指向其右节点的指针 *pNext*，这个指针将 CT 中所有叶子节点链接起来。

STO 是用来描述对象层次结构的树，用 STO.N 表示树中包含的结构数量，STO 的非叶子节点 C 用二元组 (I, S) 表示，C 除了代表对象名称外，也表示当前的层次关系：$I = \{i_1, i_2, \ldots, i_n\}$ 为 C 的子节点，即 C 的下一层的所有层次关系的集合，记作 $C.I$；$S = \{s_1, s_2, \ldots, s_n\}$，其中 s_j 为 STO 中符合层次关系 i_j 的结构数量，$1 \leqslant j \leqslant n$，记作 $C.S$；另外每个叶子节点有指针 $pChild_1, pChild_2, \ldots, pChild_n$ 分别指向 C 的下一层关系 $i_1, i_2, \ldots, i_n$；由于叶子节点表示的是原子对象的结构，其下不再有子层次，就用 C 来表示。STO 上的简单路径表达与 OEM 图类似，$pe = C_1, C_2, \ldots, C_n$，$C_i$（$i = 1, 2, \ldots, n$）为 STO 节点 C_i 的层次关系，且 C_{i+1} 为 C_i 的子节点。对于某个 pe_i，在集合中不存在任何 pe_j（$i,j = 1,2, \ldots,n$ 且 $i \neq j$）有序包含 pe_i，称 pe_i 是该集合中的最长 pe，也记作 mpe。

根据 OEM 图给出构造 STO 的描述，给定描述某个对象的 OEM 图，由于没

有环路，可以将它看成一棵树；初始 STO 为一个只有根节点(C_{root})的树，且指针 P 指向根节点，C_{root} 为 OEM 根节点的层次关系，如图 5.7 中，根节点&0 的层次关系为“premiership”；假设 OEM 中有 n 条 mpe，分别为 $mpe_1, mpe_2, \ldots, mpe_n$，$mpe_i = \{l_{i1}, l_{i2}, \ldots, l_{ik}\}$ ($1 \leqslant i \leqslant n$, k 为 mpe_i 的长度)，逐条读入 mpe_i ($i = 1, 2, \ldots, n$)，插入到 STO 中，每读入一条新的 mpe，就将指针 P 指向 STO 的根节点，插入过程如下，读入 mpe_i 中的 l_{ij} ($1 \leqslant j \leqslant n$)，假定指针 P 指向 STO 中的节点 C (I, S)，如果有 $i_x = l_{ij}$ ($i_x \in I$)，则一定存在 C 的子节点 i_x，移动指针 P 令它指向 i_x，同时修改 $s_x \in C.S$，令 s_x 增加 1；如果不存在 $i_x = l_{ij}$ ($i_x \in I$)，则为当前节点 C 创建新的子节点，并移动指针 P 到新创建的子节点，同时修改 $C.I$ 和 $C.S$，令 $C.I = C.I \cup l_{ij}$。图 5.8 给出了 STO 的构造，给定描述 *club* 对象的 OEM 图，得到三条 mpe，分别是{player, name}、{stadium, sname}和{name}，逐条插入到只有根节点 *club* 的树中得到如图所示的 STO。

当新的对象以 OEM 描述的形式插入到聚类树 CT 中，首先需要获取该对象的层次结构信息，然后从根节点起，逐层与 CT 中的节点进行相似度比较，以选择下一个要到达的节点，聚类树中的这种比较实际就是两个 STO 的相似度比较。给定两个 STO，分别记作 S_1 和 S_2，S_1 的高度为 h_1，S_2 的高度为 h_2，令 $h = \max(h_1, h_2)$；S_1 和 S_2 的相似度计算如下，分别计算 S_1 和 S_2 同层所有节点对的相似度后，得到每一层的相似度，再根据这些层相似度计算两棵树的相似度；在 STO 中，不同的层次对于树间相似度的重要性也不相同，层次越往后，重要性越低，为此，在计算树间相似度时要对不同层次的相似度分配权值 w_i ($i = 1, 2, \ldots, h$)，$0 \leqslant w_i \leqslant 1$，且 $w_i > w_{i+1}$。首先考虑两个节点的相似度，对 S_1 中的任一节点 a 和 S_2 中的任一节点 b，定义其相似度为

$$\mathrm{sim}(a, b) = \begin{cases} -1 & \text{if } a \neq b \\ 2 \mid a.I \cap b.I \mid / (\mid a.I \mid + \mid b.I \mid) & \text{if } a = b \end{cases} \tag{5.15}$$

公式(5.15)中当 a 的层次关系等于 b 时，对象 a 和 b 的相似度定义来自第三章两个分类对象相似度的定义（见第三章公式(3.5)）。假定 S_1 的第 i 层包含 m_{1i} 个节点，S_2 的第 i 层包含 m_{2i} 个节点，则 S_1 和 S_2 第 i 层的相似度为：

$$\mathrm{sim}_i(S_1, S_2) = \begin{cases} -1 & \text{if } m_{1i} = 0 \text{ or } m_{2i} = 0 \\ \dfrac{\sum_{j=1}^{m_{1i}} \sum_{k=1}^{m_{2i}} (\mathrm{sim}(a_j, b_k))}{m_{1i} m_{2i}} & \text{if } m_{1i} \neq 0 \text{ and } m_{2i} \neq 0 \end{cases} \tag{5.16}$$

公式(5.16)中 a_j 为 S_1 的第 i 层节点 ($j = 1, 2, \ldots, m_{1i}$)，b_k 为 S_2 的第 i 层节点（k

$= 1, 2, \ldots, m_{2i}$)。将公式(5.16)规格化，并分配权值，则 S_1 和 S_2 第 i 层的相似度为：

$$\mathrm{sim}_i^*(s_1, s_2) = w_i \frac{\mathrm{sim}_i(s_1, s_2) + 1}{2} \tag{5.17}$$

由此，得到 S_1 和 S_2 的相似度：

$$\mathrm{sim}(s_1, s_2) = \sum_{i=1}^{h} \mathrm{sim}_i^*(s_1, s_2) / \sum_{i=1}^{h} w_i \tag{5.18}$$

$\mathrm{sim}(S_1, S_2)$的值域为[0, 1]，该值越大则表示 S_1 和 S_2 越相似，反之，则越不相似，$\mathrm{sim}(S_1, S_2)$等于 1 表示 S_1 和 S_2 完全相同，等于 0 表示 S_1 和 S_2 完全不同。

在进行相似度计算时，如果发现第 i 层的相似度 $\mathrm{sim}_i(S_1, S_2) = -1$，则第 i 层以后所有层的相似度都为-1，原因是，如果第 i 层不相似，表明从第 i 层起，描述的不是同一类对象，因此其后所有层的相似度计算是没有意义的。下面讨论权值分配，越靠近根节点的层次分配的权值应当越大，因此，层数越小，权值越大，按下列方法取值，第一层的权值 $w_1 = 1$，第 i 层的权值 $w_i = w_{i-1} * \beta$，其中 $\beta = 1/c^{\alpha}$（$C>1$，$\alpha>0$）。

5.2.3 STO 合并

聚类过程实际就是 CT 的构造过程，而 CT 的构造过程与本文第三章描述的 CCDCS 算法相同，只是对象相似度的计算和节点修改（即合并满足最小相似度的对象）不同，相似度的定义已经在上节给出，所以本节只给出节点修改部分，有关 CT 构造过程的详情见第三章 3.3.2.3。

令读入对象的 STO 为 S_1，存在两种情况需要修改节点，一种是到达非叶子节点时，需要将 S_1 并入该节点，另一种当 S_1 到达叶子节点时，它们之间的相似度大于阈值最小相似度 *minsim* 时，需要将 S_1 合并到该叶子节点。假设合并 S_1 的节点 STO 为 S_2，S_1 的高度为 h_1，S_2 的高度为 h_2，令 $h = \max(h_1, h_2)$；合并后的节点记作 S_3：① $S_3.N = S_1.N + S_2.N$；② 从 STO 的第 1 层起，逐层合并 S_1 和 S_2 中名称相同的节点，即相同层次关系的节点，假设 n_1 和 n_2 分别为 S_1 和 S_2 中第 i 层的节点，且 $n_1 = n_2$，则 $S_3.I = S_2.I \cup S_1.I$，然后令 $S_3.S.s_j = S_2.\mathrm{num}(i_j) + S_1.\mathrm{num}(i_j)$，其中 $i_j \in S_3.I$，$j = \{1, 2, \ldots, S_3.I\}$，$S_2.\mathrm{num}(i_j)$和 $S_1.\mathrm{num}(i_j)$分别代表 i_j 在 S_2 和 S_1 中的数量，可以通过 $S_2.S$ 和 $S_1.S$ 获得，若 $S_2.S$ 或 $S_1.S$ 中不包含 i_j 的数量，则 $S_2.\mathrm{num}(i_j)$或 $S_1.\mathrm{num}(i_j)$的值为 0。

5.2.4 结构发现

STO 中 *pe* 的支持度记作 sup(*pe*)，*pe* 中一定有 $C_n = i_j$ ($i_j \in C_{n-1}.I$)，则 sup(*pe*) $= s_j$ ($s_j \in C_{n-1}.S$)，当 sup(*pe*)$\geq$*minsup* 时，*pe* 为频繁 *pe*，记作 *fpe*，否则为非频繁 *pe*。

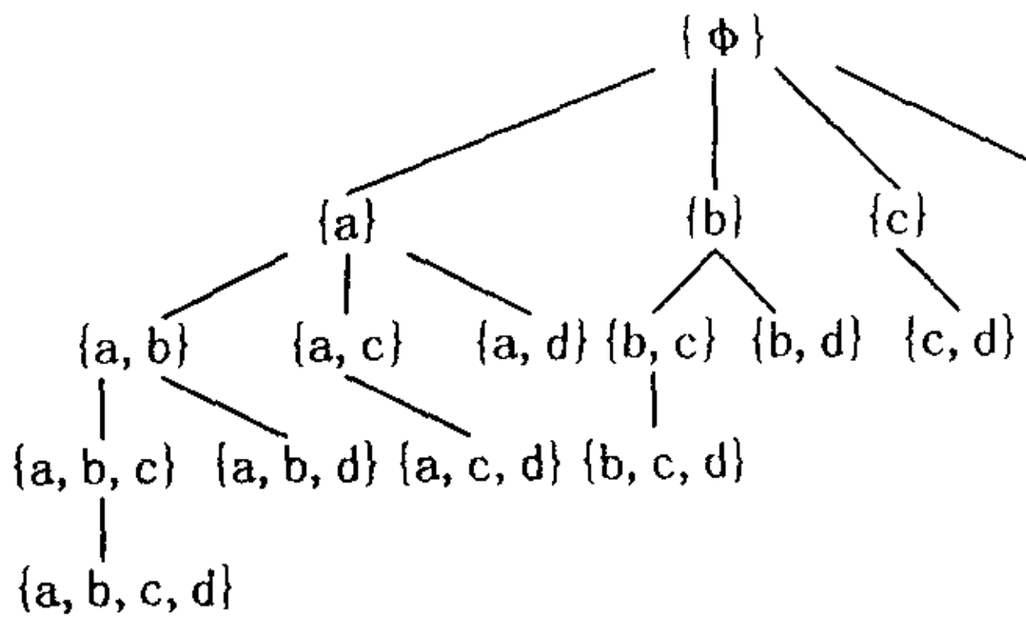

图 5.9 集合枚举树

结构发现的目的是在聚类树的每个叶子节点上发现满足兴趣度的结构，即从叶子节点的 STO 上发现满足兴趣度的结构。取 STO 上所有从根节点到叶子节点的 *pe*，即 *mpe* 作为结构发现的数据集合，这个数据集合可以看成一个交易集合 $mpe_1, mpe_2, \ldots, mpe_n$，每一条简单路径表达看作是一条交易，结构发现就是从这个集合中发现频繁的 *mpe*，记作 *fmpe*，也就相当于从交易数据库中发现最大频繁项目集。令 $T = mpe_1 \cup mpe_2 \cup \ldots \cup mpe_n$，算法通过自顶向下的生成方法构造集合枚举树[134]，以 *T* 为顶点，逐层生成候选集 *fmpecs*，通过剪枝快速生成 *fmpe*。图 5.9 给出了交易集合 *D* 的集合枚举树，已知 *T* = {a, b, c, d}，集合枚举树方便地描述出 *D* 中所有可能的交易组合（项目集），频繁项目集发现就是从集合枚举树中找到支持度大于给定阈值的项目集。

与传统的交易集合不同，*mpe* 具有层次特征，即 *mpe* 中的“项目”有序排列，因此，算法扩展了关联规则开采中发现频繁项目集的算法[135][136]，利用层次特征提高候选集 *fmpecs* 的生成和剪枝过程的效率。

将 *T* 按照层次排序，$T = \{C_{11}, C_{12}, \ldots; C_{21}, C_{22}, \ldots; \ldots; C_{n1}, C_{n2}, \ldots\}$，*h* 表示 STO 的层数，$C_{ij}$ 表示第 *i* 层的第 *j* 个节点。以 *T* 作为集合枚举树的顶点，逐层构造 *T* 的子集 *fmpecs*，即 *fmpe* 的候选集；然后对候选集 *fmpecs* 中的每条 *pe* 剪枝，剪枝的目的是尽早删除 *pe* 中的非频繁 *pe*，减少支持度的计算；接着从剪枝后的 *fmpecs* 中寻找频繁 *pe*，如果有，则删除 *fmpecs* 中该频繁 *pe* 的子集，因为采用了自顶向

下的方法最先生成的是 *mpe*，因此是 *fmpe*。① 候选项目集的生成：第 *k*+1 次的 *fmpecs* 由第 *k* 次的 *fmpecs* 生成，对任意 $pe \in fmpecs$，*pe* 具有如下形式：$pe = p_1 \cup p_2 \cup \ldots \cup p_n$，其中 p_i 为 *pe* 第 *i* 层所有元素组成的集合，*n* 为 *pe* 潜在的最大层数，当算法在进行第 *k*+1 次搜索时，则表明 *pe* 的前 *k* 层已经被搜索过，此时，对任意 $i < k+1$ 都有 $|p_i| = 1$。由于层次关系，STO 中同层的节点不会出现在一条简单路径中，所以在生成下一轮的候选集 *fmpecs* 时，直接从 *pe* 的第 *k*+1 层中取出一个元素来生成，因而对任意一个 $pe \in fmpecs$，生成 *k*+1 次 *pe*'的方法如下：$pe' = p_1 \cup p_2 \cup \ldots \cup p_k \cup \{a\} \cup p_{k+2} \cup \ldots \cup p_n$，$a \in p_{k+1}$，如果元素 a 为空，表示其后不可能跟有路径，所以这时应该删除 *k*+1 后的所有元素，即 $pe' = p_1 \cup p_2 \cup \ldots \cup p_k \cup \{a\}$。对所有的 $pe \in fmpecs$ 进行上述操作，生成下一轮的 *fmpecs*。② 剪枝：剪枝方法主要基于层次特性和以下性质：如果 *pe* 是非频繁的，那么任何包含 *pe* 的简单路径表达式都是非频繁的。对任意 $pe \in fmpecs$，$pe = p_1 \cup p_2 \cup \ldots \cup p_n$，其中 p_i 为 *pe* 第 *i* 层所有元素组成的集合，*n* 为 *pe* 潜在的最大层数，由于模式具有分层的特点，因而，在第 *k* 次搜索过程中，我们只需要计算 *pe* 中前 *k*+1 层元素所组成的路径表达式 $pe = p_1 \cup p_2 \cup \ldots \cup p_k \cup \{a\}$的支持度，$a \in p_{k+1}$，如果 $\sup(p_1 \cup p_2 \cup \ldots \cup p_k \cup \{a\}) < minsup$，则把 a 从 p_{k+1} 中删除，如果 $|p_{k+1}| = 1$ 而且其唯一的元素为空，则表明 *pe* 已到了最底端，其后不可能再有路径，此时要把 *pe* 中 p_{k+1} 以后的所有元素删除。结构发现的最后一个步骤是利用生成的 *fmpe* 构造一个满足兴趣度的 STO。

5.2.5 算法描述

算法中的聚类过程基本同第三章聚类树的构造算法（3.3.2.4），只是相似度的计算和节点合并对象的计算不同，所以这里仅给出从 STO 中发现满足兴趣度的结构算法 structure_discovery。算法中 delete(*fmpecs*, *fmpecs*[*i*])表示从 *fmpecs* 中删除元素 *fmpecs*[*i*]，prune()和 gen_mmpecs()分别为剪枝函数和 *fmpecs* 生成函数。

```
structure_discovery(T, D, minsup)
1.  k = 0, fmpecs = {T}, fmpe = {};
2.  while fmpecs ≠ {} do {
3.      prune(ref fmpecs, k, minsup, D);
4.      for i = 1 to |fmpecs| do {
5.          if (|fmpecs[i]|≤k) then {
6.              fmpe = fmpe ∪ fmpecs[i];
7.          delete(fmpecs, fmpecs[i]);  }
8.      else
9.          if (sup(fmpecs[i], D) > minsup) then {
```

```
10.            fmpe = fmpe ∪ fmpecs[i];
11.            delete(fmpecs, fmpecs[i]); }
12.    }
13.    fmpecs = gen_fmpecs(fmpecs, k);
14.    k = k + 1;
15. }
16. 删除 fmpe 中的非 mpe;
17. return fmpe;
```

```
gen_fmpecs(fmpecs, k)
1.  cs = {};
2.  for i = 1 to |fmpecs| do {
3.     m_k = {left(fmpecs[i], k)};                    //取集合 fmpecs[i]的前 k 个元素
4.     m_{k+1} = {a | a∈fmpecs[i], level(a) = k+1}; //将 fmpecs[i]中第 k+1 层的
    元素放入 m_{k+1}
5.     m_h = fmpecs[i]–m_k–m_{k+1};
6.     for j = 1 to |m_{k+1}| do {
7.        if m_{k+1}[j] = null then
8.           cs = cs ∪ m_{k+1}[j];
9.        else
10.          m = m_k ∪ m_{k+1}[j] ∪ m_h;
11.          cs = cs ∪ {m};
12.    }
13. }
14. return cs;
```

```
prune(ref fmpecs, k, minsup, D)
1.  for i = 1 to |fmpecs| do {
2.     m_k = {left(fmpecs[i], k)};                    //取集合 fmpecs[i]的前 k 个元素
3.     m_{k+1} = {a | a∈fmpecs[i], level(a) = k+1};  //将 fmpecs[i]中第 k+1 层
    的元素放入 m_{k+1}
4.     for j = 1 to |m_{k+1}| do {
5.        if (sup(m_k ∪ m_{k+1}[j], D)<minsup then
6.           delete(fmpecs[i], m_{k+1}[j]);
7.           m_{k+1} = {a | a∈fmpecs[i], level(a) = k+1}
8.        if (|m_{k+1}| = 0) then
9.           fmpecs[i] = m_k;
10.       else
```

```
11.              if (|m_{k+1}| = 1 and m_{k+1}[1] = null) then
12.                fmpecs[i] = m_k ∪ m_{k+1}[1];
13.    }
14. }
```

5.2.6 实验分析

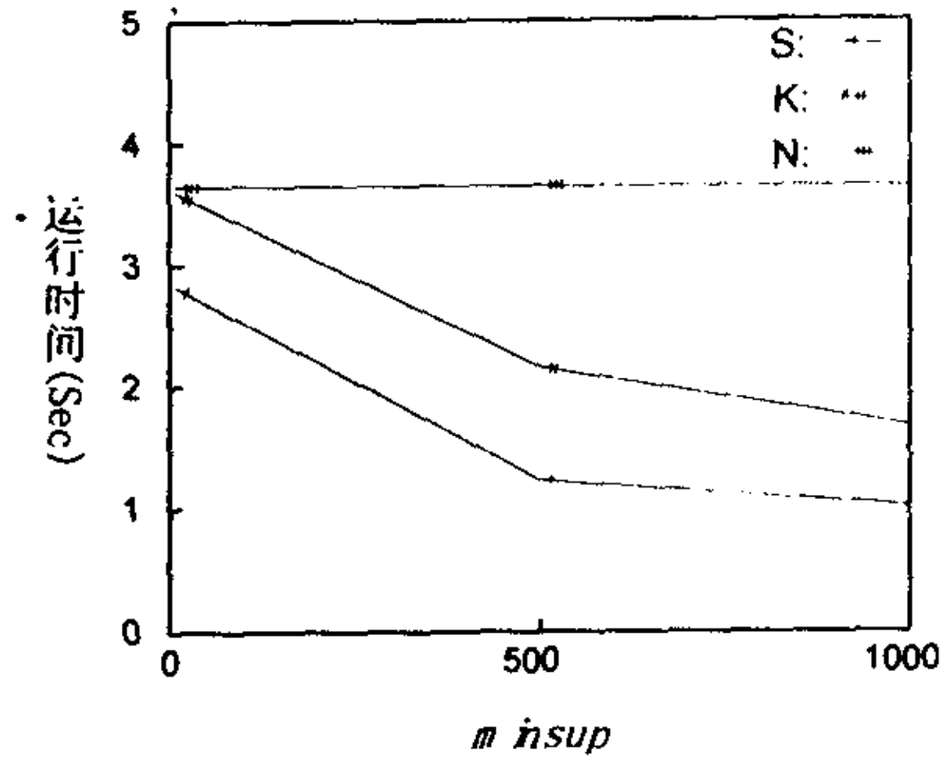

图 5.10 运行时间比较

我们选用 web 网站中文电影资料库（www.dianying.com）作为实验所需的半结构化数据，该网站以片名的首字母为索引字，总共包含 5328 部影片，每部影片是一个 HTML 页面。其中，只有单个值的项目划分为原子对象，如影片类别、地区等；包含链接、表格等的项目划分为复杂对象，如职员。实验比较算法 structure_discovery、Ke Wang 等人提出的算法[131]和 Nestorov 等提出的算法[130]，为方便描述，将这三个算法分别记作算法 S、K 和 N。计算时间的比较结果如图 5.10 所示，由于算法 N 没有最小支持度参数，所以它的运行时间为一条水平的直线，另两个算法支持度越大，计算时间越快，且算法 S 较算法 K 效率要高，这是因为聚类技术缩小了数据集合的原因。由算法 S 发现的结构和算法 K 得到的结构基本相同。

5.3 本章小结

为提高数据开采的效率，本章将聚类技术分别用于关联规则和半结构化、层次数据的结构发现上。现有的关联规则算法大都需要多次扫描数据库，而频繁的 I/O 操作会影响算法在大容量数据集合上的效率，为解决以上问题，本章提出的

```
11.              if (|m_{k+1}| = 1 and m_{k+1}[1] = null) then
12.                fmpecs[i] = m_k ∪ m_{k+1}[1];
13.    }
14. }
```

5.2.6 实验分析

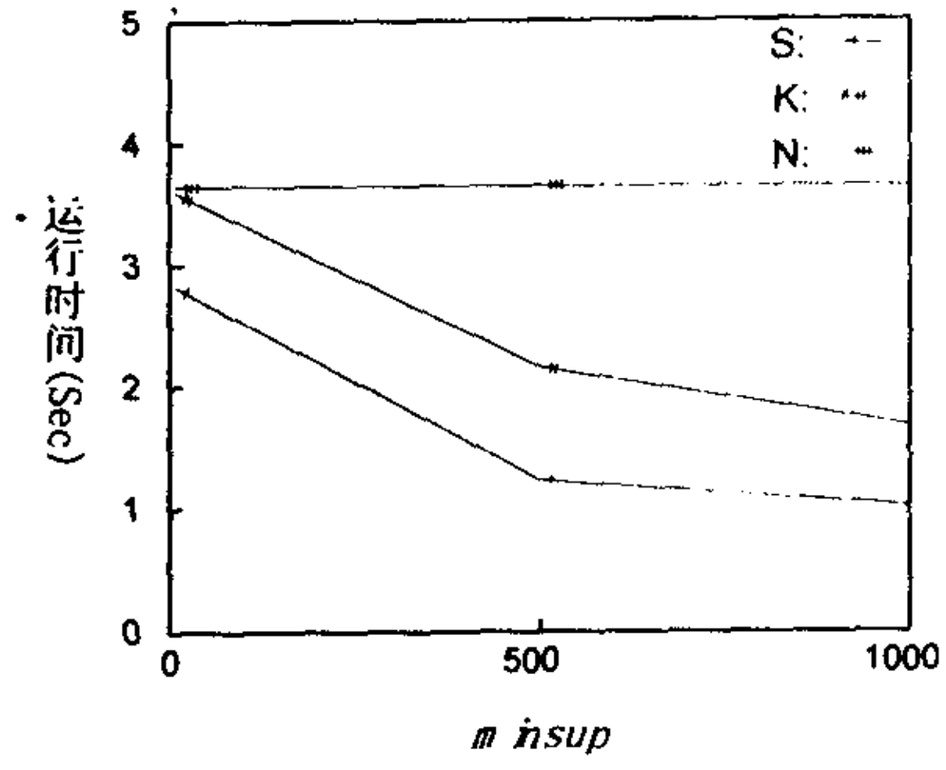

图 5.10 运行时间比较

我们选用 web 网站中文电影资料库（www.dianying.com）作为实验所需的半结构化数据，该网站以片名的首字母为索引字，总共包含 5328 部影片，每部影片是一个 HTML 页面。其中，只有单个值的项目划分为原子对象，如影片类别、地区等；包含链接、表格等的项目划分为复杂对象，如职员。实验比较算法 structure_discovery、Ke Wang 等人提出的算法[131]和 Nestorov 等提出的算法[130]，为方便描述，将这三个算法分别记作算法 S、K 和 N。计算时间的比较结果如图 5.10 所示，由于算法 N 没有最小支持度参数，所以它的运行时间为一条水平的直线，另两个算法支持度越大，计算时间越快，且算法 S 较算法 K 效率要高，这是因为聚类技术缩小了数据集合的原因。由算法 S 发现的结构和算法 K 得到的结构基本相同。

5.3 本章小结

为提高数据开采的效率，本章将聚类技术分别用于关联规则和半结构化、层次数据的结构发现上。现有的关联规则算法大都需要多次扫描数据库，而频繁的 I/O 操作会影响算法在大容量数据集合上的效率，为解决以上问题，本章提出的

算法 MARC（Mining Association Rules using Clustering）利用聚类技术通过算法交换简化交易数据库，从而减少开采算法需要处理的数据量以提高算法效率，MARC只需对数据集合扫描一遍，同时实验结果表明压缩技术对关联规则发现结果的正确性影响不大。异构数据库的集成和 Web 数据促使半结构化数据的形成，半结构化数据的特征为结构隐藏于数据中，为满足有效地检索半结构化数据中的信息，需要从中发现隐藏的结构，本文提出的 structure_discovery 算法首先将描述同一类事务对象的共有结构按照相似度聚集到一起，相似度高的结构在一个类中，同时在类上形成一个描述该类的完全结构，该结构包含了类中所有对象具有的结构；然后算法利用集合枚举树找出完全结构中满足阈值 minsup 的共有结构，从而发现半结构化、层次数据中满足用户兴趣度的结构，structure_discovery 算法和其他结构发现算法比较，效率较高，同时发现的结构同其他算法相同。

6 分布式数据开采系统

6.1 引言

数据开采的基础技术发展相当迅速，不同类型的知识开采研究也已经相当深入，将数据开采应用到企业级系统时，需要解决以下问题：① 一个完全的数据开采系统需要综合多种知识模型为企业或机构提供决策支持；② 现有的数据开采工具大都采用内置固定算法，这样系统只支持有限的数据开采过程；但是在实际应用中，不可能仅依靠单个的开采算法得到正确结果，因为在理论上，没有哪一个算法能够在所有的领域中能够比其它算法表现得更为出色，即便是在用户感兴趣的领域中，由于数据开采任务事先并不是特别明确，开采结果随着开采过程的深入逐渐形成，这就意味着大多数情况下，需要综合比较并结合多种算法得到理想的结果，因此就要求数据开采系统具有交互性和灵活性，能够提供多种多样的开采工具，并能根据用户提出的领域问题定制算法；③ 使用单策略开采系统解决问题，分析人员需要在不同的系统间切换，这引发了多系统间数据和结果的转换问题，同时分析人员需要适应不同系统的界面也是非常耗时的，多策略开采系统虽然可以在一定程度上解决这一问题，但它提供的算法选择范围也是有限的，因为这些封闭式系统没有提供开放性和扩展性的机制，但数据开采方法又是不断更新的，所以系统必须能够随时集成先进的算法，系统的开放性可以通过应用程序接口 API 实现；④ 数据开采的目的是为了优化企业和机构的商业行为，数据开采系统需要考虑如何将开采到的知识运用到与之相关的系统中，即需要为数据开采系统和其它企业决策支持系统提供一个集成环境；⑤ 由于 Internet 技术的广泛使用，企业数据的物理和逻辑位置相当分散，数据开采系统需要从不同的数据源获取数据。[137]

综合以上问题，可以得出一个完整的企业级数据开采系统需要满足完全性、扩展性、灵活性和分布式特性。完全性表明系统包含了多种分析工具和分析手段，如机器学习、可视化和统计方法等；扩展性表明系统可以通过开放式 API 随时加入新的数据开采算法，同时能够很方便地与其它系统集成；灵活性让用户可以根据需求自定义数据开采系统中的算法；分布式特性使得系统可以获取存储于不同物理位置的数据源，并且能够让 Internet 或 Intranet 上的用户方便地使用数据开采系统。

本文给出了一个具有扩展性的分布式数据开采系统框架，系统框架基于 Java

6 分布式数据开采系统

6.1 引言

数据开采的基础技术发展相当迅速，不同类型的知识开采研究也已经相当深入，将数据开采应用到企业级系统时，需要解决以下问题：① 一个完全的数据开采系统需要综合多种知识模型为企业或机构提供决策支持；② 现有的数据开采工具大都采用内置固定算法，这样系统只支持有限的数据开采过程；但是在实际应用中，不可能仅依靠单个的开采算法得到正确结果，因为在理论上，没有哪一个算法能够在所有的领域中能够比其它算法表现得更为出色，即便是在用户感兴趣的领域中，由于数据开采任务事先并不是特别明确，开采结果随着开采过程的深入逐渐形成，这就意味着大多数情况下，需要综合比较并结合多种算法得到理想的结果，因此就要求数据开采系统具有交互性和灵活性，能够提供多种多样的开采工具，并能根据用户提出的领域问题定制算法；③ 使用单策略开采系统解决问题，分析人员需要在不同的系统间切换，这引发了多系统间数据和结果的转换问题，同时分析人员需要适应不同系统的界面也是非常耗时的，多策略开采系统虽然可以在一定程度上解决这一问题，但它提供的算法选择范围也是有限的，因为这些封闭式系统没有提供开放性和扩展性的机制，但数据开采方法又是不断更新的，所以系统必须能够随时集成先进的算法，系统的开放性可以通过应用程序接口 API 实现；④ 数据开采的目的是为了优化企业和机构的商业行为，数据开采系统需要考虑如何将开采到的知识运用到与之相关的系统中，即需要为数据开采系统和其它企业决策支持系统提供一个集成环境；⑤ 由于 Internet 技术的广泛使用，企业数据的物理和逻辑位置相当分散，数据开采系统需要从不同的数据源获取数据。[137]

综合以上问题，可以得出一个完整的企业级数据开采系统需要满足完全性、扩展性、灵活性和分布式特性。完全性表明系统包含了多种分析工具和分析手段，如机器学习、可视化和统计方法等；扩展性表明系统可以通过开放式 API 随时加入新的数据开采算法，同时能够很方便地与其它系统集成；灵活性让用户可以根据需求自定义数据开采系统中的算法；分布式特性使得系统可以获取存储于不同物理位置的数据源，并且能够让 Internet 或 Intranet 上的用户方便地使用数据开采系统。

本文给出了一个具有扩展性的分布式数据开采系统框架，系统框架基于 Java

和 CORBA 等 Internet 技术构造分布式应用平台，系统中现有的数据开采算法包括本论文提出的聚类算法，其它还包括关联规则算法和序列发现算法。

6.2 分布式技术

作为分布式系统平台，Internet 上的技术已经发展得相当成熟[138]，这一技术的发展使得网络被看作一台“全球计算机”，可以在网络上任意一台计算机上访问到系统。Java 语言及 Java 提供的服务和协议等为分布式系统平台提供了可行的解决方案，并且纯 Java 组件可以通过 CORBA（Common Object Request Broker Architecture）与现有系统集成。

在分布式环境中，应用开发的过程就是组件开发的过程，一个组件就是一个分布式对象，该分布式对象不与特定的程序或计算机语言绑定在一起，组件具有封装性，对它的访问需要通过应用程序接口 API 实现。分布式框架可以方便实现灵活的可扩展系统，当需要加入新的功能时，只需要加入新的“插件”即可，Java 组件可以通过 RMI（Remote Method Invocation）集成到系统中，而非 Java 组件通过 CORBA 包装集成到系统。

Java 通过 Enterprise API 提供了构造分布式系统的平台，特点如下：

1. Enterprise JavaBeans 定义的 API 使得开发人员可以方便地创建、管理和使用基于组件的跨平台系统，它将应用并发、事务管理和资源管理等问题在逻辑上隔离开来。
2. JavaBeans 用来创建具有平台独立性的可重用的软件组件，它包括可视化 Beans，可视化 Beans 由一至多个 Java Applets 组成。
3. RMI 为不同机器或物理空间上的分布式 Java 对象或 Beans 提供通讯，用 RMI 在 Java 组件之间通讯比用 CORBA 要简便，同时它还支持将整个分布式对象作为参数传递。
4. JavaIDL（Interface Definition Language）是用 Java 实现的 CORBA 标准，为集成已有的基于 Java 的系统提供服务。
5. JDBC（Java DataBase Connectivity）为基于 Java 的系统提供统一的网络数据库访问，可以查询网络上的远程数据库，并获取具有统一格式的数据结果。

分布式组件框架为三层 Client/Server 结构，第一层是统一的客户端，为标准浏览器中的 Java Applet；中间层为应用服务器，用来集成和同步用户对后端数据

和 CORBA 等 Internet 技术构造分布式应用平台，系统中现有的数据开采算法包括本论文提出的聚类算法，其它还包括关联规则算法和序列发现算法。

6.2 分布式技术

作为分布式系统平台，Internet 上的技术已经发展得相当成熟[138]，这一技术的发展使得网络被看作一台“全球计算机”，可以在网络上任意一台计算机上访问到系统。Java 语言及 Java 提供的服务和协议等为分布式系统平台提供了可行的解决方案，并且纯 Java 组件可以通过 CORBA（Common Object Request Broker Architecture）与现有系统集成。

在分布式环境中，应用开发的过程就是组件开发的过程，一个组件就是一个分布式对象，该分布式对象不与特定的程序或计算机语言绑定在一起，组件具有封装性，对它的访问需要通过应用程序接口 API 实现。分布式框架可以方便实现灵活的可扩展系统，当需要加入新的功能时，只需要加入新的“插件”即可，Java 组件可以通过 RMI（Remote Method Invocation）集成到系统中，而非 Java 组件通过 CORBA 包装集成到系统。

Java 通过 Enterprise API 提供了构造分布式系统的平台，特点如下：

1. Enterprise JavaBeans 定义的 API 使得开发人员可以方便地创建、管理和使用基于组件的跨平台系统，它将应用并发、事务管理和资源管理等问题在逻辑上隔离开来。
2. JavaBeans 用来创建具有平台独立性的可重用的软件组件，它包括可视化 Beans，可视化 Beans 由一至多个 Java Applets 组成。
3. RMI 为不同机器或物理空间上的分布式 Java 对象或 Beans 提供通讯，用 RMI 在 Java 组件之间通讯比用 CORBA 要简便，同时它还支持将整个分布式对象作为参数传递。
4. JavaIDL（Interface Definition Language）是用 Java 实现的 CORBA 标准，为集成已有的基于 Java 的系统提供服务。
5. JDBC（Java DataBase Connectivity）为基于 Java 的系统提供统一的网络数据库访问，可以查询网络上的远程数据库，并获取具有统一格式的数据结果。

分布式组件框架为三层 Client/Server 结构，第一层是统一的客户端，为标准浏览器中的 Java Applet；中间层为应用服务器，用来集成和同步用户对后端数据

库的访问要求；第三层为数据库服务器用来存储数据。

1. Web 客户端：客户端机器不需要安装指定的操作系统、平台或其它代码，只需装有支持 Java 的 Web 浏览器，在访问分布式应用时在线下载所需的 Java 类。
2. 应用服务：利用 EJB（Enterprise JavaBeans）构造灵活的、具有可扩展性的应用系统框架，EJB 将应用逻辑封装在 Enterprise Bean 类中，由于 EJB 将处理并发及分配资源与应用逻辑隔离开来，开发人员只需考虑应用逻辑的代码实现；EJB 体系结构还支持组件接口定义和组件之间的交互，符合 API 的组件可以作为插件加入到系统中以增加系统功能；
3. 数据库服务：利用 JDBC 连接网络上任意数据库。

6.3 分布式数据开采系统

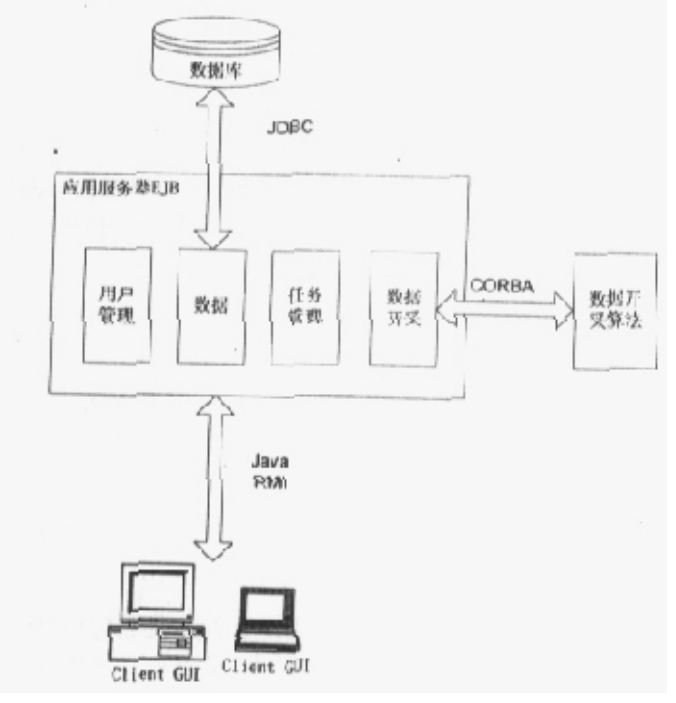

图 6.1 分布式数据开采系统体系结构

图 6.1 给出了分布式数据开采系统的体系结构，第一层为可视化的客户端，

库的访问要求；第三层为数据库服务器用来存储数据。

1. Web 客户端：客户端机器不需要安装指定的操作系统、平台或其它代码，只需装有支持 Java 的 Web 浏览器，在访问分布式应用时在线下载所需的 Java 类。
2. 应用服务：利用 EJB（Enterprise JavaBeans）构造灵活的、具有可扩展性的应用系统框架，EJB 将应用逻辑封装在 Enterprise Bean 类中，由于 EJB 将处理并发及分配资源与应用逻辑隔离开来，开发人员只需考虑应用逻辑的代码实现；EJB 体系结构还支持组件接口定义和组件之间的交互，符合 API 的组件可以作为插件加入到系统中以增加系统功能；
3. 数据库服务：利用 JDBC 连接网络上任意数据库。

6.3 分布式数据开采系统

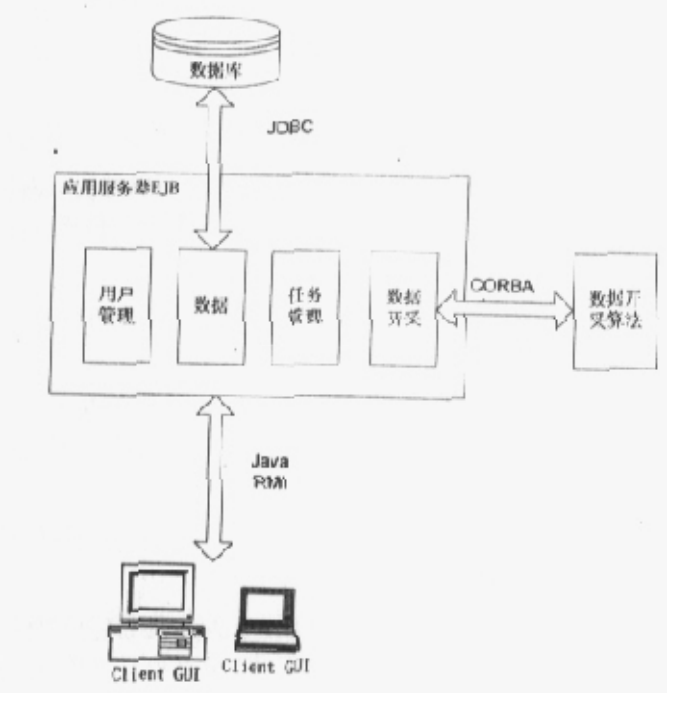

图 6.1 分布式数据开采系统体系结构

图 6.1 给出了分布式数据开采系统的体系结构，第一层为可视化的客户端，

通过 Java RMI 访问数据开采服务；第三层为网络数据库，应用服务通过 JDBC 与网络上任意数据库连接；第二层应用服务器由四个 EJB 类构成：① 用户管理：管理用户的永久对象，包括对象的用户访问或组访问控制的目录结构，主要用来管理用户数据开采任务及返回结果的永久对象；② 任务管理：客户端对数据开采任务的控制通过该 EJB 类实现，数据开采任务包括数据选择、数据转换、模型选择、模型评估等；③ 数据开采：通过 CORBA 集成数据开采算法；④ 数据：负责通过 JDBC 导入及存储数据库中的数据。

分布式数据开采系统中的数据开采工具由主要由两部分组成：关联规则开采和聚类分析，关联规则开采算法包括 Apriori、MWAR[36]、DMFI[136]和 MER[139]，聚类算法则包括 BIRCH、CCDCS 和 CHID，另外还包括 Structure_Discovery 和 MARC 等混合型的算法，算法之间的协同合作需要应用服务器中的任务管理类来完成。

6.4 本章小结

本章简要描述了分布式数据开采系统的结构框架，该系统为分布式的、可扩展的数据开采系统，通过 JDBC 访问网络上任意数据资源，并将开采结果返回到连接在网络上的任意一台计算机上，而 CORBA 技术使得系统可以随时集成新的数据开采工具，包括新的方法和算法，目前该系统包含关联规则开采和聚类分析两种知识模型。

通过 Java RMI 访问数据开采服务；第三层为网络数据库，应用服务通过 JDBC 与网络上任意数据库连接；第二层应用服务器由四个 EJB 类构成：① 用户管理：管理用户的永久对象，包括对象的用户访问或组访问控制的目录结构，主要用来管理用户数据开采任务及返回结果的永久对象；② 任务管理：客户端对数据开采任务的控制通过该 EJB 类实现，数据开采任务包括数据选择、数据转换、模型选择、模型评估等；③ 数据开采：通过 CORBA 集成数据开采算法；④ 数据：负责通过 JDBC 导入及存储数据库中的数据。

分布式数据开采系统中的数据开采工具由主要由两部分组成：关联规则开采和聚类分析，关联规则开采算法包括 Apriori、MWAR[36]、DMFI[136]和 MER[139]，聚类算法则包括 BIRCH、CCDCS 和 CHID，另外还包括 Structure_Discovery 和 MARC 等混合型的算法，算法之间的协同合作需要应用服务器中的任务管理类来完成。

6.4 本章小结

本章简要描述了分布式数据开采系统的结构框架，该系统为分布式的、可扩展的数据开采系统，通过 JDBC 访问网络上任意数据资源，并将开采结果返回到连接在网络上的任意一台计算机上，而 CORBA 技术使得系统可以随时集成新的数据开采工具，包括新的方法和算法，目前该系统包含关联规则开采和聚类分析两种知识模型。

7 总结

数据开采通过建立模型揭示隐藏在大量数据中的知识（模式和关系），而这些知识应当是隐含的、预先未知的、并且对用户具有潜在价值。通过数据开采获取的知识可以应用到以下各种领域：信息管理、查询处理、决策支持和过程控制等。数据开采的技术基础包括数据库系统、人工智能和统计学，随着数据开采的研究深入，许多研究领域，如基于知识的系统、数据可视化和情报检索等也对数据开采投入了极大关注。

数据开采是面向应用的技术，它根据应用需求选择相应的开采技术和模型。由数据开采发现的知识分类如下：关联规则、数据概括、数据分类、聚类分析、时间序列和序列发现等。作为数据开采的一项任务，聚类分析的目的是根据某种标准把大型、多维数据集合中相似的数据对象聚集到同一个类中，好的聚类结果应当使同一类中的对象相似度尽可能的高，而处于不同类之中的对象相似度尽可能的低。

在机器学习中，聚类分析也称作“无监督分类”，以区别于使用训练集的“监督分类”。机器学习中的聚类采用基于概率的增量方法，这种方法逐个读入新的对象，然后将它们分配到相应的类中，一旦确定了类别便不再轻易移动变更这些对象。增量式的方法在读入新的对象时不需要重新处理全部数据，每个对象需要从层次型结构中自顶向下按照某种标准选择“路径”从而形成节点，这些节点就是类，通常路径选取的标准都使用基于概率的方法，类的描述也采用概率来表达。统计学中的聚类分析研究主要集中在基于距离的数字属性数据的聚类上，主要聚类方法分为三种：系统聚类法、分解法和动态聚类法。系统聚类法首先将每个对象看作是一个类，然后逐个合并最相近的类，直到最终得到一个类为止；与系统聚类法相反，分解法首先将整个数据对象集合看作一个类，然后逐步分裂，直到每个类仅包含一个对象；动态聚类法分为全局方法或是半全局方法，它需要预先知道分类的数量 k，首先从数据对象集合中任意选取 k 个对象作为这 k 个类的凝聚点，然后将其它对象分配到与之最近的凝聚点所属的类中，接着不断移动交换各个类中的对象与凝聚点的位置，直到得到最优解为止，通常用一个标准函数来衡量聚类质量的好坏，即判断是否已得到最优解。

数据开采领域中，聚类分析主要是统计学和机器学习中聚类方法的应用。由于这两个领域中的聚类分析研究没有考虑大容量数据集合的问题，而且研究多集中于数字属性数据，因此，数据开采中聚类算法的研究主要集中在如何提高大型

数据集合的聚类效率、如何处理具有各种特征的数据集合，如文档数据、分类数据、高维数据等。

针对数据开采的实际应用，本文首先给出了一个处理分类属性数据的聚类算法，该算法能够有效处理大容量的数据集合；接着提出了针对高维数据集合的聚类算法，它能够发现高维空间里子空间中数据密集点的聚类；最后给出了两个算法，它们将聚类技术分别应用到关联规则发现和半结构化、层次型数据的结构发现上。总结全文，本文的创新和贡献如下：

1. 传统的聚类分析多用来聚类数字属性数据，而对于现实世界中大量存在着的分类属性数据，研究相对较少。现有的针对分类属性数据的算法需要多遍扫描数据库，对于数据开采经常处理的大容量数据，多遍 I/O 操作是一项沉重的系统开销，算法的效率会非常低。针对这些问题，本文提出了能有效处理分类属性数据的聚类算法 CCDCS。算法提出了聚类汇总的概念，使用该技术可以把规模很大的原始数据压缩到规模较小的汇总数据集合上；同时提出了一种适合汇总（或压缩）数据对象的相似度衡量标准。为了有效地存储和搜索数据，算法提出了具有层次特征的数据结构——聚类树。CCDCS 算法采用增量方法，每读入一个分类对象，首先将其转换为压缩形式，然后将它加入到聚类树中，该过程的对象路径选择采用了估价函数搜索策略，从而可以高效地构造和搜索聚类树。CCDCS 算法只扫描一遍数据库，便把原始数据集合转换到压缩数据集合，并存储到聚类树中，其后的操作都只针对压缩后的数据集合，大大提高了聚类的效率；同时由于聚类汇总中包含了足够的用于计算对象之间相似度的信息，因此数据压缩并不会对聚类结果造成影响。与其它算法在聚类结果和运行时间的比较显示，CCDCS 比其它针对分类属性数据的算法效率要高，同时聚类结果的质量也相差无几。

2. 高维数据指那些属性数量多且值域范围广的数据对象，由这些数据对象组成的数据空间维数较高，数据点分布稀疏、密度平均，因此，很难从中发现数据类；但是，在高维数据空间的子空间，由于维数较低，对象分布相对集中，从中发现数据类又是有意义的。另外，用基于距离的方法进行数据聚类时，维数的增多会使得计算对象间距离的时间开销增大，导致算法效率降低，而采用降维的方法会影响聚类结果的正确性。针对以上问题，本文提出算法 CHID，首先，它能够主动搜索 k 维空间及其所有子空间存在的聚类；其次，该算法通过数据对象的分布发现密度较高的区域从而发

现聚类，而未采用基于距离的计算方法，因此，CHID 具有扩展性，能够高效率开采大型数据集合。算法过程如下：将 n 维空间划分为大小相同的单元区域，然后采用双向搜索策略在指定的 n 维空间或其子空间上发现数据点密集的单元区域，将密集的单元区域标记为 1，其余的为 0，最后采用逐位相与的方法为这些密集单元区域聚类。算法的创新和贡献在于：① 采用了双向搜索策略来搜索子空间，该搜索策略能够有效地减少搜索空间，从而减少计算量提高算法效率；② 采用了逐位相与的方法来聚类密集单元区域，该方法只用到逐位与和位移两种机器指令，也使得算法效率得到进一步提高。

3. 算法 MARC 将聚类技术应用到关联规则的发现上，现有的关联规则算法大都需要多次扫描数据库，而频繁的 I/O 操作会影响算法在大容量数据集合上的效率。算法 MARC 利用聚类技术压缩交易数据库，再在压缩后的数据集合上发现规则，从而减少开采算法需要处理的数据量以提高算法效率，该思想在国内外文献中尚未发现，是作者的创新。由于关联规则发现需要在聚类结果上进行，所以如何将具有压缩形式的聚类结果转换成适合关联规则发现的数据形式而又尽可能不丢失信息成为问题的关键，算法提出的聚类汇总转换方法有效地解决了这一问题，利用这种方法能够得到与原数据集合相近的压缩数据集合。与其它关联规则开采算法的比较证实了采用聚类技术，MARC 确实能够提高关联规则发现效率，且不显著影响发现结果。

4. 异构数据库的集成和 Web 数据促成了半结构化数据的形成，半结构化数据的特征为结构隐藏于数据中。为满足有效地检索半结构化数据中的信息，需要从中发现隐藏的结构，本文提出的 structure_discovery 算法首先利用聚类分析将描述同一类对象的共有结构按照相似度聚集到一起，相似度高的结构在同一个类中，同时在类上形成一个描述该类的完全结构，该结构包含了类中的所有结构；然后算法利用集合枚举树找出完全结构中满足阈值 *minsup* 的共有结构，从而发现半结构化、层次数据中满足用户兴趣度的结构，structure_discovery 算法和其他结构发现算法比较，效率较高，同时就结果来看，发现的结构同其他算法相同。将聚类分析应用到半结构化数据的结构发现尚属首次。

聚类算法研究还有许多需要深入的地方，以下是进一步研究的方向：① CCDCS 算法中针对特定聚类分析任务的最小相似度和 B 阶数的选择；② MARC

算法中，最小支持度、最小置信度等参数的自动选择；③ 利用聚类方法自动分类 Web 文档等半结构化或无结构数据。

另外，数据开采在国内还处在研究阶段，随着政府部门、金融机构以及大型零售业的数据积累越来越多，对数据开采系统以及特定领域分析预测解决方案的需求一定会越来越急迫，国外对于系统产品的研究很广泛[140]-[143]，而国内在这方面的研究较少，为此，数据开采的研究人员应当考虑如何构架合理的数据开采系统，这个系统应当① 对内：能够协调各种数据开采工具协同完成不同阶段的数据开采任务；② 对外：要具有可扩展性，能够快速集成新的数据开采算法。考虑到目前 Internet 技术的普及，各种单位机构的数据大都分散在网络各处，因此，该系统也应当具有分布式特性。由于本文的研究重点是数据开采中的聚类算法研究，所以只给出了一个分布式数据开采系统的框架原理，有关分布式数据开采系统的具体实现还需要我们做更进一步的研究。

致 谢

一直记得我的导师卢正鼎教授在四年前说的话："下决心，再辛苦几年！"是的，对我来说，从准备博士候选人的资格考试到目前论文初稿完成并不是一个轻松的过程，但是，当我检视这几年的收获，无论是本专业领域的知识还是世界观都似乎比过去有了质的飞跃。所以，我首先要感谢卢老师在我面临人生重要抉择时，向我指明还有这样一条崎岖但充满花香的路；其次，要感谢卢老师在学术研究上给予的高屋建瓴式的指导以及帮助。同时，我还要感谢我的副导师胡和平教授对我在学术上的细心指导，他勤劳严谨的工作作风和乐观开朗的工作态度是我学习的榜样；最后，我和我先生还要感谢两位导师在生活上给予我们的帮助。

感谢 Tian Zhang 博士在聚类分析领域理论及资料上给予的无私帮助！

感谢武汉市第二十一中学的英语教师闸远明老师在英语语法上给予的指导。

还要感谢的是我的家人。感谢我先生路松峰对我的理解和支持，他的鼓励让我不止一次在困难面前充满自信；每当想起我的父母每次来看望我之前，都会小心翼翼地问是否会打扰到我的学习时，我都会感到深深的歉意，念亲恩，无以为报，我只希望能够成为他们的骄傲！

最后感谢我的朋友们，没有他们，生活不会如此多姿多彩！

参考文献

[1] 王珊等. 数据仓库技术与联机分析处理. 北京:科学出版社, 1998. 113-125

[2] Alex Berson, Stephen J. Smith. Data Warehousing, Data Mining, and OLAP. 北京:世界图书出版公司, 1999. 367-369

[3] William J. Frawley, Gregory Piatetsky-Shapiro, Christopher J. Matheus. Knowledge Discovery in Databases: An Overview, Knowledge Discovery in Databases. Menlo Park: AAAI Press, 1991. 1-30

[4] Ming-Syan Chen, Jiawei Han, Philip S. Yu. Data Mining: An Overview from Database Perspective. IEEE Transactions on Knowledge and Data Engineering(TKDE), 1996, 8(6): 866-883

[5] Jiawei Han, Yongjia Fu, Krzysztof et al. Knowledge Mining in Databases: An Integration of Machine Learning Methodologies with Database Technologies. Canadian Artificial Intelligence, 1996, 38(Winter): 4-8

[6] Sunita Sarawagi, Rakesh Agrawal, Nimrod Megiddo. Discovery-Driven Exploration of OLAP Data Cubes. In: Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos et al eds. Proceedings of Advances in Database Technology, the 6th International Conference on Extending Database Technology (EDBT'98). Valencia, Spain. 1998. Heidelberg: Springer, 1998. 168-182

[7] Jiawei Han, Sonny H.S. Chee, Jenny Y. Chiang. Issues for On-Line Analytical Mining of Data Warehouses. In: Laura M. Haas, Ashutosh Tiwary eds. Proceedings of 1998 SIGMOD'98 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98). Seattle, USA. 1998. New York: ACM Press, 1998. 21-25

[8] Jiawei Han. OLAP Mining: An Integration of OLAP with Data Mining. In: Proceedings of 1997 IFIP Conference on Data Semantics (DS'7). Leysin, Switzerland. 1997. Heidelberg: Springer, 1997. 1-11

[9] John R. Quinlan. Induction of Decision Trees. Machine Learning, 1986, 1: 81-106

[10] John R. Quinlan. C4.5: Programs for Machine Learning. Machine Learning, 1993, 1: 81-206

[11] Rakesh Agrawal, Sakti P. Ghosh, Tomasz Imielinski et al. An Interval Classifier for Database Mining Applications. In: Li-Yan Yuan ed. Proceedings of the 18th International Conference on Very Large Data Bases(VLDB'92). Vancouver, Canada. 1992. San Francisco: Morgan Kaufmann, 1992. 560-573

[12] Tarek M. Anwar, Howard W. Beck, Shamkant B. Navathe. Knowledge Mining by Imprecise Querying: A Classification-Based Approach. In: Forouzan Golshani ed. Proceedings of the Eighth International Conference on Data

Engineering. Tempe, Arizona. 1992. Los Alamitos: IEEE Computer Society, 1992. 622-630

[13] Philip K. Chan, Salvatore J. Stolfo. Learning Arbiter and Combiner Trees from Partitioned Data for Scaling Machine Learning. In: Usama M. Fayyad, Ramasamy Uthurusamy eds. Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95). Montreal, Canada. 1995. Menlo Park: AAAI Press, 1995. 39-44

[14] Peter Cheeseman, John Stutz. Bayesian Classification (AutoClass): Theory and Results. In: Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy eds. Advances in Knowledge Discovery and Data Mining. 1996. Menlo Park: AAAI/MIT Press, 1996. 153-180

[15] Ming-Syan Chen, Philip S. Yu. Using Multi-Attribute Predicates for Mining Classification Rules. Technical report, IBM Research Report. 1995. 1-11

[16] Jiawei Han, Yongjia Fu. Attribute-Oriented Induction in Data Mining. In: Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy eds. Advances in Knowledge Discovery and Data Mining. 1996. Menlo Park: AAAI/MIT Press, 1996. 399-421.

[17] Jiawei Han, Yongjian Fu, Wei Wang, et al. DBMiner: A System for Mining Knowledge in Large Relational Databases. In: Evangelos Simoudis, Jiawei Han, Usama M. Fayyad eds. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, USA. 1996. Menlo Park: AAAI Press, 1996. 250-255

[18] Hongjun Lu, Rudy Setioni, Huan Liu. NeuroRule: A Connectionist Approach to Data Mining. In: Umeshwar Dayal, Peter M. D. Gray, Shojiro Nishio eds. Proceedings of 21st International Conference on Very Large Data Bases (VLDB'95). Zurich, Switzerland. 1995. San Francisco: Morgan Kaufmann, 1995. 478-489

[19] Manish Mehta, Rakesh Agrawal, Jorma Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining. In: Peter M. G. Apers, Mokrane Bouzeghoub, Georges Gardarin eds. Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96). Avignon, France. 1996. Heidelberg: Springer, 1996. 18-32

[20] Wojciech. Ziarko. Rough Sets, Fuzzy Sets and Knowledge Discovery. London: Springer-Verlag, 1994. 16-23

[21] John F. Elder IV, Daryl Pregibon. A Statistical Perspective on Knowledge Discovery in Databases. In: Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy eds. Advances in Knowledge Discovery and Data Mining. Menlo Park: AAAI/MIT Press, 1996. 83-113

[22] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, et al. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. ACM SIGMOD Record, 1990, 19(2): 322-331

[23] Martin Ester, Hans-Peter Kriegel, Xiaowei Xu. Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. In: Max J. Egenhofer, John R. Herring eds. Proceedings Advances in Spatial Databases, 4th International Symposium (SSD'95). Portland, USA. Heidelberg: Springer, 1995. 67-82

[24] Douglas Fisher. Improving Inference through Conceptual Clustering. In: Forbus, K. S. H eds. Proceedings of Sixth National Conference on Artificial Intelligence. Seattle, Washington. 1987. Menlo Park: AAAI Press, 1987. 461-465

[25] Douglas Fisher. Optimization and Simplification of Hierarchical Clustering. In: Usama M. Fayyad, Ramasamy Uthurusamy eds. Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95). Montreal, Canada. 1995. Menlo Park: AAAI Press, 1995. 118-123

[26] Anil K. Jain, Richard C. Dubes. Algorithms for Clustering Data. New Jersey: Prentice Hall, 1988. 3-39

[27] Raymond T. Ng, Jiawei Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In: Jorge B. Bocca, Matthias Jarke, Carlo Zaniolo eds. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94). Santiago de Chile, Chile. 1994. San Francisco: Morgan Kaufmann, 1994. 144-155

[28] Tian Zhang, Raghu Ramakrishnan, Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: H. V. Jagadish, Inderpal Singh Mumick eds. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96). Montreal, Canada. 1996. New York: ACM Press, 1996. 103-114

[29] Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In: Peter Buneman, Sushil Jajodia eds. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93). Washington, D.C. 1993. New York: ACM Press, 1993. 207-216

[30] Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In: Jorge B. Bocca, Matthias Jarke, Carlo Zaniolo eds. Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94). Santiago de Chile, Chile. 1994. San Francisco: Morgan Kaufmann, 1994. 487-499

[31] Jiawei Han, Yongjia Fu. Discovery of Multiple-Level Association Rules from Large Databases. In: Umeshwar Dayal, Peter M. D. Gray, Shojiro Nishio eds. Proceedings of 21st International Conference on Very Large Data Bases (VLDB'95). Zurich, Switzerland. 1995. San Francisco: Morgan Kaufmann, 1995. 420-431

[32] Heikki Mannila, Hannu Toivonen, A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In: Usama M. Fayyad, Ramasamy Uthurusamy eds. Proceedings of the 12th National Conference on Artificial Intelligence. Seattle, USA. 1994. Menlo Park: AAAI Press, 1994. 181-192

[33] Jong Soo Park, Philip S. Yu, Ming-Syan Chen. Mining Association Rules with Adjustable Accuracy. In: Forouzan Golshani, Kia Makki eds. Proceedings of the Sixth International Conference on Information and Knowledge Management (CIKM'97). Las Vegas, Nevada. 1997. New York: ACM Press, 1997. 151-160

[34] Ashoka Savasere, Edward Omiecinski, Shamkant B. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In: Umeshwar Dayal, Peter M. D. Gray, Shojiro Nishio eds. Proceedings of 21st International Conference on Very Large Data Bases (VLDB'95). Zurich, Switzerland. 1995. San Francisco: Morgan Kaufmann, 1995. 432-444

[35] Ramakrishnan Srikant, Rakesh Agrawal. Mining Generalized Association Rules. In: Umeshwar Dayal, Peter M. D. Gray, Shojiro Nishio eds. Proceedings of 21st International Conference on Very Large Data Bases (VLDB'95). Zurich, Switzerland. 1995. San Francisco: Morgan Kaufmann, 1995. 407-419

[36] Songfeng Lu, Heping Hu, Fan Li. Mining weighted association rules. Intelligent Data Analysis, 2001, 5(2): 211-225

[37] 刘芳，路松峰，卢正鼎等. 一种基于限制的关联规则数据开采的算法. 华中科技大学学报. 2001, 29(3): 27-29

[38] Ming-Syan Chen, Jong Soo Park, Philip S. Yu. Efficient Data Mining for Path Traversal Patterns in Distributed Systems. In Proceedings of the 16th International Conference on Distributed Computing Systems. Hong Kong, China. 1996. Los Alamitos: IEEE Computer Society, 1996. 385-393

[39] Shiby Thomas, Sunita Sarawagi. Mining Generalized Association Rules and Sequential Patterns Using SQL Queries. In: Rakesh Agrawal, Paul E. Stolorz, Gregory Piatetsky-Shapiro eds. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98). New York City, USA. 1998. Menlo Park: AAAI Press, 1998. 344-348

[40] Rakesh Agrawal, Ramakrishnan Srikant. Mining Sequential Patterns. In: Philip

S. Yu, Arbee L. P. Chen eds. Proceedings of the 11th International Conference on Data Engineering (ICDE'95). Taipei, Taiwan. 1995. Los Alamitos: IEEE Computer Society, 1995. 3-14

[41] Ramakrishnan Srikant, Rakesh Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In: Peter M. G. Apers, Mokrane Bouzeghoub, Georges Gardarin eds. Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96). Avignon, France. 1996. Heidelberg: Springer, 1996. 3-17

[42] Stephen A. Fenner, Jack H. Lutz, Elvira Mayordomo. Weakly Useful Sequences. In: Zoltán Fülöp, Ferenc Gécseg eds. Proceedings of Automata, Languages and Programming, 22nd International Colloquium (ICALP'95). Szeged, Hungary. 1995. Heidelberg: Springer, 1995. 393-404

[43] Myra Spiliopoulou, Lukas C. Faulstich, Karsten Winkler. A Data Miner Analyzing the Navigational Behaviour of Web Users. In Proceedings of the Workshop on Machine Learning in User Modelling of the ACAI'99. Creta, Greece. 1999.

[44] Peter Buneman, Susan B. Davidson, Mary F. Fernandez, et al. Adding Structure to Unstructured Data. In: Foto N. Afrati, Phokion Kolaitis eds. Proceedings of 6th International Conference on Database Theory (ICDT'97). Delphi, Greece. 1997. Heidelberg: Springer, 1997. 336-350

[45] Masum Z. Hasan, Alberto O. endelzon, Dimitra Vista. Applying Database Visualization to the World Wide Web. SIGMOD Record, 1996, 25(4): 45-49

[46] Weiyi Meng, King-Lup Liu, Clement T. Yu et al. Determining Text Databases to Search in the Internet. In: Ashish Gupta, Oded Shmueli, Jennifer Widom eds. Proceedings of 24th International Conference on Very Large Data Bases (VLDB'98). New York City, USA. 1998. San Francisco: Morgan Kaufmann, 1998. 14-25

[47] Osmar R. Zaïane, Man Xin, Jiawei Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In: Proceedings of the IEEE Forum on Reasearch and Technology Advances in Digital Libraries (ADL'98). Santa Barbara, USA. 1998. Los Alamitos: IEEE Computer Society, 1998. 19-29

[48] Svetlozar Nestorov, Serge Abiteboul, Rajeev Motwani et al. Extracting Schema from Semistructured Data. In: Laura M. Haas, Ashutosh Tiwary eds. Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD'98). Seattle, USA. 1998. New York: ACM Press, 1998. 295-306

[49] Stéphane Grumbach, Giansalvatore Mecca. In Search of the Lost Schema. In:

Catriel Beeri, Peter Buneman eds. Proceedings of the 7th International Conference on Database Theory (ICDT'99). Jerusalem, Israel. 1999. Heidelberg: Springer, 1999. 314-331

[50] Mark Craven, Dan DiPasquo, Dayne Freitag et al. Learning to Extract Symbolic Knowledge from the World Wide Web. In Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI'98). Madison, Wisconsin. 1998. Menlo Park: AAAI Press, 1998. 509-516

[51] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini. Queries and Constraints on Semi-Structured Data. In: Matthias Jarke, Andreas Oberweis eds. Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99). Heidelberg, Germany. 1999. Heidelberg: Springer, 1999. 434-438

[52] Sanjay Kumar Madria, Sourav S. Bhowmick, Wee Keong Ng, et al. Research Issues in Web Data Mining. In: Mukesh K. Mohania, A. Min Tjoa eds. Proceedings Data Warehousing and Knowledge Discovery, First International Conference. Florence, Italy. 1999. Heidelberg: Springer, 1999. 303-312

[53] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, et al. Rewriting Regular Expressions in Semi-Structured Data. In: Catriel Beeri, Peter Buneman eds. Proceedings of 7th International Conference on Database Theory (ICDT'99) Workshop on Query Processing for Semi-Structured Data and Non-Standard Data Formats. Jerusalem, Israel. 1999. Heidelberg: Springer, 1999. 194--204

[54] Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo. Semistructured and Structured Data in the Web: Going Back and Forth. SIGMOD Record, 1997, 26(4): 16-23

[55] Myra Spiliopoulu. The Laborious Way From Data Mining to Web Log Mining. International Journal of Computer System, Science and Engineering, Special Issue on "Semantics of the Web", 1999, 14(3): 113-126

[56] Bamshad Mobasher, Namit Jain, Eui-Hong(Sam) Han, et al. Web Mining: Pattern Discovery from World Wide Web Transactions. Technical Report TR96-050, Department of Computer Science, University of Minnesota, 1996. 2-71

[57] 刘芳，胡和平. 半结构化数据的模式发现. 微型电脑应用. 2000, 16(2): 13-15

[58] 刘芳，胡和平，路松峰. 半结构化、层次数据的模式发现. 小型微型计算机系统. 2001, 22(1): 84-88

[59] Stefan Berchtold, H. V. Jagadish, Kenneth A. Ross. Independence Diagrams:

A Technique for Visual Data Mining. In: Rakesh Agrawal, Paul E. Stolorz et al eds. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98). New York City, USA. 1998. Menlo Park: AAAI Press, 1998. 139-143

[60] Daniel. A. Keim, Hans-Peter Kriegel. Issues in Visualizing Large Databases, System Demonstration. In: Michael J. Carey, Donovan A. Schneider eds. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. San Jose, USA. 1995. New York: ACM Press, 1995. 203-214

[61] Daniel A. Keim. Pixel-Oriented Database Visualizations. SIGMOD Record special issue on "Information Visualization", 1996, 25(4): 35-39

[62] Daniel A. Keim, Hans-Peter Kriegel, Thomas Seidl. Supporting Data Mining of Large Databases by Visual Feedback Queries. In Proceedings of the 10th International Conference on Data Engineering (ICDE'94). Houston, USA. 1994. Los Alamitos: IEEE Computer Society, 1994. 302-313

[63] Daniel A. Keim, Hans-Peter Kriegel. Visdb: A System for Visualizing Large Databases. In: Michael J. Carey and Donovan A. Schneider eds. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. San Jose, USA. 1995. New York: ACM Press. 1995. 482-494

[64] Daniel A. Keim, Hans Peter Kriegel. VisDB: Database Exploration Using Multidimensional Visualization. IEEE Computer Graphics and Applications, 1994, 14(5): 40-49

[65] Mihael Ankerst, Christian Elsen, Martin Ester et al. Visual Classification: An Interactive Approach to Decision Tree Construction. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, USA. 1999. New York: ACM Press, 1999. 392-396

[66] Daniel A. Keim, Hans-Peter Kriegel. Visualization Techniques for Mining Large Databases: A Comparison. IEEE Transactions on Knowledge and data Engineering (TKDE), 1996, 8(6): 923-938

[67] Bing Liu, Wynne Hsu, Ke Wang et al. Visually Aided Exploration of Interesting Association Rules. In: Ning Zhong, Lizhu Zhou eds. Proceedings of the 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD'99). Beijing, China. 1999. Heidelberg: Springer, 1999. 380-389

[68] Flip Korn, Alexandros Labrinidis, Yannis Kotidis et al. Ratio Rules: A New Paradigm for Fast, Quantifiable Data Mining. In: Ashish Gupta, Oded Shmueli, Jennifer Widom eds. Proceedings of 24th International Conference on Very Large Data Bases (VLDB'98). New York City, USA. 1998. San Francisco:

Morgan Kaufmann, 1998. 582-593

[69] Mong Li Lee, Tok Wang Ling, Hongjun Lu et al. Cleansing Data for Mining and Warehousing. In: Trevor J. M. Bench-Capon, Giovanni Soda, A. Min Tjoa eds. Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA'99). Florence, Italy. 1999. Heidelberg: Springer, 1999. 751-760

[70] Hongjun Lu, Sam Yuan Sung, Ying Lu. On Preprocessing Data for Effective Classification. In: H. V. Jagadish, Inderpal Singh Mumick eds: Proceedings of ACM SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96). Montreal, Canada. 1996. New York: ACM Press, 1996. 39-47

[71] Huan Liu, Hongjun Lu, Jun Yao. Identifying Relevant Databases for Multidatabase Mining. In: Xindong Wu, Kotagiri Ramamohanarao, Kevin B. Korb eds. Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD'98). Melbourne, Australia. 1998. Heidelberg: Springer, 1998. 210-221

[72] Bing Liu, Wynne Hsu. Post-Analysis of Learned Rules. In Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96. Portland, USA. 1996. Menlo Park: AAAI Press / The MIT Press, 1996. 828-834

[73] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen et al. Finding Interesting Rules from Large Sets of Discovered Association rules. In: Charles K. Nicholas, James Mayfield eds. Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94). Gaithersburg, USA. 1994. New York: ACM Press, 1994. 401-407

[74] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen et al. Pruning and Grouping Discovered Association Rules. In: Nada Lavrac, Stefan Wrobel eds. Proceedings of the 8th European Conference on Statistics, Machine Learning, and Knowledge Discovery in Databases (ECML'95) Workshop. Heraclion, Greece. 1995. Heidelberg: Springer, 1995. 47-52

[75] Nimrod Megiddo, Ramakrishnan Srikant. Discovering Predictive Association Rules. In: Rakesh Agrawal, Paul E. Stolorz, Gregory Piatetsky-Shapiro eds. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98). New York City, USA. 1998. Menlo Park: AAAI Press, 1998. 274-278

[76] Jiawei Han, Yongjia Fu, Wei Wang et al. DMQL: A Data Mining Query Language for Relational Databases. In H. V. Jagadish, Inderpal Singh Mumick

eds. Proceedings of 1996 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96). Montreal, Canada. 1996. New York: ACM Press, 1996. 27-33

[77] Serge Abiteboul, Dallan Quass, Jason McHugh, et al. The Lorel Query Language for Semistructured Data. International Journal of Digital Libraries, 1997, 1(1): 68-88

[78] Osmar R. Zaïane, Jiawei Han. WebML: Querying the World-Wide Web for Resources and Knowledge. In Proceedings of Workshop on Web Information and Data Management (WIDM'98). Bethesda, Maryland. 1998. New York: ACM Press, 1998. 9-12

[79] Yiming Yang. An Evaluation of Statistical Approaches to Text Categorization. Journal of Information Retrieval, 1999, 1(1/2): 67-88

[80] Brian Lent, Rakesh Agrawal, Ramakrishnan Srikant. Discovering Trends in Text Databases. In: David Heckerman, Heikki Mannila, Daryl Pregibon eds. Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97). San Diego, USA. 1997. Menlo Park: AAAI Press, 1997. 227-230

[81] Dayne Freitag. Information Extraction from HTML: Application of a General Machine Learning Approach. In Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI'98). Madison, USA. 1998. Menlo Park: AAAI Press, 1998. 517-523

[82] Ronen Feldman, Haym Hirsh. Mining Association in Text in the Presence of Background Knowledge. In: Evangelos Simoudis, Jiawei Han, Usama M. Fayyad eds. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, USA. 1996. Menlo Park: AAAI Press, 1996. 343-346

[83] Chumki Basu, Haym Hirsh, William W. Cohen. Recommending Papers by Mining the Web. In: Thomas Dean ed. Proceedings of Workshop on Machine Learning for Information Filtering, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99). Stockholm, Sweden. 1999. San Fransisco: Morgan Kaufmann, 1999. 943-952

[84] Ian H. Witten, Zane Bray, Malika Mahoui, et al. Text Mining: A New Frontier for Lossless Compression. In Proceedings of Data Compression Conference. Snowbird, Utah, USA. 1999. Los Alamitos, CA: IEEE Press, 1999. 198-207

[85] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, et al. Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases. In: Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, et al. eds. Proceedings of the 23rd

VLDB Conference. Athens, Greece. 1997. San Fransisco: Morgan Kaufmann, 1997. 446-455

[86] 卢正鼎, 刘芳. 利用文本挖掘实现 Web 智能服务. 小型微型计算机系统. 2001, 22(6): 703-705

[87] 张尧庭, 方开泰. 多元统计分析引论. 3, 北京: 科学出版社, 1997. 393-457

[88] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In: L. LeCam, J. Neyman eds. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1. Berkeley, CA. 1967. Berkeley, CA: University of California Press, 1967. 281-297

[89] Leonard Kaufman, Peter J. Rousseeuw. Finding Groups in Data – An Introduction on Cluster Analysis. Wiley Series in Probability and Mathematical Statistics, Wiley Press. 1990. 32-71

[90] Tian Zhang, Raghu Ramakrishnan. BIRCH: A New Data Clustering Algorithm and Its Applications. Data Mining and Knowledge Discovery, 1997, 1: 141-182

[91] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel et al. OPTICS: Ordering Points To Identify the Clustering Structure. In: Alex Delis, Christos Faloutsos, Shahram Ghandeharizadeh eds. Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD'99). Philadelphia, USA. 1999. New York: ACM Press, 1999. 49-60

[92] Martin Ester, Hans-Peter Kriegel, Jörg Sander et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Evangelos Simoudis, Jiawei Han, Usama M. Fayyad eds. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, USA. 1996. Menlo Park: AAAI Press, 1996. 226-231

[93] Richard O. Duda, Peter E. Hard. Pattern Classification and Scene Analysis. New York: Wiley-Interscience Publication. 1973. 79-105

[94] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In Proceedings of the 15th International Conference on Data Engineering (ICDE'99). Sydney, Australia. 1999. Los Alamitos: IEEE Computer Society, 1999. 512-521

[95] Zhexue Huang. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. In: Joan Peckham ed. Proceedings of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. Tucson, USA. 1997. New York: ACM Press, 1997. 512-523

[96] Venkatesh Ganti, Johannes Gehrke, Raghu Ramakrishnan. CACTUS-Clustering Categorical Data Using Summaries. In Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining

(KDD'99). San Diego, CA, USA. 1999. New York: ACM Press, 1999. 73-83

[97] David Gibson, Jon M. Kleinberg, Prabhakar Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. In: Ashish Gupta, Oded Shmueli, Jennifer Widom eds. Proceedings of 24th International Conference on Very Large Data Bases (VLDB'98). New York City, USA. 1998. San Francisco: Morgan Kaufmann, 1998. 311-322

[98] Zhexue Huang. Clustering Large Data Sets with Mixed Numeric and Categorical Values. In: Hong-jun Lu, Huan Liu, H. Motoda eds. Proceedings of The First Pacific-Asia Conference on Knowledge Discovery and Data Mining. Singapore. 1997. World Scientific, 1997. 21-34

[99] Ke Wang, Chu Xu, Bing Liu. Clustering Transactions Using Large Items. In Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management (CIKM'99). Kansas City, USA. 1999. New York: ACM Press, 1999. 483-490

[100] Shinichi Morishita, Teruyoshi Hishiki, Kousaku Okubo. Towards Mining Gene Expression Database. In: Kyuseok Shim, Ramakrishnan Srikant eds. Proceedings of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'99). Philadelphia, USA. 1999. 7-19

[101] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: Laura M. Haas, Ashutosh Tiwary eds. Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD'98). Seattle, USA. New York: ACM Press, 1998. 94-105

[102] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke et al. Clustering Large Datasets in Arbitrary Metric Spaces. In Proceedings of the 15th International Conference on Data Engineering (ICDE'99). Sydney, Austrialia. 1999. Los Alamitos: IEEE Computer Society, 1999. 502-511

[103] Oren Zamir, Oren Etzioni, Omid Madani et al. Fast and Intuitive Clustering of Web Documents. In: David Heckerman, Heikki Mannila, Daryl Pregibon eds. Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97). Newport Beach, USA. 1997. Menlo Park: AAAI Press, 1997. 287-290

[104] Oren Zamir, Oren Etzioni. Web Document Clustering: A Feasibility Demonstration. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98). Melbourne, Austrailia. 1998. New York: ACM Press, 1998. 46-54

[105] Jerome Moore, Eui-Hong (Sam) Han, Daniel Boley et al. Web Page Categorization and Feature Selection Using Association Rule and Principal

Component Clustering. In 7th Workshop on Information Technologies and Systems. Atlanta, USA. 1997. New York: ACM Press, 1997. 365-391

[106] Eui-Hong Han. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. PhD thesis, University of Minnesota, 1999. 38-115

[107] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar et al. Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. Data Engineering Bulletin, 1998, 21(1): 15-22

[108] Anupam Joshi, Sanjiva Weerawarana, Elias N. Houstis, et al. The Use of Neural Networks to Support Intelligent Scientific Computing. In Proceedings of the 4th IEEE International Conference on Neural Networks (IEEE ICNN'94). 1994. Los Alamitos: IEEE Computer Society, 1994. 2197-2202

[109] Anupam Joshi, Narcndran Ramakrishnan, Elias Houstis. On Neurobiological, NeuroFuzzy and Statistical Pattern Recognition Techniques. IEEE Transactions on Neural Networks, 1996, 8(1): 18-31

[110] Supot Jesse S. Jin. Applying Unsupervised Fuzzy C-Prototypes Clustering in Motion-Based Segmentation. In: Yahiko Kambayashi, Dik Lun Lee et al eds. Proceedings of the Workshops on Data Warehousing and Data Mining, Mobile Data Access, and Collaborative Work Support and Spatio-Temporal Data Management (ER'98). Singapore. 1998. Heidelberg: Springer, 1998. 580-589

[111] Anuparm· Joshi, Narendran Ramakrishnan. A Neuro-Fuzzy Approach to Agglomerative Clustering. In: Proceedings of the 6th IEEE International Conference on Neural Networks (IEEE ICNN'96). Los Alamitos: IEEE Computer Society, 1996. 1028-1033

[112] Anuparm Joshi, Sansanee Auephanwiriyakul, Raghu Krishnapuram. On Fuzzy Clustering and Content Based Access to Networked Video Databases. In Proceedings of 8th International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications (RIDE'98). Orlando, Florida. 1998. Los Alamitos: IEEE Computer Society, 1998. 42-49

[113] Brian Lent Lent, Arun N. Swami, Jennifer Widom. Clustering Association Rules. In: Alex Gray, Per-Åke Larson eds. Proceedings of the 13th International Conference on Data Engineering (ICDE'97). Birmingham U.K. 1997. Los Alamitos: IEEE Computer Society, 1997. 220-231

[114] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar. Clustering Based on Association Rule Hypergraphs. In: Joan Peckham ed. Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97). Tucson, USA. 1997. New York: ACM Press, 1997. 9-13

[115] Jon M. Kleinberg, Christos H. Papadimitriou, Prabhakar Prghavan. Segmentation Problems. In Proceedings of the Thirtieth Annual ACM

Symposium on the Theory of Computing (STOC'98). Dallas, Texas, USA. 1998. New York: ACM Press, 1998, 473-482.

[116] Martin Ester, Hans-Peter Kriegel, Jörg Sander et al. Incremental Clustering for Mining in a Data Warehousing Environment. In: Ashish Gupta, Oded Shmueli, Jennifer Widom eds. Proceedings of 24th International Conference on Very Large Data Bases (VLDB'98). New York City, USA. 1998. San Francisco: Morgan Kaufmann, 1998. 323-333

[117] Cornelis J. van Rijsbergen. Information Retrieval. London: Butterworths. 1979. 67-85

[118] Chidananda Gowda, Edwin. Diday. Symbolic Clustering Using a New Dissimilarity Measure. Pattern Recognition, 1991, 24(6): 567-578

[119] Hinrich Schütze, Craig Silverstein. Projections for Efficient Document Clustering. In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97). Philadelphia, USA. 1997. New York: ACM Press, 1997. 74-81

[120] 卢正鼎, 刘芳, 路松峰. 多媒体数据的聚簇开采. 计算机工程与应用, 2000, 36(8): 16-18

[121] Fang Liu, Zhengding Lu, Songfeng Lu. Mining association rules using clustering. Intelligent Data Analysis, 2001, 5(4): 309-326

[122] 蔡自兴, 徐光祐. 人工智能及其应用. .(2). 北京: 清华大学出版社. 1996. 60-92

[123] 林尧瑞, 马少平. 人工智能导论. 北京: 清华大学出版社. 1989. 50-100

[124] Hannu Toivonen. Sampling Large Databases for Association Rules. In: T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan et al eds. Proceedings of 22nd International Conference on Very Large Data Bases (VLDB'96). Mumbai, India. 1996. San Francisco: Morgan Kaufmann, 1996. 134-145

[125] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman et al. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: Joan Peckham ed. Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97). Tucson, USA. 1997. New York: ACM Press, 1997. 255-264

[126] Peter Buneman. Semistructured Data. In: Joan Peckham ed. Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97). Tucson, USA. 1997. New York: ACM Press, 1997. 117--121

[127] Michal Cutler, Yungming Shih, Weiyi Meng. Using the Structure of HTML Documents to Improve Retrieval. In Proceedings of USENIX Symposium on Internet Technologies and Systems (NSITS'97). Monterey, California. 1997. 241-251

[128] 刘芳, 卢正鼎. 有效地检索 HTML 文档. 小型微型计算机系统, 2000, 21(9): 986-988

[129] Serge Abiteboul. Querying Semi-Structured Data. In: Foto N. Afrati, Phokion Kolaitis eds. Proceedings of 6th International Conference on Database Theory (ICDT'97). Delphi, Greece. 1997. Heidelberg: Springer, 1997. 1-18

[130] Svetlozar Nestorov, Jeffrey Ullman, Janet Wiener, et al. Representative Objects: Concise Representations of Semistructured, Hierarchical Data. In: Alex Gray, Per-Ake Larson eds. Proceedings of the Thirteenth International Conference on Data Engineering. Birmingham U.K. 1997. IEEE, 1997. 79-90

[131] Ke Wang, Huiqing Liu. Schema Discovery for Semistructured Data. In: David Heckerman, Heikki Mannila, Daryl Pregibon eds. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97). Newport Beach, USA. 1997. Menlo Park: AAAI Press, 1997. 271-274

[132] Ke Wang, Huiqing Liu. Discovering Structural Association of Semistructured Data. IEEE Transactions on Knowledge and Data Engineering, 2000, 12(3): 353--371

[133] Yannis Papakonstantinou, Hector Garcia-Molina, Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In: Philip S. Yu, Arbee L. P. Chen eds. Proceedings of the Eleventh International Conference on Data Engineering. Taipei, Taiwan. 1995. Los Alamitos: IEEE, 1995. 251-260

[134] Ron Rymon. Search through Systematic Set Enumeration. In: Bernhard Nebel, Charles Rich, William R. Swartout eds. Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, USA. 1992. San Francisco: Morgan Kaufmann, 1992. 539-550

[135] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, et al. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. eds. Advances in Knowledge Discovery and Data Mining. Menlo Park: AAAI/MIT Press, 1996. 307-328

[136] 路松峰, 卢正鼎. 快速开采最大频繁项目集. 软件学报, 2001, 12(2): 293-297

[137] Jaturon Chattratichat, John Darlington, Yike Guo, et al. An Architecture for Distributed Enterprise Data Mining. In: Peter M. A. Sloot, Marian Bubak, A. G. Hoekstra, et al. eds. Proceedings of High-Performance Computing and Networking, 7th International Conference, HPCN Europe 1999. Amsterdam, Netherlands. 1999. Heidelberg: Springer, 1999. 573-582

[138] Quan Xia, Ling Feng, Hongjun Lu. Supporting Web-Based Database Application Development. In: Arbee L. P. Chen, Frederick H. Lochovsky eds.

Database Systems for Advanced Applications, Proceedings of the Sixth International Conference on Database Systems for Advanced Applications (DASFAA). Hsinchu, Taiwan. 1999. Los Alamitos: IEEE, 1999. 17-24

[139] 路松峰，卢正鼎. 快速开采意外规则. 计算机工程与应用，2000 36(5): 21-23.

[140] Jiawei Han, Jen-Shiun Chiang, Sonny H. S. Chee, et al. DBMiner: A System for Data Mining in Relational Databases and Data Warehouses. In Proceedings CASCON'97: Meeting of Minds. Toronto, Canada. 1997. 249-260

[141] Osmar R. Zaiane, Jiawei Han, Ze-Nian Li et al. Mining Multimedia Data. In: Proceedings of IBM's Centre for Advanced Studies Conference on Meeting of Minds (CASCON'98). 1998. 83-96

[142] Osmar R. Zaiane, Jiawei Han, Ze-Nian Li et al. MultiMediaMiner: A System Prototype for Multimedia Data Mining. In: Laura M. Haas, Ashutosh Tiwary eds. Proceedings ACM SIGMOD International Conference on Management of Data(SIGMOD'98). New York: ACM Press, 1998. 581-583

[143] Raskesh Agrawal, Manish Mehta, John C. Shafer et al. The Quest Data Mining System. In: Evangelos Simoudis, Jiawei Han, Usama M. Fayyad eds. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, USA. 1996. Menlo Park: AAAI Press, 1996. 244-249 .

附录 1 攻读学位期间发表论文目录

1. Fang Liu, Zhengding Lu, Songfeng Lu. Mining association rules using clustering. Intelligent Data Analysis, 2001, 5(4): 309-326
2. 卢正鼎, 刘芳. 利用文本挖掘实现 Web 智能服务. 小型微型计算机系统. 2001, 22(6): 703-705
3. 刘芳, 卢正鼎. 有效地检索 HTML 文档. 小型微型计算机系统, 2000, 21(9): 986-988
4. 刘芳, 胡和平, 路松峰. 半结构化、层次数据的模式发现. 小型微型计算机系统. 2001, 22(1): 84-88
5. 卢正鼎, 刘芳, 路松峰. 多媒体数据的聚簇开采. 计算机工程与应用, 2000, 36(8): 16-18
6. 刘芳, 路松峰, 卢正鼎, 胡和平. 一种基于限制的关联规则数据开采的算法. 华中科技大学学报. 2001, 29(3): 27-29
7. 刘芳, 胡和平. 半结构化数据的模式发现. 微型电脑应用. 2000, 16(2): 13-15
8. 刘芳, 阎红卫. 并行网络搜索引擎. 世界网络与多媒体. 1999, 7(7): 56-57