学校编码：10384　　分类号 _____ 密级 _____
学　　号：9631008　　UDC _____

学　位　论　文

数据库知识发现的方法研究

林 劲 松

指导教师：辜建德教授、李茂青教授

厦门大学自动化系

申请学位级别：硕　　士
专 业 名 称：系　统　工　程
论文提交日期：1999 年 5 月　日
论文答辩日期：1999 年 7 月　日
学位授予单位：厦　门　大　学
学位授予日期：1999 年　月　日

答辩委员会主席：_____

评　　阅　　人：_____

1999 年 5 月

摘　要

数据库知识发现（KDD）是综合人工智能领域机器学习技术和数据库等学科的一个当前非常活跃的研究领域。当今社会正在向数字化社会演变，目前已建立起了覆盖各行各业的数据库，且数据量还在急剧增长中，这远远超过了人类目前已有的分析和理解能力，因此从大量的数据中智能地、自动地提取出有价值的知识和信息的研究，即数据库中的知识发现，具有十分重要的现实意义。

本文首先介绍了 KDD 研究的基本问题，然后针对 KDD 中两种基本的知识模型，即关联规则模型及分类知识模型作重点的阐述及研究，并以中国证券市场为背景对象作了具体的应用研究。作为一种先期工作，研究工作的最终目的在于，在对证券市场历史数据作智能分析的基础上，最终能实现一个有效的证券决策支持系统。

本文的组织安排可分为四个部分。

第一部分由第一章构成，主要介绍了 KDD 的产生背景、概念表述、基本内容、处理过程及知识发现的主要方法，同时分析了 KDD 与相关研究领域的关系。

第二部分是对关联规则发现的研究，由第二章、第三章构成。第二章详细介绍了关联规则发现的基本问题，给出了基本的发现算法，在此基础上本文针对高频物品集的有效生成及数据库的高效扫描提出了一些改进思路，同时介绍了一种变支持度的关联规则更新算法。第三章结合证券交易数据库，提出证券市场关联规则发现的一般模型。通过对物品集作附加时间属性的扩展来改进一般模型，从而实现具有预测功能的证券市场关联规则发现。

第三部分是对分类知识发现的研究，由第四章、第五章构成。第四章重点研究了 KDD 的一种新型理论工具－粗糙集。介绍了粗糙集的分类规则生成原理。属性约简是粗糙集方法优越于其他分类发现方法的地方，本章讨论了全局约简及相对约简的计算问题。最后本文给出了一个完整的基于粗糙集的分类规则发现算法。第五章结合证券市场，提出粗糙集理论用于证券市场买卖规则分类发现的思路。

第四部分由第六章构成，对全文工作进行总结，并给出后续工作的重点及进一步研究方向。

关键词：数据库知识发现、关联规则、分类知识、粗糙集

ABSTRACT

Knowledge discovery in databases(KDD) is a rapidly emerging research field relevant to artificial intelligence, especially to machine learning techniques, and database system. Current society is marching for so-called digital society, the explosive growth of many business, government, and scientific databases have far outpaced our ability to interpret and digest this data, creating a need for a new generation of tools and techniques for automated and intelligent database analysis, and that is the goal of KDD and is undoubted of great practical value.

In this paper, we first present basic notions of KDD, then we focus on two kinds of basic knowledge models: association rules and classification rules.The application background of this paper is based on China's stock market This paper is a prework in some critical methods. To form an efficient decision support system for stock selection based on intelligent and automate analysis on history transaction data is the final goal of the whole research work.

This paper is organized as four parts, and the main content of each part is as below:

The first part includes chapter one as outlines basic notions of KDD including its basic concepts, iterative process procedure, basic discovery methods and its relationship with some related research fields.

The second part includes chapter two and three. In chapter two, some improvements based on basic algorithms of association rules discovery are presented, as increase the efficiency of generation of frequent sets and scanning speed of database. Updating algorithms on renewed data and dynamic changing minimum support are respectively presented and introduced. In chapter three, by extending ordinary item set to those with its attribute of time dimension, we can mine association relationship among different data records in database and generate association rules with forecasting function.

The third part includes chapter four and five. Rough set is a new theoretical tool of KDD, especially in classification rules mining, as is discussed in detail in chapter four. Chapter five focuses on finding buying and selling rules in stock market based on rough set theory.

The last part is chapter six. The total research work is summarized here, the inadequacies and future possible improvements are pointed out. Some important ideas of future work are also pointed out.

Key Words: KDD, association rules, classification rules, rough set

目　录

第一章 KDD 概述

§1.1 KDD 的产生背景

随着数据库技术的不断发展及数据库管理系统的广泛应用，数据库中存储的数据量急剧增大。据估计，全世界的数据量每 20 个月翻一番，人们保存如此大量的数据,一是因为计算机技术的发展使之变得方便可行，二是因为这些数据有巨大的潜在作用。然而，如何有效地使用这些数据却成为一个问题，因为常常是数据丰富而信息贫乏（Data Rich and Information Poor）。利用数据库管理系统提供的数据管理和简单的处理功能，人们可以在这些数据之上进行商业分析及科学研究，但已有的技术还不能很好地发挥这些数据的作用。一方面，数据量的爆炸性增长使传统的手工处理方法变得不切合实际。另一方面，人们已不满足于对数据的简单维护及查询，而是希望能够找到对数据进行较高层次处理的技术，从中找到规律和模式，帮助人们更好的利用数据进行决策和研究。

对数据进行深层次处理，从而得到关于数据的总体特征及对发展趋势作出预测，这是传统的数据库管理系统所无法达到的功能，因此，我们需要引进自动化程度更高、效率更好的数据处理方法。

机器学习通过对数据对象之间关系的分析可以提出隐含在数据中的模式，找到有用的信息，也即知识，避免了数据资源的浪费。正是由于实际工作的需要及相关技术的发展，将机器学习用于大型数据库的数据库中的知识发现（Knowledge Discovery in Database-KDD）技术逐渐发展起来。

§1.2 KDD 的概念表述

简单而言，KDD 是从大量数据中提取出可信的、新颖的、有效的并能被人理解的模式的高级处理过程。我们可以对上述定义作如下详细的解释。

数据：数据是指一个有关事实 F 的集合，它记录了数据有关方面的原始信息，如商场销售数据，股市交易数据，银行客户信息等。KDD 处理的数据来源于现实世界，数据可能并不规范，一般要对数据进行预处理，使之适合于知识提取。

模式：模式可视为我们所说的知识，它给出了数据的特性和数据之间的关系，是对数据包含的信息更抽象的描述。模式的表示方式很多，显式的如：“成绩优秀的学生学习都很刻苦”，非显式的如用神经网络表示的模式，它是通过网络中的连接权值体现出来的。

处理过程：KDD 是一个多步骤的对大量数据进行分析的高级处理过程，包括数据预处理，模式提取，知识提取及过程优化。知识提取往往需要多次反复，通过对相关数据的再处理及知识学习算法的优化，不断提高学习效率。

可信、新颖的：KDD 从数据中发现的模式必须有一定程度的正确性及新颖性，否则 KDD 即毫无意义。KDD 可以对已有的知识进行验证，但更为重要的是去发现新的知识或者对现有知识再拓展以得到更全面的、更具有实际意义的知识。发现的知识必须通过实践的检验并通过在实际应用中发现的问题对学习数据和策略进行修正，从而得到更精确的知识。

可被人理解：要求发现的模式能被用户理解，目前它主要是体现在简洁性上。应当将发现的知识以直观易用的方式提供给用户。

§1.3 KDD 的处理过程

综合上述对于“数据库中的知识发现”这一概念的理解，我们利用图 1.1 来直观地描述 KDD 的处理过程，并对该过程作进一步的解释。KDD 过程可粗略地理解为三部曲：数据准备（data preparation）、数据开采以及结果的解释和评估（interpretation and evaluation）。

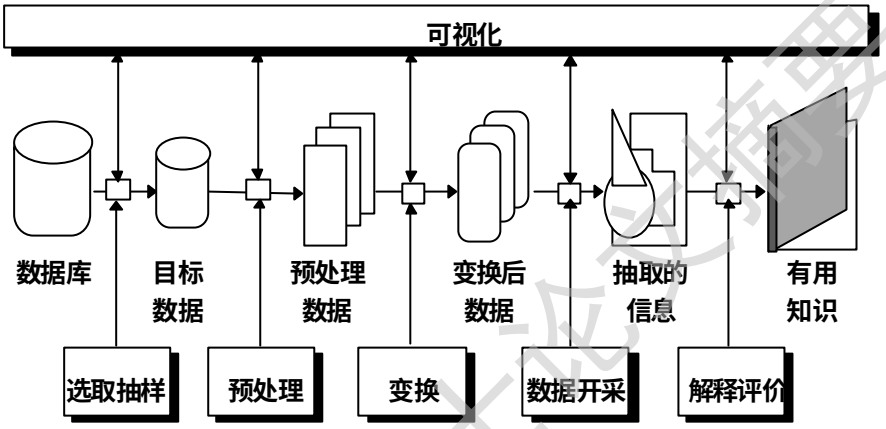


图 1.1 KDD 过程示意图

数据准备

数据准备又可分为三个子步骤：数据选取（data selection）、数据预处理（data preprocessing）和数据变换（data transformation）。

数据选取的目的是确定发现任务的操作对象，即目标数据（target data），它是根据用户的需要从原始数据库中抽取的一组数据。

数据预处理是指对数据进行再加工，一般可能包括检查数据的完整性、一致形、消除噪声、利用统计方法推导计算缺值数据、消除重复记录、完成数据类型转换（如把连续值数据转换为离散型的数据，以便于符号归纳，或是把离散型的转换为连续型的，以便于神经网络归纳）等。

数据变换的主要目的是消减数据维数或降维（dimension reduction），即从初始特征中找出真正有用的特征以减少数据开采时要考虑的特征或变量个数。

数据开采阶段

数据开采阶段首先要确定开采的任务或目的是什么，即确定 KDD 的目标，如数据总结、分类、聚类、关联规则发现或序列模式发现等。确定了开采任务后，就要决定使用什么样的开采算法。同样的任务可以用不同的算法来实现，选择实现算法有两个考虑因素：一是不同的数据有不同的特点，因此需要用与之相关的算法来开采；二是用户或实际运行系统的要求，有的用户可能希望获取描述型的（descriptive）、容易理解的知识（采用规则表示的开采方法显然要好于神经网络

之类的方法），而有的用户或系统的目的是获取预测准确度尽可能高的预测型（predictive）知识。

完成了上述准备工作后，就可以实施数据开采操作了。具体的数据开采方法将在后面章节中作较为详细的论述。

结果解释和评估

数据开采阶段发现出来的模式，经过用户或机器的评估，可能存在冗余或无关的模式，这时需要将其剔除；也有可能模式不满足用户要求，这时需要将整个发现过程退回到发现阶段之前，如重新选取数据、采用新的数据变换方法、设定新的数据开采参数值，甚至换一种开采算法（如当发现任务是分类时，有多种分类方法，不同的方法对不同的数据有不同的效果）。所以，KDD 是一个多步骤的，可能多次反复的复杂过程。

另外，KDD 由于最终是面向人类用户的，因此可能要对发现的模式进行可视化，或者把结果转换为用户易懂的另一种表示，如把分类决策树转换为“if…then…”规则。

§1.4 KDD 的分类及相关技术

1.知识的种类

究竟我们希望从数据库中发现什么样的知识呢？

R.Agrawal 指出了三种要在数据库中发现的知识[10]：分类知识、关联规则和序列模式。

分类知识发现在数据开采中是一项非常重要的任务。分类试图用有导师和无导师的方法把给定的数据集分成特定类，其目的是学会一个分类函数或分类模型（也常常称作分类器），该模型能把数据库中的数据项映射到给定类别中的某一个。分类和回归都可用于预测。预测的目的是从利用历史数据纪录中自动推导出对给定数据的推广描述，从而能对未来数据进行预测。和回归方法不同的是，分类的输出是离散的类别值，而回归的输出则是连续数值。要构造分类器，需要有一个训练样本数据集作为输入。训练集由一组数据库记录或元组构成，每个元组是一个由有关字段（又称属性或特征）值组成的特征向量，除了这些外，训练样本还有一个类别标记。一个具体样本的形式可为：(v_1，v_2，…，v_n；c)；其中 v_i 表示字段值，c 表示类别。

分类的无导师方法也称为聚类，聚类是把一组个体按照相似性归成若干类别，即“物以类聚”。它的目的是使得属于同一类别的个体之间的距离尽可能的小而不同类别上的个体间的距离尽可能的大。从而发现一个概念集用于描述数据集的特点，并且它能应用在把未知的数据对象归类。如银行需要将其信用卡用户分类以决定是否提供贷款。

关联规则发现是发现形式如下的一种蕴含或规则，$X \Rightarrow Y$，其中 X 和 Y 分别是两个物品集合，这两个物品集中没有共同的物品。用于关联规则发现的对象主要是事务型数据库（transactional databases），其中针对的应用则是售货数据，也称货篮数据(basket data)。一个事务一般由如下几个部分组成：事务处理时间，一组顾客购买的物品（items），有时也有顾客标识号（如信用卡号）。由于条形

码技术的发展，零售部门可以利用前端收款机收集存储大量的售货数据。因此，如果对这些历史事务数据进行分析，则可对顾客的购买行为提供极有价值的信息。例如，可以帮助如何摆放货架上的商品（如把顾客经常同时买的商品放在一起），帮助如何规划市场（怎样相互搭配进货）。由此可见，从事务数据中发现关联规则，对于改进零售业等商业活动的决策非常重要。在发现序列模式时，时间通常是给定的一个额外属性，问题集中在去发现动态的模式，如在股市数据及客户数据中，发现在给定的一个时间窗里经常发生的固定事件。

2.KDD 分类

KDD 的核心部分是数据开采阶段，它包含了许多的数据开采算法。有必要对这些方法进行分门别类。一个完整的算法通常包括三个部分：输入、输出和处理过程。数据开采算法的输入是数据库，算法的输出是要发现的知识或模式，算法的处理过程则涉及具体的搜索方法。从算法的输入、输出和处理过程三个角度分，我们可以确定这样几种分类标准：开采对象、开采任务、开采方法。

根据开采任务分，不外乎上述三种知识发现任务，具体而言，可能包括分类或预测模型知识发现、数据总结、数据聚类、关联规则发现、序列模式发现、依赖关系或依赖模型发现、异常和趋势发现等等。本文的重点在于关联规则发现及分类规则发现的方法研究及其应用，我们将另辟章节对其给以重点介绍。

根据开采对象分，有如下若干种数据库或数据源：关系数据库、面向对象数据库、空间数据库、时态数据库、文本数据源、多媒体数据库、异质数据库、遗产（legacy）数据库，以及环球网（Web）。

根据开采方法分，可粗分为：统计方法、机器学习方法、神经网络方法和数据库方法。统计方法中，可细分为：回归分析（多元回归、自回归等）、判别分析（贝叶斯判别、费歇尔判别、非参数判别等）、聚类分析（系统聚类、动态聚类等）、探索性分析（主元分析法、相关分析法等）等。机器学习中，可细分为：归纳学习方法（决策树、规则归纳等）、基于范例学习、遗传算法等。神经网络方法中，可细分为：前向神经网络（BP 算法等）、自组织神经网络（自组织特征映射、竞争学习等）等。数据库方法主要是多维数据分析或 OLAP 方法，另外还有面向属性的归纳方法。

数据总结的目的是对数据进行浓缩，给出它的紧凑描述。传统的也是最简单的数据总结方法是计算出数据库的各个字段上的求和值、平均值、方差值等统计值，或者用直方图、饼状图等图形方式表示。此处主要从数据泛化（data generalization）的角度来讨论数据总结。数据泛化是一种把数据库中的有关数据从低层次抽象到高层次上的过程。由于数据库上的数据或对象所包含的信息总是最原始基本的信息。人们有时希望能从较高层次的视图上处理或浏览数据，因此需要对数据进行不同层次上的泛化以适应各种查询要求。数据泛化目前主要有两种技术：多维数据分析方法和面向属性的归纳方法。多维数据分析（multidimensional data analysis）方法是一种数据仓库（dataware housing）技术，也称作联机分析处理（On-Line Analytical Processing，简称 OLAP）或数据立方体（data cube）方法。数据仓库是面向决策支持的、集成的、稳定的、不同时间的历史数据集合。决策的前提是数据分析。在数据分析中经常要用到诸如求和、总计、平均、最大、最小等汇集操作，这类操作的计算量特别大。因此一种很自然的想法是，把汇集操作结果预先计算并存储起来，以便于决策支持系统或其它 KDD 应用系统的使用。存储汇集操作结果的地方称作多维数据库，或形象地称

为数据立方体。我们知道，汇集操作涉及两项输入：要汇集的属性和汇集范围。要汇集属性是要在其上完成诸如求和等计算的属性，汇集范围相当于查询语句的 **Where** 条件语句，它由某些属性上的取值来确定。为了对数据进行灵活汇集，属性的取值有必要形成各种泛化级上的层次结构。如日期属性，可以形成“小时”、“天”、“周”、“月”、“季度”和“年”这样的越来越抽象或泛化的层次。对属性值规定了各种层次上的取值后，就可以在多维数据库上完成提升（roll-up）和下沉（drill-down）操作，提升操作能使得数据在高层次上显示，如把日期属性从“周”泛化到“月”级别上，则能以月为单位来观看数据；下沉操作则相反，可以使得用户从更细微的角度来考察数据。采用多维数据分析方法进行数据总结，它针对的是数据仓库，数据仓库存储的是脱机的历史数据。为了处理联机数据，人们又提出了一种面向属性的归纳方法。它的思路是，直接对用户感兴趣的数据视图（用一般的 SQL 查询语言即可获得）进行泛化，而不是象多维数据分析方法那样预先就存储好了泛化数据。这种数据泛化技术被称之为面向属性的归纳方法（attribute-oriented induction method）。该方法涉及的泛化操作有：属性去除操作、概念树提升、属性阈值控制、频数传播，以及其它的数据汇集操作。原始关系经过泛化操作后得到的是一个泛化关系（generalized relation），它从较高的层次上总结了在低层次上的原始关系。有了泛化关系后，就可以对它进行各种深入的操作而生成满足用户需要的知识，如在泛化关系基础上生成特性规则（characteristic rules）、判别规则（discriminant rules）、分类规则（classification rules），以及关联规则（association rules）等。面向属性的归纳方法的一个核心部件是属性背景知识，它表示为各个属性的概念层次树。面向属性的归纳方法是一种通用的技术，不仅可用于关系数据库，同样可用于面向对象数据库、空间数据库，以及其它类型的数据库。这种方法是一种泛化技术的方法，不适用于挖掘低层次上的具体模式。

§1.5 KDD 与相关研究领域的关系

综合上述的认识与理解，可以看出，KDD 就是利用机器学习的方法从数据库中提取有价值知识的过程，是数据库技术及机器学习两个学科的交叉学科。数据库技术侧重于研究对数据进行存储处理的高效率方法，机器学习侧重于研究从数据中提取知识的新方法。KDD 利用了数据库技术对数据作前端处理，利用了机器学方法从处理后的数据提取有用的知识。

机器学习和 KDD 都研究如何从数据中抽取模式和模型的理论和算法。KDD 中的不少方法就源于机器学习。从算法角度看，KDD（数据开采阶段）与机器学习相比，KDD 主要研究真实世界中的大规模数据上的学习或发现算法。从系统模型角度看，KDD 关心整个发现过程，不仅仅是数据开采算法，还强调数据的准备和发现结果的解释和评估以及人机交互等等方面。事实上，除了机器学习，KDD 和其他从数据中抽取模式或导出模型的方法之间的区别也类似，如模式识别和神经网络。

统计和 KDD 有很多研究的共同之处。数据分析是应用统计学的主要任务，特别是探索性数据分析，现在又出现了智能数据分析的概念，都是和 KDD 关系密切。统计方法既可在 KDD 中的开采阶段提供建模手段，还可在数据预处理和数据转换阶段用于消除噪声和消减维数。

数据库为 KDD 提供数据操纵管理的理论和技术基础。这方面目前与 KDD

最相关的当属盛行的数据仓库技术，建立数据仓库的目的就是为了面向决策支持和联机分析。

KDD 的发展是应用驱动，对不同应用领域的背景知识的合理利用，也会增加学习算法的效率。

KDD 作为一个新的研究领域，有特定的要求和任务（参见其概念描述），所处理的对象也与传统的统计分析方法、机器学习有极大的区别。例如从所处理的数据集大小上来看，机器学习的训练集很少超过几千条,KDD 则要处理来源于实际中的大规模数据库，数据量可能非常之大，必须重点考虑算法的效率及可扩充性。此外，来源于现实世界的数据，其完整性及一致性也很难保证，对算法必然提出更高的要求。必须在现有相关领域技术的基础上研究新的集成技术，以达到知识发现的目的。

第二章 关联规则发现

在数据库知识发现的结果中，一种重要的知识形式是规则知识，规则知识的描述能力强且易于转化为自然语言。关联规则是规则知识的一种，是 KDD 中的一项重要任务和研究重点。

关联规则的研究最早来源于对大型商场零售数据的分析。由于条形码技术的发展，零售部门可以利用前端收款机收集存储大量的售货数据。人们很自然地希望对这些历史事务数据进行分析，从而发现关于顾客购买行为的有价值的信息。例如，可以帮助如何摆放货架上的商品（如把顾客经常同时买的商品放在一起），帮助如何规划市场（怎样相互搭配进货）。有趣的“啤酒与尿布”规则，就揭示了这样一种信息。由此可见，从事务数据中发现关联规则，对于改进零售业等商业活动的决策非常重要。

关联规则在多种分析和预测任务中都有极其重要的作用。采用其技术的应用系统已在金融预测，医疗诊断，市场选择及许多应用领域取得了显著成果。因而也吸引了许多研究人员的兴趣。

关联规则发现(ARM)是本文的研究重点。我们将关联规则发现的相关思路、发现算法应用于中国证券市场，用于发现不同个股之间的关联关系，产生能够指导预测与投资的关联规则。具体问题将在下一章节中描述。

§2.1 关联规则定义

关联规则发现的一般过程可以利用形式化语言抽象地定义为，给定一组数据 F，一种语言 L，给定确定性 C 的测量尺度，使用语言 L 将一个模式描述为陈述 S，该陈述 S 揭示了 F 中一个子集 F_S 的内在关系和相应的确定性 C。显然陈述 S 比对 F_S 中所有事实的简单罗列更明确。如果某一模式是有趣的（新颖、有效、可理解的－参见 KDD 定义）和确定的（满足用户给定标准），则该模式可上升为知识。知识发现程序（算法）用于对数据库中的事实集合进行检测，根据知识判别的标准输出模式（知识）。

我们依据上述思路，给出关联规则发现相关问题的定义及其描述。

1.高频物品集（Frequent Sets）

高频物品集是产生关联规则的基础，我们首先给出它的定义。

首先给出一些概念基础。

$I=\{i_1、i_2、i_3、\ \dots\ i_m\}$为一文字集合,称为一组物品集（itemset），如零售商店出售给客户的一组物品，证券市场中的股票品种。

定义事务(Transaction)（作为 KDD 领域中的术语,不同于 DBMS 中的”事务”)[2] T 是一物品集，且其为物品集 I 的任意子集，所以有 $T \subseteq I$。和一般的集合相同，T 中不存在重复元素(物品)，此外，为方便今后算法的处理，我们还假定事务数据库中的每一事务包含的物品都经过排序。这也是事务 T 与普通集合

的不同之处。

定义 D 为一组事务集（称为事务数据库），为区分 D 中的不同事务，每一事务均有一独立标识符，称为 TID。

设 X 是一个物品集，且 $X \subseteq I$，如果一事务 T 包含 X 中的所有物品，即 $X \subseteq T$ 则称事务 T 支持 X。我们可以在 D 中找出所有支持 X 的事务，它们亦构成一个集合，其元素为各个事务的 TID，可以用 T(X)加以标记这个集合。

定义物品集 X 的支持度（support），简写为 SUPP(X).。它代表了事务集 D 中所有支持 X 的部分。我们可以事先给定一个最小支持度 Smin，如果 SUPP(X)> Smin，则称 X 为高频物品集。

一个物品集 X 被称为高频物品集，表示 X 在事务数据库(D)中的所有事务(T)中得到了足够高频程度的支持，用户给定的最小支持度 Smin 则给出了度量这种高频程度的最低标准。定义高频物品集的意义在于找到 D 中有足够的发生频度的物品集合，因为正是它们才有可能揭示不同物品之间的普遍的关联关系，才能引起用户的足够兴趣。

此外，我们还称集合势为 K（$|X| = k$）的物品集 X 为 K-物品集（K-itemset）

高频物品集具有如下三个性质，它们是构造关联规则发现算法的基础。

性质 **1**(Support for Subsets) 对于物品集 A、B，如果 $A \subseteq B$，则 supp(A)≥supp(B)，因为事务数据库 D 中，支持 B 的事务也必须支持 A。

性质 **2** (Supersets of Infreqent Sets are Infrequent) supp(A)<Smin，即物品集 A 不能满足最小支持度，则 A 的所有超集 B 也不会是高频集。因为由性质 1 可得 supp(B)≤supp(A)<Smin。

性质 **3** (Subsets of Frequent Sets are Frequent) 物品集 B 为高频集，则 B 的所有子集在 D 中也是高频集。特别的，如果 A={i_1、i_2、…i_k}是高频集，它的所有 k-1 子集都是高频集。反之不成立。

2.关联规则

关联规则是形式如下的一种规则，“在购买面包和黄油的顾客中，有 90%的人同时也买了牛奶”，简单而言，关联规则要揭示数据库中不同物品的关联关系。我们利用上节所讨论的基本概念来表述关联规则的数学形式。

R： $X \Rightarrow Y$ **；其中** $X \subseteq I, Y \subseteq I$, **且** $X \cap Y = \varnothing$

这种形式的规则有其预测的意义，即若事务数据库中的某一事务支持 X，则它也将以一定的概率支持 Y，这个概率称为规则的信任度，记作 conf（R）。因此，关联规则又可以表示为：

R：（P，S，C）

P 表示模式（如关联规则的自然语言描述），S 和 C 分别表示支持度和信任度。

上述数学形式的意义是明显的，即一个事务集 D 中，有 c%的事务，既包含 X，又包含 Y，有 s%的事务中包含 $X \cup Y$。

支持度和信任度是描述一个关联规则的两个重要参数，我们有必要详细讨论

一下这两个参数的意义，并给出他们的计算公式。

如果不考虑关联规则的支持度和可信度，那么在事务数据库中存在无穷多的关联规则。支持度限制即对事务数据库 D 中支持某一规则的事务个数的限制。也就是 D 中既满足规则的前项，又满足规则后项的事务所占的百分比。而信任度则描述了一项规则的可信程度。信任度是规则强度的度量，支持度则对应了规则的统计意义。引进支持度还因为我们只对支持度超过一定阈值的规则感兴趣，支持度太小则说明该规则并非一种普遍的规律，不具有指导意义。

有了上述认识基础，很容易得到支持度与信任度的计算公式。

规则的支持度用 supp(R)=supp(X∪Y)表示。则信任度为一条件概率。

$$\text{conf(R)=P}(Y \subseteq T \mid X \subseteq T)=\frac{p(Y \subseteq T \wedge X \subseteq T)}{p(X \subseteq T)}=\text{supp(X}\cup Y)/\text{supp(X)}$$

给定最小支持度 Smin，最小信任度 Cmin，当某一关联规则 R，其 supp(R)> Smin，conf(R)> Cmin，我们称 R 为强规则。

进一步讨论关联规则的性质。

1、$X \cap Y = \varnothing$，表示 X、Y 两个物品集互相独立，没有相同的物品项。这个条件并非必须。限定这个条件目的在于避免产生冗余和无意义的规则。很显然，$X \Rightarrow X$ (conf=1)，以及等价于 $X \Rightarrow Y$ 的 $X \Rightarrow X \cup Y$ 都是无意义的。基于同样的原因，也要求 $Y \neq \varnothing$，因为根据信任度公式：

$$\text{conf= supp(X}\cup Y)/\text{supp(X)} = \text{supp(X}\cup \varnothing)/\text{supp(X)} = 1$$

从这里的分析亦可看出，$X \Rightarrow X$ 和 $X \Rightarrow \varnothing$ 是等价的。

2、关联规则的前项与后项都不可合并。即：

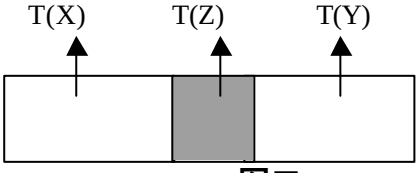
$$X \Rightarrow Z,\ Y \Rightarrow Z \not\rightarrow \quad X \cup Y \Rightarrow Z$$
$$X \Rightarrow Y,\ X \Rightarrow Z \not\rightarrow \quad X \Rightarrow Y \cup Z$$

因为 D 中的事务支持 $X \Rightarrow Z$，$Y \Rightarrow Z$，并不代表 $X \cup Y$ 在 D 中会有足够的支持度及信任度。

3、关联规则的前项不可分解，后项可分解。即：

$X \cup Y \Rightarrow Z$ 不一定能得到 $X \Rightarrow Z$，$Y \Rightarrow Z$

因为可能存在这种情形，对于 D 中的事务，Z 仅当 X、Y 出现时才出现，即 supp(Z)=supp(XY)，若有 supp(X)，supp(Y)>supp(XY)，则分解后的两个规则将不会有足够大的信任度。我们可以借用图 2 直观地解释这一情形。



图二

反之，关联规则的后项是可分解的，即

$X \Rightarrow Y \cup Z$ 可以分解得到 $X \Rightarrow Y$，$X \Rightarrow Z$

因为 $\text{supp(XY)} \geq \text{supp(XYZ)}$，$\text{supp(XZ)} \geq \text{supp(XYZ)}$，这种分解不仅成立，而且得到的规则具有更大的支持度、信任度。但在实际应用中，这种分解的意义不大，因为结果常常是冗余的。

4、关联规则不具有传递性。

从 $X \Rightarrow Y$，$Y \Rightarrow Z$，我们无法推出 $X \Rightarrow Z$

给出反例：

假设 $T(X) \subset T(Y) \subset T(Z)$，给定 Cmin，当 conf($X \Rightarrow Y$)=conf($Y \Rightarrow Z$)=Cmin 时，很容易有 conf($X \Rightarrow Z$)=$Cmin^2$< Cmin(Cmin<1)，即 $X \Rightarrow Z$ 无法满足最小信任度要求。

§2.2 关联规则的发现过程及其基本算法

关联规则发现的任务本质上就是在大型数据库中发现强规则。这一过程可以分解为如下两个子过程：

1.发现高频物品集。即在数据库中发现那些高于预设最小支持度的物品集。

2.利用高频物品集产生所有满足最小信任度的关联规则。

第一步决定了整个关联规则发现的性能，一旦寻找出高频物品集，相应的关联规则就可以用很直接的方式获得。所以，研究工作主要集中在寻找高频物品集的有效算法上。目前在关联规则发现的研究论文中，提出的好几种方法都是围绕此问题而来的。其中 Apriori 算法[13]是许多后续的关联规则发现算法的基础，我们将其作为一个典型算法给予重点介绍。

Apriori 算法流程：

L_1：={frequent 1-itemsets} //对数据库第一遍扫描

K：=2 //K 代表对数据库的扫描次数

While {L_{k-1}<>0} do

Begin

C_k：= New Candidates of size k generated form L_{k-1};//从 L_{k-1} 中产生的大小为 k 的候选物品集

For all transactions $t \in D$ do

Increment the count of all candidates in C_k that are contained in t;

L_k:=All canditates in C_k with minmum support

K=k+1

End

Answer:= $\bigcup_k L_k$

上述流程中候选物品集的产生算法可细化为：

Insert into C_{k+1}

Select a.item1,a.item2,. a.itemk,b.itemk From Lk a,Lk,b;

Where item1=b.item1,a.item2=b.item2,. a.item(k-1)=b.item(k-1),a.itemk<b.itemk

(根据性质 3 对候选集进行剪枝运算，去除不可能的候选集，减低候选集的规模大小)

forall itemset $C \in C_{k+1}$ do

forall K-subsets S of C do

if (S $\notin Lk$) then

delete C from C_{k+1}

end

算法解释：

物品集经过预先排序，在即将产生规模为 K＋1 的候选集时，以先前生成的 Lk（Lk 中的所有物品集都是规模为 K 的高频集）作为输入，在 Lk 构成的集合对中搜索出前 K－1 项物品都相同的两个不同物品集，它们的第 K 项（最大项）不同，在这里，为了不产生重复的候选集，要求第二个集合的第 K 项在物品的排序序号上要大于第一个集合第 K 项的排序序号。满足上述情形后，第二个物品集的第 K 项就作为新的扩展项加入到第一个集合中，产生规模为 K＋1 的候选集。到此为止，这个 K＋1 候选集只有两个子集被自然地确认为高频集，所以在剪枝阶段，还要对其他子集再作进一步的检验，以保证得到真正的 K＋1 候选集。

举例说明，{1、3、4、6},{1、3、4、8}是两个规模为 4 的高频集，依据上述讨论，我们可以利用它们产生规模为 5 的候选集{1、3、4、6、8}，这个候选集仍有{1、3、6、8}，{1、4、6、8}，{3、4、6、8}三个子集有待于在 Lk 中检验，若它们中的任何一个不在 Lk 中，这个新产生的高频集将被抛弃。

由上述讨论可见，Apriori 是一种宽度优先算法，通过对数据库 D 的多趟扫描（Pass）来发现所有的高频物品集，在每一趟 k 中只考虑具有同一长度 K 的所有物品集（即第 K 次扫描只考虑 K－物品集）。在第 1 趟扫描中，Apriori 算法计算物品集合 I 中所有单个物品的支持度，生成所有长度为 1 的高频物品集。在后续的每一趟 k 中，首先以前一趟中所发现的所有高频物品集为基础，生成所有新的候选物品集（Candidate Itemsets），即潜在的高频物品集，然后扫描数据库 D，计算这些高频物品集的支持度，最后确定候选物品集中哪一些真正成为高频物品集。重复上述过程直到再也发现不了新的高频物品集。

算法效率的关键在于生成较小的候选物品集，也就是尽可能不生成和计算那些不可能成为高频物品集的候选物品集。实际上，为了实现这一点，所有已知的算法都利用了高频物品集的基本性质 3，即一个高频物品集的任一子集必定也是高频物品集。

Apriori 的特点就在于：

1.在产生候选集的过程中引入了剪枝过程，减小了候选集的规模。

2.候选集的产生和候选集中每一物品集在数据库中的支持度计算是分开的。这无疑减少了数据库扫描的次数。

§2.3 基本算法的性能分析及改进

上述由 R..Agrawal 等人给出的基本算法，能够应用于从数据库中发现单层关联规则知识，但是由于在高频集的搜索上要花费大量的时间，因而在算法效率上较低，特别是当其面对大型数据库产生的大规模数据集时，处理起来性能很不理想。有必要提出新的改进算法。

本部分通过对算法作详细的分析，指出算法性能瓶颈所在，并给出算法改进的方向。

从 Apriori 算法的思路来看，对运行时间（性能）产生影响的因素主要有两个，即每一阶段中产生的候选物品集数目的大小以及对事务数据库 D 的扫描次数。尽管 Apriori 算法已经根据关联规则的性质 3 内含了候选物品集的剪枝过程，避免了许多无效候选物品的产生，但对于大规模数据集而言，每一阶段的候选物品集仍然数目巨大，尤其在前两个阶段(即 K<=2 时)。举例说明，假设 D 中含有

800 个可能的物品，它们均满足用户设定的最小支持度，则 L1 中有 800 项，利用 L1*L1 产生 C2，C2 中将会有 C_{800}^{2}(316,400)项规模为 2 的候选物品集，此后还要通过扫描数据库依次确定 C2 中每一物品集的支持度，以最终得到 L2。K>2 时，虽然 Ck，Lk 中的数目会大大减少，但耗费处理器的时间仍然相当之大。

此外，在规则挖掘过程中，为了最终聚焦到用户感兴趣的，有效的关联规则，我们还需要不断地调整支持度，而支持度的每一次调整都将导致上述过程的重复进行，而不能利用先前已完成过程的有用信息。

基于上述分析，算法改进应从提高候选物品集的产生效率及尽量减少数据库的扫描次数入手。

本文给出一些提高数据库扫描效率的思路。

1.剩余元组优化技术

在数据库扫描过程中，依据所剩余未扫描元组的个数，我们可以在扫描完整个数据库之前即知道某个候选集最终不可能成为高频集，从而提前结束扫描，并将该集合从候选集中删除。依据如下：

假设 X＋Y 是由规模为 K 的高频集 X 扩展而来的候选集，D 中有 x 项记录包含 X，X＋Y 的当前支持度计数为 s，且扫描进行到含有 X 的第 c 个记录，则 D 中最多还有(x-c)个记录可能包含 X＋Y，比较（x-c+s）和最小支持度 minsupp（此处用 D 中包含某个候选集的记录数目代表）的值。如果(x-c-s)<minsupp，说明 X+Y 的支持度不能满足要求，它不可能成为最终的高频集，此时可将其从候选集中删除，此后不再针对其作扫描计数。

2.候选集计数判定过程优化

D 中的每一条记录都需要对某一个候选集作检验，判定物品项中是否包含了这个候选集，判定为真值时，才对相应的候选集的计数器作加一的操作。由于是独立地检验判定每个候选集，当候选集的各个项目之间包含大量的重复元素时，这一过程就包含了大量的重复判定，浪费了处理时间。举例说明，对于{A,B}及{A,C}两个候选集合，它们拥有相同的元素（物品）A，D 中的每一记录都要对 A 作两次的判定。

此处我们引入树的数据结构来避免这一重复，加速判定过程。

在每次对 D 扫描之前，以 D 中的每个物品生成一个结点作为树的根，候选物品集的每一个候选者生成一个叶结点；如果某个物品在一个候选者中，则保证在该物品所在的根结点到此叶结点之间有一条链相连。扫描开始，每读入一条记录，将各个物品的取值置入物品所对应的根结点，物品存在取值为 1，反之为 0。若某物品取值为 0，则与该根结点所相连的叶结点取值为 false（所有叶结点的初值为 true）。最后检查所有叶结点的取值情况。对那些取值为 true 的叶结点，表明其所对应的候选集存在于 D 中的当前记录里，相应的支持度计数器加一。以下给出一个具体例子的图示描述。

D 中有 A，B，C，D 四种可能物品，候选物品集为{ABC,ABD,BCD,ACD}，则可生成如图 1 的树结构。假设扫描过程中，读入一个记录为{1,1,1,0}，此时树结构如图 2，由于 D 物品取值为 0，则与 D 相关的三个叶结点 ABD,BCD,ACD 取值应为 false，如图 3 所示。即我们将只对{ABC}这个候选者作计数操作。

算法描述如下：

for D 中的每一个物品 do