软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(12):2965−2980 [doi: 10.3724/SP.J.1001.2011.03907]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种高效的多层和概化关联规则挖掘方法*

毛宇星+, 陈彤兵, 施伯乐

(复旦大学 计算机科学技术学院,上海 200433)

Efficient Method for Mining Multiple-Level and Generalized Association Rules

MAO Yu-Xing+, CHEN Tong-Bing, SHI Bai-Le

(School of Computer Science, Fudan University, Shanghai 200433, China)

+ Corresponding author: E-mail: myuxing@fudan.edu.cn

Mao YX, Chen TB, Shi BL. Efficient method for mining multiple-level and generalized association rules. *Journal of Software*, 2011,22(12):2965−2980. http://www.jos.org.cn/1000-9825/3907.htm

Abstract: This paper proposes a idea for mining multiple-level and generalized association rules. First, an item correlation model is set up, based on the domain knowledge and clusters the items according to their correlation. Secondly, the transaction database, based on the item clusters, are reduced which make the transaction database smaller. Finally, the partitioned transaction databases are projected onto a compact structure called AFOPT-tree and find the frequent itemsets from the AFOPT. Based on the proposed idea, this paper proposes a top-down algorithm TD-CBP-MLARM and a bottom-up algorithm BU-CBP-MLARM to mine the multiple-level association rules. Additionally, this paper extends the idea to a generalized mining association rule and gives a new efficient algorithm CBP-GARM. The experiments show that the proposed algorithms not only corrects and completes mining results, but also outperform the well-known and current algorithms in mining effectiveness.

Key words: taxonomy data; multiple-level association rule; generalized association rule; hierarchical clustering; reduction

摘 要: 通过对分类数据的深入研究,提出了一种高效的多层关联规则挖掘方法:首先,根据分类数据所在的领域知识构建基于领域知识的项相关性模型 DICM(domain knowledge-based item correlation model),并通过该模型对分类数据的项进行层次聚类;然后,基于项的聚类结果对事务数据库进行约简划分;最后,将约简划分后的事务数据库映射至一种压缩的 AFOPT 树形结构,并通过遍历 AFOPT 树替代原事务数据库来挖掘频繁项集.由于缩小了事务数据库规模,并采用了压缩的AFOPT结构,所提出的方法有效地节省了算法的I/O时间,极大地提升了多层关联规则的挖掘效率.基于该方法,给出了一种自顶向下的多层关联规则挖掘算法 TD-CBP-MLARM 和一种自底向上的多层关联规则挖掘算法 BU-CBP-MLARM.此外,还将该挖掘方法成功扩展至概化关联规则挖掘领域,提出了一种高效的概化关联规则挖掘算法 CBP-GARM.通过大量人工随机生成数据的实验证明,所提出的多层和概化关联规则挖掘算法不仅可以确保频繁项集挖掘结果的正确性和完整性,还比现有同类最新算法具有更好的挖掘效率和扩展性.

关键词: 分类数据;多层关联规则;概化关联规则;层次聚类;约简划分

中图法分类号: TP311 **文献标识码**: A

* 基金项目: 国家自然科学基金重大研究计划重点项目(90818023); 国家重点基础研究发展计划(973)(2005CB321905)
收稿时间: 2009-11-23; 定稿时间: 2010-07-05

关联规则挖掘(association rule mining,简称 ARM)[1−4]旨在从大量事务数据库中发现事务之间有趣的关联关系,以揭示隐藏其中的客户行为模式.传统关联规则挖掘的一个重要前提假设是:构成事务的项都是单层概念的.然而在实际应用领域中,事务的项却大多具有层次概念.如图 1 所示的零售业数据,其中“面包”按照商品成分的不同可以分为“白面包”和“全麦面包”,这时“全麦面包”与“面包”之间就存在一种层次分类关系.基于分类数据的关联规则不仅普遍存在,而且可以提供比传统关联规则更丰富、更具参考价值的信息.

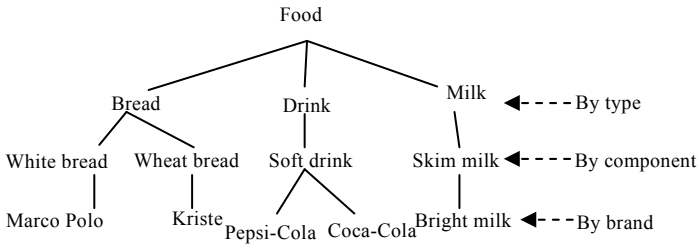

Fig.1 Example of the taxonomy data in retail trade

图 1 零售业商品分类数据的例子

分类数据关联规则挖掘通常分为多层关联规则(MLARM)和概化关联规则挖掘(GARM)两种.从 Han[5]和 Srikant[6]等人最早提出分类数据关联规则挖掘问题以来,基于分类数据的关联规则挖掘已取得了一系列研究成果[7,8].Pramudiono 等人提出了一种基于 FP-Growth[3]思想的算法 FP-tax[9],其基本思想是:将扩展后的事务数据库映射成为一棵压缩的 FP-tree 树,由于树的每个节点记载着对应项的计数信息,从而可以实现无需产生候选项集而直接找出频繁项集的目的.与需要通过多次扫描事务数据库,并通过计数来判定项集是否频繁的基于 Apriori 思想[1]的早期算法相比,FP-tax 算法无疑使分类数据关联规则的挖掘效率有了较大的提升.但是,该算法仍存在以下不足:

- 假设所有事务项之间的关系都是未知和平等的,使得分类数据关联规则的挖掘成为一种完全无监督的学习过程.
- 基于 FP-Growth 挖掘方法自身的局限性.该方法基于频繁 1-项集的降序方式来对事务数据库中的项进行排序,然后通过遍历事务数据库生成 FP-tree.然而,这种 FP-tree 结构在结合自底向上遍历策略寻找每个项集的前缀频繁项集时,由于无法重用原始的 FP-tree 而必须不断循环新建以该项为前缀的条件子 FP-tree,从而使算法增加了大量额外开销.

针对现有算法存在的以上不足,本文从分类数据的内在特征和相关领域知识入手,结合最新挖掘方法的研究成果,提出了一种新的多层关联规则挖掘方法,其基本思想是:

- 利用分类数据所在的领域知识来构建项与项之间的相关性模型(domain-based item correlation model).基于该相关性模型构建的相关性函数(correlation function)可以实现对现有通用相关性函数的有效修正,使之更加适合于分类数据项之间相关性的度量.
- 基于该相关性函数对分类数据的项进行层次聚类(hierarchical clustering),即根据用户给定的控制阈值将相关性较高的项尽量聚成一类,然后根据项的聚类结果对事务数据库进行约简划分.划分后事务数据库的交易列表中只保留来自同一聚类中的项,从而可以缩小事务数据库的规模,节省了挖掘算法扫描事务数据库的 I/O 时间.
- 将一种新的压缩前缀树数据结构 AFOPT-tree[4]扩展至分类数据挖掘领域,并根据分类数据的特性提出了一组优化规则,使之更加适合于分类数据的挖掘,可进一步提升了算法的效率.

基于上述核心思想,本文提出并实现了一种自顶向下的多层关联规则挖掘算法 TD-CBP-MLARM 和一种自底向上的多层关联规则挖掘算法 BU-CBP-MLARM.此外,本文还将这一思想成功扩展至概化关联规则挖掘领域,提出并实现了一种高效的概化关联规则挖掘算法 CBP-GARM.表明本文提出的方法对于分类数据关联规

则挖掘的两个主要领域均具有普遍的适用性.此外,通过大量人工随机生成数据的实验,其结果表明,本文提出的多层和概化关联规则挖掘算法不仅可以确保频繁项集挖掘结果的正确性和完整性,还比现有同类最新挖掘算法具有更好的执行效率和扩展性.

本文第 1 节给出问题描述.第 2 节给出多层和概化关联规则挖掘方法及示例.第 3 节给出具体算法描述及时间复杂度分析.第 4 节是实验结果.最后是总结和展望.

1 问题描述

事务数据库是由一组交易组成,通常可以表示为 $D=\{t_1,t_2,\ldots,t_n\}$,其中,交易 $t_i(i=1,2,\ldots,n)$由唯一的交易标识(TID)和一组项列表(itemlist)组成.当项集 $A=\{i_1,i_2,\ldots,i_k\}$中的所有项均在交易 t_i 中时,则称交易 t_i 包含项集 A,记为 $t_i \supseteq A$.包含项集 A 的交易集合表示为 $D_A=\{t_i|A\subseteq t_i,t_i\in D\}$.项集 A 的支持度是指事务数据库中包含该项集的交易所占的比例,记为 $Support(A)=|D_A|/|D|$.其中,$|D_A|$是包含项集 A 的交易数,$|D|$是事务数据库的总交易数.对于用户定义的最小支持度阈值$\sigma_{\min sup}$,如果 $Support(I)\geqslant\sigma_{\min sup}$,则称项集 I 是频繁项集.分类数据通常可以用分类层次树进行描述.

定义 1(分类层次树). 分类层次树记为 $Tax=(V,E)$,其中,V 是所有节点(项)的集合,E 是所有边的集合,V_0 是分类层次树的根节点(root).对于任意简单路径 $P=(v_0,v_1,\ldots,v_n)$,则有:

(1) v_{n-1} 是 v_n 的父项(parent item);反之,v_n 是 v_{n-1} 的子项(child item).v_n 的所有父项的集合记为 $parent(v_n)$,而 v_{n-1} 的所有子项的集合记为 $child(v_{n-1})$;

(2) $v_0,\ldots,v_{n-1}$ 是 v_n 的祖先项(ancestor item).v_n 的所有祖先项的集合记为 $ancestor(v_n)$;反之,$v_1,\ldots,v_n$ 是 v_0 的后代项(descendant item),而 v_0 的所有后代项的集合记为 $descendant(v_0)$;

(3) 对于任意节点 x,y,z,如果 $x\in child(z)$且 $y\in child(z)$,则称 x,y 是兄弟项(brother item);

(4) 节点 v 的深度是指该项在分类层次树中所处的层次数,记为 $depth(v)$;

(5) 任何深度相同的节点为堂兄项(sibling item);

(6) 任何无子项的节点为叶子项(leaf item);

(7) 除根和叶子节点之外的任意节点称为概化项(g-item).

定义 2(同层项集、同层频繁项集). 在分类数据中,对于任意项集 $A=\{i_1,i_2,\ldots,i_k\}$,如果满足 $depth(i_1)=depth(i_2)=\ldots=depth(i_k)=\ell$,则称项集 A 为同层项集,记为 $A[\ell,k]$.其中,k 表示第ℓ层项集 A 的长度(即项集中项的个数).如果 $Support(A[\ell,k])\geqslant\sigma_{\min sup}[\ell]$(第$\ell$层的最小支持度阈值),则称项集 $I[\ell,k]$是同层频繁项集.

问题描述 1:多层关联规则挖掘,指从给定分类数据中逐层寻找所有支持度大于用户给定阈值(层最小支持度)的同层项集之间所形成的关联规则.

定义 3(概化项集、频繁概化项集). 在分类数据中,任何不存在祖先后代关系的项组成的项集称为概化项集,记为 I_G.如果 $Support(I_G)\geqslant\sigma_{\min sup}$(最小支持度阈值),则称概化项集 I_G 是频繁概化项集.

问题描述 2:概化关联规则挖掘,指从给定分类数据中寻找所有支持度大于用户给定阈值(最小支持度)的概化项集之间所形成的关联规则.

2 多层和概化关联规则挖掘方法

2.1 基本思想

分类数据所属的领域往往蕴含了大量的先验知识,这些知识经过抽取和提炼通常可以用于指导关联规则的挖掘过程,使得关联规则挖掘从无监督学习转变为半监督甚至有监督学习.图 2 给出了本文提出方法的主要思想,从中可以清晰地看出本文的挖掘方法主要包括 3 个阶段:

- 项的相关性模型构建阶段.主要任务是根据分类数据所在的领域知识构建了项与项之间的相关性模型,并基于该模型对通用的相关性函数进行了修正,提出了一种新的基于领域知识的相关性函数,从而实现了利用领域知识来指导后续的频繁项集挖掘.

- 事务数据库约简划分阶段.主要任务是两个:(1) 根据自顶向下或自底向上的遍历策略,基于相关性函数对在分类层次树每层的项进行层次聚类.即从分类层次树的叶子层或第 1 层的项开始聚类,然后利用分类数据的特点迭代生成其余层项的聚类;(2) 基于每层项的聚类结果对该层对应的事务数据库进行约简划分,使得原始数据库被划分为多个相对较小规模的子事务数据库.
- 频繁项集挖掘阶段.将每层经过约简划分后的多个子事务数据库分别扫描映射成一种新的压缩 AFOPT-tree 树形结构,然后通过扫描各 AFOPT-tree 而不是子事务数据库来产生最终的频繁项集.

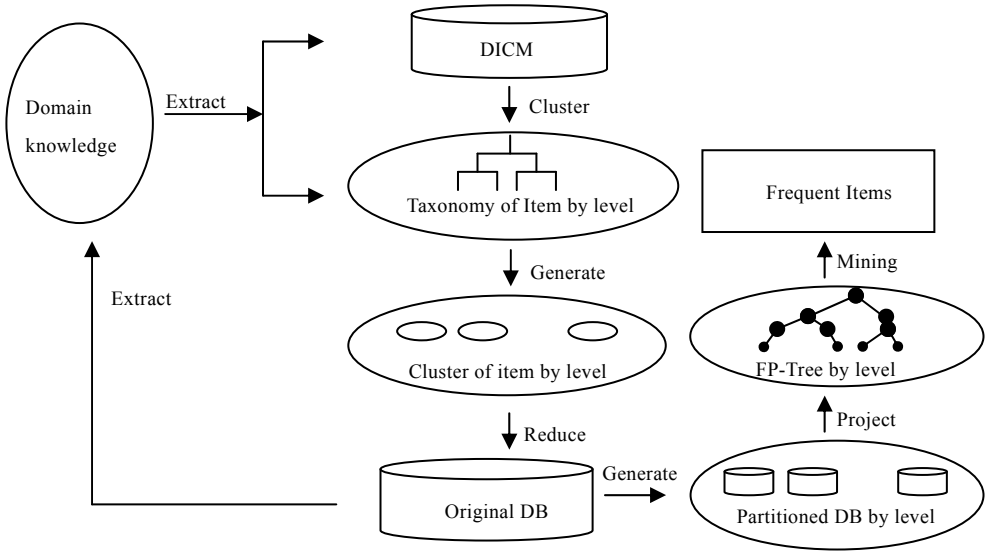

Fig.2 A new method framework of mining the taxonomy data

图 2 本文提出的多层关联规则挖掘方法整体框架

2.2 利用领域知识来约简划分事务数据库

2.2.1 基于领域知识的项相关性模型的构建

能否实现对项的有效聚类是本文提出的多层关联规则挖掘方法的关键技术之一,而相关性函数(correlation function)的选择则是聚类的核心问题之一.对于项之间的相关性度量,一种最简单的方法是直接采用一些通用的相关性函数,比如 Dice 系数、Jaccard 系数、Cosine 系数或 Overlap 系数等.当然,也可以直接用 2-项集支持度形式来表示,具体形式为:

$$Cor_D(i,j)=\frac{|D_{i\cap j}|}{|D|} \tag{1}$$

其中,D 为事务数据库.

以上的项相关性函数因为其简单和归一化特点而被普遍采用,但这些项相关性函数都是仅仅基于有限的训练数据来度量项之间的相关性,由于训练数据的有限性,使得度量出的相关性并不总能够有效地反映该领域中数据项之间的真正相关性.这种数据项之间相关性度量的误差会在后续的工作(比如分类、聚类等)中一直传播,从而严重影响后续工作.而对于具体应用领域来讲,往往存在大量先验知识,并经过领域专家的长期积累而形成领域知识.领域知识的内容非常广泛,其中就包括了本文重点讨论和应用的事务数据库中项之间的相关性关系.如果能够引入领域知识来修正通用的相关性函数,就可以克服通用相关性函数的不足.因此,为了更好地引入领域知识,本文提出了一种新的基于领域知识的项相关性模型:

定义 4(基于领域知识的项相关性模型(domain knowledge-based item correlation model,简称 DICM)). 基于领域知识的项相关性模型 DICM 由正相关矩阵 PICM 和负相关矩阵 NICM 组成,其中,PICM 和 NICM 均为

$m\times m$ 的矩阵,m 是分类层次树中第 l 层项的数量.矩阵 PICM 和 NICM 中各项的取值定义如下:

$$PICM(i,j)=\begin{cases}\alpha, & i,j\text{之间具有明确的正关系}\\ \gamma, & i,j\text{之间的正关系不明确}\end{cases},$$

$$NICM(i,j)=\begin{cases}\beta, & i,j\text{之间具有明确的负关系}\\ \gamma, & i,j\text{之间的负关系不明确}\end{cases}.$$

其中,(α,β,γ)是相关性模型参数,通常由领域专家根据该领域的知识经验决定.一般情况下,γ=0.5,$\alpha\in$(0.5,1), $\beta\in$[0,0.5).

本文以图 3 所示的金融业交易数据库为例来说明领域知识相关性模型的构建方法.假设金融交易是按照业务品种用数字 0~9 来进行编码的,编码长度为 4,*表示交易码的该位可以是任意数字.经过编码后的金融交易具有非常明显的分类层次特征,其局部的分类结构如图 3 所示.假设 00**表示个人类交易,01**表示企业类交易,02**表示银行卡类交易,由于根据金融业的领域知识,企业客户和个人客户的账户一般是分列管理的,只有少数经过授权或联动交易才能导致这两类账户的交易一起执行,而个人客户或企业客户和银行卡交易一起执行的概率相对较高.基于这些业务规则结合金融领域专家的经验,就可以形成先验的基于领域知识的相关性模型,图 4 给出了部分项之间的相关性模型示例.

Transaction ID (TID)	Finance transaction code (items list)
001	0001, 0010, 0100, 0101
002	0100, 0101
003	0010, 0100, 0101
004	0001
005	0100, 0200

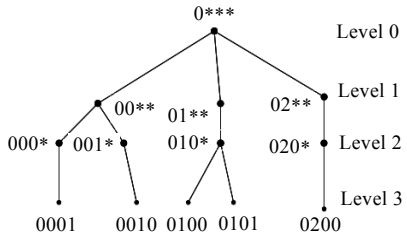

Fig.3 An example of financial transaction database and the corresponding taxonomy

图 3 一个金融交易数据库的例子及其对应的交易分类层次结构

$$PICM=\begin{bmatrix} & 0001 & 0010 & 0100 & 0101 & 0200\\ 0001 & - & - & - & - & -\\ 0010 & 0.8 & - & - & - & -\\ 0100 & 0.6 & 0.6 & - & - & -\\ 0101 & 0.6 & 0.6 & 0.8 & - & -\\ 0200 & 0.8 & 1 & 0.6 & 0.6 & - \end{bmatrix},\quad NICM=\begin{bmatrix} & 0101 & 0201 & 0301 & 0401 & 0901\\ 0101 & - & - & - & - & -\\ 0201 & 0.3 & - & - & - & -\\ 0301 & 0.5 & 0.4 & - & - & -\\ 0401 & 0.3 & 0.5 & 0.4 & - & -\\ 0901 & 0 & 0.4 & 0.3 & 0.5 & - \end{bmatrix}$$

Fig.4 An example of the financial-based correlation model

图 4 一个金融领域相关性模型示例

基于上述领域知识形成的相关性模型,就可以形成基于领域知识的相关性函数,其定义如下:

定义 5(基于领域知识的项相关性函数(domain-based item correlation function)).基于领域知识的项相关性函数定义如下:

$$Cor_{DICM}(i,j)=\frac{PICM(i,j)}{PICM(i,j)+NICM(i,j)} \tag{2}$$

其中,i 和 j 为任意项.有了基于领域知识的项相关性函数.本文对通用的相关性函数(1)进行优化,修正后的相关性函数定义如下:

定义 6(基于 DICM 的项相关性函数). 经过相关性模型 DICM 修正后,分类数据项 i 和 j 之间的相关度函数定义如下:

$$Cor(i,j)=Cor_D(i,j)\times Cor_{DICM}(i,j) \tag{3}$$

以上基于 DICM 的相关性函数主要优点是:

- 在寻找每层的频繁项集时,2-项集的支持度是必须计算的,直接引入 2-项集支持度作为相似度函数,既达到表示项之间相关性的目的,又可以有效减少因为引入其他相关性函数而需要的额外计算开销;
- 通过引入基于领域约束模型的相关性 $Cor_{DICM}(i,j)$,可以有效地避免度量函数完全依赖于特定训练数据集的缺点.

2.2.2 基于项相关性函数的聚类方法

有了合适的相关性函数,接下来就是如何对分类数据中的项进行聚类.本文提出了一种基于项相关性函数的层次聚类方法 AHC(agglomerative hierarchical clustering),该方法采用自顶向下或自底向上的策略,从分类数据第 1 层或最底层(叶子层)的项开始进行聚类,直至叶子层的上一层或第 1 层为止.以图 3 的金融交易数据库为例进行说明,并假设第 1 层~第 3 层的层递减最小支持度$\sigma_{\min sup}[l]$分别为 0.4,0.25,0.21,取$\delta=\sigma_{\min sup}[3]=0.21$, $Cor_{DICM}=1$.不失一般性,对第 1 层项的层次聚类具体过程如下:

(1) 首先,将所有频繁 1-项集中每个项单独划为一类,即:{00**},{01**}和{02**};

(2) 根据公式(3)计算出两两项之间的相关性.由于 Cor(00**,01**)=2/5=0.4≥δ,所以,01**与 02**应进行合并.此时,第 1 层的项被划分成两类:{00**,01**}和{02**};

(3) 根据层次聚类的思想,进一步考察{00**,01**}与{02**}两个类之间是否可以进一步合并.而由于 Cor(00**,02**)=1/5=0.2 和 Cor(01**,02**)=1/5=0.2 都小于δ,所以,00**与 02**,01**与 02**均无法合并.至此,第 1 层项最终聚为两个类:C_1={00**,01**}和 C_2={02**}.

2.2.3 利用聚类结果对事务数据库约简划分方法

对分类数据的项进行聚类,其主要目的是为了对数据库进行约简划分.本文提出的对数据库进行约简方法,其本质是一种特殊的“划分(partitioning)”方法.与传统划分方法[10]将事务数据库集按交易记录划分不同,约简划分的方法是对交易记录中的项进行约简划分,从而可以达到将相关性高的项划入同一事务数据库集的目的.这一点正是传统划分方法所无法实现的.本文给出对事务数据库 D 进行约简划分的规则如下:

规则 1(事务数据库约简划分). 如果第l层的项划分成类 $C_1,C_2,\ldots,C_k$,那么对应可将事务数据库 D_λ划分成 $D_{\lambda 1}, D_{\lambda 2},\ldots, D_{\lambda k}$,使得每个 $D_{\lambda i}$ ($i\in[1,k]$)只保留 $C_i(i\in[1,k])$中所有项及其后代项.

以图 3 的金融交易数据库为例来进行说明.根据第 1 层项的划分结果,对相应的事务数据库 D 进行约简划分,形成事务数据库 D_1 和 D_2,使 D_1 只包含 C_1 中的项 00**,01**及其后代项 000*,001*,010*,0001,0010,0100 和 0101,而将项 02**及其后代项 020*,0200 进行剪枝;同理,使 D_2 只包含 C_2 的项 02**及其后代项 020*,0200.经过第 1 层约简划分后的事务数据库 D_1 和 D_2 如图 5 所示.一般情况下,经过约简后会形成多个子事务数据库,每个子事务数据库的规模均小于原事务数据库.

Original transaction database D

TID	Itemslist
001	0001, 0010, 0100, 0101
002	0001, 0101
003	0010, 0100, 0101
004	0001
005	0100, 0200

Partitioned database D_1

TID	Itemslist
001	0001, 0010, 0100, 0101
002	0001, 0101
003	0010, 0100, 0101
004	0001
005	0100

Partitioned database D_2

TID	Itemslist
005	0200

Fig.5 Partitioned database after pruning

图 5 对事务数据库约简划分的结果

2.3 基于分类数据的频繁项集挖掘方法AFOPT-tax

与现有将事务数据库映射成 FP-tree 压缩结构的算法不同,作为多层和概化关联规则方法的核心部分,本文将一种新的压缩结构 AFOPT-tree[4]引入分类数据多层关联规则的挖掘,提出一种适用于分类数据挖掘的 AFOPT-tax 方法.与 FP-tree 结构相比,采用 AFOPT 结构结合自顶向下的遍历策略具有更小的时间开销,根据文献[4]的思想,证明如下.

定理 1. 给定分类层次树 Tax,分别映射至 AFOPT-tree 和 FP-tree 结构.假定访问 AFOPT-tree 和 FP-tree 任意一个节点的时间开销相同,那么使用 AFOPT-tree 结构结合自顶向下的遍历策略的时间开销少于使用 FP-tree 结构结合自底向上的遍历策略.

证明:首先,考察基于给定分类层次树的多层频繁项集挖掘情况,假设任意第 ℓ 层对应的事务数据库 D_λ 映射至 FP-tree 后,其包含的叶子节点数为 n_{fp},则自底向上遍历该 FP 树需要遍历 n_{fp} 个分枝,按照遍历顺序,这 n_{fp} 个分枝按叶子节点支持度升序排列,如果按照这些分枝来建立前缀树,正好是本文所述的 AFOPT-tree 结构,由于本文使用自顶向下的遍历策略,每个节点只遍历一次,遍历的节点数正好是 AFOPT-tree 的节点数,小于上述 n_{fp} 个分枝长度之和.由于多层频繁项集挖掘是针对分类层次树逐层进行的,因此整个分类层次树的多层频繁项集挖掘使用 AFOPT-tree 结构结合自顶向下遍历策略的时间开销要少于使用 FP-tree 结构结合自底向上的遍历策略.

其次,考察基于给定分类层次树的概化频繁项集挖掘情况,与多层频繁项集挖掘的不同仅在于,对应的为经过扩展后的事务数据库 D',其中包含了所有的项及其祖先项,而建树方式和遍历策略均同前者.因而可以推出,整个分类层次树的概化频繁项集挖掘使用 AFOPT-tree 结构结合自顶向下的遍历策略的时间开销要少于使用 FP-tree 结构结合自底向上的遍历策略.

综上所述,结论成立.AFOPT-tax 方法的优化措施主要包括以下 3 个方面:

优化规则 1. 引入子树合并操作来进一步压缩 AFOPT-tree 规模

在考察完每个项之后,通过执行其对应子树的合并(merge)操作来进一步缩小 AFOPT-tree 规模.在自上向下、深度优先的遍历策略下,在每次考察并产生项 X 为前缀的所有频繁项集后,除了把 X 对应的节点从 AFOPT-tree 中剪枝之外,还可以增加执行一次该节点子树的合并操作.所谓子树合并操作是指:将 X 节点的所有子节点与 X 节点的所有兄弟节点进行比对,如果存在同名项则合并该节点,并累计节点计数;如果不同,则直接插入作为 X 节点父节点的一个新的子节点.一般情况下,虽然该步骤会带来一定额外开销,但与对 AFOPT-tree 压缩后对算法效率的提升相比,总体上仍是有利的.

优化规则 2. 利用头表辅助结构避免条件子 AFOPT-tree 的重复生成

本文在 AFOPT-tree 结构的基础上引入头表(header-table)和项的子头表(sub-header-table)两种辅助数据结构,使得在产生项 X 为前缀的所有频繁项集时,可以直接利用子头表的指针来定位 AFOPT-tree 树中的节点(项),从而实现原始 AFOPT-tree 的重复利用,避免重新构造条件子 AFOPT-tree 的额外开销.

优化规则 3. 利用分类特性来实现过滤剪枝

在引入项子头表的同时,本文还根据分类数据的特性引入了过滤剪枝的优化措施.在项 X 的子头表 H_X 插入扩展项集 Y 时,如果该扩展项集 Y 与项 X 存在祖先与后代关系,则不插入该项集 Y,也不再继续寻找以 XY 为前缀的后续扩展项集.这是因为根据定义 3 可知,XY 不可能构成概化频繁项集.该优化规则可以有效减少以该项为前缀的所有频繁项集的候选项集规模.

AFOPT-tax 作为一个频繁项集的核心挖掘算法,以独立过程方式被多层和概化关联规则挖掘主体算法所调用.该算法主要分为数据预处理、AFOPT-tree 构建和频繁项集挖掘这 3 个主要步骤.以图 3 的金融交易数据库为例,假设第 3 层的最小支持度 $\sigma_{minsup}[3]=0.21$.不失一般性,以第 3 层为例来说明其工作过程.

步骤 1. 寻找所有的频繁 1-项集,并按照其升序对事务数据库进行重新排序和剪枝.

首先,第 1 次扫描事务数据库,针对每个交易的所有项插入其概化项,然后计算产生包含概化项在内的所有频繁 1-项集.由于 0001,0010,0100,0101,0200 的支持度分别为 2/5,2/5,4/5,3/5,1/5.所以,除了项 0200 其支持度小于最小支持度不频繁之外,其余均为频繁 1-项集.频繁 1-项集的升序排列依次为:(0001:2),(0010:2),(0101:3)和

(0100:4).其中,冒号后数字表示该项的支持度.然后,第 2 次扫描事务数据库,根据该升序对数据库的每个交易列表中的项进行重新排序,并对不频繁的 1-项集进行剪枝.其中,由于项 0200 不频繁,所以从交易 TID 为 005 的项列表中剪枝.经过重新排序和剪枝后的事务数据库如图 6 中的左图所示.

步骤 2. 扫描事务数据库,构建 AFOPT-tree 及头表.

AFOPT 结构包括 AFOPT-tree 和头表 *H* 两部分,其中,AFOPT-tree 为一种前缀树结果,每个节点包括项、计数、子节点指针及兄弟指针等域,通常可简化表示为(项:计数).而头表则包括了项、计数、第 1 个节点指针.以下主要说明 AFOPT-tree 的构建过程:首先以空节点作为 AFOPT-tree 的根,然后在第 2 次扫描事务数据库的同时,从事务数据库中读取 *TID*=001 的交易{0001,0010,0101,0100},依次插入 AFOPT-tree,并对每个节点进行计数,分别为(0001:1),(0010:1),(0101:1)和(0100:1).然后,再从事务数据库中读取 *TID*=002 交易{0101,0100},此时若没有相同的 0101 节点存在,则从根开始依次插入.同理,读取 *TID*=003 交易进行插入.在读取 *TID*=004 交易{0001}时,由于 AFOPT-tree 已经存在 0001 节点,则与之进行合并,并累计该项的计数为 2.按照以上规则,再依次读取事务数据库的剩余交易记录并插入,最终形成完整的 AFOPT-tree 如图 6 中的右图所示.

TID	Itemslist	Itemslist (after sorting and pruning)
001	0001, 0010, 0100, 0101	0001, 0010, 0100, 0101
002	0001, 0101	0001, 0101
003	0010, 0100, 0101	0010, 0100, 0101
004	0001	0001
005	0100, 0200	0100

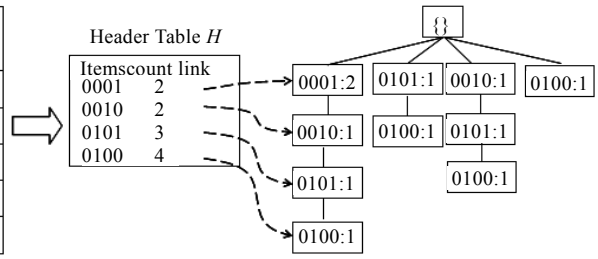

Fig.6 Transaction database (left) and AFOPT-tree (right) after resorting and pruning

图 6 经过排序和剪枝后的事务数据库(左)及其对应的 AFOPT 树(右)

步骤 3. 按照自顶向下、深度优先遍历策略,依次查找所有的频繁项集.

由于采用了自顶向下、深度优先的遍历策略,首先考察头表 *H* 的第 1 个项 0001.具体工作过程是:遍历从节点 0001 开始至叶子的所有路径,由于以 0001 为前缀的所有扩展项 0010,0101,0100 中计数均为 1(即 1/5=0.2,小于最小支持度).因此,节点 0001 的子头表中没有项,不再继续寻找以 0001 为前缀的其他频繁项集.在考察头表中的第 2 个项 0010 之前,需要对以节点 0001 为根的子树进行合并操作,然后对节点 0001 节点进行剪枝,以进一步压缩 AFOPT-tree 的规模.具体过程为:节点 0001 子树的分支为{0010,0101,0100},由于在 0001 兄弟节点中存在与其子树节点 0010 相同的节点,因此可以对节点 0010 进行合并,并累计节点 0010 的计数.同理,对节点 0101,0100 也进行合并,并累计节点计数,然后对节点 0001 的子树进行剪枝.这里只需置逻辑标志,而不需真正物理删除.对节点 0001 子树合并和剪枝操作过程如图 7 所示.此时,经过剪枝和合并后的 AFOPT-tree 规模一般要小于最初的 AFOPT-tree,节点数从 10 个减少至 6 个,如图 8 所示.

在考察完节点 0001 之后,将依次考察头表 *H* 中剩余的项 0010,0101,0100,并分别进行遍历挖掘、子树合并及剪枝等操作,最终找出所有分类数据第 3 层的频繁项集.

受篇幅所限,有关 AFOPT-tax 算法的具体描述可参见文献[11].

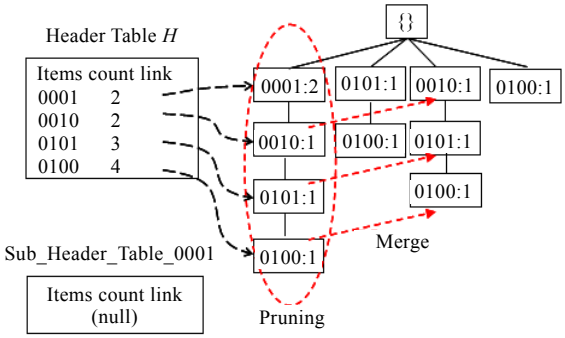

Fig.7 Subtree merge and pruning of item 0001

图 7 节点 0001 的挖掘、子树合并及剪枝示意图

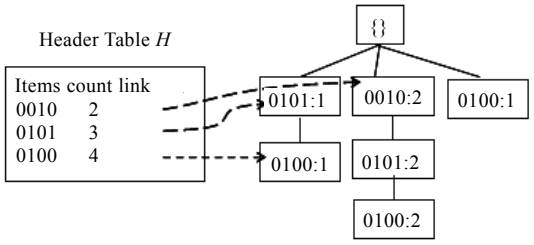

Fig.8 AFOPT-tree after pruning of subtree of item 0001

图 8 节点 0001 子树剪枝后的 AFOPT-tree

3 算法描述及时间复杂度分析

3.1 多层关联规则挖掘算法

基于以上基本思想,本文首先提出了一种自顶向下的多层关联规则挖掘算法 TD-CBP-MLARM,其完整描述如下.

算法 1. TD-CBP-MLARM.

输入:分类数据 Tax,最大层数 Max*level*;原始事务数据库 D_0;用户定义的层最小支持度$\sigma_{\min sup}[\ell]$,用户定义的相关性阈值δ.其中,$D_{\lambda i}$表示第ℓ层的第 i 个事务数据库;

输出:第ℓ层的频繁项集 $LL(\ell)$.

1) **For** (第 1 层开始直至叶子层, $D_{\lambda i}$初始化为 D_0) **do** {
2) 调用过程 $AFOPT\text{-}tax(D_{\lambda i},\sigma_{\min sup}[\ell],LL(\ell))$来挖掘第$\ell$层的频繁项集
3) **If** 不是叶子层 **then** {
4) 分类层次树第ℓ层的每个项初始化为独立的类 $C_{\lambda 1}, C_{\lambda 2},..., C_{\lambda n}$;
5) **For** 每一对类之间 **do** {
6) **If** (相关性函数 $Cor(C_{\lambda i},C_{\lambda j}\geqslant\delta)$ **then** {
7) 将 $C_{\lambda i}$和 $C_{\lambda j}$聚成一类 $C_{\lambda i}$ } //end **if**
8) } //end **for**
9) **For** 根据每个类 $C_{\lambda 1}, C_{\lambda 2},..., C_{\lambda m}$,只要 $D_{(\lambda-1)i}$非空 **do** {
10) 对第ℓ–1 层的每个事务数据库 $D_{(\lambda-1)i}$约简划分成 $D_{\lambda 1}, D_{\lambda 2},...,D_{\lambda}$;
11) } //end **for**
12) } //end **if**
13) } //end **for**

时间复杂度分析:不失一般性,假定多层挖掘的核心算法基于 AFOPT-tax 思想,其算法的时间复杂度为 t_2;本文提出算法的时间复杂度为 t_1.设分类层次树的第ℓ层含 n_0 个节点,各节点平均扇出度为 w,并设分类树的深度为 d.根据项与项的相关度函数,如果该层的 n_0 项被聚成了 k 类,则本文算法的时间复杂度为 $t_1(n_0)+t_2(|D_1|)+t_2(|D_2|)+\ldots+t_2(|D_k|)$,而直接用 AFOPT-tax 算法挖掘,时间复杂度为 $t_2(|D|)$.其中:t_1 是该层 n_0 项划分成 k 类的时间复杂度,为 $O(n_0\times(n_0-1)/2\log(n_0\times(n_0-1)))$;$t_2$ 是使用 AFOPT-tax 算法挖掘的时间复杂度,与 AFOPT-tree 的大小有关.由于是第 1 层,n_0很小,$n_0\approx\lceil|I|/w^{d-1}\rceil$(上取整).由此可见,$t_1$耗费的时间较少,只要有划分,$t_2(|D|)$所耗费的时间就会比 $t_2(|D_{11}|)+t_2(|D_{12}|)+\ldots+t_2(|D_{1k}|)$多,这是因为分而治之的后者节约了较多的 I/O 时间;针对分类层次树的第 2 层,如果直接用 AFOPT-tax 算法,时间复杂度为 $t_2(|D_{11}|)+t_2(|D_{12}|)+\ldots+t_2(|D_{1k}|)$,由于本文算法是在第 1 层划分剪枝后的小事务数据库上进行,其时间复杂度为 $k\times t_1(n_0\times w/k)+t_2(|D_{211}|)+t_2(|D_{212}|)+\ldots+t_2(|D_{21k}|)+\ldots+t_2(|D_{2k1}|)+t_2(|D_{2k2}|)+\ldots+$

$t_2(|D_{2kk}|)$.以此类推,直至第 ℓ 层,如直接用 AFOPT-tax,算法的时间复杂度为

$$t_2(|D_{(\lambda-1)1\ldots1}|)+t_2(|D_{(\lambda-1)1\ldots2}|)+\ldots+t_2(|D_{(\lambda-1)1\ldots k}|)+\ldots+t_2(|D_{(\lambda-1)k\ldots1}|)+t_2(|D_{(\lambda-1)k\ldots2}|)+\ldots+t_2(|D_{(\lambda-1)k\ldots k}|).$$

而 TD-CBP-MLARM 的时间复杂度为

$$k^{\lambda-1}\times t_1(n_0\times w^{\lambda}/k^{\lambda})+t_2(|D_{\lambda1\ldots1}|)+t_2(|D_{\lambda1\ldots2}|)+\ldots+t_2(|D_{\lambda1\ldots k}|)+\ldots+t_2(|D_{\lambda k\ldots1}|)+t_2(|D_{\lambda k\ldots2}|)+\ldots+t_2(|D_{\lambda k\ldots k}|).$$

由于算法中 t_1 只和与之对应的项多少有关,其中 $|I|=n_0\times w^d$,远小于 $|D|$,由于用于聚类的额外时间开销很少,所以在一般情形下,聚类并对原数据库进行约简有助于总体挖掘效率的提升.

以上本文提出的思想同样也适用于自底向上的遍历策略.由于从叶子层开始入手挖掘,因为叶子层节点通常较多,算法最初的聚类和约简划分效果也会更好,这点从最后实验部分得到了验证.

算法 2. BU-CBP-MLARM.

输入:分类数据 Tax,最大层数 Max*level*;原始事务数据库 D_0;用户定义的层最小支持度 $\sigma_{\min sup}[\ell]$,用户定义的相关性阈值 δ.其中, $D_{\ell i}$ 表示第 ℓ 层的第 i 个事务数据库;

输出:第 ℓ 层的频繁项集 $LL(\ell)$.

1) **For** (叶子层直至第 1 层) **do** {
2) **If** 不是叶子层 **then** {
3) 分类层次树第 ℓ 层的每个项初始化为独立的类 $C_{\lambda1},C_{\lambda2},\ldots,C_{\lambda n}$;
4) **For** 每一对类之间 **do** {
5) **If** (相关性函数 $Cor(C_{\lambda i},C_{\lambda j})\geqslant\delta$) **then** {
6) 将 $C_{\lambda i}$ 和 $C_{\lambda j}$ 聚成一类 $C_{\lambda i}$ } //end **if**
7) } //end **for**
8) **For** 根据每个类 $C_{\lambda1},C_{\lambda2},\ldots,C_{\lambda m}$ **do** {
9) 对原始事务数据库 D_0 约简划分成 $D_{\lambda1},D_{\lambda2},\ldots,D_{\lambda m}$;
10) } //end **for**
11) } // end **if**
12) 调用过程 *AFOPT-tax*($D_{\ell i}$,$\sigma_{\min sup}[\ell]$,$LL(\ell)$)来挖掘第 ℓ 层的频繁项集
13) } //end **for**

时间复杂度分析:本算法从划分和挖掘过程来看,复杂度同 TD-CBP-MLARM 算法.由于在对原交易数据库的约简划分顺序上的差异,挖掘效率会与 TD-CBP-MLARM 算法有所不同.

3.2 概化关联规则挖掘算法

针对分类数据挖掘更一般的形式——概化关联规则的挖掘问题,以上主要思想也同样适用,其工作过程如下:

步骤 1. 对原始数据库进行扩展概化项处理.

由于概化关联规则的挖掘需要考虑所有的项及其概化项,因此,首先需对原始事务数据库的项进行扩展处理,即在交易列表中的项添加其所有的概化项.仍以图 3 的金融交易数据库为例,添加概化项后,事务数据库如表 1.

Table 1 Financial database after adding the generalized items

表 1 金融交易数据库添加概化项后结果

TID	Itemlist	Itemlist with generlized items
001	0001, 0010, 0100, 0101	0001, 0010, 0100, 0101, 000*, 001*, 010*, 00**, 01**
002	0100, 0101	0100, 0101, 010*, 01**
003	0010, 0100, 0101	0010, 0100, 0101, 001*, 010*, 00**, 01**
004	0001	0001, 000*, 00**
005	0100, 0200	0100, 0200, 010*, 020*, 01**, 02**

步骤 2. 对分类数据的所有项进行聚类,对原始数据库进行约简划分.

根据第 2.2.2 节描述的方法,首先对分类数据的所有项进行层次聚类.这时,相关性函数的值可以通过扫描原始事务数据库计算出 1-项集和 2-项集支持度来间接获取.然后根据项的聚类结果,参照第 2.2.3 节的方法对添加概化

项后的事务数据库进行约简划分.此时在约简时,项与其概化项是平等的,即在约简某概化项时,其后代项仍然保留而不是约简.

步骤 3. 挖掘所有的频繁项集.

基于约简划分后的事务数据库,将事务数据库分别映射成 AFOPT-tree 结构,然后根据第 2.3 节描述的方法挖掘出所有的频繁项集.由于项及其概化项的同时存在,在利用 AFOPT-tree 结果挖掘频繁项集时可利用优化规则 3,以减少项集前缀项集的数量.

下面给出 CBP-GARM 算法的具体描述.

算法 3. CBP-GARM.

输入:分类数据分类 Tax;初始事务数据库 D_0;用户定义的全局最小支持度σ_{minsup};用户定义的相关性阈值δ;

输出:概化频繁项集 L.

1) **For** 对于 D_0 中的所有交易 **do** {
2) 对每个交易的项添加其所有的祖先项
3) } //end **for**
4) 将分类层次树每个项初始化为独立的类 $C_{\lambda 1},C_{\lambda 2},...,C_{\lambda n}$;
5) **For** 每一对类之间 **do** {
6) **If** (相关性函数 $Cor(C_{\lambda i},C_{\lambda j})\geq\delta$) **then** {
7) 将 $C_{\lambda i}$ 和 $C_{\lambda j}$ 聚成一类 $C_{\lambda i}$ } //end **if**
8) } //end **for**
9) **For** 根据每个类 $C_{\lambda 1},C_{\lambda 2},...,C_{\lambda m}$ **do** {
10) 对原始事务数据库 D_0 约简划分成 $D_{\lambda 1},D_{\lambda 2},...,D_{\lambda m}$;
11) } //end for
12) 调用过程 *AFOPT-tax*($D_{\lambda i}$,$\sigma_{\min sup}$,L)来挖掘频繁项集

时间复杂度分析:本算法共包含以下几个部分:

- 第 1)行~第 4)行的时间复杂度为 $O(|D|\times d\times|L|)$($|D|$为事务数据库所包含的事务数,d 表示分类树的深度,$|L|$表示事务数据库 D 中所有事务的最大长度);
- 第 4)行初始化过程的时间复杂度为 $O(|I|)$($|I|$表示分类交易树所包含的项的个数);
- 第 5)行~第 8)行聚类过程的时间复杂度为 $O(I\times(I-1)/2\log(I\times(I-1)))$;
- 第 9)行~第 11)行约简划分的时间复杂度为 $O(|L|\times d\times|D|)$;
- 设第 12)行的时间复杂度为 O_1.

综上所述,则该算法的时间复杂度为 $O(|L|\times|I|\times|D|)+m\times O_1(|D_{\lambda i}|)$,与直接用 AFOPT-tax 挖掘原始交易数据库的时间复杂度 $O_1(|D'|)$相比(此处的 D'为扩展后的数据库),其挖掘效率有一定幅度的提升.

4 实 验

4.1 实验数据及设置

为了验证本文提出算法的正确性、高效性和扩展性,本文的实验数据集是利用关联规则挖掘领域权威的 IBM 实验室[1]提供的数据生成器生成的人工随机实验数据集,其命名规则为 DxTxRxIx,具体控制参数含义及其缺省值详见表 2.

Table 2 Default value of synthetic datasets

表 2 控制参数及缺省值

Parameter	Description	Default value
$\|D\|$	Number of transactions	10K~100K
$\|T\|$	Average size of the transactions	5~15
$\|I\|$	Average size of the maximal potentially frequent itemsets	2~10
N	Number of items	1K~100K
R	Number of roots	250~1K
L	Number of levels	5~10
F	Fanout	5~25

实验的运行环境为 HP DL380 G3 服务器,Xeon 2.8GHZ 双 CPU,2G 内存,操作系统为 Windows Server 2003,数据库系统为 ORACLE 9i,算法实现均采用 JAVA 语言.本文选取了多层关联规则挖掘对比算法包括:基于 Apriori 思想的 ML_T2L1 算法[5]和最新的基于 FP-Growth 思想的 ML_FP-Growth 算法.选取的概化关联规则挖掘对比算法包括:基于 Apriori 思想的 Cumulate 算法[6]、基于 Eclat 思想的 SET 算法[8]和最新的基于 FP-Growth 思想的 TD-FP-tax 算法[9].由于各种算法的内存消耗差异不大,且服务器内存已不是主要瓶颈,以下主要针对算法执行时间进行比较.

4.2 多层关联规则挖掘算法性能比较

本文从支持度变化、数据库规模变化和分类数据根的数量变化这 3 方面,对各多层关联规则挖掘算法的性能进行了实验比较.由于各算法性能差异较大,所有实验结果图示均采用了对数坐标方式.

- 支持度变化情况下的性能比较

本文选取了两个分类规模不同的人工随机生成数据集 D30T4R1000I10 和 D20T4R100I10,考察了在 5 组不同的层递减支持度(2.4,1.2,0.6,0.3),(2,1,0.5,0.25),(1.6,0.8,0.4,0.2),(1.2,0.6,0.3,0.15)和(0.8,0.4,0.2,0.1)情况下的性能表现.从图 9 的实验结果可知,在支持度较大的情况下,TD-CBP-MLARM 算法与 ML-FP-Growth 算法性能基本相当.这是由于在频繁项集数量相对较少的情况下,从分类数据自顶向下的划分效果不够明显.而 BU-CBP-MLARM 算法由于从分类数据自底向上开始划分,因而划分效果和执行效率相对较好.随着最小支持度的逐步减小,频繁项集数量也相应增加,对事务数据库的约简划分效果逐步明显.无论是 TD-CBP-MLARM 还是 BU-CBP-MLARM 算法的性能优势也逐步体现.在最小层递减支持度为(0.8,0.4,0.2,0.1)时,挖掘产生多层频繁项集共 3 506 个,TD-CBP-MLARM 算法执行时间约为 59 766ms,BU-CBP-MLARM 算法执行时间约为 73 750ms,分别只有 ML-FP-Growth 算法执行时间 119 547ms 的 49.99%和 61.69%,性能提高约 1 倍;只有 ML_T2L1 算法执行时间 17 341 187ms 的 0.34%和 0.43%左右,性能提高超过两个数量级.同时还可以看到,在事务数据库中项集规模较大的情况下,TD-CBP-MLARM 和 BU-CBP-MLARM 算法由于约简划分效果逐步趋同而执行时间趋于接近.

- 数据库规模变化情况下的性能比较

在 DxT4R1000I10 的参数环境下,本文基于两组不同的层递减支持度(2.4,1.2,0.6,0.3)和(1.6,0.8,0.4,0.2),选取了 6 组不同的数据库交易数(5k,10k,15k,20k,25k,30k)来考察各算法的扩展性.从图 10 的实验结果可知,BU-CBP-MLARM 算法性能最优,在交易数达到 30k 时,执行时间约为 13 703ms,只有 ML-FP-Growth 算法执行时间 48 922ms 的 28%,性能提高超过 3 倍.而 TD-CBP-MLARM 算法与 ML-FP-Growth 算法比较接近,执行时间约为 ML-FP-Growth 的 70%~90%.但都只有 ML_T2L1 算法执行时间 1 921 359ms 的 0.7%左右,性能提高超过两个数量级.另外,从图 10 的右图与左图比较可知,在相同数据库规模的情况下,随着层递减支持度的变小,TD-CBP-MLARM 算法性能优势也会逐步变优.所以,TD-CBP-MLARM 和 BU-CBP-MLARM 算法更加适合于大规模数据库的挖掘,因而具有更好的扩展性.

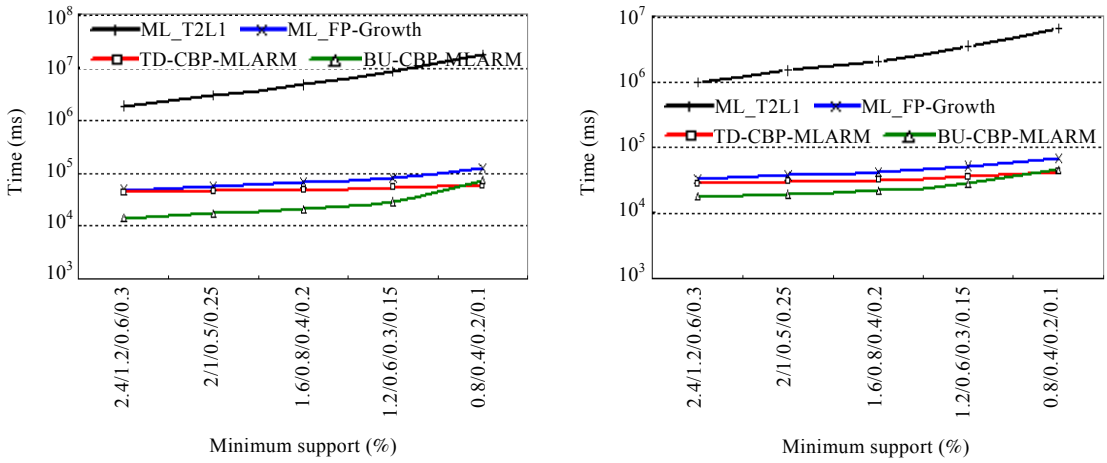

Fig.9 Dataset D30T4R1000I10 (left) and D20T4R100I10 (right)

图 9 基于数据集 D30T4R1000I10(左)和 D20T4R100I10(右)

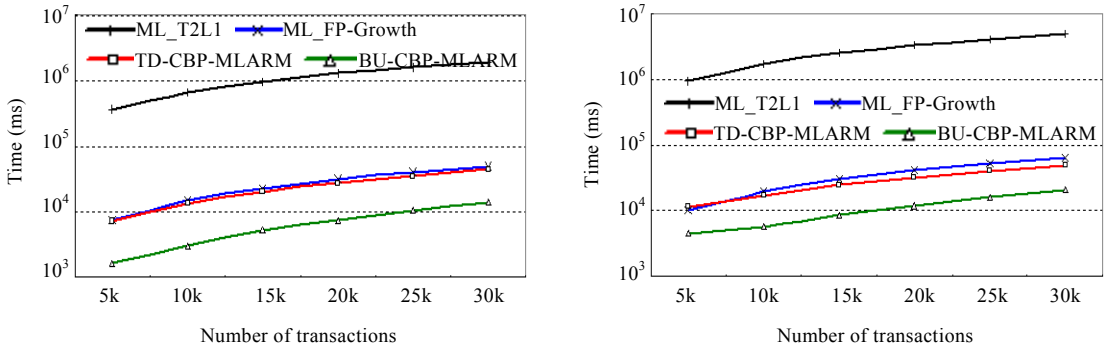

Fig.10 Minimum support of (2.4,1.2,0.6,0.3) (left) and (1.6,0.8,0.4,0.2) (right)

图 10 基于支持度(2.4,1.2,0.6,0.3)(左)和(1.6,0.8,0.4,0.2)(右)

- 根数量变化情况下的性能比较

另一种考察算法扩展性的方法,是改变分类数据根的数量.在 D20T4RxI10 的参数环境下,本文基于两组不同的层递减支持度(2.4,1.2,0.6,0.3)和(1.6,0.8,0.4,0.2),选取 6 组不同的根的数量,分别为 100,200,400,700,1 000 和 2 000 个.从图 11 的实验结果可知,在根数量较小的情况下,各种算法的性能基本相当.这是由于根的数量相对较少,大部分交易都比较集中,对事务数据库的约简划分效果不够明显.随着根数量逐步增加,交易相对分散在更多的根之下,项的聚类效果逐步明显,以 BU-CBP-MLARM 算法性能为最优.在根数量达到 2 000 个时,其执行时间只有 ML-FP-Growth 算法的 20%~50%左右,性能提高超过 1 倍~5 倍.而 TD-CBP-MLARM 算法与 ML-FP-Growth 算法比较接近,执行时间约为 ML-FP-Growth 的 55%~80%.但这两个算法都只有 ML_T2L1 算法的 1%左右,性能提高两个数量级.另外,从图 11 的右图与左图比较可知,在相同数据库规模的情况下,随着根数量的增加,TD-CBP-MLARM 算法的性能会逐步体现.因而,TD-CBP-MLARM 和 BU-CBP-MLARM 算法更加适合于规模大且数据分布相对分散的数据库挖掘.

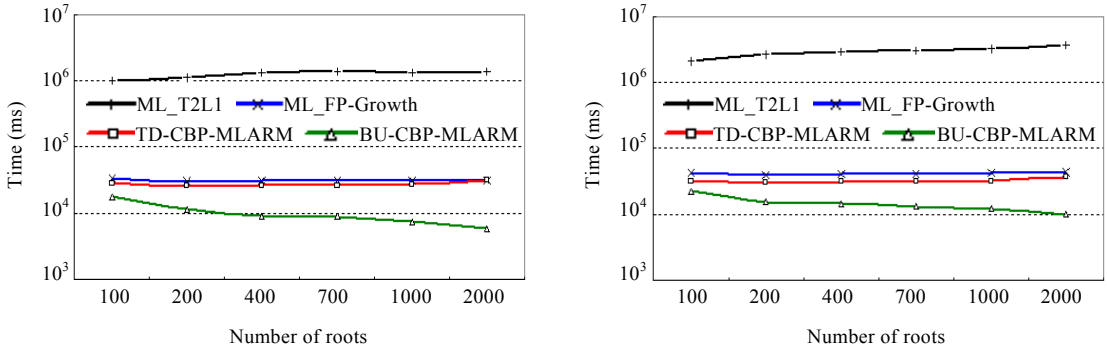

Fig.11 Minimum support of (2.4,1.2,0.6,0.3) (left) and (1.6,0.8,0.4,0.2) (right)

图 11 基于支持度(2.4,1.2,0.6,0.3)(左)和(1.6,0.8,0.4,0.2)(右)

4.3 概化关联规则挖掘算法性能比较

以上是对多层关联规则挖掘算法的性能比较,本文还将从支持度变化、数据库规模变化方面,对概化关联规则挖掘算法的性能进行了实验比较.

- 支持度变化情况下的性能比较

本组实验选取了两个不同规模的人工随机数据集 D10T4R1000I10,D25T4R1000I10,考察了在 4 组不同最小支持度 1,0.5,0.3 和 0.2 情况下的性能表现.从图 12 的实验结果可知,在数据库规模较小的情况下(左图),CBP-GARM 算法的执行效率虽然比 SET 算法和 Cumulate 算法具有明显优势,但与 TD-FP-tax 算法比较接近.在数据库规模较大的情况下(右图),随着支持度的不断减小,CBP-GARM 算法的优势开始显现,执行时间只有 TD-FP-tax 算法的 50%~60%,提高约 1 倍,只有 SET 算法的 30%~50%和 Cumulate 算法的 5%~10%,分别提高 1 倍~3 倍和 10 倍~20 倍.以上实验结果表明,CBP-GARM 算法对具有大规模频繁项集的数据库挖掘具有相对更好的性能表现.

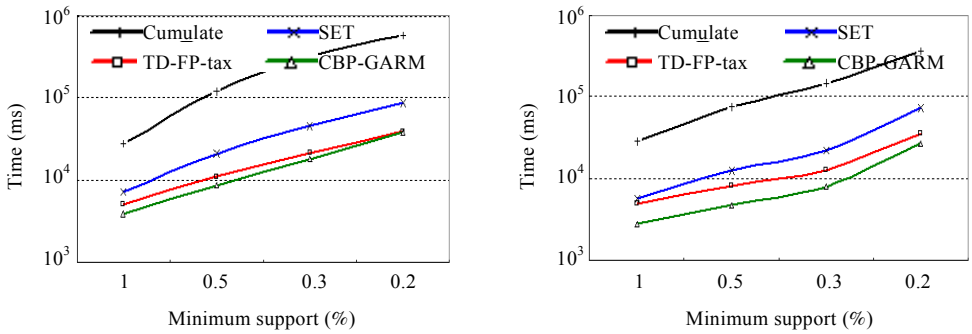

Fig.12 Dataset D20kT6I1 (left) and D50KT4I10 (right)

图 12 基于数据集 D20kT6I1(左)和 D50KT4I10(右)

- 数据库规模变化情况下的性能比较

下面考察各种算法的扩展性情况.本文基于人工数据集 DxT4R1000I10,在两个不同最小支持度 0.5%和 0.3%的情况下,分别选取了 5 组不同的数据库记录数,分别为 10k,15k,20k,30k 和 45k.从图 13 的实验结果可知,随着事务数据库交易数的增加,CBP-GARM 算法始终保持对各算法的效率优势,执行时间只有 TD-FP-tax 算法的 50%~80%,只有 SET 算法的 35%~50%,只有 Cumulate 算法的 5%~10%,提高约 10~20 倍.另外,从图 13 的实验结果可知,CBP-GARM 算法曲线比 TD-FP-tax 算法更为平缓,因而具有较好的扩展性.

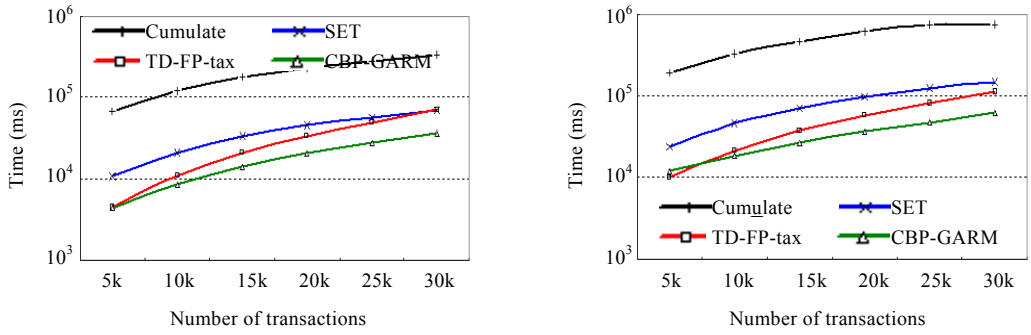

Fig.13 Running time on DxT4I10

图 13 基于 DxT4I10 的执行时间比较

通过以上实验表明,本文提出的方法对于多层和概化关联规则挖掘都取得了较好的效果.相对而言,在概化关联规则挖掘时,算法性能提升相对有限.这是因为只对事务数据库进行一次约简划分,而不像多层关联规则挖掘时,可以进行逐层约减划分.针对这个问题,可以考虑将概化关联规则也进行“分层”挖掘,即按不同长度的项集依次进行挖掘,可以有效地利用约减划分和其他优化规则的效果,达到进一步提升算法效率的目的.

5 总结和展望

本文通过对分类数据特性的深入研究,利用领域知识来构建项与项之间相关性模型,并基于相关性模型对分类数据的项进行聚类,根据项的聚类结果实现对事务数据库进行约简划分,从而节省挖掘算法扫描事务数据库的 I/O 时间.其次,利用层间智慧思想来指导多层关联规则挖掘,将当前层的聚类或划分结果来有效指导下一层的聚类或划分,从而减少部分重复工作的开销.最后,基于一种新的 AFOPT 数据结构来挖掘频繁项集,由于在寻找每个项集的前缀频繁项集时可以有效利用原始的 AFOPT 树,从而有效节约了循环新建条件子树的开销.从大量实验结果来看,本文提出的多层关联规则挖掘算法 TD-CBP-MLARM 和 BU-CBP-MLARM 以及概化关联规则挖掘算法 CBP-GARM 比最新的同类算法具有更好的挖掘效率和扩展性.在接下来的工作中,我们将把上述思想进一步拓展至分类数据的最大集、封闭集以及序列模式挖掘领域.

References:

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. of the Int’l Conf. on Very Large Data Bases (VLDB). Santiago, 1994. 487−499.

[2] Zaki MJ. Scalable algorithms for association mining. IEEE Trans. on Knowledge and Data Engineering (TKDE), 2000.12(3): 372−373. [doi: 10.1109/69.846291]

[3] Han J, Pei J, Yin YW. Mining frequent patterns without candidate generation. In: Proc. of the ACM Annual Conf. on Management of Data (SIGMOD). 2000. 1−12. [doi: 10.1145/342009.335372]

[4] Liu GM, Lu HJ, Lou WW, Xu YB, Yu JX. Efficient mining of frequent patterns using ascending frequency ordered prefix-tree. Data Mining and Knowledge Discovery (DMKD), 2004,9(3):249−274. [doi: 10.1023/B:DAMI.0000041128.59011.53]

[5] Han JW, Fu YQ. Discovery of multiple-level association rules from large databases. In: Proc. of the Int’l Conf. on Very Large Data Bases (VLDB). 1995. 420−431.

[6] Srikant R, Agrawal R. Mining generalized association rules. In: Proc. of the Int’l Conf. on Very Large Data Bases (VLDB). 1995. 407−419.

[7] Hipp J, Myka A, Wirth R, Guntzer U. A new algorithm for faster mining of generalized association rules. In: Proc. of the European Symp. on Principles of Data Mining and Knowledge Discovery (PKDD). 1998. 74−82. [doi: 10.1007/BFb0094807]

[8] Sriphaew K, Theeramunkong T. Fast algorithm for mining generalized frequent patterns of generalized association rules. IEICE Trans. on Information and Systems (TOIS), 2004,E87-D(3):761−770.

[9] Pramudiono I, Kitsuregawa M. FP-Tax: Tree structure based generalized association rule mining. In: Proc. of the ACM Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). 2004. 60−63.

[10] Savasere A, Omiecinski E, Navathe SB. An efficient algorithm for mining association rules in large databases. In: Proc. of the 21st Int'l Conf. on Very Large Data Bases (VLDB). 1995. 432−444.

[11] Mao YX, Shi BL. AFOPT-Tax: An efficient method for mining generalized frequent itemsets. In: Proc. of the 2nd Asian Conf. on Intelligent Information and Database Systems (ACIIDS). 2010. 82−92. [doi: 10.1007/978-3-642-12145-6_9]

毛宇星(1971－),男,上海人,博士生,主要研究领域为数据库,数据挖掘.

陈彤兵(1968－),男,博士,讲师,主要研究领域为数据库,软件工程.

施伯乐(1936－),男,教授,博士生导师,CCF高级会员,主要研究领域为数据库,知识库理论及应用.