维普资讯 http://www.cqvip.com

基于抽样的分布式约束性关联规则挖掘算法研究*)

李 宏 陈松乔 杜剑峰 陈建二
(中南大学 信息科学与工程学院 长沙 410083)

摘 要 本文采用抽样的方法,在基于约束的 Eclat 类算法(例如 EclatA 和 EclatM)的基础上,提出了一种分布式约束性关联规则的挖掘算法——DMCASE 算法。本算法在各数据站点上对一个较小的样本采用基于约束的 Eclat 类算法,挖掘局部约束频繁项集,采用归纳学习的方法归并所有局部约束频繁项集,产生全局约束频繁项集。只需 1 次扫描数据库,挖掘效率较高。实验证明:该算法是一种十分有效的解决基于约束条件下的分布式关联规则挖掘算法。

关键词 数据挖掘,约束性关联规则,抽样

Algorithm Analysis for Anti-monotone and Monotone Constraints Association Rules Mining

LI Hong CHEN Song-Qiao DU Jian-Feng CHEN Jian-Er
(School of Information Science&Engineering, Central South University, Changsha 410083)

Abstract An algorithm for distributed mining association rules with constraints, called DMCASE, is presented using sampling and constrained Eclat algorithm. At each database sites, sampling algorithm and constrained Eclat algorithm are implemented. And the local frequent itemsets that satisfied constraints are developed. They then are combined to global frequent itemsets that satisfied constraints based on learning from induction. DMCASE algorithm scans the whole database only once. It also is an algorithm with high efficiency. Results from our experiments show that the algorithm is an effective way to resolve the problem of distributed mining association rules with constraints.

Keywords Data mining, Association rules with constraints, Sampling

1 引言

关联规则挖掘是数据挖掘的重要研究内容。约束关联规则挖掘算法能够从大量的频繁项集中筛选出有用的规则,提高挖掘效率。随着分布式计算环境的日益普遍,研究分布式环境下基于约束的关联规则挖掘具有重要意义。目前,关联规则挖掘算法包括 Apriori 算法[1]、PARTITION 算法[7]、FP-growth 算法[4]、Eclat 算法[11]和抽样算法[10]等。并行和分布式关联规则挖掘算法主要有 CD 算法[2]、FDM 算法[3]、DDM 算法[8]和 D-Sampling 算法[9]等。目前可用的基于约束条件下的关联规则挖掘算法包括 CAP 算法[5]、CFG-约束频繁模式增长[6]和基于约束的 Eclat 类算法[13](例如反单调约束关联规则挖掘算法——EclatA 算法、单调约束关联规则挖掘算法——EclatM 算法)。分布式约束性关联规则挖掘算法还比较少。

Tovienven 提出的抽样方法[10]对数据库的一个较小的样本进行挖掘,用剩余的部分验证结果,只需 1 次扫描数据库,挖掘效率较高。本文采用了抽样的方法,在基于约束的 Eclat 类算法的基础上,提出了一种分布式约束性关联规则的挖掘算法——DMCASE 算法。DMCASE 算法在各数据站点上对一个较小的样本采用基于约束的 Eclat 类算法挖掘局部约束频繁项集,采用归纳学习的方法归并所有局部约束频繁项集,产生全局约束频繁项集。算法只需 1 次扫描数据库,无需对候选项集进行剪枝,占用内存较小。

2 相关概念与描述

目前,应用最广泛的约束是反单调约束和单调约束。

定义 1 反单调约束(Anti-monotone Constraint):给定一个约束条件 C_a,如果一个项集 S 不满足 C_a,S 的任何超集也不可能满足 C_a,则称约束条件 C_a 是反单调的。

定义 2 单调性约束(Monotone Constraint):给定一个约束条件 C_m,如果一个项集 S 满足 C_m,S 的任何超集也满足 C_m,则称约束条件 C_m 是单调的。

利用反单调约束和单调约束可以减少挖掘的时间和空间,提高挖掘效率。如著名的 Apriori 性质(频繁项集的任何非空子集也必然是频繁的)就是反单调的。这个性质用于 Apriori 算法的每次迭代,可以大大减少候选项集的个数,压缩关联规则的搜索空间。

其它几种约束的概念请参见文[6]。

设事务数据库 DB 存放在分布式系统 $S(S^1, S^2, \cdots, S^n, n$ 是站点数)上。各站点上的数据分别为 $\{DB^1, DB^2, \cdots, DB^n\}$。对任一项集 X,X. sup 和 X. sup^i 分别表示项集 X 在 DB 和 DB^i 中的支持度计数。min_sup 表示最小支持度阈值,$local_sup^i$ ($local_sup^i = min_sup \times DB^i$)表示 DB^i 上的局部支持度计数。局部约束频繁项集表示局部站点上的所有满足局部支持度和约束条件 C 的局部频繁项集,全局约束频繁项集表示在整个数据库 DB 上的所有满足最小支持度和约束条件 C 的频繁项集。分布式挖掘约束性关联规则的主要任务就是在给定最小支持度的情况下在多个站点上挖掘满足约束条件 C

*)国家自然科学基金(60433020)资助。李 宏 副教授,从事 KDD、分布式处理技术和信号处理研究;陈松乔 教授,博士生导师,从事决策支持和软件工程研究;杜剑峰 工程师,硕士生,从事数据挖掘研究;陈建二 长江学者,从事计算机算法复杂性研究。

维普资讯 http://www.cqvip.com

的全局频繁项集。

3 相关算法描述

3.1 抽样[10]

定义3 负边界：一个频繁项集的负边界是指它的任一子集均是频繁项集，但其本身并不频繁。例如：I={A,B,C}，假设频繁项集 L={{A},{B},{C},{A,B}}，则 $Bd^-(L)$={{B,C},{A,C}}。例如，对于集合{B,C}来说，它的任一子集{B}和{C}均属于 L，但其本身不属于 L。在关联规则挖掘的层次算法中，负边界是产生的候选项集中的所有非频繁项集。

抽样算法的主要思想是取一个比 min_sup 还小的支持度阈值 low_sup，在抽样数据 s 上用层次算法求出支持度大于 low_sup 的所有项集的集合，记为 S(假设 S 是满足 min_sup 的频繁项集的超集)，并得出负边界 $Bd^-(S)$。扫描数据库 DB 的剩余部分，计算 $S\cup Bd^-(S)$在整个数据库 DB 上的实际支持度。如果负边界 $Bd^-(S)$中均是非频繁项集，则 S 是频繁项集的超集。输出其中所有满足 min_sup 的项集，否则报告出现错误。如果 $Bd^-(S)$中存在某个频繁项集 Y，则 Y 的超集也可能是频繁的。解决的方法是将 Y 加到 S 中(即 $S=S\cup Y$)，再求出负边界 $Bd^-(S)$，再扫描数据库 DB，再验证，直到 $Bd^-(S)$中均为非频繁项集为止。这需要进行第 2 次数据库扫描。抽样方法是牺牲了一些精度来换取效率。当数据量比较大时，使用抽样方法具有很高的效率，但不能保证挖掘出所有满足最小支持度阈值的频繁项集。

定理1 给定一个属性集 X 和一个随机样本 s，在给定的误差范围 ε 及误差超出此值的最大概率 δ 情况下，s 的大小是：

$$|s|\geqslant\frac{1}{2\varepsilon^2}\ln\frac{2}{\sigma} \tag{1}$$

定理2 给定一个频繁项集 X 和一个随机样本 s，在给定的误差范围 ε 及误差超出此值的最大概率 δ 情况下，low_sup 的大小是：

$$low_suup<\min_sup-\frac{1}{2|s|}\ln\frac{1}{\sigma} \tag{2}$$

详细的定理证明请参见文[10]。

算法1 抽样

输入：数据库 DB，最小支持度阈值 min_sup，抽样样本使用的支持度阈值 low_sup(low_sup 的值要小于 min_sup)。

输出：DB 中满足 min_sup 的所有频繁项集及其支持度计算，或错误记录。

(1)从 DB 中随机抽取样本 s。

(2)S=F(s,low_sup)；//计算 s 中所有满足 low_sup 的项集。

(3)F={X∈S∪Bd^-(S)|X. sup≥min_sup}；//计算 S∪Bd^-(S)在 DB 上的实际支持度。

(4)对于所有 X∈F，输出 X 和 fr(X,DB)。

(5)如果出现错误(关于错误的详细说明请见 4.2 中的完备性分析)，则输出错误记录。

3.2 基于 Eclat 算法的约束关联规则挖掘算法实现[13]

利用 Eclat 算法的概念格理论和等价类划分方法，能够将约束条件融入到基于垂直数据分布的关联规则挖掘算法中。该算法采用了垂直数据库结构，利用基于前缀的等价关系将概念格划分为较小的子概念格。在自底向上的频繁集搜索过程中，同时进行约束条件的检验。算法的详细描述请参见文[13]。

引理1[11] 由等价关系 θ_k 生成的等价类$[X]_{\theta_k}$是 P(I)的1个子概念格。

定理1 Eclat 算法中的概念格和等价类划分方法可以确保利用自底向上的搜索方法遍历所有满足约束条件的频繁项集。

证明：由引理1可得出，对于数据垂直分布，可以采用概念格和等价类划分生成完备的频繁项集 L。令所有满足约束条件的频繁项集的集合为 S，根据约束的定义可知 S⊆L。因此，可以将约束整合到等价类的自底向上的搜索过程中，在发现频繁项集的同时进行约束条件的检验。

例如，根据反单调约束和 Eclat 算法的特点，只要在自底向上的搜索算法中验证支持度计数时，同时对约束条件进行判断，就可以挖掘出所有满足反单调约束条件的频繁项集。详细的反单调约束关联规则挖掘算法 EclatA 描述如下：

算法2 EclatA 算法

```
Algorithm Bottom_Up(S){
For all atoms A_i∈S do
Ti=∅;
For all atoms A_j∈S, with j>i do
  R=A_i∪A_j;
  L(R)=L(A_i)∩L(A_j);
  If σ(R)≥min_sup and C(R)==true then
    T_i=T_i∪{R}; F_|R|=F_|R|∪{R};
For all T_i≠∅ do Bottom_Up(T_i);}
```

其中，S 为等价类；A_i 为原子；σ(R)为项集 R 的支持度计数。若 C(R)==true，则表明项集 R 满足约束条件 C；若 C(R)==false，则表明项集 R 不满足约束条件 C；$F_{|R|}$ 为输出的频繁项集。

4 基于抽样的分布式约束关联规则挖掘算法实现

DMCASE 算法的主要思想是：首先在各分站点上进行随机抽样，抽取的事务集合为 s。对于事务集合 s，使用一个小于各站点的局部支持度计数(local_sup^i)的支持度计数 low_sup^i，采用基于约束的 Eclat 算法计算所有满足 low_sup^i 的项集 S^i。如果用户想要挖掘出所有满足约束条件的全局频繁项集，则按照 4.2 进行完备性分析，验证并补充抽样算法中丢失的局部约束频繁项集。将最终得到的满足约束条件的局部频繁项集记为 F^i。最后用归纳学习的方法归并 F^i($F=\bigcup_{i=1}^{n}F^i$)，计算 F 中所有满足全局支持度阈值 min_sup 和约束条件的所有全局频繁项集。

4.1 数据结构

为了确保只扫描1次数据库，我们采用了前缀树结构和位矩阵作为基本的数据结构。前缀树的每个节点代表1个项集及其支持度计数，其子女是以该节点所代表的项集为前缀的项集，每个项目所在的节点和它所有的子孙就构成了1个子概念格。例如在图2(a)中，项集{A}所在的节点的子女代表了以{A}为前缀的项集({AC},{AD},{AT},{AW})，项集{A}及其所有子孙({AC},{AD},{AT},{AW},{ATW})就构成了1个子概念格(即等价类[A])。

根据本算法，前缀树的构建和遍历均采用深度优先的方法。如果1个节点是满足约束条件的频繁项集，则向下扩展，计算以该项集为前缀的子节点是否是满足约束条件的频繁项集。如果是，则继续向下扩展，否则转向另一个前缀。如图2

维普资讯 http://www.cqvip.com

(a)中,初始构建前缀树时,首先构建项目 A 及其所有子孙节点,然后再依次构建项目 C、D、T、W 的子孙节点。对于项目 A,首先构建项集{AC},由于{AC}不满足约束条件,因此不向下扩展;然后构建项集{AD},而{AD}不频繁,因此也不再向下扩展;其次是项集{AT},{AT}是满足约束条件的频繁项集,则向下扩展为{ATW},依此类推。

每个项目在事务中的 tid-list 存储在 1 个位向量中,所有位向量组成 1 个位矩阵。位向量由"0"和"1"组成,位的个数等于事务的个数。如果项目在某个事务中,则代表该事务的位为"1",否则为"0"。如表 2 中,站点 1 上的项目 A 的 tid-list 为{1,3,5,6},则代表项目 A 的位向量为"101011"。项目 C 的 tid-list 为{1,2,6},则代表项目 C 的位向量为"110001"。项目的支持度计数为位向量中"1"的个数。如项目 A 在站点 1 上的支持度为 4,项目 C 在站点 1 上的支持度为 3。由于在本算法中采用了 Eclat 算法,因此项集的位向量通过原子项的位向量的与运算产生。如站点 1 上的项集{AC}的位向量为"100001",支持度计数为 2。因此只要在算法的初始阶段就对数据库进行扫描,建立起对应的位矩阵,当挖掘过程中产生候选项集时,就不必再重复扫描数据库,只需对位矩阵中的相应位向量进行与计算即可得出该候选项集的支持度计数。

4.2 抽样完备性分析

由于抽样算法是对一个随机抽取的数据库样本进行挖掘,因此在选取的样本 s 较小、数据库的数据倾斜严重或设置的 low_sup 较大等情况下,采用抽样算法不会产生完备的频繁项集,存在丢失少数频繁项集的现象,即抽样算法报告存在错误。下面详细描述抽样算法的错误情况和解决方法。

(1)负边界 $Bd^-(S)$ 中存在某个频繁项集 Y,即项集 Y 存在于样本 s 中,虽然在样本 s 中是不频繁的,但其在整个数据库中是频繁的。此外,如果项集 Y 是频繁的,则 Y 的超集也可能是频繁的。文[10]中提到的解决方法是将 Y 加到 S 中(即 $S=S\cup Y$),再求出负边界 $Bd^-(S)$,再扫描数据库 DB,再验证,直到 $Bd^-(S)$ 中的项集的支持度计数均小于 min_sup 为止。这就需要进行第 2 次数据库扫描,验证 Y 和 Y 的超集是否频繁。在 DMCASE 算法中,同样采用这种方法。但是由于采用了位矩阵,因此查找项集 Y 及其超集的支持度计数并不需要再次扫描数据库,只需进行对应位向量的与操作即可,真正做到了只需 1 次扫描数据库。

(2)S 及其负边界 $Bd^-(S)$ 中均不存在某个频繁项集 Y,即样本 s 中不包括项集 Y,因此采用抽样算法不会产生频繁项集 Y 及其超集。产生这种情况的主要原因是抽取的样本较小或数据库中存在数据倾斜的问题。文[10]中采用逐渐扩大样本或重复随机抽取样本挖掘来解决这种问题,这就需要不断扫描数据库。从上述分析可以看出,该频繁项集 Y 必是频繁 1 项集或频繁 1 项集的超集。在 DMCASE 算法中,解决的方法是在各站点进行抽样挖掘后,无论是否需要负边界验证,都对剩余数据进行扫描。如果在剩余数据中存在某个满足约束的候选 1 项集的支持度计数大于给定的局部支持度,则对该 1 项集及其所有超集进行验证。根据本算法的数据结构,只需将该 1 项集插入到前缀树的根节点下,并以其为前缀,将其所有超集及支持度计数作为子孙对应插入到前缀树中。由于每个 1 项集在事务中的 tid-list 都存储在 1 个位向量中,因此以上操作不需再次扫描数据库。

综上所述,用户可以根据情况选择是否进行完备性测试。如果允许牺牲一些精度来换取高效率,用户可以不进行完备性测试;反之则选择进行完备性测试。如果进行完备性测试,则对各分站点上抽取的样本,使用一个小于各站点的局部支持度计数($local_sup^i$)的支持度计数 low_sup^i,以及采用基于约束的 Eclat 类算法进行挖掘的方法,能够保证各站点挖掘出的 S^i 是局部频繁项集的超集;而根据分布式数据库特有的数据倾斜特性,全局频繁项集必定在某个分站点上是局部频繁的,即各分站点的局部频繁项集的并集是全局频繁项集的超集,因此各站点挖掘出的 S^i 的并集必定是全局频繁项集的超集。此外,根据约束条件和 Eclat 算法的特点,只要在自底向上的频繁集搜索过程中,同时对约束条件进行判断,就可以挖掘出所有满足约束条件的频繁项集。因此,在这种情况下,我们采用抽样方法和基于约束的 Eclat 类算法能够保证产生所有满足约束条件的全局频繁项集。

4.3 归纳学习

人工智能中的归纳学习是一种重要的机器学习方法,是指学习元对环境提供的大量输入信息进行推理归类和对共性的分析,抽象出一般的概念和规则[14]。在 DMCASE 算法中,采用归纳学习来归并各站点产生的局部约束频繁项集,产生最终的全局约束频繁项集。利用归纳学习发现频繁项集的框架如图 1 所示。

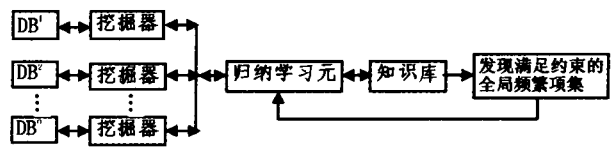


图 1 归纳学习挖掘框架

从图 1 中可以看出:各站点上挖掘出的满足约束条件的局部频繁项集作为归纳学习元的输入信息,归纳学习元对这些局部约束频繁项集的信息(如项集的支持度计数、所在的站点等)进行归纳和对比分析,并将得出的结果送入知识库。对于每一个从挖掘器中产生的项集 X,归纳学习元都将在知识库中查找其信息。如果项集 X 在知识库中存在,且来自不同站点,则将其支持度计数与知识库中该项集的支持度计数相加,并送回知识库;反之则将其信息输入到知识库中。待所有满足约束的局部频繁项集均学习后,归纳学习元再次查找知识库,若存在某个项集缺少在某些站点上的支持度计数,则归纳学习元将该项集向其它站点广播,采集其在其它站点上的支持度计数。知识库中存储的项集的信息用于发现满足约束的全局频繁项集。在学习过程中,已发现的全局约束频繁项集还将反馈回归纳学习元,用于指导进一步的学习。例如,若某个从挖掘器输入的项集已是全局约束频繁项集,则归纳学习元不必将其与知识库中的信息进行对比,可直接送入知识库,进行支持度计数的加操作即可。可任选 1 个站点作为归纳学习的元站点。各站点不需同步传输。

4.4 DMCASE 算法描述

算法 3 DMCASE 算法

输入:在站点 S^i 上的数据集 DB^i,约束条件 C,最小支持度阈值 min_sup^i,各站点的局部支持度计数 $local_sup^i$,抽样用的支持度计数 low_sup^i($low_sup^i<local_sup^i$)。

输出:DB 中满足约束条件 C 的所有全局频繁项集 F

```
Algorithm DMCASE{
  For(i=1,i++,i<n){
    draw a random sample s^i from DB^i;// 从 DB^i 中随机抽取样本 s^i
    S^i=F(s^i,low_sup^i); // 用基于约束的 Eclat 类算法计算 s^i 中所有满足 low_sup^i 的局部约束项集
    If need to find all constrainted frequent items then{ // 完备性验
```

维普资讯 http://www.cqvip.com

```
        证
        F^i={X∈S^i∪Bd^-(S^i)|X.sup^i≥local_sup^i}; //计算 S^i∪Bd^-(S^i)在 DB^i 上的实际支持度
        Add the lost frequent 1-itemset and its super itemsets that satisfy constraint C to F^i;//添加丢失的满足约束条件的频繁 1 项集及其超集到 F^i
    }}
    // 以下代码在归纳学习的元站点 DB^i 上执行
    receive all F^i;//
    F=∪_{i=1}^{n} F^i; // 用归纳学习产生全局约束频繁项集
}
```

5 实例分析及实验结果

5.1 实例分析

假设事务数据库 DB 存在于 3 个数据站点上，每个数据站点上的数据集分别为 DB^1，DB^2 和 DB^3，每个 DB^i 上有 6 个事务，事务中的项按字典顺序存放。表 1 为事务数据库 DB。令最小支持度阈值 min_sup 为 9，约束条件 C=sum(S)≤180 为反单调约束，各项的值分别为：⟨A(50)，C(150)，D(10)，T(20)，W(30)⟩。假设各分站点上随机抽取的事务集合为 s。为方便起见，设各站点使用的 local_sup(局部支持度计数)、low_sup(s 中使用的支持度计数)和抽取的事务个数均相同，$local_sup^i$ 为 3，low_sup^i 为 2，s 包括 3 个事务。图 2 和图 3 描述了实际的挖掘过程。

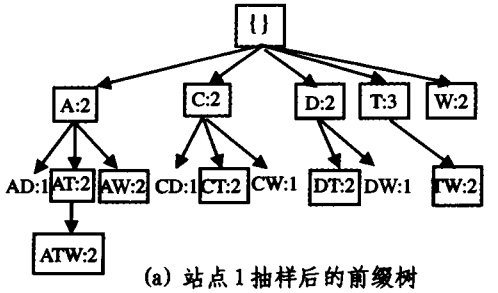
表 1 事务数据库

Site	TID	Items	Sampled TID	Sampled Items
DB^1	1	A,C,T,W		
	2	C,D,T	1	A,C,T,W
	3	A,T,W	2	C,D,T
	4	D,T,W	5	A,D,T,W
	5	A,D,T,W		
	6	A,C,D,W		
DB^2	1	C,T,W		
	2	A,C,D	1	C,T,W
	3	A,C,T,W	5	A,C,D,W
	4	C,D,W	6	A,C,D,T
	5	A,C,D,W		
	6	A,C,D,T		
DB^3	1	C,D,W		
	2	A,D,W	1	C,D,W
	3	A,C,D,W	2	A,D,W
	4	A,C,D,T,W	3	A,C,D,W
	5	C,D,T		
	6	A,C,D,T		

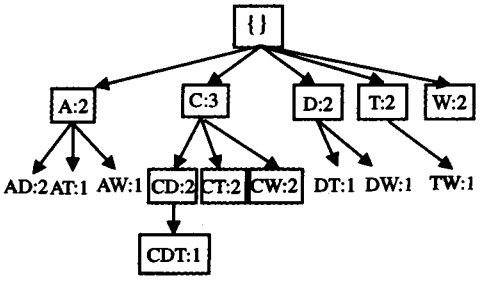
(1)挖掘各站点上的满足约束条件的局部频繁项集

首先，在各分站点上对抽样集合 s^i 按照 DMCASE 算法构建前缀树，前缀树的每个节点代表一个项集及其支持度计数。如图 2(a)中，A:2 表示项集{A}的支持度计数为 2。图中实线方框中的项集为满足反单调约束的频繁项集，如项集{A}；其它为满足反单调约束的非频繁项集，如项集{AD}。对于反单调约束，可将负边界的定义扩展到满足反单调约束的非频繁项集。图 2(a)、(b)、(c)分别为站点 1、站点 2 和站点 3 上由抽样集合 s 构成的前缀树。令 S^i 是各站点上的 s^i 中满足约束条件 C 和 low_sup^i 的项集，$Bd^-(S^i)$ 是对应的负边界(候选项集中满足反单调约束的非频繁项集)，则从图 2(a)、(b)、(c)分别可以看出：S^1 = {{A}，{AT}，{ATW}，{AW}，{C}，{CT}，{D}，{DT}，{T}，{TW}，{W}}，Bd-(S1) = {{AD}，{CD}，{CW}，{DW}}；S^2 = {{A}，{AD}，{C}，{CD}，{CT}，{CW}，{D}，{T}，{W}}，Bd-(S2) = {{AT}，{AW}，{CDT}，{DT}，{DW}，{TW}}；S^3 = {{A}，{AD}，{ADW}，{AW}，{C}，{CD}，{CW}，{D}，{DW}，{W}}，$Bd^-(S^3)=\varnothing$。

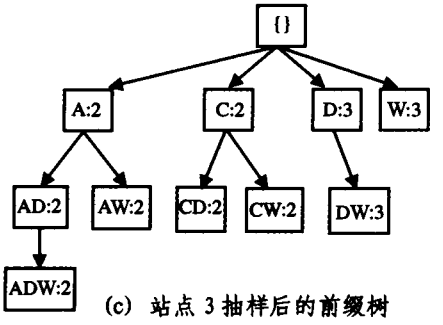
如果需要进行完备性测试，则扫描各站点的剩余数据库，计算 $S^i \cup Bd^-(S^i)$ 在 DB^i 上的实际支持度，并改变前缀树上相应节点的支持度计数值，利用负边界对结果进行验证。同时，添加抽样丢失的满足约束条件的频繁 1 项集及其超集。进行完备性测试后的前缀树如图 3 所示。



(a) 站点 1 抽样后的前缀树



(b) 站点 2 抽样后的前缀树



(c) 站点 3 抽样后的前缀树

图 2

从图 3(a)中可以看出，负边界 $Bd^-(S^1)$ 中的项集{DW}在站点 1 上是局部频繁的，即在站点 1 的抽样算法执行过程中出现了丢失局部频繁项集的现象，则应将项集{DW}加入集合 S^1 中，并验证其超集是否满足约束条件 C 和 $local_sup^1$。结果是其超集并不局部频繁(由于本算法中采用了位矩阵，因此查找项集{DW}及其超集的支持度计数并不需要再次扫描数据库，只需进行对于位向量的与操作即可)。从图 3(b)中可以看出，抽样算法在站点 2 上一次成功。比较图 2(c)和图 3(c)，可以看出在站点 3 上抽取的样本中丢失了满足约束条件的频繁 1 项集{T}，因此{T}及其超集被丢失。根据 4.2 中的完备性分析，DMCASE 算法能够找到项集{T}及其所有满足约束条件的频繁超集，并将其添加到集合 S^3 中。

最后，计算各站点上的满足约束条件的局部频繁项集的集合 F^i。从图 2(a)、(b)、(c)分别可以看出：F^1 = {{A}，{AT}，{ATW}，{AW}，{C}，{D}，{DT}，{DW}，{T}，{TW}，{W}}；F^2 = {{A}，{AD}，{C}，{CD}，{CT}，{CW}，{D}，{T}，{W}}；F^3 = {{A}，{AD}，{ADW}，{W}，{C}，{CD}，{CW}，{D}，{DW}，{W}，{T}，{TC}，{TCD}，{TD}}。

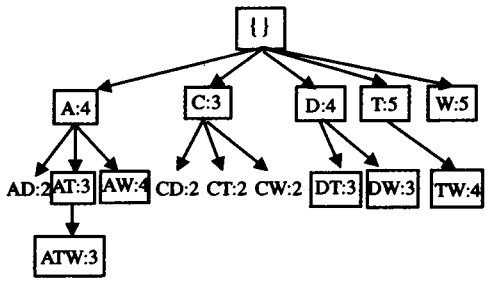
(2)采用归纳学习的方法挖掘满足约束条件的全局频繁项集

任选 1 个站点作为归纳学习的元站点，归并各站点的满

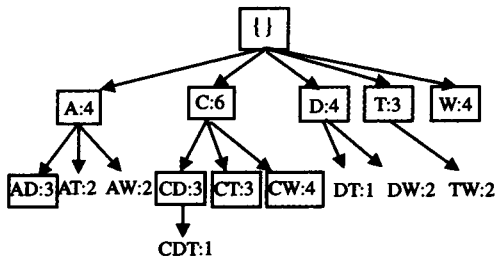
维普资讯 http://www.cqvip.com

足约束条件的局部频繁项集(即 $F=\bigcup_{i=1}^{n}F^i$),并根据 min_sup 筛选出满足约束条件的全局频繁项集的集合 F。

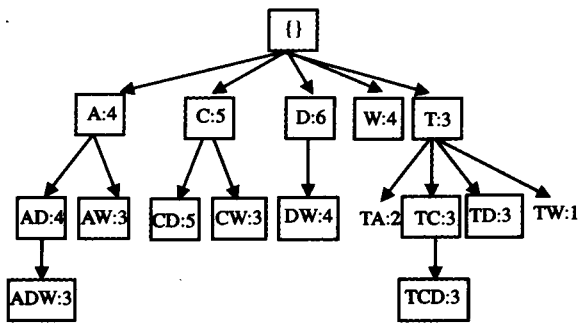
F={{A},{AD},{AW},{C},{CD},{CW},{D},{DW},{T},{W}}



(a) 站点 1 进行完备性验证后的前缀树



(b) 站点 2 进行完备性验证后的前缀树



(c) 站点 3 进行完备性验证后的前缀树

图 3

5.2 实验结果

为了验证算法的性能,用 VC++ 6.0 对 DMCASE 算法进行了测试。操作系统为 Windows2000 Server,CPU 为 P4 1.8GHz,内存为 512MB。实验数据集采用文[2]的合成数据方法。在每个站点上分别合成相同参数的数据集。算法采用随机抽样的方法,抽取的样本大小 s 和 low_sup 分别按照式(1)和式(2)进行取值(令 $\varepsilon=0.01,\delta=0.001$)。为了更好地测试 DMCASE 算法的性能,对 DMCASE 算法和 SampleC 算法以及 FDMC 算法进行了对比测试。这里的 SampleC 算法是指在抽样算法的基础上添加了对约束条件的处理,用于挖掘局部约束频繁项集,并采用归纳学习的方法进行归并;而 FDMC 算法是指在 FDM 算法中添加了对约束条件的处理。上述 3 种算法采用相同的通信机制。表 3 描述了当数据集为 T10I8D400K、最小支持度阈值 min_sup=0.25%、反单调约束 C=sum(S.price)≤500 时,DMCASE 算法、Sample-C 算法和 FDM-C 算法的比较情况。图 4 为上述 3 算法在执行时间方面的比较情况。

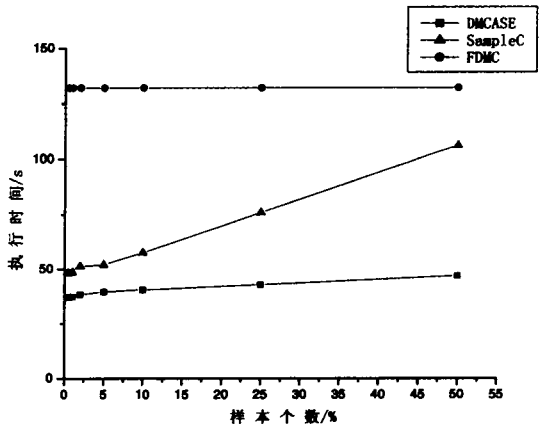


图 4 DMCASE 算法、SampleC 算法和 FDMC 算法的执行时间比较

表 3 DMCASE 算法、SampleC 算法和 FDMC 算法的比较情况

Sample Size(K)	Sample Percent(%)	low_sup (%)	Frequent Size	Accuracy (%)	Run Time(s)		
					DMCASE	SampleC	FDMC
200	50	0.23	1079	100	46.753	106.187	132.11
100	25	0.22	1079	100	42.733	75.828	132.11
40	10	0.21	1051	97.40	40.473	57.546	132.11
20	5	0.20	1076	99.72	39.603	51.953	132.11
8	2	0.15	1071	99.25	38.38	51.188	132.11
4	1	0.14	1062	98.42	37.27	48.484	132.11
2	0.5	0.13	990	91.75	37.01	48.281	132.11

从上述实验结果可以看出:

(1)DMCASE 算法的挖掘效率较高。DMCASE 算法的执行时间要远远小于 SampleC 算法和 FDMC 算法。SampleC 算法是在层次算法的基础上采用了抽样技术,DMCASE 算法则是在 Eclat 算法的基础上进行抽样。这是由于 Eclat 算法具有传统的层次算法无法比拟的优越性能。此外,由于 FDMC 算法没有采用抽样,因此整个挖掘时间几乎高于 DMCASE 算法 3 倍。随着样本大小的减少,DMCASE 算法挖掘的执行时间也随之减少,FDMC 算法的执行时间保持不变。

(2)DMCASE 算法的执行精度较高。即使当抽样的样本仅为整个数据库的 0.5%时,也能得到 91.75%的挖掘准确率,而且挖掘的精度下降得比较缓慢。样本大小为 50%与 0.5%相比,挖掘的准确度仅下降 8.25%。

5.3 算法性能分析

与其它算法相比,DMCASE 算法具有以下几个特点:

(1)使用了抽样方法,只需扫描 1 次数据库,大大提高了

维普资讯 http://www.cqvip.com

挖掘效率。

(2)在各站点上,使用基于约束的 Eclat 算法来挖掘满足约束的局部频繁项集。采用了垂直数据库结构,利用基于前缀的等价关系将概念格划分为较小的子概念格,在自底向上的频繁集搜索过程中,同时进行约束条件的检验。算法无需复杂的 hash 数据结构;在生成候选项集后,不需再进行剪枝;可以在内存中独立处理每一个子网格,挖掘效率较高,占用的内存比较小。

(3)DMCASE 算法在各站点均采用前缀树和位矩阵作为存储结构,使算法真正做到了只需 1 次扫描数据库。

(4)各站点独立挖掘满足约束条件的局部频繁项集,各站点之间不需要同步。

(5)采用 4.2 中的完备性验证方法,能够使用抽样算法挖掘出完备的满足约束条件的全局频繁项集。

(6)采用归纳学习的方法归并满足约束条件的局部频繁项集。各站点不需同步传输挖掘出的信息,归纳学习元通过不断地接收信息和学习发现满足约束的全局频繁项集,大大降低了网络的通信代价。

结束语 本文将抽样与基于约束的 Eclat 类算法相结合,提出了一种分布式约束性关联规则的挖掘算法——DMCASE 算法。与其它同类算法相比,DMCASE 算法具有挖掘效率高、只需 1 次扫描数据库等优越性能。本文仅以反单调约束为例,描述了 DMCASE 算法的实现。DMCASE 算法同样适用于基于单调性约束、简洁性约束和可转变约束的分布式环境下关联规则的挖掘问题。如对于单调性约束,只需在 DMCASE 算法中将负边界设定为非频繁项集即可。进一步的工作是如何将 DMCASE 算法在多种约束条件下的实现方法更加完善和优化,并且降低网络通信代价、提高归纳学习的效率以及将挖掘结果做到最优。

参 考 文 献

1 Agrawal R, Shafer J C. Parallel mining of Association rules: Design, Implementation, and Experience [J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6): 962～969

2 Agrawal R, Srikant R. Fast algorithms for mining association rules [A]. In: Proceedings of the 20th International Conference on Very Large Databases [C], Santiago, Chile, 1994. 487～499

3 Cheung D W, Han J, Ng V T, et al. A Fast Distributed algorithm for Mining Association Rules [A]. In: Proceedings of 1996 International Conference on Parallel and Distributed Information Systems [C]. Miami Beach, 1996. 31～44

4 Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [A]. In: Proceedings of the ACM-SIGMOD Conference on Management of Data [C], Dallas, ACM Press, 2000. 1～12

5 Ng R T, Lakshmanan L V S, Han J, et al. Exploratory mining and pruning optimizations of constrained association rules [A]. In: Proc. 1998 ACM-SIGMOD Int Conf Management of Data [C]. Seattle, Washington, 1998. 13～24

6 Pei J, Han J. Can we push more constrained into frequent pattern mining? [A]. In: Proc. 2000 Int Conf Knowledge Discovery and Data Mining [C]. Boston, MA, 2000. 350～354

7 Savasere A, Omiecinski E, Navathe S. An efficient algorithm for mining association rules in large databases [A]. In: Proc. of the VLDB Conference [C]. Zurich, 1995. 432～444

8 Schuster A, Wollff R. Communication-efficient Distributed Mining of Association Rules. In: Proc. of the 2001 ACM-SIGMOD Int Conf Management of Data [C]. Santa Barbara, 2001. 473～484

9 Schuster A, Wollff R, Trock D. A High-Performance Distributed Algorithm for Mining Association Rules [A]. In: Proc. of the 2001 ACM-SIGMOD Int Conf Management of Data [C]. Santa Barbara, 2003. 473～484

10 Toivonen H. Sampling large databases for association rules [A]. In: Proc. of the 22nd Int'l Conf on Very Large DataBases [C]. Mumbai, 1996. 134～145

11 Zaki M J. Scalable algorithms for association mining [J]. IEEE Transactions Knowledge and Data Engineering, 2000, 12(3): 372～390

12 王春花,黄厚宽. 利用抽样技术分布式开采可变精度的关联规则[J]. 计算机研究与发展,2000,37(9):1101～1106

13 杜剑峰,李宏,陈松乔,陈建二. 单调和反单调约束条件下关联规则的挖掘算法分析[J]. 计算机科学,2005,32(6):142～144

14 蔡智兴,徐光佑. 人工智能及其应用. 第二版[M]. 北京:清华大学出版社,1995

(上接第 110 页)

务 T_i^1 的所有写操作不影响事务 T_k^2 的所有读操作(或事务 T_k^2 的所有写操作不影响事务 T_i^1 的所有读操作)。

事务 T_i 的操作 op_i^t 可以看作是 T_i 的一个子事务处理。因此,执行顺序 $H=(\Gamma,\rightarrow)$ 中的任何两个事务 T_i, T_j,如果 Validation(T_i)< Validation(T_j),必有 commit[$T_i(a)$] > Read[$T_j(a)$],说明乐观嵌套工作流事务的执行是无级联调度的。

结束语 乐观嵌套事务模型在协同工作流管理系统 Synchroflow 中实现事务管理。协同工作流管理系统采用信牌驱动工作流模型来实现对工作流程的建模。我们在工作流建模模型的操作原语与乐观嵌套事务模型的操作原语间建立了二者之间的对应关系,通过对应关系能够把工作流过程定义转化成为乐观嵌套事务树或事务森林。这种转换使得事务工作流执行过程中,其操作原语和乐观事务模型的操作原语是一致的,工作流活动的转移控制与乐观嵌套事务模型的子嵌套事务的生成过程及其表示方式也是一致的。

协同工作流管理系统中采取流程运行时分别对流程与活动进行实例化的策略,而工作流活动或活动组对应了工作流事务或嵌套事务,因此,工作流事务的时间戳可在工作流活动实例化时指定,并且根据工作流活动的类型差异,对工作流事务的时间戳的持久性进行分类管理,来保证长时间执行事务的正确执行。乐观嵌套事务模型在事务工作流应用过程中处理能力强,并且在定义嵌套事务时简单方便,易于实现。

参 考 文 献

1 Schuldt H, Alonso G, Beeri C, Schek H J. Atomicity and Isolation for Transaction Processes. ACM Transaction on Database Systems, 2002, 27(1): 63～116

2 丁柯,金蓓弘,冯玉琳. 事务工作流的建模和分析. 计算机学报,2003,26(10)

3 Wachter H, Reuter A. The ConTract Model, Database Transaction Models for Advanced Applications. In: A. H. Elmagarmid, ed. Morgan Kaufmann, San Jose, eds. 1992. 123～158

4 沈备军,陈诚,居德华. 基于规则的软件过程事务模型工作. 软件学报,2002,13(1)

5 Thomasian A. Distributed Optimistic Concurrency Control Methods for High- Performance Transaction Processing. IEEE Transaction on Knowledge and Data Engineering, 1998, 10(1)

6 Kung H T, Robinson J T. On Optimistic Methods for Concurrency Control. ACM Transaction on Database Systems, 1981, 6(2): 213～226

7 董云卫,郝克刚. 一种乐观嵌套工作流事务模型. 计算机科学,2005,32(8)

8 Eliot J, Moss B. Nested Transaction: an Approach to Reliable Distributed Computing: [MIT/LCS/RT-260]. 1981

9 Bertino E, Catania B, Ferrari E. A Nested Transaction Model for Multilevel Secure Database Management System. ACM Transaction on Information and System Security, 2001, 4(4): 321～370

10. Weikum G, Deacom A, Schaad W, Schek H. Open Nested Transaction in Federated Database System. Data Engineering IEEE Computer Society, 1992, 16(2)