

一种基于上三角频繁项集矩阵的频繁模式挖掘算法

王文正¹, 王文平¹, 许映秋², 谈英姿³

(1 东南大学 经济管理学院, 江苏 南京 211189; 2 东南大学 机械工程学院, 江苏 南京 211189;

3 东南大学 自动化学院, 江苏 南京 211189)

摘 要: 提出了一种高效挖掘数据的频繁项目集模式的算法 FIA. 该算法采用一种二进制符号来表示数据, 在仅扫描数据库一次之后, 建立起二进制向量与上三角频繁项集矩阵, 根据两者来产生出频繁项集. 从而有效地缩小了搜索空间, 加快了处理速度. 通过实验表明, FIA 算法比 Apriori 算法更有效.

关键词: 数据挖掘; 关联规则; 逻辑运算; 频繁项集; 上三角频繁矩阵

中图分类号: TP301

文献标识码: A

文章编号: 1000-7180(2010)09-0138-06

Algorithms for Mining Frequent Itemsets Based on Upper Triangular Frequent Matrix

WANG Wen-zheng¹, WANG Wen-ping¹, XU Ying-qiu², TAN Ying-zi³

(School of Economics and Management, Southeast University, Nanjing 211189, China;

2 School of Mechanical Engineering, Southeast University, Nanjing 211189, China;

3 School of Automation, Southeast University, Nanjing 211189, China)

Abstract: The traditional algorithms for mining association frequent patterns generate conditional sub tables, which costs much runtime and memory space. To solve these problems, a new algorithm FIA (Frequent Itemset Algorithm) is proposed. The FIA algorithm adopts a binary of symbols to compress the store data. The algorithm using logic of symbols to express data in a database, which only after one scan, and establish a binary vector and upper triangular frequent matrix, according to the both to produce a set of frequently. Thereby effectively narrowing the search space, speed up the processing speed. Through analysis showed that, FIA algorithm more effective than Apriori algorithm.

Key words: data mining; association rule; logical operation; frequent itemset; upper triangular frequent matrix

1 引言

关联规则是数据挖掘的重要内容之一, 发现频繁项集是其应用中的关键技术和步骤^[1-2]. 第一个对关联规则进行研究的是 R. Agrawal(1993)提出的 AIS 算法^[2], 通过多次扫描数据库生成的候选强项集来产生频繁项集; 1994 年, 他又提出了经典算法 Apriori 算法, 把关联规则挖掘分为两个阶段, 并引

入修剪候选集的方法来产生频繁项集; 其后很多学者提出了许多 Apriori 算法的变体, 如引进了 Hash 方法、划分技术、采样方法, 旨在提高算法挖掘规则的效率; Jiawei Han(2000)等提出 FP-growth 算法^[3], 将数据的重要信息压缩在一个称为频繁模式树(FP-Tree)的数据结构中, 然后基于 FP-Tree 生成数据集中所有的频繁项集. 这些 Apriori 系列算法的主要缺陷在于产生了数量巨大的候选项集、多

收稿日期: 2009-05-26; 修回日期: 2009-08-10

基金项目: 国家自然科学基金项目(70571013); 教育部新世纪优秀人才支持计划项目(NECT-06-0471); 江苏省教育厅高等教育改革研究课题

层次重复扫描数据库检验这些候选项集是否为频繁项目集;而频繁模式树算法的增长需要生成大量的条件模式树,并采用递归的方法,需要大量的执行时间和储存空间.

为解决 Apriori 系列算法存在的问题,Show - Jane Yen(2002) 等提出基于关联图的算法^[4],构造关联图来展示项目集之间的关系,使用向量逻辑“与”运算遍历关联图来产生频繁集;李哲(2003)等提出元向量法挖掘关联规则^[5],通过向量的逻辑运算寻找频繁元向量集,根据有关阈值挑选出有趣且有效的强关联规则;陈耿(2004)等提出 B-Gen 算法^[6],对挖掘对象进行“或”、“与”、“异或”等逻辑运算操作以生成二进制形式的候选频繁项集及相应的计算支持数;胡慧蓉(2005)等提出 SLIG 算法^[7],引入了项目可辨识向量及其“与”运算,将频繁项目集的产生过程转化为项目集的关系矩阵中向量运算过程.但这些相关研究是将一个事务转化为一个二进制组,并在产生频繁项集之前仍存在候选项集.

文中提出的 FIA 算法是利用逻辑“与”运算构建上三角频繁项集矩阵,通过分析该矩阵来生成高层频繁项集,避免候选项集的出现.

2 问题描述

2.1 关联规则描述

设 $I = \{i_1, i_2, \dots, i_n\}$ 是由 n 个不同的项目组成的集合.一条关联规则就是形如 $X \Rightarrow Y$ 的蕴含式,其中 $X \subseteq I, Y \subseteq I$ 且 $X \cap Y = \phi$.

定义 1 对于关联规则 $X \Rightarrow Y$, 其支持度为 $\text{Sup}(X \Rightarrow Y) = P(X \cup Y)$; 其置信度是 $\text{Conf}(X \Rightarrow Y) = P(X/Y)$.

定义 2 项目集 $\{i_1, i_2, \dots, i_k\}$ 在数据库 DB 中的频度是指其包含该集的事务数,如果它的支持度不小于给定的最小支持度阈值 Min_sup , 则称其为 DB 中的一个频繁 k -项集,记为 L_k .

定理 1 频繁项目集的所有非空子集一定是频繁项集.

定理 2 非频繁项集的超集一定不是频繁项集.

推论 1 $\{i_1, i_2, \dots, i_k, i_t\} \in L_{k+1} (t \geq k \geq 2)$, 则有 $\forall j (1 \leq j \leq k), \{i_j, i_t\} \in L_2$, 即 $\{i_1, i_2, \dots, i_k, i_t\} \in L_{k+1} (k \geq 2)$ 的必要条件是:① $\{i_1, i_t\}, \{i_2, i_t\}, \dots, \{i_k, i_t\}$ 都必须属于 L_2 ;② 能与 i_t 组成频繁 2-项集且比 i_t 编号小的项目至少要有 k 个.

多数算法把关联规则的挖掘分解为两个过程:

第一,产生频繁项集;第二,产生关联规则.其中第一个任务是关键^[1],文中的重点也是如何解决第一个任务上.

2.2 文中的思想

文中在前人的研究基础之上,把一个项目作为一个二进制向量,利用逻辑“与”运算来发现频繁项集模式,不需要迭代产生候选项集,仅需扫描数据库一次,得到二进制向量转化表及上三角频繁项集矩阵,利用该矩阵可以迅速得出任意两行对应的项目集,再通过扩展低层频繁集的最后一个项目来生成高层的频繁项目集,对该矩阵进行循环操作,直到没有“与”运算为止.并在此过程中采用优化策略,有效地缩小了搜索与储存空间,加快处理速度.

3 上三角频繁项集矩阵的构造

3.1 二进制向量与逻辑运算

根据定义 2 可知频繁项目集的定义和二进制的逻辑“与”运算有着本质的相似,其中频繁项目集实质是事务数据库中公共项目集的集合;逻辑“与”运算定义为集合 X 和 Y 的“与”,即 X, Y 两集合的所有共同的元素构成的一个新集合;所以可采用二进制表示频繁项集.

在事务数据库 DB 中,对于某一项目来说,要么在一个事务中,要么不在这一个事务中.因此,用二进制中的“1”或“0”来表示事务中是否包含该项,如果包含,该项目对应位上的值用“1”来表示,否则用“0”来表示.据此,可以在二进制向量的意义下定义关联规则的相关概念.

定义 3 设 $I = \{1, 2, \dots, n\}$, 项目 i 的二进制向量 B_i 定义为: $B_i = (b_{i1}, b_{i2}, \dots, b_{ij}, \dots, b_{im})$, 其中: m 为数据库中事务的总个数.

定义 4 项目 i 的支持度计数为 $\text{Sup_count}(B_i) = \sum_{j=1}^m b_{ij}$, 即 B_i 中“1”的个数;其支持度记为 $\text{Sup}(B_i) = \text{Sup_count}(B_i)/m$.

定义 5 项目集 $\{i_1, i_2, \dots, i_k\} (1 < k < n)$ 的支持度即是二进制向量 $B_1 \wedge B_2 \wedge \dots \wedge B_k$ 的支持度 $\text{Sup}(B_1 \wedge B_2 \wedge \dots \wedge B_k)$, 表示项目 $i_1, i_2, \dots, i_k$ 同时出现的频度.

二进制向量中每个位上代表某一固定的项,故一个事务可以用唯一的一个二进制数来表示,这样就完成了从事务数据库到用二进制向量的转化.

3.2 编码策略

文中利用逻辑“与”向量运算挖掘频繁项集的

算法主要是将事务数据库中的全部事务进行编码,然后再对各个项目进行逻辑“与”运算产生频繁项集.

编码原则:根据数据库中涉及到的项目数确定二进制向量的数目,根据数据库中的事务的数目决定编码长度,每个事务中的一个值在代码中对应一个位置.根据定义3可知,如果一个项目 i 在第 j 个事务中出现,那么项目 i 代码的第 j 位上设为“1”,否则设为“0”.例如, $DB = \{\{A, B, C\}, \{A, C, D\}, \{A, B, E, F\}, \{A, B, C, D, F\}, \{B, C, D, E\}\}$ 表示一个事务数据库 DB .为了便于描述,文中用1~6代替 $A \sim F$.经过对 DB 进行一遍扫描,得到全部项目的二进制向量如表1所示,并根据定义4可计算出其支持度数.

表1 项目二进制向量转化表

项目	向量标识	二进制向量	支持度数
A	B_1	(11110)	4
B	B_2	(10111)	4
C	B_3	(11011)	4
D	B_4	(01011)	3
E	B_5	(00101)	2
F	B_6	(00110)	2

从表1可知项目的支持度数分别为4、4、4、3、2、2.若设 $\text{Min_sup} = 40\%$,则最小支持度阈值为 $5 \times 40\% = 2$, DB 的频繁1-项集为 $L_1 = \{A, B, C, D, E, F\}$.

3.3 构造上三角频繁项集矩阵

定义6 设频繁1-项目集中的项目集为 $\{\{1\}, \{2\}, \dots, \{f-1\}, \{f\}\}$,项目总数是 f 个,则上三角频繁项目集矩阵(Upper Triangular Frequent Matrix, UTFM)可以定义为:

$$\text{UTFM}_{f-1, f-1} =$$

$$\begin{pmatrix} Z_{12} & \cdots & Z_{1i} & Z_{1,i+1} & \cdots & Z_{1j} & Z_{1,j+1} & \cdots & Z_{1f} \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & Z_{i-1,i} & Z_{i-1,i+1} & \cdots & Z_{i-1,j} & Z_{i-1,j+1} & \cdots & Z_{i-1,f} \\ & & & Z_{i,i+1} & \cdots & Z_{i,j} & Z_{i,j+1} & \cdots & Z_{i,f} \\ & & & & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & & & & Z_{j-1,j} & Z_{j-1,j+1} & \cdots & Z_{j-1,f} \\ & & & & & & Z_{j,j+1} & \cdots & Z_{j,f} \\ & & & & & & & \ddots & \vdots \\ & & & & & & & & Z_{f-1,f} \end{pmatrix}$$

其中:

$$Z_{ij} =$$

$$\begin{cases} 1; & \text{当 } i < j \text{ 且 } \text{Sup}(B_i \cap B_j) \geq \text{Min_sup} \\ 0; & \text{当 } i < j \text{ 且 } \text{Sup}(B_i \cap B_j) < \text{Min_sup} \end{cases}$$

对 UTFM 的几点说明:

(1) 由定理2可知,如果项目 i 不是频繁1-项集,那么 i 也不可能是频繁2-项集中的一个项目.所以该矩阵中的最大列数与行数不是数据库中全部项目的个数而是小于等于频繁1-项集中的个数.

(2) 由于 $B_i \wedge B_j = B_j \wedge B_i$,故矩阵存在着对称关系.文中构建的 $(f-1) \times (f-1)$ 矩阵除去第一行与最后一列是对称矩阵,其中包含了第2个到第 $(f-2)$ 个项目之间的全部关系,将 $(1,2) \cdots (1,f)$ 的值补于第一行,将 $(2,f) \cdots (f-1,f)$ 的值补于最后一列,这样就组成了上三角频繁项集矩阵.

(3) 该矩阵是 $(f-1) \times (f-1)$ 矩阵而不是 $f \times f$ 矩阵或 $(f-1) \times f$ 矩阵,是因为 $\text{UTFM}[i,j] = Z_{ij}$ 并且 $i < j$,对于对称矩阵 $f \times f$ 最后一行只存在一个可记录的矩阵值 $Z_{ff} = 0$,在本矩阵中 $Z_{ff} = 0$ 不起作用,且第 f 个项目是项目集中编号最大的项目,不存在编号比它还大的项目组成频繁2-项集,故可以去掉最后一行,构成 $(f-1) \times f$ 矩阵;在 $(f-1) \times f$ 矩阵中,对角线上为0,没有特殊意义,所以可以将纵坐标向后退一位,这样组成矩阵的对角线上也存储着有价值的信息,同时这样也减少 $(2f-1)$ 储存空间.

(4) 由定义6可知,如果 $Z_{ij} = 1$,则项目集 $\{i, j\}$ 为频繁2-项集;反之,如果 $Z_{ij} = 0$,则项目集 $\{i, j\}$ 肯定不是频繁2-项集,同时项目集 $\{i, j\}$ 也不会出现在比频繁2-项集高的频繁项集中.

(5) 文中构造的 UTFM 思路可以简单归结为:采用上三角矩阵表达方式,即当 $i < j$,对 B_i 与 B_j 进行逻辑“与”运算,并记录下来运算后二进制中“1”的个数以及项目信息,记为 Z_{ij} .将 Z_{ij} 与最小支持度阈值进行比较,如果 Z_{ij} 不小于最小支持度阈值,则记 $\text{UTFM}[i,j] = 1$,否则记 $\text{UTFM}[i,j] = 0$.

4 基于上三角频繁项集矩阵的算法

4.1 FIA 算法思路

文中提出的 FIA(Frequent Itemsets Algorithm)算法是将每一项目作为一个二进制向量,该算法共分为三个步骤:第一步是生成各个项目的二进制向量以及频繁1-项集,并记录相关的信息;第二步是构造成上三角频繁项集矩阵以及产生频繁2-项集;第三步是通过对二进制向量进行逻辑“与”运算并利用上三角频繁项集矩阵来生成高层频繁项集.

4.2 频繁项集生成算法

4.2.1 生成项目二进制向量及1-频繁项集

本步骤通过扫描一次数据库后得到每一项目的二进制向量并生成频繁1-项集,同时根据定义4生成计算每个项目的支持度计数,逐次判断项目是否为频繁1-项集.这一步骤的伪代码如下:

算法1 FIA-1 扫描一次数据库 DB,生成频繁1-项集.

- (1) Initialize the $B_i = 0$;
- (2) For ($i = 1; i \leq n; i++$) do
- (3) For ($j = 1; j \leq m; j++$) do
- (4) If the item i in the j transaction then
- (5) $\text{Sup_count}(i) + 1$
- (6) $b_{ij} = 1$
- (7) End if
- (8) Next j
- (9) If $\text{Sup_count}(i) \geq \text{Min_sup}$ then
- (10) $L_1 \cup = \{i\}$
- (11) Next i

4.2.2 构建上三角频繁项集矩阵

本步骤算法是在 FIA-1 的基础上进行的,根据定义6构建了一个上三角频繁项目集矩阵,其中任何两个向量的运算采用第一步骤中的二进制向量.根据上三角频繁项目矩阵可以得出 DB 的频繁2-项集.算法的伪代码如下:

算法2 FIA-2 构造上三角频繁项目集矩阵,生成频繁2-项集

- (1) Initialize the $\text{UTFM}[i, j] = 0$; //初始化上三角频繁项集矩阵,使其矩阵值为0
- (2) For ($i = 1; i \leq f; i++$) begin // f 是频繁1-项集中项目的个数
- (3) For ($j = i + 1; j \leq f - 1; j++$) begin
- (4) If $\text{Sup}(B_i \wedge B_j) \geq \text{Min_sup}$ then //若项目 i 和 j 的支持度不小于最小支持,则 $\{i, j\}$ 为频繁2-项目集
- (5) $L_2 \cup = \{i, j\}$;
- (6) $\text{UTFM}[i, j] = 1$;
- (7) End if
- (8) Next j
- (9) Next i

例如,表2所示的是文中 DB 的 UTFM,由该矩阵可以方便地得出频繁2-项集为 $L_2 = \{\{A, B\}, \{A, C\}, \{A, D\}, \{A, F\}, \{B, C\}, \{B, D\}, \{B, E\}, \{B, F\}, \{C, D\}\}$.

表2 表1的上三角频繁项集矩阵

UTFM _{5,5}	B	C	D	E	F
A	1	1	1	0	1
B		1	1	1	1
C			1	0	0
D				0	0
E					0

4.2.3 生成 k -频繁项集($k \geq 3$)

依据 UTFM 的值来判断是否能从 $(k-1)$ -项集($k \geq 3$)扩展成 k -项集,并判断是否为频繁 k -项集.判断的具体方法是:利用频繁 $(k-1)$ -项集的最后一项与比其编号大的项目 k 项目扩展,如果频繁 $(k-1)$ -项集中每个项目与 k 项目组成的2项集对应的 UTFM 值都是“1”,根据定理2的推论可知本次扩展得到 k -项集;再对这 k 个向量进行逻辑“与”运算,得到的向量中的“1”的个数如果不小于最小支持度阈值,那么就是一个 k -项集,否则不能扩展成 k -项集.例如, $\{i_1, i_2, \dots, i_{k-2}, i_{k-1}\} \in L_{k-1}$, i_t 是 i_{k-1} 比编号大的项目,若 $\text{UTFM}[i_{k-1}, i_t] = 1$,且满足 $\text{UTFM}[i_1, i_t] = \text{UTFM}[i_2, i_t] = \dots = \text{UTFM}[i_{k-1}, i_t] = 1$,则可以将 $\{i_1, i_2, \dots, i_{k-2}, i_{k-1}, i_t\}$ 扩展为 k -项集;如果对项目集 $\{i_1, i_2, \dots, i_{k-1}, i_t\}$ 进行逻辑“与”运算 $\text{Sup}(B_1 \wedge B_2 \wedge \dots \wedge B_{k-2} \wedge B_{k-1} \wedge B_t)$,其结果不小于最小支持度阈值,则称该项目集为频繁 k -项集.按照这样思路进行扩展直到最后一个编号的项目或者到没有新的频繁集为止.算法的伪代码如下:

算法3 FIA- k 扫描一次数据库 DB,生成频繁 k -项集.

- (1) For ($k = 3; k \leq n; k++$) do
- (2) Until to $L_{k-1} = \phi$
- (3) Take one itemsets $\{i_1, i_2, \dots, i_{k-2}, i_{k-1}\} \in L_{k-1}$
- (4) For ($t = k; t \leq f; t++$)
- (5) If $\text{UTFM}[i_{k-1}, i_t] = 1$ then
- (6) If $\text{UTFM}[i_1, i_t] = \dots = \text{UTFM}[i_{k-1}, i_t] = 1$ and $\text{Sup}(B_1 \wedge \dots \wedge B_{k-1} \wedge B_t) \geq \text{Min_sup}$ then
- (7) $L_k \cup = \{i_1, i_2, \dots, i_{k-1}, i_t\}$;
- (8) End if
- (9) Next t
- (10) Take Next L_{k-1}

(11) Next k

根据该算法、依靠 UTM 矩阵对数据库进行扩展,生成高频繁项集的过程可用表 3、表 4 演示如下:

表 3 生成频繁 3-项集

频繁 2-项集	后移一位	频繁 2-项集	是否可组成 3-项集	是否为频繁 3-项集
{A, B}	{B, C}	{A, C}	{A, B, C}	{A, B, C}
	{B, D}	{A, D}	{A, B, D}	×
	{B, E}	×		
	{B, F}	{A, F}	{A, B, F}	{A, B, F}
{A, C}	{C, D}	{A, D}	{A, C, D}	{A, C, D}
	{C, E}	×		
	{C, F}	×		
{A, D}	{D, E}	×		
	{D, F}	×		
{A, F}	×			
{B, C}	{C, D}	{B, D}	{B, C, D}	{B, C, D}
	{C, E}	×		
	{C, F}	×		
{B, D}	{D, E}	×		
	{D, F}	×		
{B, E}	{E, F}	×		
{B, F}	×			
{C, D}	{D, E}	×		
	{D, F}	×		

注: {X, Y} × 表示 {X, Y} 是非频繁 ($k-1$)-项集; × 表示同行之后的栏内信息不需要再考虑。

表 4 生成频繁 4-项集

频繁 3-项集	后移一位	频繁 2-项集	是否组成 4-项集	是否为频繁 4-项集
{A, B, C}	{C, D}	{A, D}	{A, B, C, D}	×
		{B, D}		
	{C, E}	×		
	{C, F}	×		
{A, B, F}	×			
{A, C, D}	{D, E}	×		
	{D, F}	×		
{B, C, D}	{D, E}	×		
	{D, F}	×		

由表 3 可得出频繁 3-项集 $L_3 = \{\{A, B, C\}, \{A, B, F\}, \{A, C, D\}, \{B, C, D\}\}$ 。由表 4 可知 $L_4 = \phi$, 没有频繁 4-项集, 算法到此结束。

4.3 优化策略

(1) 忽略非频繁 1-项目集

根据定理 2 可知, 频繁项集满足反单调性质: 即非频繁项集的超集不是频繁项集。所以在结束 FIA-1 算法后, 对那些不包括在频繁 1-项集中的项目进行删减, 即在后面的操作中, 不再涉及到全部的项目, 仅仅对频繁 1-项集中的项目进行运算。

(2) 优化上三角频繁项集矩阵

根据定理 2 频繁项集单调性质和上三角频繁项集矩阵的定义可知, 如果某行或者某列的标记全是“0”, 那么这些组合的超集不可能是频繁项目集, 所以可以将这些行与列删除掉, 减少该矩阵的储存容量, 同时也减少了扫描矩阵的次数。例如表 2 的上三角频繁项集矩阵中第 4 行和第 5 行标记的全是“0”, 所以可以删除这两行, 故表 2 可以优化为表 5。

表 5 优化后的上三角频繁项集矩阵

UTFM _{3,5}	B	C	D	E	F
A	1	1	1	0	1
B		1	1	1	1
C			1	0	0

5 性能分析

5.1 FIA 与 Apriori 算法的性能对比

(1) 时间复杂度分析

Apriori 算法包括了连接与剪枝 2 个步骤, 在生成频繁 k -项集的过程中需要 k 次扫描整个数据库; FIA 算法只需扫描数据库一次, 故可大幅度减少 I/O 次数。在构建起上三角频繁项集矩阵后, 求频繁 k -项集时, 通过扩展某个频繁 ($k-1$)-项集最后一个项目之后是否存在于上三角频繁项集矩阵中, 如果存在则是 k -项集, 再通过逻辑“与”运算来判断是否为频繁 k -项集; 若不存在则就不能生成频繁 k -项集。通过该矩阵进行判断能否组成 k -项集, 有效的减低了逻辑“与”运算的次数。从而使 FIA 算法比 Apriori 算法具有更好的时间特性。

(2) 空间复杂度分析

Apriori 算法采用的是候选项集产生频繁项集的途径, 在此过程中需要大量的内存空间供其剪枝和生成频繁项集使用; 而 FIA 算法不需要存储候选项集, 仅需布尔值“1”或“0”来储存各个项集, 以位方式构建上三角频繁项集矩阵, 其代价是该矩阵占据了最大 $n(n-1)/2$ bytes 的空间, 相对于随频繁项集长度增加而呈指数级增长的 Apriori 算法来说, 其空间利用率得到大大提高, 同时此结构减少了对原始事务数据库的读操作, 使用逻辑“与”操作大大降

低了空间的占用量;该算法生成的频繁 k -项集之后立即释放频繁 $(k-1)$ -项集,并会立即对于新的待产生的 k -项集判断其是否为一个频繁项集,如果不是,就不会生成该 k -项集,这样也明显的减少了内存空间的增加.这样就有效地节省了存储空间,具有良好的空间特性.

5.2 数据集测试算法的性能

本模拟实验是在内存 512G, Windows XP 操作系统的环境下,采用 Visual Basic 语言来实现 Apriori 算法与 FIA 算法.测试数据是 UCI 机器学习数据集 Mushroom,该数据集是从频繁项集挖掘实现资料库下载^[8],共有 8125 条事务记录,记录长度是 23.测试的结果如图 1 所示.

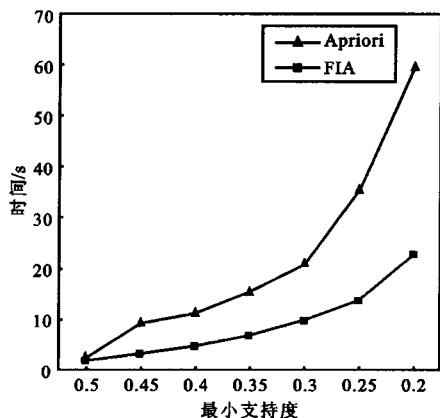


图1 Apriori 算法与 FIA 算法的时间对比

由图 1 可知,当事务记录一定时,FIA 算法的效率比经典 Apriori 算法的效率要高,当最小支持度较大时,两种算法的效率区别不是很大,当最小支持度阈值变小时,FIA 算法比 Apriori 算法的优势较大.

6 结束语

文中提出了一种基于上三角频繁项集矩阵的频繁项目集挖掘算法.该算法基于上三角频繁项集矩阵,只需扫描事务数据库一次,将扫描数据库的次数降低到最少;对事务数据库中事务按照项目进行向量生成,通过逻辑运算操作,可以方便计算出项目集的支持度,降低了项目集支持度的计算难度;不产生候选项目集,也不要逐层迭代的方法,降低了 Apriori 算法的瓶颈问题,提高了频繁项集的发现效率;

通过上三角频繁项集矩阵进行高一层频繁项集扩展,可以较明晰的得出各频繁项集.下一步就如何将该算法应用到多维及动态更新的关联规则挖掘中进行研究.

参考文献:

- [1] Agrawal R, Skikant R. Fast algorithms for mining association rules in large database[C]//Proceedings of 20th International Conference on Very Large Data Bases. Santiago: Morgan Kaufmann, 1994:478-499.
- [2] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large database[C]//Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'93). New York: ACM Press, 1993:207-216.
- [3] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]//Proceedings of the ACM SIGMOD internal Conference on Management of Data. Dalas Texas:ACM Press,2000:1-12.
- [4] Show - Jane Yen, Arbee L P Chen. A graph based approach for discovering various types of association rules[J]. IEEE Transactions on Knowledge and Data Engineering, 2001,13(5):839-845.
- [5] 张哲,杨兆中,庞炳章.大型数据库中关联规则的向量法挖掘[J].计算机工程,2003(8):97-99.
- [6] 陈耿,朱玉全,杨鹤标,等.关联规则挖掘中若干关键技术的研究[J].计算机研究与发展,2005,42(10):1785-1789.
- [7] 胡慧蓉,王周敬.一种基于关系矩阵的关联规则快速挖掘算法[J].计算机应用,2005(7):1577-1579.
- [8] Frequent Itemset Mining Implementations Repository [EB/OL].[2008-09-01]. <http://fimi.cs.helsinki.fi>.

作者简介:

王文正 男,(1983-),硕士研究生.研究方向为企业网络、数据挖掘.

王文平 女,(1966-),教授,博士生导师.研究方向为知识型企业变革管理、集群成长与网络分析、复杂系统分析.

许映秋 女,(1962-),教授,硕士生导师.研究方向为工业工程.

谈英姿 女,(1969-),副教授,硕士生导师.研究方向为人工智能.