基于矩阵算法的序列模式挖掘研究

卢博礼,张小平,王翰虎
(贵州大学 计算机科学与信息学院,贵州 贵阳 550025)

摘 要:序列模式挖掘中几种算法的缺点:都要进行多次扫描数据库,CPU要进行多次I/O操作。这成为序列挖掘中的一大瓶颈,使得算法在实际应用中的效率不高。文中提出一种矩阵算法,即在一次扫描数据库时,根据扫描数据建立由0和1组成的事务矩阵。接下来的大序列、序列模式等都是通过矩阵的列向量对应元素的相乘运算和简单的加法运算而得到。从而使算法得到进一步优化,提高了CPU的使用率,解决了序列挖掘中的瓶颈问题。本算法通过大量的数据实验,证明了算法确实有效地优化了算法的时间复杂度。

关键词:序列模式挖掘;序列模式;大序列;矩阵算法;连接运算
中图分类号:TP311.13 **文献标识码**:A **文章编号**:1673-629X(2011)01-0120-05

Research on Sequential Pattern Mining Based on Algorithm of Matrix

LU Bo-li, ZHANG Xiao-ping, WANG Han-hu
(College of Computer Science and Information, Guizhou University, Guiyang 550025, China)

Abstract: Based on several shortcomings of algorithm of the sequential pattern mining. Itmust scan the database many times, so a number of CPU to I / O operation, it become bottlenecks. The efficiency of algorithm is not high in practical applications. The paper presents an algorithm of matrix, that is, in a scan database, in accordance with scan data to establish the matrix of affairs which are composition of 0 and 1. Then, large sequence and sequential patterns are all out pass the vector of matrix multiplication operator corresponding to the elements and simple addition operations have been. So that the algorithm has been further optimized to improve the CPU rate of utilization. The algorithm use a large number of experimental data to prove that the algorithm is indeed effective to optimize the algorithm of complexity of time.

Key words: sequential pattern mining; sequential pattern; large sequence; algorithm of matrix; connected computing

0 引 言

序列模式挖掘又称序列挖掘[1],主要是找出序列数据库中那些在序列集合中出现频率超过最小支持度[2]或者用户指定的最小支持阈值[2]的子序列。其目的是想通过在含有交易时间属性的客户交易数据库中发现频繁项目序列,通过频繁项目序列找出一定的时间段内客户购买活动的规律[2,3]。序列模式的挖掘主要是基于大项集[4]的挖掘,主要思想是:

首先,把交易数据库转换为以客户序列为记录的数据库;其次,利用大项集搜索算法计算大项集,作为一阶大序列;再次,以一阶大序列为基础,依次搜索所有阶的大序列;最后,从大序列中删除子序列,最终得到序列模式。

从以上过程可以看出,序列模式挖掘一般分为五个步骤[2,5],它们分别是排序阶段[5]、大项集阶段[5]、转换阶段[5]、序列阶段[5]、选最大阶段[5]。序列模式挖掘主要来源于关联规则挖掘[3,6,7],是关联规则应用的延伸。

在关联规则挖掘Apriori算法[2,8]的基础上,由Agrawal和Strikant[2]等人提出几种序列模式的挖掘算法,即AprioriAll[5]、AprioriSome[5]及DynamicSome[5]。但这几种算法都要进行多次扫描数据库,扫描数据库便成为序列模式挖掘中的一大瓶颈,多次扫描数据库更是如此。

文中提出的矩阵算法主要是解决多次扫描数据库的瓶颈问题。在介绍矩阵算法之前先简要介绍AprioriAll算法和AprioriSome算法以便进行对比。

收稿日期:2010-05-22;**修回日期**:2010-08-04
基金项目:贵州省2008年省级信息化专项资金项目(0830);贵州省科技计划工业攻关课题(黔科合GY字[2008]3035)
作者简介:卢博礼(1978-),男,贵州黔西人,硕士研究生,CCF会员,研究方向为数据库技术与软件工程;张小平,研究员,硕士生导师,研究方向为数据库技术与软件工程;王翰虎,教授,硕士生导师,研究方向为数据库系统、分布式系统、面向对象方法。

1　基本概念及示例数据库

1.1　基本概念

基本概念如下:

序列:序列是项集的一个非空有序集合,也就是一个集合的集合,记为$\{s_1,s_2,\cdots s_n\}$,其中s_i是一个项的集合,表示在同一条交易中出现的商品。

客户序列:把一个客户的所有交易按照时间的升序排列成一个序列$<t_1,t_2,\cdots,t_n>$,称为客户序列或顾客序列(customer sequence[9])。

支持阈值[4,10]:主要从统计学的角度来理解,表示项集或客户序列在统计意义上的最低重要性。如果交易数据库的客户数量是固定的,可以用最小支持数[2,6]minsup[9,11]来代替支持阈值。如果项集 X 的支持数不小于最小支持数,即 X. sup≥minsup(文中设最小支持数 minsup=2)。则称 X 是大项集(large itemset)或频繁项集[4](frequent itemset[9,11])。如果序列 A 的支持数不小于最小支持数,即 A. sup≥minsup,则 A 是大序列[7](large sequence)或频繁序列(frequent sequence)。

序列模式[7,12]:具有最小支持数的最大序列称为序列模式。它必须满足两个条件:首先,支持数大于等最小支持数;其次,不是其它大序列的子序列[2]。

1.2　示例数据库

表 1 经过预处理后得到交易数据库 Ds;对表 2 进行大项集的映射并计算出支持数后得到表 3;对表 3 进行转换得出换后的数据库 Dt。

表 1　示例数据库

客户标识	客户序列
1	<(40),(100)>
2	<(20,30),(40),(50,70,80)>
3	<(40,60,80)>
4	<(40),(50,80),(100)>
5	<(100)>

表 2　预处理后的交易数据库 Ds

客户标识	交易标识	项集
1	1	40
1	2	100
2	1	20,30
2	2	40
2	3	50,70,80
3	1	40,60,80
4	1	40
4	2	50,80
4	3	100
5	1	100

表 3　大项集的映射(最小支持度 minsup=2)

大项集	支持数	映射后的整数值
(40)	4	1
(50)	2	2
(80)	3	3
(50,80)	2	4
(100)	3	5

2　AprioriAll 算法分析

AprioriAll 算法的思想主要来源于关联规则挖掘中的 Apriori 算法,它是 Apriori 在序列模式挖掘中的扩展应用。它和 Apriori 一样,也是一个要进行多次扫描数据库的算法。在每一次扫描中都利用前一遍的大序列来产生候选序列[2,4],在扫描完整个数据库后再来进行支持度的检测。在第一遍的扫描中,大项集阶段的输出被用来初始化 1 阶大序列的集合。在每次遍历中,从一个由大序列组成的种子集[2,7]开始。利用这个种子集,可以产生新的潜在的大序列。在第一次遍历前,所有在大项集阶段得到的 1 阶大序列组成了种子集。AprioriAll 算法描述如下:

表 4　转换后的数据库 Dt

客户标识	客户客户序列	大项集表示的客户序列	转换后的序列*
1	<(40),(100)>	<{(40)},{(100)}>	<{1},{5}>
2	<(20,30),(40),(50,70,80)>	<{(40)},{(50),(80),(50,80)}>	<{1},{2,3,4}>
3	<(40,60,80)>	<{(40,80)}>	<{1,3}>
4	<(40),(50,80),(100)>	<{(40)},{(50),(80),(50,80)},{(100)}>	<{1},{2,3,4},{5}>
5	<(100)>	<{(100)}>	<{5}>

输入:序列数据库 Dt

输出:所有最长序列[5]

L1 = {frenquent items};

For (k=2; Lk-1≠Ø;k++) do

Ck=Get_candidate(Lk-1);

For all customer sequence c∈Dt do

Ct=subset(Ck,c);

For all s∈Ct do s. sup=s. sup+1; end for

End for;

Lk={s∈Ck|s. sup≥minsup};

End for

Lk=UkLk;

本算法中候选序列的调用 Get_candidate()[5]函数来实现,这样大大减少候选序列的个数。AprioriAll 利用了 Apriori 算法的思想,但在候选序列的产生和频繁序列生成方面需要考虑序列元素的特点后再作相应的

处理。AprioriAll 的算法性能分析如下：

1)缺少时间限制[2]：需要用户的干预来指定序列模式中相邻元素之间的时间间隔[5,7]。

2)严格事务的定义[2]：客户的一次购买行为中所购买的所有物品必须包含在一个事务中。

3)没有分类层次[2]：只能在项目的原始级别上进行挖掘[2,5]。

4)多次扫描数据库：在每一次进行支持度的检测时都要扫描数据库。

基于 AprioriAll 算法的这些缺点，于是产生了 AprioriSome 算法。

3 AprioriSome 算法分析

AprioriSome 算法描述如下：

输入：序列数据库 DT

输出：所有最长序列[5]

```
//前向搜索阶段
Lk={frequent items};
C1=L1;
Last=1;
For (k=2;Ck-1≠Ø and Llast≠Ø;k=k+1) do
If (Lk-1 存在) then
Ck=Get_candidate(Ck-1);
End if
If k=nest(last) then
For all customer sequence c∈Dt do
Dt=subset(Ck,c);
For all s∈Ct do s.sup=s.sup+1;
End for
End for
End if
Lk={s∈Ct| s.sup≥minsup};
Last=k;
End for
//后向搜索阶段
For(k-1;k≥1;k=k-1) do
IfLk 不存在 then
Delete all sequence in Ck contained in Li(i>k);
For all customer sequence c∈Dt do
Ct=subset(Ck,c);
For all s∈Ct do s.sup=s.sup+1; end for
Lk={s∈Ct| s.sup≥minsup};
End for
Else
Delete all sequence in Lk contained in Lk(i>k);
end if
End for
Lk=UkLk;
```

从上面两个算法可以看出，AprioriSome 算法是 AprioriAll 算法的改进，它分为前推和回溯两个阶段。确实时间上比 AprioriAll 要节省很多，但 AprioriSome 还是要进行多次扫描数据库。基于以上两个算法都要多次扫描数据库，多次扫描数据库便成为序列挖掘的一大瓶颈。为了解决这一瓶颈问题，文中提出了一种只对数据库进行一次扫描的矩阵算法。

4 矩阵算法

4.1 矩阵生成

令 $I=\{i_1,i_2,\cdots i_n\}$ 为映射后的整数值，$T=\{t_1,t_2\cdots t_m\}$ 为转换后的序列。初始矩阵 C_1 按如下规则生成。矩阵 C_1 的元素 $\{e_{ij}\}$ 定义如下：

$$e_{ij}=\begin{cases}1,若\ i_j\in t_i\\0,若\ i_j\notin t_i\end{cases}$$

其中 $i=1,2,\cdots m;j=1,2\cdots n$。

$$e_{m+1j}=\sum_{i=1}^{m}e_{ij},其中\ j=1,2,\cdots n。$$

下面通过表 4 中的数据来说明矩阵的生成过程。

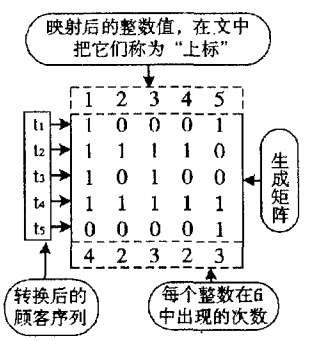

图 1 矩阵的生成

在图 1 中，最上面一行元素是映射后的整数值，把它们称为“上标”，它们不是矩阵的元素。矩阵中最下面一行元素是生成矩阵中每个列向量中前 m 个元素中“1”的个数，最左边的一列（t_1、t_2、t_3、t_4、t_5）为转换后的客户序列，中间由“0”和“1”组成的部分为生成矩阵，其生成规则为：当映射后的整数值（1、2、3、4、5）在转换后的客户序列（t_1、t_2、t_3、t_4、t_5）中出现时对应元素值为“1”，否则为“0”。应用这个矩阵可以产生各阶大序列候选矩阵 C_i 和各阶大序列矩阵 L_i，并最终生成序列模式。

4.2 向量的连接运算

规定两个向量的连接运算如下：进行连接运算的两个列向量中，把它们的前 m 个对应元素进行相乘的积作为所产生的新向量相对应的前 m 个元素，而新向量的第 $m+1$ 个元素为新向量中前 m 个元素中值为“1”的元素的个数，新向量的上标为进行连接运算的所有向量对应的上标。例如上面的矩阵的第一个列向

量和第二个列向量,第三个列向量和第五个列向量进行连接运算的过程(见图 2)。

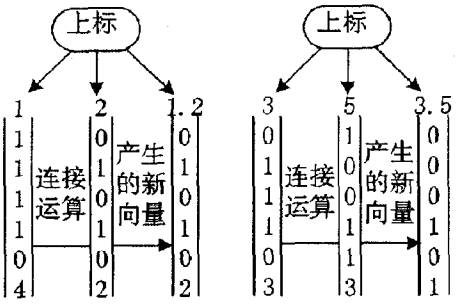

图 2　向量的连接运算

4.3　矩阵算法描述

(1)在初始矩阵 C_1 中找出每个列向量的第 $m+1$ 个元素值大于或等于最小支持数 minsup 的这些列向量,把这些满足条件的列向量重新组成 1 阶大序列矩阵 L_1。

(2) 由 L_i 进行自连接运算产生 $i+1$ 阶候选矩阵 C_{i+1},在候选矩阵 C_{i+1} 的每个列向量中找出第 $m+1$ 个元素值大于或等于最小支持数 minsup 的这些列向量,把这些满足条件的列向量重新组成 $i+1$ 阶大序列矩阵 L_{i+1}。

(3) 重复步骤(2) 直到 L_i 为空。

用以上矩阵为例,说明一下矩阵算法的过程。假设最小支持数 minsup = 2,从初始矩阵 C_1 中选出满足最小支持数 minsup 的列向量,即最后一行元素大于 2 的列向量组成 L_1。

$$C_1=\begin{vmatrix}1&2&3&4&5\\1&0&0&0&1\\1&1&1&1&0\\1&0&1&0&0\\1&1&1&1&1\\0&0&0&0&1\\4&2&3&2&3\end{vmatrix}\quad L_1=C_1=\begin{vmatrix}1&2&3&4&5\\1&0&0&0&1\\1&1&1&1&0\\1&0&1&0&0\\1&1&1&1&1\\0&0&0&0&1\\4&2&3&2&3\end{vmatrix}$$

由于 C_1 中的每个列向量的最后一个元素的值都大于最小支持数 minsup,所以 $L_1=C_1$。由 L_1 进行自连接运算得出 C_2。

$C_2=$

1.2	1.3	1.4	1.5	2.3	2.4	2.5	3.4	3.5	4.5
0	0	0	1	0	0	0	0	0	0
1	1	1	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
2	3	2	2	2	2	1	2	1	1

再由 C_2 中选出满足最小支持数 minsup 的列向量组成 L_2。

$$L_2=\begin{vmatrix}1.2&1.3&1.4&1.5&2.3&2.4&3.4\\0&0&0&1&0&0&0\\1&1&1&0&1&1&1\\0&1&0&0&0&0&0\\1&1&1&1&1&1&1\\0&0&0&0&0&0&0\\2&3&2&2&2&2&2\end{vmatrix}$$

由 L_2 进行自连接运算得出 C_3。

$C_3=$

1.2.3	1.2.4	1.2.5	1.3.4	1.3.5	1.4.5	2.3.4
0	0	0	0	0	0	0
1	1	0	1	0	0	1
0	0	0	0	0	0	0
1	1	1	1	1	1	1
0	0	0	0	0	0	0
2	2	1	2	1	1	2

再由 C_3 中选出满足最小支持数 minsup 的列向量组成 L_3。

$$L_3=\begin{vmatrix}1.2.3&1.2.4&1.3.4&2.3.4\\0&0&0&0\\1&1&1&1\\0&0&0&0\\1&1&1&1\\0&0&0&0\\2&2&2&2\end{vmatrix}$$

由 L_3 进行自连接运算得出 C_4。再由 C_4 中选出满足最小支持数 minsup 的向量组成 L_4。由于 C_4 中只有一个列向量,不能产生 C_5,从而也不能生成 L_5,所以算法终止。

$$L_4=\begin{vmatrix}1.2.3.4\\0\\1\\0\\1\\0\\2\end{vmatrix}\quad L_4=C_4=\begin{vmatrix}1.2.3.4\\0\\1\\0\\1\\0\\2\end{vmatrix}$$

从以上算法过程可以看出 L_1、L_2、L_3、L_4 四个矩阵的上标就对应着大序列。

4.4　删除子序列

(1) 在矩阵 L_1,L_2,L_3 中去掉矩阵 L_4 中上标的所有子集所对应的上标。L_4 的上标为“1.2.3.4”,它的所有子序列为“1”、“2”、“3”、“4”、“1.2”、“1.3”、“1.4”“2.3”、“2.4”、“1.2.3”、“1.2.4”、“2.3.4”。这样,L_3 中的上标全部被去掉。L_2 中只剩下上标“1.5”;L_1 中只剩下上标“5”。

(2) 在矩阵 L_1 中去掉矩阵 L_2 中剩下的上标“1.5”

的所有子集所对应的上标;这样,L_1 中的所有上标全部被去掉。

(3) 在 L_1、L_2、L_3、L_4 四个矩阵的所有上标中只剩下上标“1.2.3.4”和“1.5”。

4.5 大序列的产生

L_1、L_2、L_3、L_4 中剩下的所有上标的集合就是所要求的序列模式。进行上述操作后,产生的序列模式为:{< 1 2 3 4 >, < 1 5 >},执行过程如下所示:

在 L_3 中删除 L_4 的所有子集;

L_4 =

1.2.3.4	1.2.3	1.2.4	1.3.4	2.3.4
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
2	2	2	2	2

在 L_2 中删除 L_4 的所有子集,在 L_1 中删除 L_4 和 L_2 的所有子集。

1.2	1.3	1.4	1.5	2.3	2.4	3.4	1 2 3 4 5
0	0	0	1	0	0	0	1 0 0 0 1
1	1	1	0	1	1	1	1 1 1 1 0
0	1	0	0	0	0	0	1 0 1 0 0
1	1	1	1	1	1	1	1 1 1 1 1
0	0	0	0	0	0	0	0 0 0 0 1
2	3	2	2	2	2	2	4 2 3 2 3

5 实验结果分析

根据以上算法步骤的分析与研究,矩阵算法与传统序列模式挖掘算法相比,主要是矩阵的计算。而对各 K 阶项集的支持度的计算显得尤其简便,无需对数据库进行多趟扫描。本算法在联想电脑(T6400—2.0G,2G 内存)上使用 Java 进行实验,采用的数据库是来自某百货销售公司的销售数据,约 5 万条。同时在同一台计算机上对同样的数据库采用 AprioriSome 算法进行实验,结果发现两种算法的结果是相同的,这验证了矩阵算法的正确性。但在时间上,矩阵算法的运行时间明显比 AprionSome 算法少。主要是矩阵算法充分利用向量计算的优势,它只扫描事务数据库库一次且直接通过布尔运算计算项集的支持度,比经典的 AprioriSome 算法占用内存空间小,I/O 操作少,执行速度快。

6 结束语

提出了一个新的算法——矩阵算法。在实际的应用中,大型交易事务数据库的使用非常普遍,并且交易数据也非常巨大,扫描数据库成为序列模式挖掘中的一大瓶颈,多次扫描数据库更是如此。使用矩阵进行序列模式挖掘相对于 AprioriAll 和 AprioriSome 算法而言,只对事务数据库进行一次扫描,根据扫描结果建立由 0 和 1 组成的初始矩阵。并且计算机在存储初始矩阵时使用的是 0 和 1 这种布尔数据(以位方式存储即可),因此在样本记录较大时通常会占用更少的空间。各阶大序列都只在矩阵上进行运算,支持度的检测只要使用矩阵的最后一个行向量的每个元素值与最小支持度 minsup 进行比较即可。不用再进行多次数据库扫描,从而减少 CPU 的 I/O 操作次数,提高了 CPU 的利用率,使算法在时间复杂度上有了较大的优势。实验表明,文中提出的算法是有效、可行的。下一步研究的主要目标是在实际应用中对算法进行进一步测试,使其便于推广应用。

参考文献:

[1] 夏 岩,倪世宏,王彦鸿. 动态划分序列模式挖掘算法[J]. 计算机仿真,2009(2):127-130.

[2] 陈 安,陈 宁,周龙骧,等. 数据挖掘技术及应用[M]. 北京:科学出版社,2006.

[3] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//In Proceedings of the 24th International Conference on Very Large Databases(VLDB'98). San Francisco, CA: Morgan Kaufmann, 1998:478 - 499.

[4] 夏明波,王晓川,孙永强,等. 序列模式挖掘算法研究[J]. 计算机技术与发展,2006,16(4):4-10.

[5] 毛国军,段立娟,王 实,等. 数据挖掘原理与算法[M]. 北京:清华大学出版社,2007.

[6] 史 原,鲁汉榕,罗 菁,等. 基于规模约简和多支持度的关联规则挖掘[J]. 计算机工程与设计,2006(21):4105-4107.

[7] 袁玉波,杨传胜,黄廷祝,等. 数据挖掘与最优化技术及其应用[M]. 北京:科学出版社, 2007

[8] 高 杰,李绍军,钱 锋. 挖掘关联规则 AprioriTid 算法的改进[J]. 计算机工程与应用, 2007(7):188-190.

[9] Agrawal R, Imielinksi T, Swami A. Minig association rules between sets of items in large database[C]//The 1993 ACM SIGMOD Conference. Washington DC, USA:[s. n.],1993.

[10] 刘洪辉,吴岳芬. 用户行为模式挖掘问题的研究[J]. 计算机技术与发展,2006,16(5):85-92.

[11] Aleman-Meza B, Halaschek C, Arpinar I B. Context-aware semantic association ranking[C]//Cruz I F, Kashyap V, Decker S, et al. Proc. of the 1st int'l Workshop on Semantic Web and Databases. Co-located with VLDB2003. Berlin, Gemlany: Humboldt-University, 2003:33-50.

[12] 张 洋,陈未如,陈珊珊. 并发序列模式挖掘方法研究[J]. 计算机应用,2009(11):3096-3099.