基于 2- 序列矩阵的序列模式增量式更新研究 *

吴永俊,郑 诚,赵建伟

(安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039)

摘 要:在序列模式挖掘相关研究中,增量式挖掘是序列模式挖掘中的难点和热点。在分析了2-序列矩阵的相关特性和理论基础上,提出了一种基于2-序列矩阵的序列模式增量挖掘算法SPI_2SM,该算法充分应用了先前挖掘的结果,减少了对数据库的扫描和查找次数,减少了空间开销,提高了挖掘效率。

关键词:数据挖掘;2-序列矩阵;序列模式;增量更新

中图分类号:TP311 **文献标识码**:A

Research on incremental updating for mining sequential patterns based on 2-sequence matrix

WU Yong Jun, ZHENG Cheng, ZHAO Jian Wei

(Educational Department Key Laboratory of Intelligent Computing & Signal Processing, Anhui University, Hefei 230039, China)

Abstract: In the research of sequence pattern mining, the incremental mining of sequential pattern mining is difficult and hot. After analyzing related characteristics and theoretical of 2-sequence matrix, this paper proposed SPI_2SM (Sequential Pattern Incremental Mining based on a 2-sequence matrix) algorithm. SPI_2SM uses the results of the previous mining to reduce the database scanning and seeking as well as reduce space costs, so the efficiency of mining is enhanced.

Key words: data mining; 2-sequence matrix; sequence pattern; incremental update

数据挖掘是从海量的数据中找出人们感兴趣的数据模式,称为频繁项集。近年来,数据挖掘的重要分支——序列模式挖掘[1]成为数据挖掘中的研究热点,随着多种针对序列模式的挖掘算法不断提出,时空效率也不断提高。序列模式挖掘与传统的数据挖掘相比具有很多自己的特点,最大的区别是序列数据具有时间属性。序列模式挖掘效率的挑战主要是在多次扫描序列数据库以及产生候选序列模式的时空开销上,而参考文献[2]中提出的基于2-序列矩阵的挖掘方法可以显著地提高挖掘效率,该方法主要是减少对序列数据库的扫描次数,同时通过编码也提高了序列模式的存储效率。

事实上,序列模式挖掘无论是从早期的 AprioriAll 和 AprioriSome 算法还是到后来的 FreeSpan 和 PrefixSpan 算法[3]等,都是基于静态数据库的频繁序列的挖掘,而实际上序列数据库是时刻变化的,使得之前已经发现的序列模式不再频繁,而从更新的序列数据中也可能发现新的序列模式,这就要不断更新原来的序列数据库,这就是序列模式挖掘中的增量式更新挖掘[4]。但是从新的数据库中挖掘序列模式最大的挑战是能不能充分利用之前的挖掘信息来减少对原数据库的扫描次数,进而提高整个更新算法的时空效率。所以能否高效地挖掘出动态序列数据库中的序列模式是衡量增量式更新挖掘算法优秀与否的主要指标。

目前在增量式挖掘中比较高效实用的2个算法是参考文献[5-6]中提到的算法 ISE 和 FASTUP,但是这2个算法也有很多不足,比如要多次扫描数据库以及处理大规模的候选新序列的开销巨大等等,这些都是严重影响挖掘算法效率的因素。本文提出的 SPI_2SM(Sequence Pattern Incremental Ming Based on 2-Sequence Matrix)算法主要是基于以下2点考虑:(1) 通过基于2-序列矩阵的序列模式挖掘方法对序列数据库中各序列编码可

* 基金项目:安徽省高等学校省级自然科学研究重点项目(KJ2009A57)

以节省大量查找时间和序列的存储空间。(2)通过构建序列支持数表(SupTable),在应用2-序列矩阵编码基础上对增加的新序列数据挖掘结果并入原序列模式集合,从而对原序列模式集进行更新。该算法有以下优点:

(1)在挖掘原序列数据库和新序列数据库时只需扫描1遍数据库。

(2)通过对序列的编码可以迅速方便合并序列模式集合。

(3)不需要提前提供最小支持度,即可以提供任意支持度挖掘。

表1所示为已有的序列数据库DB,表2所示为要增加的序列数据库DB^+。

表1 序列数据库DB

CID	Sequence
1	12 131
2	2 311
3	121
4	23 213
5	321

表2 新增序列数据库DB^+

CID	Sequence
1	41 213
2	2 134
3	3 211

1 相关概念和理论

1.1 基本概念

对于给定的交易数据集$DB(CID,TID,Itemset)$,其中CID是客户编号,TID是交易时间,$Itemset$是交易项集,设$I=\{i_1,i_2,\cdots,i_n\}$是由n个不同的项目组成的集合,$Itemset$是I的非空子集。该交易数据集DB可以转换成二元组$DB(CID,Sequence)$表示交易序列。对于序列s来说,支持s的顾客数称为序列s的支持数,支持数所占全部顾客数的比值称为序列s的支持度,设min sup为最小支持度,所有支持度大于或等于min sup的序列称之为频繁序列,也称为序列模式。

1.2 2-序列及2-序列矩阵理论

为了便于应用2-序列矩阵给序列压缩编码和挖掘,把序列s分为两部分$s=<X:Y>$,其中X的长度为2,称为2-序列,即令$s=<s_1s_2\cdots s_k>=<s_1s_2:s_3\cdots s_k>$。这样每个序列可以化为以2-序列为首的序列并产生多个以2-序列为首的子序列。如把表1中每个序列的所有2-序列都列出如表3所示。

为了更好地应用2-序列,把$<X:Y>$中Y部分也分解成多个2-序列,具体的做法如下:

$$Y=<y_1y_2\cdots y_l>=\begin{cases}<y_1y_2:y_3y_4:\cdots:y_{l-1}y_l>, & l\text{为偶数}\\ <y_1y_2:y_3y_4:\cdots y_{l-1}:y_l>, & l\text{为奇数}\end{cases}\quad(1)$$

为了提高对2-序列的存储效率,使用2-序列矩阵对2-序列编码,2-序列矩阵的表示如图1所示。

表3 DB中的所有2-序列

CID	Sequence	All 2-Sequences
1	12 131	<12:131><11:31><13:1><11><21:31><23:1><21><31>
2	2 311	<23:11><21:1><21><31:1><31><11>
3	121	<12:1><11><21>
4	23 213	<23:213><22:13><21:3><23><32:13><31:3><33><13>
5	321	<32:1><31><21>

	1	2	3	4	5
1	11	12	13	14	15
2	21	22	23	24	25
3	31	32	33	34	35
4	41	42	43	44	45
5	51	52	53	54	55

图1 2-序列矩阵

对于该矩阵中的元素按照(11,12,21,22,13,23,31,32,33,14,24,34,41,42,43,44,15,25,35,45,51,52,53,54,55)顺序排列而不是(11,12,13,14,15,21,22,23,24,25,31,32,33,34,35,41,42,43,44,45,51,52,53,54,55)的顺序。2-序列矩阵中的元素可以和2-序列相对应,如12对应于<12>,而它的排列位置索引是2,对于2-序列$<i_1,i_2>$的索引$index$按照以下计算方法计算:

$$index=\begin{cases}i_1\times i_2, & i_1=i_2\\ i_1\times(i_1-1)+i_2, & i_1>i_2\\ i_1+(i_2-1)^2, & i_1<i_2\end{cases}\quad(2)$$

例如,<32>的索引就是$3\times(3-1)+2=8$,<35>的索引就是$3+(5-1)^2=19$。这样做的目的主要是考虑当2-序列中增加了1个项$item$,对于之前的2-序列索引顺序不用改变,如增加了序列<26>,则根据(2)式算得其索引为27,而其他的2-序列顺序未改变,这就为增量更新挖掘带来了便利。

2 SPI_2SM算法

2.1 存储编码

2-序列矩阵最大的优点即是通过对序列数据的压缩编码来提高对序列数据的存储效率,如对于序列<23>需要存储空间为2,而通过2-序列矩阵编码<23>的索引为6,这样可以把<23>转换为<6>,其需要存储空间即为1,从而减少了存储空间。

对于长度为N的序列,使用传统的存储技术需要N个单位空间,而使用2-序列矩阵后只需要约$N/2$个单位空间即可,如通过2-序列计算索引序列<2,5,3,4,1,1>可转换为<18,12,1>,同时为了使得索引值与1-序列有所区别,对于1-序列不用编码,直接在1-序列前加入"-"以示区别,如<6>可表示成<-6>。

2.2 SPI_2SM算法描述

设有2个序列数据库DB和DB^+,前者是原序列数据库,后者是要增加更新的序列数据库,SPI_2SM算法的主要目的是基于2-序列矩阵,使从DB^+中挖掘的数

据能够在更少的时间和空间内更新到 DB 的挖掘结果中。算法具体描述如下：

输入：原序列数据库 DB、增加序列数据库 DB^+。

输出：更新后的序列模式挖掘结果 Result。

(1)从序列数据库中读取一条交易序列 sequence。

(2)对每一条序列 sequence 列举出所有的 2-序列 X 以及剩余序列 Y，使用(1)式对 Y 进行分解。

(3)用 2-序列矩阵计算出 X 和 Y 的索引并对其进行编码。

(4)对 DB 中所有序列重复以上 3 步，并计算所有 X 和 Y 的支持数，构建序列支持数表 SupTable1。

(5)对序列数据库 DB^+ 重复以上 4 步，构建序列支持数表 SupTable2。

(6)从 SupTable2 中读出一条编码后的序列及其支持数与 SupTable1 进行比较，并入 SupTable1 中，从 SupTable2 中删除该序列。

具体做法如下：

①对于 $s=<X:Y>\in$ SupTable2，若在 SupTable1 中找到相同的 X，则把 X 的支持数加到 SupTable1 中的相对应 X 的支持数上，再比较两序列的剩余部分 Y，找到相同的序列支持数相加，否则直接在 SupTable1 中添加新的 Y。

②若在 SupTable1 中找不到相同的 X，通过计算 s 的索引将其插入 SupTable1 中的相应位置。

(7)重复(6)直至 SupTable2 为空，得到更新后的序列支持数表 SupTable。

(8)用户根据需要决定最小支持度 minSup，从 SupTable 中找出所有满足 minSup 的序列模式，即更新后的挖掘结果 Result；

(9)算法结束。

2.3 算法举例

为了便于说明 SPI_2SM 算法，对表 1 和表 2 进行增量式更新挖掘。

在 2.2 节已经对表 1 中序列的所有 2-序列列举结果形成表 3，现在对其 y 进行分解并通过 2-序列矩阵编码，如表 4 所示。

表 4 DB 中编码后的 2-序列

CID	Sequence	All encoded 2-Sequences
1	12131	<2:5:-1><1:7><5:-1><1> <3:7><6:-1><3><7>
2	2311	<6:1><3:-1><3><7:-1><7><1>
3	121	<2:-1><1><3>
4	23213	<6:3:-3><4:5><3:-3><6> <8:5><7:-3><9><5>
5	321	<8:-1><7><3>

根据表 4 中 2-序列的索引构建 DB 序列支持数表 SupTable1，如表 5 所示。

表 5 序列支持数表 SupTable1

X: support	Y: support
<1>:3	<7>:1
<2>:2	<-1>:1 <5, -1>:1
<3>:5	<-1>:1 <-3>:1 <7>:1
<4>:1	<5>:1
<5>:2	<-1>:1
<6>:3	<-1>:1 <1>:1
<7>:4	<-1>:1 <-3>:1
<8>:1	<-1>:1
<9>:1	

同理，构建 DB^+ 的序列支持数表 SupTable2，如表 6 所示。

表 6 序列支持数表 SupTable2

X: support	Y: support	X: support	Y: support
<1>:2	<-3>:1	<8>:1	<1>:1
<2>:1	<5>:1	<10>:1	
<3>:3	<-3>:1 <12>:1	<11>:1	
<5>:2	<-4>:1	<13>:1	<3>:1 <3, -3>:1
<6>:2	<-4>:1	<14>:1	<5>:1
<7>:1	<-1>:1		

按照 2-序列矩阵编码，表 6 中所有序列都是编码后的序列，如<12>表示序列<3，4>。

根据 SPI_2SM 算法(6)把表 6 并入表 5 得到增加更新后的序列支持数表 SupTable，如表 7 所示。

表 7 更新后的序列支持数表 SupTable

X: support	Y: support	X: support	Y: support
<1>:5	<7>:1 <-3>:1	<8>:2	<1>:1 <-1>:1
<2>:3	<5>:1 <-1>:1 <5, -1>:1	<9>:1	
<3>:8	<-3>:2 <12>:1 <-1>:1 <7>:1	<10>:1	
<4>:1	<5>:1	<11>:1	
<5>:4	<-1>:1 <-4>:1	<13>:1	<3>:1 <3, -3>:1
<6>:5	<-1>:1 <1>:1 <-4>:1	<14>:1	<5>:1
<7>:5	<-1>:2 <-3>:1		

可以根据需要决定挖掘的最小支持数 min sup，如 min sup=2，通过表 7 迅速地挖掘出序列<1>，<2>，<3>，<5>，<6>，<7>，<8>，<3，-3>，<7，-1>为频繁序列，利用 2-序列矩阵逆运算得到在 minsup=2 挖掘出的序列模式为<11>，<12>，<21>，<13>，<23>，<31>，<32>，<213>，<311>。也可以通过设置不同的 minsup 挖掘不同的序列模式。

3 实验结果

3.1 实验环境及数据

实验使用双核 3.0 GHz Pentium4 PC 机器，内存为 1 GB，采用 Windows XP 操作系统，基于 java+eclipse 编程环境。

实验数据采用由 IBM 数据生成器产生，该生成器是目前序列模式挖掘研究中广泛认可的数据生成器，通过设置不同的参数来产生不同的实验数据。可以具体指定这些参数：序列(客户)数目、每个序列的平均项目集(交易)数目、每个项目集中的平均项目个数以及数据集中的不同项目数等。

3.2 实验结果

目前在增量式挖掘中比较典型的算法是参考文献[7-8]中提到的 ISE 和 FASTUP 算法，本实验数据集参数的具体含义见参考文献[1]，*DB* 采用 D10C10T2.5N5 数据集，DB^+采用 D1C5T3N0.5 数据集，实验结果如图 2、图 3 所示：

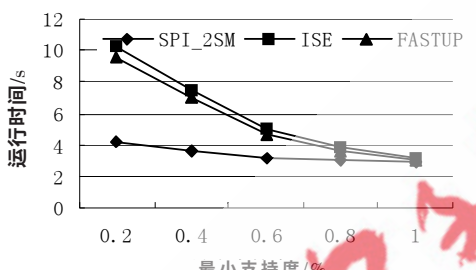

图 2 算法执行时间

从图中可以看出，当最小支持度较小的情况下，本文提出的 SPI_2SM 算法无论从时间效率还是空间效率都要比另外 2 个算法优秀，尤其是空间效率大大提高，说明了本算法的可靠性和高效性。

在分析了 2-序列及 2-序列矩阵相关性质和理论的基础上，提出了序列模式增量更新挖掘算法 SPI_2SM，该算法充分利用了 2-序列矩阵的编码功能。

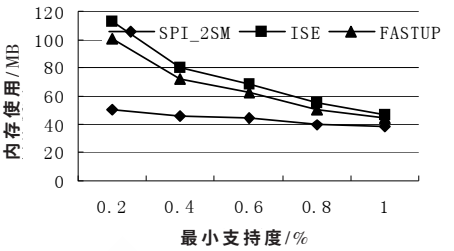

图 3 算法执行内存需求

即提高了序列的存储效率，也提高了挖掘序列模式的时间效率，同时该算法不需要事前提供最小支持度，可以挖掘任意最小支持度的序列模式。实验证明，该算法易于扩展，是有效和高效的。

参考文献

[1] AGRAWAL R, SRIKANT R. Mining sequential pattern [C]//Proceedings of the 11th International Conference on Data Engineering. Taipei: IEEE Computer Society Press, 1995: 3-14.

[2] HSIEH C Y, YANG D L, WU J P. An efficient sequential pattern mining algorithm based on the 2-sequence matrix[C]. Proceedings of 2008 IEEE International Conference on Data Mining Workshops. Taipei: [s.n.]: 583-591.

[3] 陈卓，杨炳儒.序列模式挖掘综述[J].计算机应用研究 2008，25(7)：1960-1976.

[4] 彭慧丽，张啸剑，张亚.一种快速增量式频繁访问序列挖掘算法[J].计算机工程与应用，2009，45(3)：138-140.

[5] MASSEGLIA F, PONCELET P, TEISSEURE M. Incremental mining of sequential patterns in large database[J]. Data and Knowledge Engineering, 2003, 46(1): 97-121.

[6] LIN M Y, LEE S Y. Incremental update on sequential patterns in large databases[C]//Proceedings of 10th IEEE International Conference on Tools with Artificial Intelligence. Taipei: [s.n.], 2001: 24-31.

(收稿日期：2009-07-12)