一种挖掘带时间约束序列模式的改进算法

胡学钢,张圆圆
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:针对带时间约束的序列模式,提出了一种改进的挖掘算法 TSPM,克服了传统的序列模式挖掘方法时空开销大,结果数量巨大且缺少针对性的缺陷. 算法引入图结构表示频繁 2 序列,仅需扫描一次数据库,即可将与挖掘任务相关的信息映射到图中,图结构的表示使得挖掘过程可以充分利用项目之间的次序关系,提高了频繁序列的生成效率. 另外算法利用序列的位置信息计算支持度,降低了处理时间约束的复杂性,避免了反复测试序列包含的过程. 实验证明,该算法较传统的序列模式发现算法在时间和空间性能上具有优越性.

关键词:数据挖掘;序列模式;时间约束
中图分类号:TP182 **文献标识码**:A **文章编号**:1673-4785(2007)02-0089-05

An improved algorithm for mining sequential patterns with time constraints

HU Xue-gang, ZHANG Yuan-yuan
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: An improved time constrained sequential pattern mining algorithm (TSPM) is proposed, overcoming the problem of traditional sequential mining algorithm whose performance is poor, and result is numerous and short of pertinence. Graph is introduced to express the frequent 2-sequence. It need scan the transaction database only once, then mapping information related to the mining task into graph. The graph representation can fully utilize the property of item order in the mining process, thus improving the generating efficiency of frequent sequences. Besides it makes use of the positional information of sequence to count support, therefore reducing the complexity of time constraints processing, and avoiding the process of testing whether a candidate sequence is contained in a data sequence. Experimental results prove the superiority of the algorithm in time and space performance.

Keywords: data mining; sequential pattern; time constrain

Rakesh Agrawal 等对超市数据进行分析时首先提出了序列模式(sequential patterns),发现了这一 KDD 分支. 序列模式挖掘的主要任务是找出随着时间或是特定顺序经常发生的模式. 关于其挖掘算法已经有很多研究[1-6].

传统的序列模式挖掘过程与用户的交互限制在较低的范围,仅需指定抽取模式的最小支持度,最终生成的模式数量很大,然而由于缺少针对性,有用的却很少. 因而需要进行大量的筛选抽取出有用的模式. 为此,文献[2]将序列模式的基本模型进行了扩展,加入了时间约束、滑动时间窗口和分类层次,提出了泛化序列模式发现问题并给出了相应的 GSP 算法. 其中时间约束是一种最基本且实用的约束,通过限制模式中相邻元素间的最大和最小时间间隔(max_gap 和 min_gap),避免用户不感兴趣的模式产生. 为处理时间约束,GSP 算法通过一个前推阶段(forward phase)和回溯阶段(backward phase)来判断一个客户序列是否包含一个候选序列. 每次"前推"或"回溯"都要对序列元素的时间标签进行运算、判断,计算量较大,并且该算法需要反复扫描数据库,扫描的次数取决于数据库中最长频繁序列的长度,如果数据库很大且最长频繁序列很长,则磁盘

收稿日期:2006-06-20.
基金项目:安徽省自然科学基金资助项目(050420207).

I/O 开销会很高.

本文针对带时间约束的序列模式挖掘提出了一种改进算法 TSPM (time-constrained sequential pattern mining),主要策略是用图来表示所有的频繁 2 序列,再利用频繁 2 序列进行时序连接逐层扩展生成更长的序列. 考虑到时间约束只存在于相邻元素之间,这种连接方式简化了处理时间约束的过程,并且图中过滤了不可能成为频繁 2 序列的组合,因而在生成更长频繁序列的过程中,大大减少了需要处理的数据量. 实验证明,该算法是精确和有效的.

1 基本概念

1.1 序列模式的基本模型

定义 1 非空集合 $I = \{i_1, i_2, \cdots, i_m\}$ 称为项集(itemset),其中 i_k ($k = 1, \cdots, m$) 称为项(item).

定义 2 序列(sequence)是项集的有序表,记为 $\alpha = <a_1, a_2, \cdots a_n>$,其中,$a_i$ ($i = 1, \cdots, n$) 为项集,称为序列的元素. 含有 k 个项的序列长度为 k,称为 k 序列($k = \sum |a_i|$).

定义 3 令序列 $\alpha = <a_1, a_2, \cdots a_n>$,序列 $\beta = <b_1, b_2, \cdots b_m>$,若存在整数 $i_1 < i_2 \cdots < i_n$,使得 $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \cdots, a_n \subseteq b_{in}$,则称序列 α 是序列 β 的子序列,或序列 β 包含序列 α.

定义 4 包含某个序列的客户序列在所有客户序列中所占的比例称为这个序列的支持度. 计算序列的支持度时,如果一个序列在同一个客户序列中多次出现,按一次计算.

用于序列模式挖掘的交易数据库 D 由客户序列组成. 客户序列是客户的所有交易按时间的升序排列,以客户号 Cid 标识,客户序列中的每笔交易以时间 Tid 标识. 给定交易数据库 D,序列模式挖掘的问题就是发现 D 中所有满足最小支持度的频繁序列,每一个这样的频繁序列代表了一个序列模式(a sequential pattern).

1.2 时间约束

对于仅给定最小支持度的序列模式挖掘,可能产生大量的模式,其中许多模式用户并不感兴趣. 时间约束的加入,使得序列模式的定义更加灵活实用. 用户可以根据需要限制所挖掘序列模式中相邻元素间的时间间隔,避免了大量冗余模式的产生,使得挖掘过程更有效率.

给定用户指定的最大时间间隔 max_gap 和最小时间间隔 min_gap,序列 $s = <s_1, s_2, \cdots s_n>$ 被客户序列 $d = <d_1, d_2, \cdots d_m>$ 包含被重新定义如下:

1) s 是 d 的子序列;

2) transaction_time (s_i) - transaction_time (s_{i-1}) > min_gap, $1 < i \leq n$;

3) transaction_time (si) - transaction_time (s_{i-1}) max_gap, $1 < i \leq n$.

由于引入了最大时间间隔 max_gap,包含序列 s 的客户序列 d 并不一定包含 s 的所有子序列,但是 d 一定包含 s 的所有连续子序列[2].

定义 5 对给定的序列 $s = <s_1 \cdots s_n>$ 及其子序列 c,如果满足以下条件之一,则 c 是 s 的连续子序列:

1) c 是由删除 s 中第一个元素或最后一个元素中的某一项目得到的;

2) c 是由删除 s 中任一个具有多于两个项目的元素中的某一项目得到的;

3) c 是 c 的连续子序列,c 是 s 的连续子序列.

2 TSPM 算法

2.1 算法采用的关键技术

首先,引入图结构表示序列信息,加速频繁序列的产生. 算法中,以频繁项为顶点,每一频繁 2 序列对应一条边构造图,再基于这张图逐层生成所有的频繁序列. 方法是对已发现的频繁 $k-1$ 序列,从图中获得其末项对应的顶点,以该顶点出发的频繁 2 序列覆盖了此频繁 $k-1$ 序列扩展成频繁 k 序列的所有可能,因此将频繁 $k-1$ 序列和这些频繁 2 序列连接,即用 2 序列扩展 $k-1$ 序列的末项(包括项扩展和序列扩展),生成候选频繁 k 序列.

另外,算法利用序列的位置信息计算支持度. 如果顾客序列 c 包含序列 s,s 在 c 中的位置约定为 c 中包含 s 的最后一个元素的交易标识符,以(Cid, Tid)表示. s 在 c 中可能有多个位置,例如 $<ab>$ 在 Cid = 20 的序列 $<a(ab)(c)(bd)>$ 中的位置有 2 个,分别为(20,2)和(20,4). s 在所有客户序列中的位置组成了 s 的位置集合 $P(s)$. 计算该集合中不同的 Cid 值,得到序列 s 的支持数. 这样,在判断候选 $k+1$ 序列是否频繁时,只需计算出其位置集合,省去了反复测试序列是否被客户序列包含的过程,并且只需判断生成它的频繁 k 序列和频繁 2 序列之间是否满足时间约束,避免了前推/回溯,简化了时间约束的处理.

2.2 算法描述

算法对数据库一次扫描,转换原交易数据库,得到项目的位置集合. 再通过频繁项的自连接和位置集合的计算得到所有的频繁 2 序列及其位置集合,

以此构造频繁 2 序列图,用邻接矩阵表示. 基于这张图,通过对频繁 $k-1$ 序列的末项进行频繁 2 序列扩展,逐层生成所有满足条件的频繁 k 序列. 如此循环,直到没有新的频繁序列产生.

算法框架:

输入:交易数据库 D,最小支持数 min_sup,最小时间间隔 min_gap,最大时间间隔 max_gap.

输出:所有满足 min_sup,min_gap 和 max_gap 的序列模式.

1) Scan D once to find all frequent items (1-sequences) and their positional data.

2) Generate all frequent 2-sequences and add their positional data to matrix IM and SM for extension.

3) for ($k=3; L_{k-1} \neq \Phi; k++$) {

4) $L_k = \Phi$;

5) for each sequence p in L_{k-1} {

6) for IM 和 SM 中 p 项所在行中的非空元素 {

7) 以该元素对应的频繁 2 序列扩展 p 得到 k 序列 p' 及其位置集合;

8) if sup(p') ≥ min_sup

9) $L_k = L_k \cup \{p'\}$; } }

10) Answer = $\cup L_k$ }

以表 1 中的交易数据库为例,来说明上述过程. 扫描数据库一遍得到项目及其位置集如表 2 所示. 位置集合中不同 Cid 个数即为项目的支持数. 假设给定最小支持数为 2,得到频繁项也就是频繁 1 序列为 <a>:2、<b>:3、<c>:3、<d>:3、<e>:2.

表 1　交易数据库

Table 1　Transaction database

Cid	tid	item
1	1	a
1	2	b, d
1	3	c
1	4	e
2	1	a
2	2	a, d
2	3	b
2	4	c
2	5	e
3	1	c
3	2	b, d
3	3	f

表 2　项目及其位置集

Table 2　Item and its position

item	P(s)
a	(1,1) (2,1) (2,2)
b	(1,2) (2,3) (3,2)
c	(1,3) (2,4) (3,1)
d	(1,2) (2,2) (3,2)
e	(1,4) (2,5)
f	(3,3)

2.2.1　图的构造及表示

频繁 2 序列的生成,通过频繁 1 序列的自连接. 但是不需要再次扫描数据库来计算它们的支持度,而是利用频繁 1 序列的位置集合来构造频繁 2 序列的位置集合.

对项 i 和项 j 的连接而成的序列,考虑项连接和序列连接 2 种情况:

1) 项连接 i 和 j 形成序列 <(i, j)>,其位置集合由 i 和 j 的位置中完全相同的位置组成,例如对表 2 中的项 b 和 d 进行项连接形成 2 序列 <(b, d)>,b 和 d 的位置集合中有相同的位置 (3,2),所以序列 <(b, d)> 的位置集合为{(3,2)};

2) 序列连接 i 和 j 形成序列 <i, j>,其位置集合由 j 的位置中能够找到 i 的某一位置与之匹配满足 (i.Cid = j.Cid) and (min_sup < b.tid - a.tid ≤ max_sup) 的位置组成. 例如对表 2 中的项 a 和 b 进行序列连接形成序列 2 序列 <(a, b)>,a 的位置 (1,1) 和 b 的位置 (1,2) 匹配得到 <a, b> 的一个位置 (1,2),a 的位置 (2,1) 和 b 的位置 (2,3) 相匹配得到 <a, b> 的一个位置 (2,3),所以 <a, b> 的位置集合为{(1,2) (2,3)}.

通过对位置集合的计算可判断 2 序列是否频繁. 用图表示所有的频繁 2 序列,图中的顶点对应频繁项,每条边对应一个频繁 2 序列,以其位置集合作为边的权值. 用邻接矩阵表示图,行表示序列的起点,列表示序列终点.

项连接和序列连接而成的频繁 2 序列代表了 2 种类型的边,可分别用于序列的项扩展和序列扩展,建立项扩展矩阵 IM 和序列扩展矩阵 SM 来分别表示. SM 表示 <i, j> 型边,其维数是频繁项的个数. IM 表示 <(i, j)> 型边,由于同一交易中的项目不重复且不区分顺序,所以 IM 是一个三角矩阵,维数为频繁项目数减 1.

以表 2 为例,给定 min_sup = 2,min_gap = 0,max_gap = 2. 所有频繁项两两连接得到的频繁 2-序列及其位置集合表示在图中得到 SM 和 IM 如表

3 和表 4 所示.

表 3 序列扩展矩阵 SM

Table 3 Sequence-extention matrix SM

	a	b	c	d	e
a		(1,2) (2,3)	(1,3) (2,4)	(1,2) (2,2)	
b			(1,3) (2,4)		(1,4) (2,5)
c					(1,4) (2,5)
d			(1,3) (2,4)		
e					

表 4 项扩展矩阵 IM

Table 4 Item-extention matrix IM

	b	c	d	e
a				
b			(1,2) (3,2)	
c				
d				

2.2.2 基于图(SM 和 IM)逐层挖掘所有序列模式

由 SM 和 IM,可以得到某一项可以向哪些项扩展,形成频繁 2 序列,并且详细记录了频繁 2 序列在客户序列中的位置. 这些提供了由频繁 $k-1$ 序列向频繁 k 序列扩展的所有信息. 由频繁 $k-1$ 序列生成候选频繁 k 序列的方法如下:

对频繁 $k-1$ 序列 p 的最后一项,扫描它所在 SM 和 IM 中所在的行,其中非空元素代表了可以对该 $k-1$ 序列的末项进行扩展生成频繁 k 序列的频繁 2 序列,分为序列扩展和项扩展 2 种. SM 作序列扩展,IM 作项扩展. 即如果序列 p 的最后一项为 i,

1) 序列扩展:扫描 SM 第 i 行非空元素 SM$[i][j]$,把 j 作为独立的项集(j)追加到 p 末形成序列 p',p 的位置由 $<(i,j)>$ 的位置中能够找到 p 的某一位置与之匹配满足条件$(p.\mathrm{Cid} = <i,j>.\mathrm{Cid})$ and$(\min_\sup < <(i,j)>.\mathrm{tid} - p.\mathrm{tid} \leq \max_\sup)$ 的位置组成.

2) 项扩展:扫描 IM 第 i 行非空元素 IM$[i][j]$,把 j 作为最后一项追加到 p 的末项集中形成 p',p' 的位置集合由 $<(i,j)>$ 和序列 p 的完全相同的位置组成.

以上一节的例子,比如对 $<ab>$ 进行序列扩展,扫描 SM$[b][c]$可获得一个 3 序列 $<abc>$ 及其位置集合$\{(1,3),(2,4)\}$,扫描 SM$[b][e]$可获得一个 3 序列 $<abe>$ 及其位置集合$\{(1,4),(2,5)\}$. 对 $<ab>$ 进行项扩展,扫描 IM$[b][d]$可获得一个 3 序列 $<a(bd)>$ 及其位置集合 $\{(1,2)\}$.

对于生成的候选序列,可以通过对位置集合中不同 Cid 的计数判断其是否频繁. 这种对频繁 k 序列进行频繁 2 序列扩展的方式较传统的连接方式降低了候选序列生成的数量和需要搜索的空间,并且可以生成所有的频繁序列,可以证明其可行性,

定理 1 通过对频繁 k 序列进行频繁 2 序列扩展,可以产生所有可能的频繁 $k+1$ 序列.

证明 (反证)假设有一频繁 $k+1$ 序列不能通过频繁 k 序列进行频繁 2 序列扩展获得,那么或者其前 k 项构成的 k 序列不频繁或者其最后 2 项构成的 2 序列不频繁,但根据定义 5,其前 k 项构成的 k 序列和其最后 2 项构成的 2 序列均为此 $k+1$ 序列的连续子序列,如果序列频繁,它的连续子序列必然都频繁[12],矛盾. 因此,对频繁 k 序列进行频繁 2 序列扩展,可以产生所有可能的频繁 $k+1$ 序列.

3 分析与实验

3.1 算法性能分析

与经典的针对时间约束的序列模式挖掘比较,TSPM 方法有以下特点:

1) 将与挖掘任务相关的信息表示在图中. 从而减少了扫描原始交易数据库的次数;

2) 通过子序列位置集合上的先后的连接得到候选序列的支持度,避免了判断候选序列与顾客序列之间的包含关系,并简化了其中对时间约束的处理;

3) 经典算法使用的是水平数据库的形式,在算法的第一步需要将交易数据库转换为顾客序列数据库,而 TSPM 算法直接由交易数据库构造项目的位置集合,不需要做这样的转换.

3.2 实验结果

通过试验,将本文提出的 TSPM 算法与经典的 GSP 算法进行比较,GSP 算法是目前序列模式挖掘领域中广泛认可的一种算法,该算法已被引入 IBM 公司的数据挖掘产品中. 在 P Ⅲ1.2 Gbps,256 MB 内存的机子上,以 JDK1.4.2 为开发环境进行试验. 实验数据库由 IBM 数据生成器生成. 在数据库 D1C5T3N0.05S4I1.25 中,取最小支持度为 1%～10%,min_gap = 0,max_gap = 10,运行 TSPM 算法和 GSP 算法. TSPM 算法与 GSP 算法得到的结果相同,这说明它是正确的. 同时还对它们的运算效率做了初步的比较,实验结果如图 1 和图 2 所示. 实验证明随着支持度的增加,在时间性能上 TSPM 算法具有明显的优势,在空间性能上 TSPM 算法相对于 GSP 算法在最小支持度较低时略有改善,在最小支持度较高时相差不多. 空间上,TSPM 不需要像 GSP 那样保存大量的候选序列,但需要保存一张频繁序列图,因此,为了进一步改善其空间性能,对于

如何降低图的规模还有待于进一步研究.

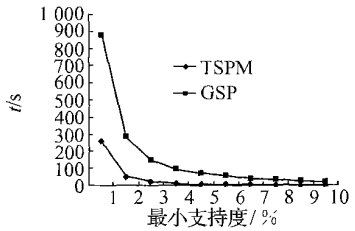

图 1　TSPM 和 GSP 执行时间比较

Fig. 1　Comparison of time performance

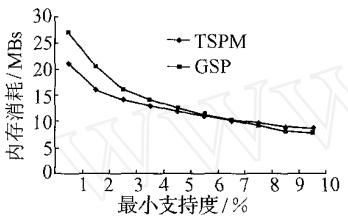

图 2　TSPM 和 GSP 空间性能比较

Fig. 2　Comparison of memory performance

4　结束语

序列模式发现是近几年越来越受到关注的研究方向. 本文提出的带时间约束的序列模式发现算法 TSPM 采用了图的数据结构和纵向数据存储格式,克服了经典算法在时空性能方面的不足,具有一定的优越性. 未来的研究包括如何进一步降低图的规模,如何加入有效的约束条件以及与闭合序列相结合,在改善时空性能的同时挖掘出更加有价值的序列模式.

参考文献:

[1] AGRAWAL R, SRIKANT R. Mining sequential patterns[A]. In: Proc of the 11st Int Conf on Data Engineering[C]. Taipei, China, 1995.

[2] SRIKANT R, AGRAWAL R. Mining sequential patterns: generalizations and performance improvements [A]. In: Proc of the Fifth Int. Conf. on Extending Database Technology (EDBT) [C]. Avignon, France, 1996.

[3] SPADE M. J. An Efficient algorithm for mining frequent sequences [A]. In: Proceeding of Machine Learning Journal, Special issue on Unsupervised Learning[C]. [s. l.], 2003.

[4] MASSEGLIA F, CATHALA F, PONCELET P. The psp approach for mining sequential patterns [A]. In: Proc. 1998 European symp. Principle of data mining and knowledge discovery (PKDD '98) [C]. Nantes, France, 1998.

[5] HAN Peijian. FreeSpan: frequent pattern-projected sequential pattern mining [A]. In: Proc 2000 Int Conf Knowledge Discovery and Data Mining (KDD. 00) [C]. Boston, 2000.

[6] PEI J, HAN J, MORTAZAVI-ASL B, PINTO H, CHEN Q, DAYAL U, HSU M C. PrefixSpan: mining sequential patterns efficiently by prefix - projected pattern growth[A]. In: Proc 2001 Int Conf Data Engineering (ICDE '01) [C]. Heidelberg, Germany, 2001.

[7] 邓明荣,叶福根,史　烈,潘云鹤. 挖掘泛化序列模式的一种有效方法[J]. 浙江大学学报,2002,29(4):415 - 422.
DENG Mingrong, YE Fugen, SHI Lie, PAN Yunhe. Efficient algorithm for mining generalized sequential patterns[J]. Journal of Zhejiang University, 2002, 29(4): 415 - 42.

[8] 朱立运,朱建秋. 带时间特征的序列模式挖掘算法 TESP [J]. 计算机工程,2004,30(10),51 - 54.
ZHU Liyun, ZHU Jianqiu. Time-enriched equential pattern mining algorithm TESP[J]. Computer Engineering, 2004, 30(10), 51 - 54.

[9] 周　斌,吴泉源. 序列模式挖掘的一种渐进算法[J]. 计算机学报,1999,22(10):882 - 887.
ZHOU Bin, WU Quanyuan. An incremental algorithm for mining sequential pattern [J]. Chinese Journal of Computers, 1999, 22(10): 882 - 887.

[10] 陈金玉,樊兴华. 序列模式的一种挖掘算法[J]. 重庆大学学报(自然科学版),2001,24(1):92 - 94.
CHEN Jinyu, FAN Xinghua. Algorithm for mining sequential pattern [J]. Journal of Chongqing University (Natural Science Edition), 2001, 24(1): 92 - 94.

[11] 刘月波,陆阶平,刘同明. 基于 CTID 序列模式的一种改进算法[J]. 微机发展,2005,15(3):20 - 22.
LIU Yuebo, LU Jieping, LIU Tongming. An improved algorithm for mining sequential patterns based on CTID [J]. Microcomputer Development, 2005, 15(3): 20 - 22.

作者简介:

胡学钢,男,1961 年生,教授,主要研究方向为知识工程、数据挖掘、数据结构. 主持及参加国家自然基金课题、国家教委博士点专项基金课题、安徽省自然基金课题、安徽省教委基金课题等多项课题. 发表论文 20 多篇,出版著作多部.

E-mail: jsjxhuxg@hfut.edu.cn.

张圆圆,女,1982 年生,硕士研究生,主要研究方向为知识工程、数据挖掘.

E-mail: zhangyuanyuan0401@sina.com.