一种基于序列末项位置信息的序列模式挖掘算法*

缪裕青，吴孔玲，朱晓雁，苏 杰
（桂林电子科技大学 计算机科学与工程学院，广西 桂林 541004）

摘 要：针对 PrefixSpan 算法中反复扫描投影数据库寻找局部频繁项并重复构造挖掘大量重复投影数据库的不足，提出一种基于序列末项位置信息的序列模式挖掘算法 SPM-LIPT。通过连接 2-序列位置信息表（LIPT）找到序列模式的下一项，实现序列模式增长，避免对投影数据库反复扫描；同时通过检查相同末项序列首位置信息表（SLIFPT）进行前向剪枝；消除大量重复投影的构建。最后通过实验证明了算法的有效性。

关键词：数据挖掘；序列模式挖掘；位置信息；投影数据库
中图分类号：TP391 **文献标志码**：A **文章编号**：1001-3695(2012)07-2505-04
doi:10.3969/j.issn.1001-3695.2012.07.027

Sequential pattern mining algorithm based on last item location information of sequence

MIAO Yu-qing, WU Kong-ling, ZHU Xiao-yan, SU Jie
(*School of Computer Science & Engineering, Guilin University of Electronic Technology, Guilin Guangxi* 541004, *China*)

Abstract: In order to solve the defects of repeatedly scanning projection database looking for local frequent item and producing, mining large number of duplicated project databases in PrefixSpan algorithm, this paper proposed the SPM-LIPT algorithm for sequential pattern mining. By connecting the 2-sequence LIPT (last item position table), the algorithm found the next item of the sequence, realized sequential pattern growth and avoided repeatedly scanning projection database. At the same time, it also could avoid producing and mining large number of duplicated project databases by checking SLIFPT (same last item first position table) prior to pruning. Experiments show that the algorithm is effective.

Key words: data mining; sequential pattern mining; position information; projected database

序列模式挖掘是数据挖掘的一个重要研究领域，有非常广泛的应用前景，近年来已在生物信息学、医学诊疗、购物信息分析、客户行为分析等多个领域取得了良好的经济效益和社会效益。自 Agrawal 等人[1]提出序列模式挖掘以来，各国研究人员学者不断提出新算法或是改进现有算法缺点，以求得高效挖掘序列模式。

现有的序列模式挖掘算法基本上可以分为两类，一类是以 AprioriAll[1]、GSP[2]、SPADE[3]为代表的基于 Apriori 性质逐层发现方法；另一类是以 PrefixSpan[4]为代表的模式增长方法。基于 Apriori 性质逐层发现方法在挖掘过程中需要多次遍历原始数据库计算支持度，且会生成大量的候选序列模式，这是影响算法效率的主要原因。SPADE 算法由于采用数据的垂直表示，减少了对原始数据库的扫描，通过连接项的位置信息表挖掘序列模式，被认为是基于 Apriori 性质算法中运行速度最快的算法。Pei 提出的 PrefixSpan 算法基于分而治之的思想，通过投影，把搜索空间划分成更小的多个投影数据库，然后在投影数据库中挖掘频繁项，以构成新的作为下次投影的前缀的序列模式。依次递归地构造挖掘投影数据库，直到挖掘出所有的序列模式。PrefixSpan 算法通过投影将大的数据集分割成小的数据集，且挖掘过程不生成候选序列，效率大大提高，被公认为是挖掘大型数据库时的首选算法。然而它需要递归的构造大量投影数据库，开销巨大，同时大量重复投影数据库的存在，使得重复对投影数据库进行扫描挖掘，降低了算法效率。

本文提出的基于序列末项位置信息算法 SPM-LIPT（sequence pattern mining based last item position table）通过构建扫描 2-序列信息，实现 $K-1$ 序列位置信息与 2-序列位置信息连接，生成 K-序列以及相应位置信息，避免 PrefixSpan 算法中多次扫描投影数据库寻找局部频繁项；同时受 BIDE 算法前向与后向剪枝策略的启发，构建相同末项序列首位置信息表，通过检查相同末项序列首位置信息表进行前向剪枝，避免 PrefixSpan 中重复对投影数据库的扫描挖掘。

1 基本术语与问题描述

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是所有不同属性的集合，每个属性又称为项（item）。项集（itemset）是项组成的非空集合，是 I 的子集。

序列（sequence）是项集的有序表，记做 $S=\langle s_1,s_2,\cdots,s_n\rangle$。其中 $s_k(1\leqslant k\leqslant n)$ 是非空项集，称为序列的元素（element）。序

收稿日期：2011-11-20；**修回日期**：2011-12-31 **基金项目**：广西可信软件重点实验室开放基金；广西研究生科研创新项目（2011105950812M22）

作者简介：缪裕青（1966-），女，副教授，博士，主要研究方向为数据挖掘、生物数据挖掘（miaoyuqing@guet.edu.cn）；吴孔玲（1986-），女，硕士，主要研究方向为数据挖掘、序列模式挖掘；朱晓雁（1979-），女，硕士，主要研究方向为管理学、营销管理；苏杰（1986-），男，硕士，主要研究方向为数据挖掘、文本挖掘.

列包含的项的个数称为序列的长度,长度为 K 的序列记为 K-序列。

给定两个序列 $A=\langle a_1,a_2,\cdots,a_m\rangle$ 和 $B=\langle b_1,b_2,\cdots,b_n\rangle$,如果存在一组整数 $j_1<j_2<\cdots<j_m$,使得 a_1 包含于 b_{j_1},a_2 包含于 b_{j_2},a_m 包含于 b_{j_m},则称序列 A 包含于序列 B,A 是 B 的子序列,记为 $A\subseteq B$。

序列数据库 D 是元组(S_{id},S)的集合。其中,S_{id}是序列号,S 是序列。如果序列 T 是 S 的子序列(即 $T\subseteq S$),称元组(S_{id},S)包含序列 T。序列 T 在序列数据库 D 中的支持度是数据库中包含 T 的元组个数,即 $\text{support}D(T)=|\{(S_{id},S)|(S_{id},S)\in D\wedge T\subseteq S\}|$,记做 support($T$)。给定支持度阈值 ξ,如果序列 T 在序列数据库中的支持度不低于 ξ,则称序列 T 为序列模式,长度为 l 的序列模式记为 l-模式。

前缀:设每个元素中的所有项目按照字典序排列。给定序 $\alpha=\langle e_1,e_2,\cdots,e_n\rangle$,$\beta=\langle e_1',e_2',\cdots,e_m'\rangle$($m\leqslant n$),如果 $e_i'=e_i(i\leqslant m-1)$,$e_m'\subseteq e_m$,并且(e_m-e_m')中的项目均在 e_m' 中项目的后面,则称 β 是 α 的前缀。例如表 1 所示的序列数据库中,序列 CAA 是序列 $CAABC$ 的前缀,而序列 AA 则不是。

投影:给定序列 α 和 β,如果 β 是 α 的子序列,则 α 关于 β 的投影 α' 必须满足:β 是 α' 的前缀,α' 是 α 的满足上述条件的最大子序列。例如表 1 中,序列 BC 是序列 $ABCB$ 的子序列,则序列 $ABCB$ 关于序列 BC 的投影是 BCB。

后缀:序列 α 关于子序列 $\beta=\langle e_1,e_2,\cdots,e_{m-1},e_m'\rangle$ 的投影为 $\alpha'=\langle e_1,e_2,\cdots,e_n\rangle$($n\geqslant m$),则序列 α 关于子序列 β 的后缀为$\langle e_m'',e_{m+1},\cdots,e_n\rangle$。其中 $e_m''=(e_m-e_m')$。例如表 1 中,序列 $ABCB$ 关于子序列 BC 的后缀为 B。

投影数据库:设 α 为序列数据库 S 中的一个序列模式,则 α 的投影数据库为 S 中所有以 α 为前缀的序列相对于 α 的后缀,记为 $S|\alpha$。例如对于表 1 的序列数据库,假设 AB 是一个序列模式,则 AB 的投影数据库如表 2 所示。

表 1 序列数据库

S_{id}	序列
S_1	$CAABC$
S_2	$ABCB$
S_3	$CABC$
S_4	$ABBCA$

表 2 AB 的投影数据库

S_{id}	序列
S_1	C
S_2	CB
S_3	C
S_4	BCA

序列模式挖掘就是要找出序列数据库中所有的序列模式,即支持度不小于用户给定的支持度阈值的所有序列。

2 SPM-LIPT 算法

2.1 形式化定义

连接:若序列 $\alpha=\langle e_1,e_2,\cdots,e_n\rangle$,$\beta=\langle b_1,b_2\rangle$,若 $e_n=b_1$,则序列 α 与 β 可以进行连接,连接后得到新序列 $\gamma=\langle e_1,e_2,\cdots,e_n,b_2\rangle$,即找到序列的下一项。

序列末项位置信息 LIPT(last item position table):LIPT 由序列号和位置号两个属性构成。序列号表示末项所在的序列,位置号表示末项在序列中出现的位置,体现序列的时间关系。若某项不存在于某序列中,则对应的位置号属性值为 0。例如对表 1 所示的序列数据库,序列 A 的 LIPT 如表 3 所示,序列 AB 的 LIPT 如表 4 所示。

表 3 序列 A 的 LIPT

S_{id}	Ipid
S_1	2,3
S_2	1
S_3	2
S_4	1,5

表 4 序列 AB 的 LIPT

S_{id}	Ipid
S_1	4
S_2	2,4
S_3	3
S_4	2,3

相同末项序列首位置信息 SLIFPT(same last item first position table):由所有含有相同最后项的序列的首位置信息组成,包括两个属性。序列号表示末项所在的序列,位置号表示末项出现的首位置,以某项 α 为末项的序列首位置信息表记为 α-SLIFPT。例如表 1 所示的示例数据库,假设最小支持度为 2,则所有的序列模式为 $S_{fs}=\{A:4,AA:2,CA:3,B:4,AB:4,BB:2,CB:3,ABB:2,CAB:2,C:4,AC:4,BC:4,CC:2,ABC:4,CAC:2,CBC:2,CABC:2\}$,则可以构造三个分别以 A、B、C 为末项的 SLIFPT。A 的 SLIFPT 如表 5 所示,B 的 SLIFPT 如表 6 所示。

表 5 A-SLIFPT

S_{id}	S_1	S_2	S_3	S_4
Fpid	2	1	2	1
	3	0	0	5
	2	0	2	5

表 6 B-SLIFPT

S_{id}	S_1	S_2	S_3	S_4
Fpid	4	2	3	2
	0	4	0	3
	4	0	3	0
	4	4	3	0

说明:以 A 为末项的序列有 A、AA、CA。序列 A 在 S_1、S_2、S_3、S_4 中首位置为 2、1、2、1。序列 AA 在 S_1、S_2、S_3、S_4 中首位置为 3、0、0、5。序列 CA 在 S_1、S_2、S_3、S_4 中首位置为 2、0、2、5。填入表格即得表 5。同理可得到表 6。

前缀搜索树:为满足以下条件的树:a)根节点为空;b)除根节点外,其他节点代表从根节点到当前节点的序列;c)按照字母排序,左兄弟节点小于右兄弟节点。图 1 为运用 PrefixSpan 对表 1 所示的数据库进行挖掘得到的前缀搜索树。

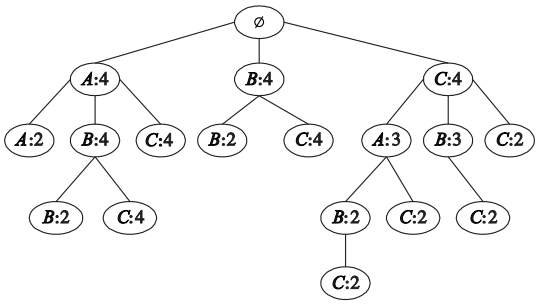

图1 前缀搜索树

定理 在进行深度优先挖掘时,若前缀搜索树中当前节点的首位置信息已存在于某一相同末项序列首位置信息表中,则对当前节点可以进行剪枝。

证明 因为当前节点的首位置信息已存在于某一末项 SLIFPT 中,则根据投影库的定义,可知以当前节点代表的序列为前缀的投影数据库与已产生的某序列的投影数据库相同,且已进行了挖掘,所以当前节点无须再进行挖掘,利用先前的挖掘结果,产生相应的序列模式。

2.2 SPM-LIPT 算法描述

SPM-LIPT 算法采用深度优先策略,主要思想是:首先检查当前节点(设深度为 $K-1$)的首位置信息是否存在于某项的 SLIFPT 中,如存在,则进行剪枝;否则通过与 2-序列的连接,找

到序列的下一项,序列长度增 1。连接主要是通过当前 K-1 序列 LIPT 和 2-序列 LIPT 的比对进行的,同时生成 K-序列的 LIPT。它遵循以下规则:对于相同的 S_{id},a)若有一方的 Ipid 为 0,则新生成的相对应的 Ipid 为 0;b)若 K-1 序列 LTPT 中 Ipid 大于 2-序列 LTPT 中的 Ipid,则新生成的相对应的 Ipid 为 0;c)若 K-1 序列 LTPT 中 Ipid 小于 2-序列 LTPT 中的 Ipid,则将比较后的多种可能结果存放于相对应的 Ipid 中,如图 2 所示。

AB	
S_{id}	Ipid
S_1	4
S_2	2,4
S_3	3
S_4	2,3

＋

BC	
S_{id}	Ipid
S_1	5
S_2	3
S_3	4
S_4	4

＝

ABC	
S_{id}	Ipid
S_1	5
S_2	3
S_3	4
S_4	4

图2　序列*AB*与*BC*连接生成*ABC*

算法的主要步骤如下:

a)扫描数据库一次,找出所有的 N 个频繁项,即频繁 1-序列。

b)再次扫描数据库,记录各频繁项出现的首位置信息,并对各频繁项进行投影,得到 N 个投影数据库。

c)对各个投影数据库扫描找出局部频繁项,构建局部频繁项的 LIPT,即得到所有的 2-序列 LIPT。

d)在前缀搜索中对当前节点(设深度为 $K-1$),首先判断它的首位置信息是否已经存在于某项的 SLIFPT 中,若存在,则进行剪枝,无须进行后继挖掘,同时更新前缀搜索树;若不存在,则把首位置信息加入 SLIFPT 中,把节点插入前缀搜索,同时向前搜索与 2-序列 LIPT 进行连接,找到序列下一节点,序列长度增加 1。

算法描述如下:

输入:序列数据 S 和最小支持度 min_sup。
输出:前缀搜索树 T。

1)扫描数据库,得到所有频繁项,即 1-序列,记其组成的集合为 α
2)for 每个频繁项 a(a∈α)
记录 a 首次出现的位置信息,构造 a 的投影库,构建局部频繁项的 LIPT,即得到 2-序列 LIPT。同时也得到局部频繁项,记其集合为 β。
3)end for
4)创建频繁项头表 header,使其指向相应的局部频繁项集 β
5)初始化搜索树以及 SLIFPT 表
6)for 每个频繁项 b(b∈α)
调用 SPM(b,b-SLIFPT,b-LIPT,T)
7)end for
子程序:SPM(b,b-SLIFPT,b-LIPT,T)
输入:前缀树 T,当前节点 b,以 b 为末项的 SLIFPT 表,b 所代表的序列的 LIPT。
输出:更新的前缀树 T。
8)从 b-LIPT 获取首位置信息,判断是否已存在于 b-SLIFPT
9)如存在,则对 b 进行剪枝,更新前缀树 T
10)如不存在,则将 b 的首位置信息插入 b-SLIFPT 中,将 b 插入前缀树 T 中,调用 Join(b, header,b- LIPT)
11)若能找到序列的下一下 b′递归调用 SPM(b′,b′-SLIFPT,b′-LIPT,T)
12)若找不到下一项,则返回上一层节点 b″,调用 Join(b″,header,b″-LIPT),遍历局部频繁项集 β 中未遍历频繁项
13)若已遍历完或是找不到下一项,转至 12),否则转至 11)
子程序:Join(b, header, b-LIPT)
输入:当前节点 b, 频繁项头表 header,b 所代表的序列 LIPT。
输出:序列下一点节点 b′, b′所代表的序列 LIPT。
14)从频繁项头表寻找与 b 相等的项,没有则序列增长结束
15)若找到,则对其指向的局部频繁项集 β 的每一个元素 b′,将 b-LIPT 与 b′的记录进行比较连接,生成 b′-LIPT。

2.3　实例说明

下面以表 1 所示的示例数据库,假设最小 min_sup = 2,来说明 SPM-LIPT 算法的挖掘过程。

a)扫描数据库,得到频繁项,即 1-序列,A:4,B:4,C:4。

b)对于各频繁项,记录其首位置信息,对其进行投影,构造局部频繁项 LIPT,即得到 2-序列 LIPT。对频繁项 A,记录其在对应序列首位置信息 2、1、2、1,同时得到序列 AA、AB、AC 的 LIPT,如图 3 所示。

AA	
S_1	3
S_2	0
S_3	0
S_4	5

AB	
S_1	4
S_2	2, 4
S_3	3
S_4	2, 3

AC	
S_1	5
S_2	3
S_3	4
S_4	4

图3　GPI图

c)初始化搜索树以及以 A、B、C 为末项的 SLIFPT。

d)对于每个频繁项 b,递归调用函数 SPM,最后输出前缀树。函数 SPM 主要由两部分构成:a)剪枝判断;b)向前搜索,寻找序列下一项。对于 A,首先检查它的首位置信息是否已经存在于 A-SLIFPT 中,此时 A-SLIFPT 为空,所以不进行剪枝,把 2、1、2、1 加入到 A-SLIFPT 中,把 A 插入到搜索树中。此时的 SLIFPT 和前缀搜索树如图 4 所示。然后向前搜索,找到下一项 A,递归调用 SPM,如此依次下去,即可得到整个数据库的前缀搜索树。图 5 是得到的最终前缀树和各 SLIFPT。

S_{id}	S_1	S_2	S_3	S_4
Fpid	1	2	1	2

图4　*A*-SLIFPT表与前缀搜索树

A-SLIFPT				
S_{id}	S_1	S_2	S_3	S_4
	2	1	2	1
Fpid	3	0	0	5
	2	0	2	5

C-SLIFPT				
S_{id}	S_1	S_2	S_3	S_4
	5	3	4	4
Fpid	1	3	1	4
	5	0	4	0

B-SLIFPT				
S_{id}	S_1	S_2	S_3	S_4
	4	2	3	2
Fpid	0	4	0	3
	4	0	3	0
	4	4	3	0

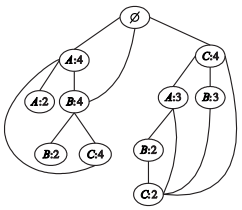

图5　实例数据库最终各项SLIFPT与最终前缀搜索树

3　实验结果

本文实验环境为 Intel® Core(™) i3-2100 CPU 2.1 GHz,2.00 GB 内存,Windows 7 操作系统;算法采用 C++语言编写,在 Visual C++ 6.0 环境下编译。实验数据采用 IBM Quest Synthetic Data Generator 生成,对数据集 D5T1.5N0.1I1.25 和 D10T1N0.2I1.25 进行测试。测试数据参数说明如下:D 表示顾客数,默认值为 1 000;C 表示每个顾客的平均事务数,默认值为 10;T 表示每个事务平均项数;N 为总项数,为 1 000;I 表示最长模式平均长度,设为 1.25。实验结果如图 6、7 所示。

图 6、7 中,取支持度为 11%、12%、13%、14%、15%、16%、17%、18%、19%,分别对数据集 D10T1N0.2I1.25、D5T1.5N0.1I1.25 进行测试。从图中可以看出,SPM-LIPT 算法在执行时间上优于 PrefixSpan 算法,但随着支持度的增加,优势减弱;同

时挖掘 D5T1.5N0.1I1.25 长序列数据集的时间消耗多于挖掘 D10T1N0.2I1.25 的短序列数据集。在长序列数据集中,SPM-LIPT 算法时间效率优势更加明显。这是因为在短序列数据集中,存在着大量的 2-序列模式,SPM-LIPT 算法在进行连接时要不断查找,扫描剪枝;在长序列数据集中,则存在大量的重复投影数据库,PrefixSpan 算法反复构建扫描投影数据库,而 SPM-LIPT 可以通过连接快速实现增长。支持度的增加使投影数据大量减少,PrefixSpan 算法执行加快,SPM-LIPT 算法进行剪枝所扫描的时间增加,优势减弱。实验证明,SPM-LIPT 算法较适合处理含有大量长序列、相识度较高的数据集,在时间执行效率上优于 PrefixSpan 算法,符合理论分析。

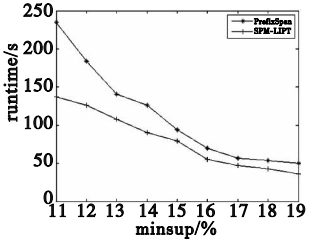

图6 D10T1N0.2I1.25数据集执行时间比较

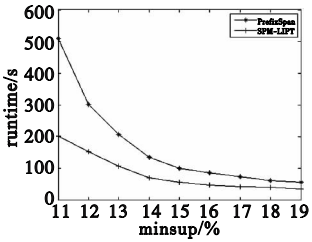

图7 D5T1.5N0.1I1.25数据集执行时间比较

4 结束语

随着应用的推广,序列模式挖掘成为研究的热点。本文在分析当前序列模式挖掘主要算法的基础上,提出基于已发现序列模式的末项位置信息算法 SPM-LIPT。通过分析与实验,证明了 SPM-LIPT 算法的有效性。下一步将考虑如何与闭合序列相结合,有效挖掘出闭合序列模式,进一步压缩挖掘结果。

参考文献:

[1] AGRAWAL R, SRIKANT R. Mining sequential patterns[C]//Proc of the 11th International Conference on Data Engineering. Washington DC:IEEE Computer Society,1995:3-14.

[2] SRIKANT R, AGRAWAL R. Mining sequential patterns:generalizations and performance improvements[C]//Proc of the 5th International Conference on Extending Database Technology. Washington DC: IEEE Computer Society,1996: 3-17.

[3] ZAKI M J. SPADE: an efficient algorithm for mining frequent sequences [J]. Machine Learning Journal,2001,42(1):31-60.

[4] PEI Jian, HAN Jia-wei, MORTAZAVI-ASL B, *et al*. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth[C]//Proc of the 17th International Conference on Data Engineering. Washington DC:IEEE Computer Society,2001:215-224.

[5] MASSEGLIA F, CATHALA F, PONCELET P. The PSP approach for mining sequential patterns[C]//Proc of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery. Berlin: Springer-Verlag,1998:174-184.

[6] ZHANG Ming-hua, KAO Ben, YIP C L, *et al*. A GSP-based efficient algorithm for mining frequent sequences[C]//Proc of International Conference on Artificial Intelligence. 2001.

[7] HAN Jia-wei, PEI Jian, MORTAZAVI-ASL B, *et al*. FreeSpan: frequent pattern-projected sequential pattern mining[C]//Proc of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM Press,2000:355-359.

[8] LIN M Y, LEE S Y. Fast discovery of sequential patterns by memory indexing[C]//Proc of the 4th International Conference of Data Warehousing and Knowledge Discovery. London, UK: Springer-Verlag, 2002:150-160.

[9] HSIEH Chia-ying, YANH Don-lin, WU Jung-pin. An efficient sequential pattern mining algorithm based on the 2-sequence matrix [C]//Proc of IEEE International Conference on Data Mining. Washington DC:IEEE Computer Society,2008:583-591.

[10] LOEKITO E, BAILEY J, PEI Jian. A binary decision diagram based approach for mining frequent subsequences[J]. Knowledge and Information Systems,2010,24(2):235-268.

[11] ZHANG Wen-zhen, MIAO Yu-qing. An improved algorithm based on suffix projection in sequential pattern mining[C]//Proc of International Conference on Computer and Computational Intelligence. 2010: 597-600.

[12] WANG Jian-yong, HAN Jia-wei. BIDE: efficient mining of frequent closed sequences[C]//Proc of the 20th International Conference on Data Engineering. Washington DC: IEEE Computer Society,2004:79-90.