doi:10.16104/j.issn.1673-1891.2019.04.015

基于投影二维表的最大频繁模式挖掘算法

王利军
(安徽经济管理学院信息工程系,合肥 230031)

摘要:PITable-MAX是基于投影二维表的最大频繁模式挖掘算法。算法只需要遍历一次事务数据库,并将数据信息存入投影数据库中,在挖掘事务项时才会从投影数据库中提取相关的数据信息生成投影二维表,从而减少对内存空间的占用,算法虽然采用递归方式,但组合策略可实现减少条件投影二维表的规模,以达到减少空间;并结合减挖策略来减少挖掘事务项的个数,以达到提高挖掘效率的目的;实验验证了算法的可行性和优越性。

关键词:投影二维表;算法改进;最大频繁项集
中图分类号:TP301.6 **文献标志码**:A **文章编号**:1673-1891(2019)03-0070-05

Maximum Frequent Pattern Mining Algorithm Based on Projection Two-Dimensional Tables

WANG Lijun
(Department of Information Engineering, Anhui Institute of Economics Management, Hefei 230031,China)

Abstract: PITable-MAX is a maximum frequent pattern mining algorithm based on projection two-dimensional tables. The algorithm only needs to traverse the transaction database once and store the data information in the projection database, and the related data information can be extracted from the projection database to generate a projection two-dimensional table when mining the transaction item, thus reducing the occupancy of memory space. Although the algorithm adopts recursive method, the combination strategy can reduce the size of conditional projection two-dimensional tables to reduce the space, and the number of mining items is reduced by combining the reduction strategy to improve the efficiency of mining. The experiment verifies the feasibility and superiority of the algorithm.

Keywords: projection two-dimensional table; algorithm improvement; maximal frequent item set

0 引言

经典的挖掘最大频繁模式的算法有FP-Max[1]、DMFIA[2]、MaxMiner[3]和MAFIA[4]等。FP-Max算法需要生成初始FP-Tree[5],且要长期存放于内存,占据较大的空间资源,另外算法采用递归挖掘,需要产生大量的条件模式树,进一步加剧对空间资源的占用。DMFIA算法也是基于FP-Tree树结构进行挖掘的,但该算法会产生过多冗余的候选项目集,影响算法的执行效率。每种经典的算法都有各自的优势,但仍存在改进的空间。

许多学者提出了一些新的改进方案,比如PFPMax算法[6]、BDRFI算法[7]等。笔者提出一种基于投影数据库而生成的投影二维表结构来存储相关事务项的数据信息和统计数值,利用该结构来进行挖掘,并采用减挖策略和组合策略加快挖掘效率,以达到提高执行效率的目的。这是一种基于投影二维表的最大频繁模式挖掘算法PITable-MAX。最后以实验方式验证了算法的可行性和优越性。

1 相关理论和举例

1.1 投影数据库

为了减少系统空间资源的浪费,笔者引入了投影数据库,它是将事务数据库划分成一系列较小的数据库,即投影数据库,并将这些投影数据库存放在磁盘空间上,直到挖掘某个事务项时才会基于相应的投影数据库形成投影二维表,若有必要可递归投影并挖掘。那如何形成投影数据库呢?首先,第

收稿日期:2019-09-10
基金项目:安徽省高校自然科学重点项目:基于spark分布式计算平台的高校教学大数据分析方法研究(KJ2019A0965)。
作者简介:王利军(1983—),男,安徽旌德人,讲师,硕士,研究方向:数据挖掘、计算机应用。

1次扫描事务数据库D,删除事务中的非频繁项并排序,预处理事务数据库D'也采用二维表来存储数据信息,该二维表包含3个部分,分别是频繁项名称区、事务信息区和总统计区。将频繁1-项集按支持度总数降序排序得到$x_1,x_2,\cdots,x_{n-1},x_n$填入预处理事务数据库D'的频繁项名称区,并将支持度计数总数填入到总统计区的相应位置。事务信息区的所有单元格的默认值为0,若事务包含相应的事务项则将相应的单元格的值更改为1,这样即可得到预处理事务数据库D'。其次,根据D'的频繁项总数n创建一个堆栈组$Z\{x_1,x_2,\cdots,x_{n-1},x_n\}$,为了减少投影数据库占用的磁盘空间的容量和重复投影,D'中的每一条事务只会投影到一个投影数据库中,根据D'中的事务的最后事务项进行投影,将以x_1为最后事务项的事务去除x_1后投影到堆栈x_1中,以此类推,这样即可得到初始投影数据库。初始投影数据库中只有堆栈x_n中包含的以x_n结尾的数据信息是完整的,而其他堆栈在挖掘过程中将进一步完善。

1.2 投影二维表

投影二维表是基于投影数据库而生成的二维表结构,该结构与预处理事务数据库D'相同,即包括3个部分,分别为频繁项名称区、事务信息区和总统计区。该二维表的频繁项名称区用于存放事务项名称,但不包含当前事务项和之后的事务项,如堆栈x_i的投影二维表的频繁项名称区为$x_1,x_2,\cdots,x_{i-1}$,事务信息区的所有单元格的默认值为0,再将投影数据库的事务逐条添加到事务信息区,若事务包含的事务项的单元格更改为1。然后对每个频繁项所在列中1的个数进行统计并填入总统计区。设最小支持度计数为minSup,删除总统计区中数值小于minSup的列进行删除,至此投影二维表制作完成。投影二维表只包含正在挖掘事务项的相关数据信息和统计,占用的内存空间较小,且挖掘完该事务项后即可立刻清除,从而减少对系统空间的浪费。

1.3 算法研究

1.3.1 基本方案

若项头表L包含1-频繁项$x_1,x_2,\cdots,x_{n-1},x_n$频繁项,则$MFS|x_i(1\leqslant i\leqslant n)$表示以$x_i$结尾的最大频繁项集的集合,因此挖掘最大频繁项集可以转换成挖掘以这些频繁项为结尾的最大频繁模式,则$MFS=MFS|x_1\cup MFS|x_2\cup\cdots\cup MFS|x_{n-1}\cup MFS|x_n$。

从堆栈x_n开始构建投影二维表PITable$\{x_n\}$,并进行挖掘,若有必要,可递归投影和挖掘,挖掘堆栈x_n完成后将堆栈x_n中每一个事务继续采用相同方法投影到其他堆栈(除堆栈x_n),接着基于堆栈x_{n-1}中投影数据库构建投影二维表,挖掘完成后将堆栈x_{n-1}中每一个事务继续投影到其他投影(除堆栈x_n和x_{n-1}),以此类推,直到堆栈x_1的投影数据库也被挖掘,算法执行完毕,合并所有的挖掘结果。

1.3.2 组合策略

定义1:组合项集在挖掘项头表L中事务项x_i时,若该项的投影二维表的总统计区的数值等于项头表L中的事务项x_i的总支持度计数,则说明这些频繁项始终跟事务项x_i同时出现,故可以将这些频繁项进行合并形成项集,支持度计数为x_i的总支持度计数。

组合策略:当挖掘项头表L中的事务项x_i时,若存在组合项集,则提前获取组合项集并与事务项x_i合并作为后缀项集,这样可以快速获得最大频繁项集,若需要递归挖掘,则在生成条件投影数据时不需要再提取组合项集包含的项的数据,加快建表速度和减少系统空间的浪费。

结合基本方案和组合策略,挖掘项头表L中的事务项x_i时,首先创建投影二维表PITable$\{x_n\}$并获取组合项集,再与事务项x_i进行合并作为后缀项集,支持度计数为事务项x_i的总支持度计数。即使组合项集为空,也不影响后缀项集的形成。

若除了组合项集包含的事务项后PITable$\{x_n\}$没有其他频繁项时,则最大频繁项集为后缀项集;若除了组合项集包含的事务项后PITable$\{x_n\}$只包含一个频繁项时,则最大频繁项集为后缀项集与该事务项进行合并,支持度计数为该事务项在PITable$\{x_n\}$中总统计区中数值;若除了组合项集包含的事务项后PITable$\{x_n\}$仍包含2个或以上的频繁项时,则需要以这些频繁项为数据信息,递归生成条件投影数据库,再创建条件投影二维表进行挖掘。

1.3.3 减挖策略

在组合策略的基础上,可以继续采用优化方案来进一步加快挖掘效率,具体描述如下:在挖掘完预处理事务数据库D'中事务项$x_i(1\leqslant i\leqslant n)$后,立即形成项集$\{x_1,x_2,\cdots,x_{n-1}\}$,检测该项集是否是最大频繁项集中某一个项集的子集,如果是,则挖掘算法可以终止,因为随后挖掘肯定不会得到新的最大频繁项集,因此无需再投影和挖掘,从而达到减少投影次数和挖掘的事务项个数加快挖掘效率的目的。

1.4 举例说明

设置最小支持度计数为3,预处理事务数据库D'如表1所示,得到的项头表L如表1的频繁项目区所示。

表1 预处理事务数据库D'

a	b	c	d	e	f	g	h	频繁项名称区
1	1	1	1	0	0	1	1	
1	1	1	1	1	1	0	0	
1	1	0	0	0	0	0	0	
1	1	1	1	1	1	0	0	
0	1	0	1	0	0	1	0	
1	1	0	0	0	0	0	1	事务信息区
1	0	1	0	1	0	1	0	
1	0	1	1	1	1	0	0	
1	1	0	0	0	0	0	1	
0	1	0	1	0	0	1	0	
1	0	1	1	1	1	0	0	
1	0	1	0	1	0	1	0	
1	0	1	1	1	1	0	0	
11	8	8	8	7	5	5	3	总统计区

挖掘事务项*h*,{*h*}为后缀项集,事务项*h*的投影二维表如表2所示,读取D'中的*h*的支持度计数为3,发现投影二维表中的*a*和*b*的总统计区中计数也为3,根据组合策略将*a*、*b*合并为组合项集{*a,b,*:3};则候选事务项组*CIArray*{*h*}=Ø,无需递归挖掘,将组合项集与后缀项集合并,支持度计数为*h*的总支持度计数得到{*a,b,h*:3},再判断{*a,b,h*:3}是否为*MFS*的子集,此时*MFS*=Ø,可直接加入*MFS*,事务项*h*挖掘完毕。

表2 事务项*h*的投影二维表

a	b	频繁项名
1	1	
1	1	事务信息区
1	1	
3	3	总统计区

挖掘事务项*g*之前,先判断项集{*a,b,c,d,e,f,g*}是否是*MFS*的子集,此处不是它的子集,则可以继续挖掘。首先将事务项*h*的投影数据库继续投影来完善其他堆栈,投影完成后只有堆栈*g*的投影数据库的数据信息完整,此时根据事务项*g*的投影数据库生成投影二维表如表3所示,此处组合项集为空,故无法使用组合策略,存在*a*、*b*、*c*和*d*的事务项计数为3,等于最小支持度计数,则候选事务项组*CIArray*{*g*}={*a,b,c,d*},从故需要递归生成条件投影数据库和条件投影二维表再进行挖掘。

表3 后缀项集{g}的投影二维表

a	b	c	d	频繁项名称区
1	1	1	1	
1	0	1	0	
1	0	1	0	事务信息区
0	1	0	1	
0	1	0	1	
3	3	3	3	总统计区

现对*CIArray*中最后一项*d*进行挖掘,此时的后缀项集为{*g,d*},条件投影二维表如表4所示,获得组合项集{*b*:3},则候选事务项组*CIArray*{*g,d*}=Ø,无需递归挖掘,直接形成候选最大频繁项集{*d,b,g*:3},经检测该项集不是*MFS*={{*a,b,h*:3}}的子集,因此加入*MFS*。继续投影数据信息。

表4 后缀项集{*g,d*}的条件投影二维表

b	频繁项名称区
1	
1	事务信息区
1	
3	总统计区

再对*CIArray*中的*c*进行挖掘,同理获得*CIArray*{*g,c*}=Ø,无需递归挖掘,经检测获得最大频繁项集{*a,c,g*:3}加入*MFS*。

再对*CIArray*中的*b*进行挖掘,同理获得*CIArry*{*g,b*}= Ø,无需递归挖掘,直接形成候选最大频繁项集{*b,g*:3},经检测该项集是*MFS*的子集,故舍弃。

最后对*CIArray*中的*a*进行挖掘,因条件投影二维表为空,则直接形成候选最大频繁项集{*a,g*:3},经检测{*a,g*:3}是*MFS*的子集,故舍弃。事务项*g*挖掘完毕。

同理挖掘事务项*f*,获得形成候选最大频繁项集{*a,c,d,e,f*:5},经检测可加入*MFS*。

同理挖掘事务项*e*,获得组合项集{*a,c*:7};只有事务项*d*的支持度计数符合要求,故候选事务项组*CIArray*{*e*}={*d*},因只有一项可直接形成候选最大频繁项集{*a,c,d,e*:5},经检测该项集为*MFS*的子集故舍弃。

同理挖掘事务项*d*,因候选事务组包含3个事务项,故需要递归挖掘,挖掘过程不再叙述,获得最大频繁项集{*a,b,c,d*:3},经检测可加入*MFS*。

因项集{*a,b,c*}是*MFS*的子集,故停止挖掘,算法执行完毕,最终得到的*MFS*={{*a,b,h*:3},{*d,b,g*:3},{*a,c,g*:3},{*a,c,d,e,f*:5},{*a,b,c,d*:3}}。

2 伪算法描述

笔者提出了一种基于投影二维表的最大频繁项集挖掘算法PITable-MAX,伪代码如下所述:

初始化:最大频繁项集集合MAX_FI =Φ

输入:最小支持度计数MIN_SUP,D'为预处理

事务数据库

输出：MAX_FI

调用 PITable-MAX (Φ,D', MIN_SUP);

Procedure PITable-MAX(u,d',min_sup) //定义 PITable-MAX 函数

输入：形式参数 1 为 u 表示后缀项集

形式参数 2 为 d' 表示预处理事务数据库或投影二维表

形式参数 3 为最小支持度计数 min_sup

输出：最大的频繁项集集合 max_fi

生成投影数据库，获得堆栈总数为 n

for (i=n;i>=1;i--){ //从的最后一个开始挖掘

$u=x_i$;

基于堆栈 xi 创建投影二维表 PITable{ u };

调用 Mine-MaxFc(u, PITable{u},min_sup); //调用挖掘函数

项集 low= $x_1 \cup x_2 \cup \cdots \cup x_{i-1}$;

If low 是 max_fi 的子集 then Retrun;//根据减挖策略可跳出函数停止挖掘

将堆栈 x_i 中项目根据最后项分配到相应的堆栈并去除最后项 //继续投影

}

Procedure Mine-MaxFc(u,PITable{u},min_sup) //定义 Mine-MaxFc 函数

形式参数 1 为 u 表示后缀项集

形式参数 2 PITable{u} 为 u 的投影二维表

形式参数 3 为最小支持度计数 min_sup

{

获得 PITable{u} 的组合项集 BItemsets{u} 和候选事务组 CIArray{u};

确定候选事务组中事务项个数 m;

if m==0 {

将 u ∪ BItemsets{u}，支持度计数为 BItemsets{u} 的计数;

If u∪BItemsets{u} 不是 MFS 的子集 Then{

加入 MFS;}

}

else if m==1 {

获取 CIArray{u} 中唯一项 only 和支持度计数;

将 only∪u∪BItemsets{u}，支持度计数为唯一项的支持度计数;

if only ∪ u ∪ BItemsets{u} 不是 MFS 的子集 Then{

加入 MFS;}

}

else{

从 PITable{u} 中提取候选事务组包含事务项的数据信息;

u=u∪BItemsets{u};

形成条件预处理事务数据库 d'{u};

递归调用 PITable-MAX(u, d'{u},min_sup);

}

}

3 算法性能测试

采用 PITable-MAX、DMFIA 和 FP-Max 共 3 种不同的算法对相同数据库集（表 5）进行挖掘比较，最终发现 3 种算法挖掘出的最大频繁项集一致，从而验证了 PITable-MAX 的可行性，通过图表的方式显示挖掘的时间效率和空间效率，分析图表验证 PITable-MAX 算法的优越性。实验使用 Pumsb 数据集和 T10I4D100K 数据集作为验证数据，采用 JAVA 语言编写程序进行验证。

表 5　数据集信息

数据集名	事务总数	事务项数	平均长度	实验阈值范围
Pumsb	49 046	2 113	74	0.6% ~ 0.75%
T10I4D100K	100 000	999	10	0.25%- ~ 0.4%

PITable-MAX 算法在 2 种类型的数据集上都具有较好的执行效率执行时间比较情况如图 1 所示，PITable-MAX 算法的时间效率明显优于 FP-Max 和 DMFIA 算法，随着支持度阈值减少，PITable-MAX 算法的优越性越来越明显，另外，内存消耗情况如图 2 所示，PITable-MAX 算法采用了投影数据库的方式进行挖掘，在内存使用上相比较于 FP-MAX 和 DMFIA 算法具有明显的优势，因此 PITable-Max 算法更加优越。

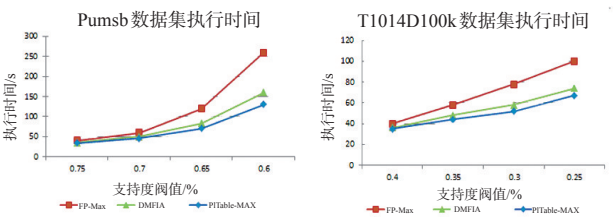

图 1　执行时间比较图

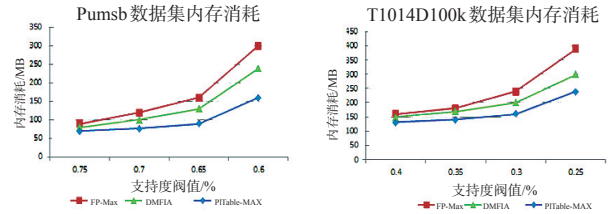

图 2　内存消耗比较图

4 结论与展望

笔者提出的PITable-MAX算法只需要遍历一次事务数据库,并将数据信息采用投影数据库的方式存入磁盘,挖掘到某个事务项时只会提取相关的数据信息,从而减少了内存的损耗,结合减挖策略和组合策略,最终达到减少空间消耗和加快挖掘效率的目的。下一步的工作是进一步优化系统空间的使用,采用更加科学的结构来存储数据信息。

参考文献:

[1] GRAHNE G,ZHU J F. High performance mining of maximal frequent itemsets[C]//Proceedings of the 6th SIAM International Workshop on High Performance,2003:135-144.

[2] 宋余庆,朱玉全,孙志挥,等.基于FP-Tree的最大频繁项目集挖掘及更新算法[J].软件学报,2003,14(9):1586-1592.

[3] BAYARDO R J.Efficiently mining long patterns from databases[C]//ACM SIGMOD Record,ACM,1998:85-93.

[4] BURDICK D,CALIMLIM M,GEHRKE J.Mafia:a maximal frequent itemset algorithm for transactional database[A].In:Stuart Feldman ed,Proceeding of the 17th International Conference on Data Engineering[C].Washington:IEEE Computer Society Press, 2001:443-452.

[5] HAN J,PEI J,YIN Y.Ming frequent patterns without candidate generation[C]//ACM SIGMOD Record,ACM,2000,29(2):1-12.

[6] 马达,王佳强.一种基于压缩FP-树的最大频繁项集挖掘算法[J].长春理工大学学报(自然科学版),2009,32(3):457-461.

[7] 钱雪忠,惠亮.关联规则中基于降维的最大频繁模式挖掘算法[J].计算机应用,2011,31(5):1339-1343.

(责任编辑:蒋召雪)

(上接第19页)

[27] 李丹,李颖.毕节地区马铃薯癌肿病发生流行的生物学因子研究[J].新疆大学学报:自然科学版,2004:147-149.

[28] SALAMAN R N, LESLEY J W.Genetic studies in potatoes;the inheritance of immunity to wart disease[J].Journal of Genetics, 1923,13(2):177-186.

[29] HEHL R,FAURIE E,HESSELBACH J,et al.TMV resistance gene N homologues are linked to Synchytrium endobioticum resistance in potato[J].Theoretical and Applied Genetics,1999,98(3-4):379-386.

[30] BRUGMANS B,HUTTEN R G B,ROOKMAKER A N O,et al.Exploitation of a marker dense linkage map of potato for positional cloning of a wart disease resistance gene[J].Theoretical and applied genetics,2006,112(2):269-277.

[31] GROTH J,SONG Y,KELLERMANN A,et al.Molecular characterisation of resistance against potato wart races 1,2,6 and 18 in a tetraploid population of potato (Solanum tuberosum subsp. tuberosum)[J].Journal of applied genetics,2013,54(2):169-178.

[32] PUTNAM M L,SINDERMANN A B.Eradication of potato wart disease from Maryland[J]. American Journal of Potato Research,1994,71(11):743-747.

[33] SALAVA J.Analysis of ribosomal DNA sequences of Synchytrium endobioticum (Schilberzsky) Percival[J].Acta fytotechnica et zootechnica (online), roč. 7, 2004, č. Mimoriadne č í slo, 2004.

[34] 李丹.毕节地区马铃薯癌肿病疫情监测及治理[J].植物检疫,2005,19(1): 31-33.

[35] 李丹,刘红梅,陆绍炎,等.筛选抗病品种防治内生集壶菌的研究[J].种子,2012,31(6): 112-114.

[36] HILARIO E.The restriction enzyme target approach to genotyping by sequencing (GBS)[J]. Plant Genotyping:Methods and Protocols,2015:271-279.

[37] ROWAN B A,SEYMOUR D K,CHAE E,et al.Methods for Genotyping-by-Sequencing[J]. Genotyping:Methods and Protocols,2017:221-242.

[38] SUMMERS C F,GULLIFORD C M,CARLSON C H,et al.Identification of genetic variation between obligate plant pathogens Pseudoperonospora cubensis and P.humuli using RNA sequencing and genotyping-by-sequencing[J].PloS one,2015,10(11): e0143665.

[39] HANSEN Z R,EVERTS K L,Fry W E,et al.Genetic Variation within Clonal Lineages of Phytophthora infestans Revealed through Genotyping-By-Sequencing, and Implications for Late Blight Epidemiology[J].PloS One,2016,11(11):e0165690.

[40] Potato Genome Sequencing Consortium. Genome sequence and analysis of the tuber crop potato[J].Nature,2011,475(7355): 189-195.

(责任编辑:曲继鹏)