一种改进的基于矩阵的频繁项集挖掘算法

张笑达,徐立臻
(东南大学 计算机系,江苏 南京 211189)

摘 要:如何从海量数据信息中挖掘出有用的关联规则已经成为人们广泛关注的问题,而在关联规则挖掘中,首要的问题就是如何高效地挖掘出频繁项集。针对已有FIMM算法作出改进,提出了一种改进的基于矩阵的频繁项集挖掘算法N-FIMM,该算法在FIMM基础上去除大量冗余的非频繁项集的项集,减少计算可能频繁项集的工作量,同时缩小了矩阵规模,提高了空间效率。通过对矩阵操作,一次性地产生所有的频繁项集。试验结果表明,该算法对已有的基于矩阵的频繁项集挖掘算法有了很大的改进,提高了挖掘效率。

关键词:数据挖掘;频繁项集;FIMM
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2010)04-0093-04

An Advanced Frequent Itemsets Mining Algorithm Based on Matrix

ZHANG Xiao-da, XU Li-zhen
(Computer College, Southeast University, Nanjing 211189, China)

Abstract: How to mine the useful association rule from large numbers of data information has been a main problem concerned widely. While in the mining of association rule, the chief question is how to mine out the frequent itemsets effectively. An advanced frequent itemsets mining algorithm based on FIMM is proposed in this paper. Through deleting unnecessary itemsets, the algorithm greatly demi-nish the workload of getting itemsets as well as minify the scale of matrix. Thus, the efficiency of space utility is largely improved. By operating on the matrix, all the frequent itemsets are generated one-time, it is proved by experiments that this algorithm is much more effective the frequent itemsets mining algorithm based on matrix existing. It will improve the efficiency of data mining.

Key words: data mining; frequent itemset; FIMM

0 引 言

关联规则挖掘是数据挖掘的一个重要方面,其目的是从海量的数据中挖掘出满足用户兴趣的依赖关系,关联规则挖掘的核心是挖掘频繁项集。Rekesh Agrawal等人提出关联规则发现算法后[1],该技术被众多学者广泛研究,出现了许多相关算法:以Apriori算法为代表的采用逐层搜索的迭代方式和不产生候选项集的FP-growth算法[2]。

目前,已有研究人员提出了基于矩阵的频繁项集挖掘算法FIMM(Frequent Itemsets Mining based on Matrix)[3],该算法只需扫描数据库一次便构造出频繁项集矩阵,避免了产生大量候选项集的瓶颈。然而在构造矩阵的过程中,此算法需计算出数据库可能产生的所有项集,因此要进行大量计算,而且构造出的频繁项集矩阵规模异常庞大,算法空间效率低。假设数据库的事务中包含10个项,可能产生的所有项集数就达到2^{10}个,这已经是一个不小的数字!现实中数据库中包含的项目数可能是100、1000,甚至更多,由此产生的所有可能项集数将不可想象,因此数据库所有可能项集的计算量和矩阵规模也将成为FIMM算法的瓶颈。

针对FIMM算法存在的问题,提出了一种改进的基于矩阵的频繁项集挖掘算法(N-FIMM),该算法只需扫描数据库一次就能把事务数据库转化成频繁项集矩阵,在构造频繁项集矩阵的过程中先去除那些必不可能成为频繁项集的项集和频繁1-项集,这样就大大减小了计算项集的工作量,同时缩小了矩阵的规模,然后对项集向量进行累加操作以实现对项集的计数,由此一次性产生所有频繁项集,避免了FIMM算法的瓶颈,有效提高了挖掘效率。

1 N-FIMM算法

1.1 基本概念

设$I=\{i_1,\cdots,i_m\}$是项的集合,事务数据库$D=$

收稿日期:2009-07-14;**修回日期**:2009-11-04
作者简介:张笑达(1986-),女,硕士研究生,研究方向为移动数据库的发展与应用;徐立臻,教授,研究方向为移动数据库。

$\{T_1,\cdots,T_n\}$,其中,每个事务 T 是项的集合,$T\subseteq I$。如果 $X\subseteq T$,则称 X 是个项集。如果 X 中有 k 个元素,则称 X 为 k - 项集。对于一个项集 X,如果其支持度大于等于用户给定的最小支持度阈值 minsup,则 X 为频繁项集。其中,支持度指包含项目集的事务在 D 中所占的百分比。

关联规则就是类似于 $X\Rightarrow Y$ 的一些蕴含式,其中 $X\subseteq T$,$Y\subseteq T$ 且 $X\cap Y=\varnothing$。规则 $X\Rightarrow Y$ 的支持度就是指事务数据库 D 中同时包含 X 和 Y 的事务的数量与所有事务数量的比率。关联规则挖掘问题就是通过最小支持度和最小信任度在一个事务数据库中寻找强关联规则的过程,划分为2个子问题:

(1) 发现频繁项目集;

(2) 在频繁项目集中生成关联规则。

关联规则挖掘的核心,也就是文中重点研究的问题就是如何快速而高效地挖掘出事务数据库中的所有频繁项目集。

定义1 设长度为零的项集为空项集,记为 $\varnothing$。

定义2 已知项的集合 $I=\{i_1,\cdots,i_m\}$,事务数据库 D,则 D 中的一个事务 T 产生的所有项集的集合称为 T 的项集集合,记为 $IS(T)$,每个项集集合中都包含空项集,则一个事务 T 共可产生 $2^{|T|}$ 个项集。

由此可知,包含 m 个项的事务数据库产生的所有可能的项集的个数为 2^m。因为事务数据库中有 2^m 个不同的项集,所以有必要把这些项集按照一定的方式排列从而得出哪些为满足条件的频繁项集。通过采用循环迭代增加项作为模式后缀的方式排列所有项集,这种排列方法更便于理解和计算[4]。

定义3 设事务数据库 D 中的项集为 $I=\{i_1,\cdots,i_m\}$,令 P_0 表示空项集 $\varnothing$,项集的排列可递归定义为

$$\begin{cases}P_0=\varnothing\\ P_k=P_{2^p-(k\bmod 2^p)-1}\oplus i_p\end{cases}\tag{1}$$

其中,$p=\lfloor \mathrm{lb}k\rfloor+1$,$0<k<2^m$,$\oplus$ 表示连接符号。

定理1 设事务数据库 D 中的项集为 $I=\{i_1,i_2,\cdots,i_m\}$,所有项集按定义3的方式排列,则项集 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 可以通过增加项 i_p 为项集 $p_0,p_1,\cdots,p_{2^{p-1}-1}$ 的后缀得到,其中 $p=1,2,\cdots,m$。

证明:根据式(1)可知,项集 P_k 由 $p_{2^p-(k\bmod 2^p)-1}$ 与项 i_p 连接构成,如果给定 p 的值(从1变化到 m),则能够得到 k 的值为 $2^p-1,2^p-2,\cdots,2^{p-1}$,令 $k=2^p-1$,则 $P_k=p_{2^p-1}=p_{2^p-((2^p-1)\bmod 2^p)-1}\oplus i_p=p_0\oplus i_p$,类似的 $p_{2^p-2},\cdots,p_{2^{p-1}}$ 可以由 i_p 连接到项集 $p_1,\cdots,p_{2^{p-1}-1}$ 的后缀得到。

一个事务 T 产生的项集与事务数据库 D 产生的所有项集有着密切的联系。

定理2 设事务数据库 D 中的项集为 $I=\{i_1,\cdots,i_m\}$,其中事务 T 产生的项集集合为 $IS(T)$,如果项 i_p 不包含于事务 T,则项集 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 不包含于 $IS(T)$,否则 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 是否包含于 $IS(T)$ 分别依赖于项集 $p_0,p_1,\cdots,p_{2^{p-1}-1}$ 是否包含于 $IS(T)$,其中 $p=1,2,\cdots,m$。

证明:该定理用于判断项集 P_k 是否在事务 T 的项集集合 $IS(T)$ 中。先证明 i_p 不在事务 T 中的情况。根据定义2,$IS(T)$ 中的每个项集都是由 T 中的项组成的,又根据定理1,如果给定 p 值,则可以得到由 i_p 组成的项集 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$,因此若 i_p 不在事务 T 中,则项集 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 不在 $IS(T)$ 中。再证明 i_p 在事务 T 中的情况。项集 P_k 是否在 $IS(T)$ 中与项集 $p_{2^p-(k\bmod 2^p)-1}$ 是否在 $IS(T)$ 中有关,根据上面证明 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 由 i_p 连接到 $p_0,p_1,\cdots,p_{2^{p-1}-1}$ 的后缀得到。因此,如果在 i_p 事务 T 中,则项集 $p_{2^p-1},p_{2^p-2},\cdots,p_{2^{p-1}}$ 是否在 $IS(T)$ 中依赖于项集 $p_0,p_1,\cdots,p_{2^{p-1}-1}$ 是否在 $IS(T)$ 中。

引理1 设事务数据库 D 中的项集为 $I=\{i_1,\cdots,i_m\}$,若 $i_p\notin L_1$(L_1 为频繁1-项集,$p=1,2,\cdots,m$),那么根据定理1,由 i_p 组成的项集 P_k($k=2^p-1,2^p-2,\cdots,2^{p-1}$)一定不是频繁项集[5]。

证明:假设某一 $i_p\notin L_1$,$p=1,2,\cdots,m$,但由 i_p 组成的那些项集中有项集 P_k 为频繁项集。根据频繁项集的定义,包含 P_k 中每个项的事务数必大于等于最小支持数阈值 minss,因为 i_p 是 P_k 中的项,所以包含 i_p 的事务数也大于等于最小支持数阈值 minss,然而这与 $i_p\notin L_1$ 矛盾,所以引理1成立。

定理3 设事务数据库 D 中项集集合为 p_0,p_1,p_{2^m-1},根据定理1,若项集 P_k 为非频繁项集,则由 P_k 构成的那些项集也必为非频繁项集。

证明:反证法。假设项集频繁项集 P_l 由非频繁项集 P_k 构成,根据频繁项集的定义,包含 P_l 的事务数必大于等于最小支持数阈值 minss,因为 P_k 是 P_l 中一部分,所以包含 P_k 的事务数也大于等于最小支持数阈值 minss,然而这与 P_k 是非频繁项集矛盾,所以定理3成立。

1.2 算法描述

N-FIMM算法的基本思想是:构造一个频繁项集矩阵,行用来存储所有的事务,而矩阵列的规模是不确定的。前 $|L_1|$ 列按照字母表顺序存储所有频繁项。然后根据式(1)、引理1和定理3,先去除一定不为频繁

项集的项集。同时,由于频繁 1 - 项集即为前 $|L_1|$ 列中的频繁项,没有必要重复列出,故也去除掉,得到所有 $2^m - 1$ 个项集中剩余的那些项集(不包括空项集 $\varnothing$),记为 P_r(设 P_r 中有 r 个项集)。后 r 列根据式(1)中项集的排列方式存储 P_r 中的所有项集。若某项属于某事务则置交叉位为 1,否则为 0;若某项集包含于某事务产生的项集,则置交叉位为 1,否则为 0。矩阵最后增加一行用于累加存储项集的列中 1 的个数,计数值大于等于最小支持度阈值的项集为频繁项集。

矩阵 $M[n+1, |L_1|+r]$ 形如

	i_{j_1}	i_{j_2}	…	$i_{j_{|L_1|}}$	p_{l_1}	p_{l_2}	p_{l_3}	…	p_{l_r}
T_1	1	1	…	0	1	1	1	…	0
T_2	1	0	…	1	0	1	0	…	1
…	…	…	…	…	…	…	…	…	…
T_n	0	1	…	0	1	1	1	…	0
sum	…	…	…	…	3	7	9	…	5

算法　改进的基于矩阵的频繁项集挖掘算法(N - FIMM)

输入 事务数据库 D,最小支持数 minss

输出 所有频繁项集

扫描 D,把所有事务置为矩阵的行,把 L_1 中的项置矩阵的前 $|L_1|$ 列,把根据引理 1 和定理 3 求出的 P_r 中的项置矩阵的后 r 列,形成频繁项集矩阵 $M[n+1, |L_1|+r]$。

```
for(i=1; i<=n; i++){
for(j=1; j<=|L1|;j++)
if(项 ij in 事务 Ti)then
Mij=1;
else
Mij=0;
//根据定理 2,对矩阵中的每个项集 Pk,判断其是否在事务 T 产生的项集集合中
for(k=1;k<=r; k++)
if(项集 pl1 in 事务 Ti 的项集集合 IS(Ti)then
Mi(k+|L1|)=1;
else
Mi(k+|L1|)=0;
}
//累加各项集的计数,计数值大于等于最小支持数阈值的项集为频繁项集
for(i=1; i<=n; i++){
for(j=|L1|+1; j<=|L1|;j++)
if(Mij等于 1)then
sum[n+1][j]++ ;
}
for(k=|L1|+1;k<=|L1|+r;k++)
if(sum[n+1][k]>=minss)then
p1k-|L1| 是频繁项集;
```

2　实例分析

设事务数据库 $D = \{T_1, \cdots, T_{10}\}$ 如表 1 所示,项集为 $I = \{a,b,c,d,e,f\}$,最小支持度阈值 min_sup 为 40%,依据 N - FIMM 算法计算数据库 D 的所有频繁项集。

表 1　事务数据库 D

TID	项数表
T_1	*abcf*
T_2	*abcef*
T_3	*bcef*
T_4	*bde*
T_5	*acef*
T_6	*bef*
T_7	*abcef*
T_8	*abdf*
T_9	*bcdef*
T_{10}	*bcef*

(1):根据频繁项集的定义,很容易得出频繁 1 - 项集 L_1 为 a,b,c,e,f。

(2):根据(1)式、引理 1、定理 3,去除必不为频繁项集的项集 $p_2, p_{23}, p_{40}, \cdots, p_{55}$,同时去除频繁 1 - 项集 $p_1, p_3, p_7, p_{31}, p_{63}$,计算出项集集合 Ps_r 中的各个项集(此处不一一列出)。

(3):构造出频繁矩阵 $M[11,5+26]$,并计算矩阵中各项集的总计数。

(4):得到数据库 D 中所有频繁项集:

L_1 为 a,b,c,e,f;

L_2 为 $\{a,b\}, \{b,c\}, \{a,c\}, \{c,e\}, \{b,e\}, \{e,f\}, \{c,f\}, \{b,f\}, \{a,f\}$;

L_3 为 $\{b,c,e\}, \{b,e,f\}, \{c,e,f\}, \{a,c,f\}, \{b,c,f\}, \{a,b,f\}$;

L_4 为 $\{b,c,e,f\}$。

3　算法分析

N - FIMM 算法是根据矩阵理论提出的,应用集合论,令矩阵的行存储所有的事务,前 $|L_1|$ 列按照字母表顺序存储所有频繁项,在去除了必不为频繁项集的项集后,得到项集集合 p_r,后 r 列存储 p_r 中所有项集,所以该算法在理论上是正确的。而且由于矩阵存储的主要数据类型是布尔值,可以按位方式进行存储,在计算上是可行的。

该算法在构造矩阵的过程中,去除了必不为频繁

项集的项集和频繁 1 - 项集，大大减少了计算数据库可能产生的所有项集的工作量，同时也大大缩小了矩阵的规模，节约了存储空间，相对于 FIMM 算法在时间和空间复杂度上有了很大的改进。

挖掘频繁项集算法的效率取决于扫描事务数据库的次数和产生可能项集的迭代次数，传统的 FIMM 算法[6]存在以下缺点：

①在生成所有可能项集的过程中需进行大量的计算，庞大的项集数量成为算法的瓶颈。

②矩阵规模过大，空间效率低。

而 N - FIMM 算法只需要扫描事务数据库一次，整个过程分为 2 个阶段：

(1)扫描事务数据库一次，去除不必要的项集后，把事务数据库转化为矩阵并为矩阵赋值，为矩阵赋值的计算次数为 $n(|L_1|+r)$。

(2) 累计矩阵后 r 列的值并求频繁项集，累加操作的次数为 $n \times r$，所以 N - FIMM 算法的时间复杂度为 $O(n \times r)$。

由于 N - FIMM 算法在极大程度上减少了访问数据库的次数，同时去除了不必要的项集，避免了求所有可能项集的瓶颈，缩小了矩阵规模，这些使得 N - FIMM 算法较之传统的 FIMM 算法节省了大量的时间和空间，由此一次性产生所有频繁项集，有效提高了挖掘效率。

4 试验结果

为了验证该算法性能，以 FIMM 算法作为试验对比算法进行性能分析。试验环境：CPU Intel(R) 1.80GHz，512 MB 内存和 Windows XP 操作系统，测试数据库为 SQL Server 2000 所带的 Foodmark 数据库，样本数据库包括 10 项，50 项，100 项，200 项，1000 项 5 中情况，试验结果如图 1 所示。

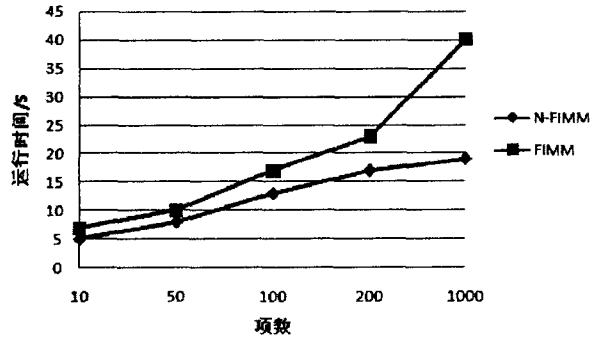

图 1 随项数变化的运行时间

图 1 给出了两种算法在固定最小支持度下随数据库中项数增加的变化结果，两种算法花费的时间随项数的增加成上升趋势，但 N - FIMM 算法上升的非常慢。图中纵坐标亦可代表空间消耗，表示算法随项数的增加空间花费成上升趋势，但 N - FIMM 算法上升的非常慢。

5 结束语

文中在已有的 FIMM 算法[7]基础上提出了改进的基于矩阵的频繁项集挖掘算法 N - FIMM，它只需扫描事务数据库一次，把事务数据库转化为频繁项集矩阵，在构造矩阵的过程中避免了不必要的项集计算，从而大大减小了矩阵的规模。在对矩阵挖掘频繁项集时，无需进行连接和剪枝操作，而是根据每列的项集计数过滤掉小于支持度阈值的项集，进而发现所有的频繁项集，当支持度阈值发生变化时，不用重新扫描数据库和重新构造矩阵，就能很容易地挖掘频繁项集，因此，该算法也适合增量挖掘频繁项集。

有了数据库的所有频繁项集，就可以进一步根据关联规则形成算法得出满足用户兴趣度的有用的关联规则，并依据关联规则作出相应的决策[8]。所以频繁项集挖掘是对数据库进行关联规则挖掘的关键部分。笔者的下一步工作是研究根据已有频繁项集生成关联规则的算法。

参考文献：

[1] Agrawal R, Imielinski T, Swami A. Mining Association Rules Between Sets of Items in Large Databases [C]//Proc. of ACMSIGMOD Int'1 Conf. on Management of Data. Washington D. C. USA: [s. n.], 1993.

[2] Han Jiawei, Pei Jian, Yin Yiwei. Mining Frequent Patterns Without Candidate Generation [C]//Proc. of the 2000 ACM - SIGMOD Int'1 Conf. on Management of Data. Dallas, TX, USA: [s. n], 2000.

[3] 张忠平，李 岩，杨 静. 基于矩阵的频繁项集挖掘算法[J]. 计算机工程，2009，35(1)：84 - 86.

[4] 张忠平，郑为夷. 基于事务树的最大频繁项集挖掘算法[J]. 计算机工程，2009，35(15)：97 - 100.

[5] 陈 晨，鞠时光. 基于改进 FP - tree 的最大频繁项集挖掘算法[J]. 计算机工程与设计，2009，30(24)：6236 - 6239.

[6] 江 晗，贾 洄，徐 峰. 基于频繁项集挖掘最大频繁项集和频繁闭项集[J]. 计算机工程与应用，2008，44(28)：146 - 148.

[7] Wu Fan. A New Approach to Mine Frequent Patterns Using Item - transformation Methods [J]. Information Systems, 2007, 32(7): 1056 - 1072.

[8] 董祥和，仲丛友，董荣和. 有趣 Web 日志关联规则挖掘算法[J]. 计算机工程与设计，2009，30(4)：1036 - 1038.