doi：10.11830/ISSN.1000-5013.201510043

事务约简和 2 项集支持度矩阵快速剪枝的 Apriori 改进算法

张健，刘韶涛

（华侨大学 计算机科学与技术学院，福建 厦门 361021）

摘要： 在 Apriori 算法的改进算法 M-Apriori 基础上，为了进一步减少不必要的数据库扫描，引入事务约简技术，提出一种改进的 MR-Apriori 算法. 考虑到 M-Apriori 算法会产生大量候选项集，为了实现对候选项集快速剪枝，加入一个自定义的 2 项集支持度矩阵，提出第 2 种改进的 MP-Apriori 算法. 将事务约简和 2 项集矩阵快速剪枝一起引入到 M-Apriori 算法中，提出第 3 种改进的 MRP-Apriori 算法. 最后，在 mushroom 数据集上进行实验. 结果表明：加入事务约简的 MR-Apriori 算法和加入 2 项集矩阵快速剪枝的 MP-Apriori 算法，运行时间相比原 M-Apriori 算法都有较大缩减，而同时结合两种优化策略的 MRP-Apriori 算法运行时间最短，验证了这两种优化策略的有效性.

关键词： 关联规则；Apriori 算法；频繁项集；支持度矩阵

中图分类号： TP 311 **文献标志码：** A **文章编号：** 1000-5013(2017)05-0727-05

Improved Apriori Algorithm for Quickly Prune by Combining Transaction Reduction With Two-Item Set Support Matrix

ZHANG Jian, LIU Shaotao

(College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China)

Abstract: Based on the M-Apriori algprithm, an improved version of the Apriori algorithm, a transaction reduction technique is introduced and an improved algorithm, MR-Apriori, is proposed in the paper in order to further reduce unnessary database scans; Meanwhile, considering that the M-Apriori algorithm generates large amount of candidate itemsets during the running process, so as to quickly prune the candidate itemsets, a self-defined two-item set support matrix is added and a second improved algorithm, MP-Aproiri, is proposed in the paper. Then transaction reduction, accompanied by two-item set support matrix which is used to quickly prune the candidate itemsets, are combined together and a third improved algorithm, MRP-Aproiri, is proposed in the paper. Finally, an experiment is conducted on the mushroom dataset, the result shows that the MR-Apriori algorithm which uses the transaction reduction and the MP-Apriori algorithm which uses the two-item set support matrix that can quickly prune the candidate itemsets, is much faster than the M-Apriori algorithm, and the MRP-Apriori algotirhm which combines these two optimization strategies together gets the shortest time, therefore, it proves that these two optimization strategies are efficient.

Keywords: association rule; Apriori algorithm; frequent itemset; support matrix

收稿日期： 2015-10-21

通信作者： 刘韶涛(1969-)，男，副教授，主要从事软件体系结构与软件复用的研究. E-mail：shaotaol@hqu.edu.cn.

基金项目： 福建省科技计划重大项目(2011H6016)

关联规则挖掘是数据挖掘任务中一个重要的研究方面,旨在挖掘出数据库中潜在的关联关系[1-8]. Apriori 算法[1]是关联规则挖掘中最经典的算法,该算法基于"产生测试"框架,采用逐层迭代的方法得到频繁项集. Apriori 算法思想简单,但也存在算法效率低的缺点. 针对算法需要频繁扫描数据库的缺点,Al-Maolegi 等[9]提出 M-Apriori 算法. M-Apriori 算法采用了一种新的改进思路,大大减少了扫描数据库次数. 但是,这种改进虽然减少了数据库扫描次数,但却存在很多不必要的扫描. Singh 等[10]则是通过事务约简改进 Apriori 算法. 纪怀猛[11]提出频繁 2 项集支持矩阵对候选项集快速剪枝的方法,对 M-Apriori 算法进行优化,优化后的算法命名为 MP-Apriori 算法. 本文结合这两个优化算法,提出 MRP-Apriori 算法.

1 Apriori 算法及 M-Apriori 算法

1.1 Apriori 算法及其改进

Apriori 算法[1]采用宽度优先搜索的策略,算法有以下 2 个步骤.

步骤 1 扫描数据库,计算得到 1 项集的支持度,删去不满足最小支持度的项集,得到频繁 1 项集的集合.

步骤 2 由频繁 1 项集得到频繁 2 项集,频繁 2 项集得到频繁 3 项集,如此循环,直到不能找到频繁项集为止.

步骤 2 分为连接和剪枝. 连接是频繁 $k-1$ 项集 L_{k-1} 与自身进行连接,条件是两者前 $k-2$ 个项都相同(称为可连接的),最后一个元素不同,连接得到的结果是两者前 $k-2$ 个项加上按字典顺序排列的两者的最后一个元素,即得到候选 k 项集 C_k. 剪枝运用 Apriori 性质[1],即频繁项集的所有非空子集也都是频繁的,对候选 k 项集 C_k 进行剪枝,剪枝后得到频繁 k 项集 L_k.

Apriori 算法简单,但是它仍比较低效. 原因主要有以下 3 个方面.

1) 需要多次扫描数据库.

2) 产生大量中间候选项集.

3) 候选项集求支持度时,需要和各条事务进行模式匹配,比较费时.

针对 Apriori 算法的这 3 点不足,国内外学者从各个方面来改进它的效率. DHP 算法[2]引入 Hash 技术来减少候选 2 项集的生成;Partition 算法[3]采用划分的办法,把数据库划分为若干个子库,在子库上求出局部频繁项集,最后再汇总求出全局频繁项集;Samping 算法[4]随机选择一部分数据库样本,用这部分数据上的频繁项集代表全局频繁项集;DIC 算法[5]在扫描的不同点添加候选项集,从而可以动态对项集进行评估,进一步减少数据库扫描次数. 陈江平等[6]引入概率的方法对 Apriori 算法进行改进;黄建明等[7]将数据库转换为十字链表的方式存储,使得扫描数据库的次数减少到了 1 次;刘维晓等[8]在 Apriori 中加入用户兴趣项进行改进,从而大范围缩减数据库容量.

1.2 M-Apriori 算法

Apriori 算法中,为了产生频繁 k 项集,需要保持大量候选项集,特别是当支持度很小的时候. 因此,为了减少在扫描数据库确定频繁项集上花费的时间,M-Apriori 算法[9]提出了一种新的改进的思路. 在第 1 次扫描数据库时,M-Apriori 算法保存频繁 1 项集 L_1 中的每个项、对应的支持度及每个项所出现的事务 ID 号的集合. 在计算 k 项集支持度时,M-Apriori 算法先将 k 项集划分为 k 个 1 项集;接着,根据频繁 1 项集 L_1 比较这 k 个 1 项集的支持度大小;最后,选择从其中支持度最小的事务 ID 集合所对应的事务中扫描计算此 k 项集的支持度,从而大大减少扫描数据库次数. 改进后 MR-Apriori 算法示例,如图 1 所示.

图 1 中:D 为原始数据库;L_1 为 M-Apriori 算法得到频繁 1 项集结果;实线为删去事务中单个项;虚线为删除整条事务. 要产生频繁 2 项集$\{I_1, I_2\}$,先比较 I_1 和 I_2 的支持度的大小,I_1 的支持度为 5,小于 I_2 的支持度 7,扫描从 I_1 所对应的事务 ID 集合$\{T_1, T_3, T_7, T_9, T_{10}\}$所对应的事务 $T_1, T_3, T_7, T_9, T_{10}$,计算$\{I_1, I_2\}$的支持度,这样本来需要扫描整个数据库,现在只用扫描其中的 5 条,减少了扫描次数. 同理,要产生频繁 3 项集$\{I_1, I_2, I_3\}$时,比较 I_1, I_2 和 I_3 的支持度,扫描从支持度最小的即 I_1 所对应

的事务 ID 集合$\{T_1,T_3,T_7,T_9,T_{10}\}$所对应的事务 T_1,T_3,T_7,T_9,T_{10}，计算$\{I_1,I_2,I_3\}$的支持度.

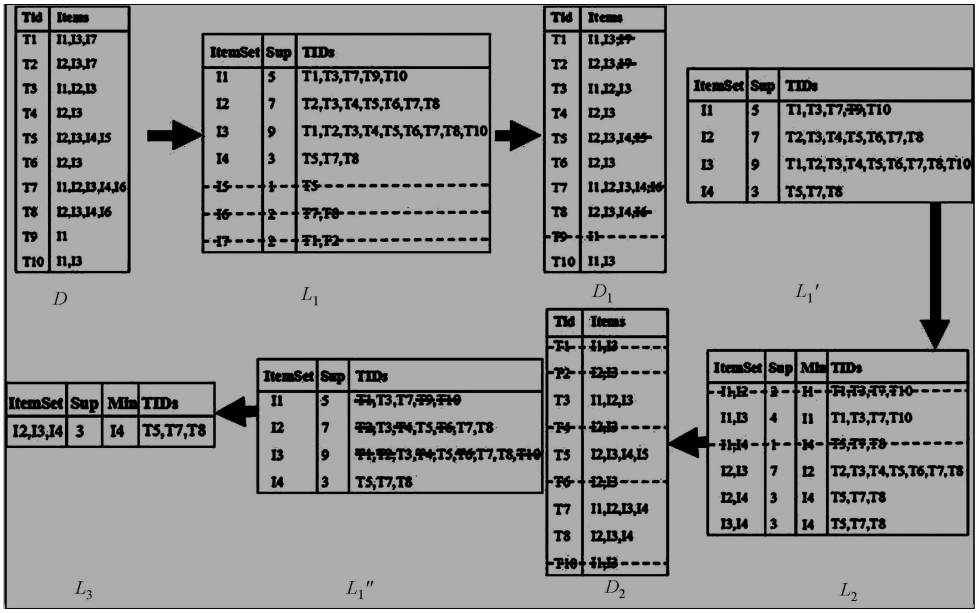

图 1　改进后 MR-Apriori 算法示例

Fig. 1　Example of improved MR-Apriori algorithm

2　M-Apriori 算法的优化

2.1　MR-Apriori 算法

文献[10]为了改进 Apriori 算法，用两点优化(性质 1,2)：从数据库中删除某个值；删除某条事务.

性质 1　当扫描第 $k(k\geqslant 2)$次时，从数据库中删除在频繁 $k-1$ 项集 L_{k-1}，但不删除在频繁 k 项集 L_k 的项.

性质 2　如果扫描第 $k(k\geqslant 2)$次时，某事务项目数小于 k，则可以将其从数据库中删除.

需要注意这两点优化是在第 $k(k\geqslant 2)$次扫描时结合在一起作用的，且有先后顺序，先用性质 1 删除单个项，再判断修改后的数据库中各项事务长度是否小于 $k(k\geqslant 2)$，删除小于 $k(k\geqslant 2)$的事务.

将这两点优化加入到 M-Apriori 中，提出 MR-Apriori 算法. MR-Apriori 算法在 M-Apriori 算法的基础上，加入事务约简技术. 在 M-Apriori 算法扫描数据库前，分别判断性质 1,2 是否成立，从而对数据库进行约简. 然后，在约简后的数据库上执行 M-Apriori 算法的后续步骤. 为了计算 k 项集支持度，根据频繁 1 项集 L_1 比较 k 个 1 项集的支持度大小，扫描支持度最小的事务 ID 集合所对应的事务中，计算 k 项集的支持度. 保存频繁 1 项集 L_1 中的每个项，每个项对应的支持度及每个项所出现的事务 ID 号的集合，由于 MR-Apriori 算法对事务进行了约简，相应的 L_1 的事务 ID 号集合这一项也要相应修改.

产生频繁 2 项集 L_2. 假设产生$\{I_1,I_2\}$，在原 M-Apriori 中，比较频繁 1 项集中 I_1 和 I_2 的支持度大小. 然后，扫描支持度小的对应的事务 ID 集合中的事务. 在改进的算法中，做数据库的缩减，先根据性质 1 删除不在 L_1 中的项，即从数据库 D 中删除 I_5、I_6、I_7. 由于 $k=1$，根据性质 2，删除事务长度小于 1 的集合，即从数据库 D 删去事务 T_9. 同时，相应也从 L_1 中删除 T_9，得到修改后的数据库 D_1 和修改后的 L'_1(图 1).

根据修改后的频繁 1 项集 L'_1 比较 I_1 和 I_2 对应的支持度大小，由于 I_1 支持度小，故只需扫描 T_1，T_3，T_7，T_{10}(L'_1 中对应的)查找 2 项集$\{I_1,I_2\}$，最终生成频繁 2 项集 L_2.

产生频繁 3 项集. 此时 $k=3$，由于性质 1 不成立，性质 2 成立，根据性质 2 从数据库 D_1 中删除事务长度小于 3 的事务，即从 D_1 中删去 T_1，T_2，T_4，T_6，T_{10}. 同时，也从修改后频繁 1 项集 L'_1 中删去这些事务，得到修改后的数据库 D_2 和修改后的频繁 1 项集 L''_1. 然后，重复上面过程，删除非频繁项得到频繁 3 项集 C_3.

2.2 MP-Apriori 算法

Apriori 算法产生大量候选集,尤其是候选 2 项集. 候选集需要剪枝步骤才能生成频繁项集,M-Apriori 算法采用的还是 Apriori 算法的剪枝原理,即 $\forall c \in C_k$,判断 c 的 k 个$(k-1)$-子集是否都在 L_{k-1} 中,若找到一个$(k-1)$-子集不在 L_{k-1} 中就淘汰 c. 因为这个过程会多次扫描 L_{k-1},特别是当生成 C_k 很大时,算法的效率并不理想[12]. 优化 1 虽然约简数据库,也能一定程度上由于数据库的减少而使生成的候选集数目减少,但从本质上说仍是基于传统 Apriori 算法的剪枝原理产生候选集,并没有充分改进候选集的剪枝过程.

定义 matrix[MaxItemId][MaxItemId]的 2 项集支持度矩阵是一个三角矩阵(全部元素位于次对角线上方),其中,MaxItemId 为数据库中所有项的最大值,MaxItemId 为 7,矩阵初始元素全部为 0. MP-Apriori 算法有如下 3 个步骤.

步骤 1 扫描数据库,构造 2 项集的支持度矩阵. 以数据库中的项作为矩阵相应的行标和列标,如果扫描到一条事务中包含有$\{I_i, I_j\}$2 项集,则对矩阵进行一次填充.

矩阵元素填充规则为:如果行下标 i 小于列下标 j,则 matrix[MaxItemId$-j$][i]$+=1$;否则,matrix[MaxItemId -1][j]$+=1$. 填充后 2 项集支持度矩阵,如图 2 所示.

0	1	1	2	0	0	0
0	1	2	2	2	0	
0	0	1	1	1		
0	1	3	3			
0	4	7				
0	2					
0						

图 2 填充后 2 项集支持度矩阵

Fig. 2 Two item support matrix after being filled

步骤 2 产生频繁 2 项集,由于第一步已经用矩阵保存了 2 项集和其对应的支持度,因此,只需要连接频繁 1 项集 L_1 和频繁 1 项集 L_1. 然后,从矩阵中得到连接后得到 2 项集的支持度,若支持度大于等于最小支持度,则加入到频繁 2 项集中. 获取元素值的方法为:如果行下标 i 小于列下标 j,则对应元素值为 matrix[maxItemId $-j$][i]位置的值;否则,为 matrix[maxItemId-1][j]位置的值.

步骤 3 产生频繁 $k(k \geqslant 3)$项集,和 M-Apriori 算法中步骤基本一致,唯一不同的就是在算法的剪枝步. 在 M-Apriori 算法中的剪枝步骤,要不断扫描数据库,确定候选 k 项集 C_k 的每一个$(k-1)$项非空真子集是否都是频繁的,从而确定频繁 k 项集 L_k,而为了运用 2 项集支持度矩阵对 C_k 进行快速剪枝,在原剪枝前面加入了一个预判断. 方法是在进行原 Apriori 剪枝前,先将要进行连接的两个项集的各自最后一项进行连接得到二项集;然后,从矩阵中得到该二项集的支持度. 如果小于最小支持度,可以把两个连接的候选项集的连接项剪掉. 通过这样的预先判断,只有矩阵元素的值大于等于最小支持度时,才需要检查此连接项的所有$(k-1)$项子集是否都是频繁的,从而可以大大提高算法效率.

2.3 MRP-Apriori 算法

优化 1,2 从两个不同的方向对 M-Apriori 算法进行优化改进,MRP-Apriori 算法将优化 1,2 综合到一起加入到 M-Apriori 算法中.

MRP-Apriori 算法前面步骤和 MR-Apriori 算法相同,得到修改后的数据库 D_1 和修改后的频繁项目集 L'_1. 不同的是,在 MR-Apriori 算法中加入了 2 项集支持度矩阵优化,扫描数据库,填充矩阵元素. 对数据库进行修改,生成 2 项集支持度矩阵是以修改后的数据库 D_1 为依据的. 后面的算法步骤和优化 1 算法 MR-Apriori 相同,唯一区别的是,加入 2 项集支持度矩阵进行快速剪枝,在算法的剪枝步之前加入了一个预判断,即在后面频繁 $k-1$ 项集连接生成候选 $k(k>2)$项集时,先将要进行连接的两个项集的各自最后一项进行连接得到二项集,然后,从矩阵中得到该二项集的支持度. 判断连接后的 2 项集支持度和最小支持度的关系,若小于,则直接预先剪掉这个 k 项集.

3 实验结果与分析

实验平台为 Intel(R) Core(TM) i5-3470,主频为 3.20 GHz;内存为 8 GB,Windows 7 旗舰版 64 位 SP1. 采用的编程语言为 Java,开发环境为 Eclipse 4.4.0. 实验数据集为 fimi 网站([http://fimi.ua.ac.be/])上的蘑菇数据集,它共有 8 124 条事务,119 个属性,事务平均长度为 23 项. 不同支持度下的运行时间对比,如图 3 所示. 图 3 中:t 为运行时间;$\eta_{\min}$ 为最小支持度. 由图 3 可知:MR-Apriori 算法由

于对在 M-Apriori 算法的基础上加入事务约简，使得运行效率较 M-Apriori 算法有所提高. MP-Apriori 算法引进了 2 项集支持度矩阵，虽然需要消耗一定的存储空间，但是它的效率提高比较明显，比较显著地提高算法的效率. MRP-Apriori 算法则综合了以上两个算法的优点，因此，它的效率是最高的.

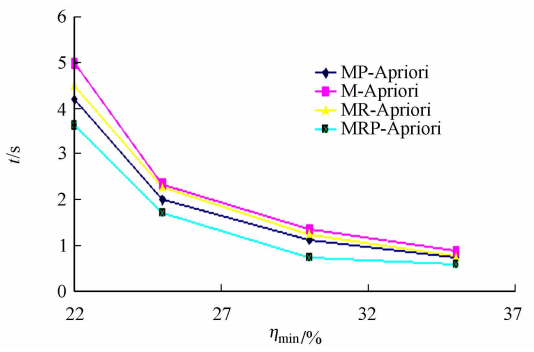

图 3　不同支持度下的运行时间对比

Fig. 3　Comparation of running time in different support

4　结论

在改进的 M-Apriori 算法，加入了事务约简优化，提出 MR-Apriori 算法，在算法计算候选项集支持度时减少了原数据库中记录的数目，从而进一步减少了数据库的次数，提高算法的效率. 扫描加入了 2 项集支持度矩阵快速剪枝优化，能快速对候选项目集进行快速剪枝，而不用像原算法那样需要检查候选 k 项集的所有$(k-1)$项子集是否都是频繁的，从而提高了效率. 最后，结合前两点优化，把提高效率的这两方面结合到一起，提出了 MRP-Apriori 算法，再用实验验证了这 3 种算法的效率.

然而，第 1 点优化会对数据库进行修改，需要花费精力对数据库进行维护，降低算法效率. 下一步将考虑使用其他办法减少数据库扫描次数，或在编写代码时，设置标志位，跳过这些需要被删除的事务，而不是直接将其删去，从而提高效率. 第 2 点优化中，当数据库事务平均长度很大时，矩阵会迅速增大，还可能会存在大量的零元素，从而会对算法效率有所影响. 下一步将对此问题的改进方法进行研究.

参考文献：

[1] AGRAWAL R，SRIKANT R. Fast algorithms for mining association rules[C]// Proc 20th Int Conf Very Large Data Bases. Santiago：VLDB，1994：487-499.

[2] PARK J S，CHEN M S，YU P S. An effective hash-based algorithm for mining association rules[J]. ACM，1995：175-186.

[3] SAVASERE A，OMIECINSKI E R，NAVATHE S B. An efficient algorithm for mining association rules in large databases[C]// International Conference on Very Large Data Bases. [S. l.]：Morgan Kaufmann Publishers Inc，1995：432-444.

[4] TOIVONEN H. Sampling large databases for association rules[C]// Proceedings of the 22nd VLDB Conference. Mumbai：[s. n.]，1996：134-145.

[5] BRIN S，MOTWANI R，ULLMAN J D，*et al*. Dynamic itemset counting and implication rules for market basket data [J]. ACM SIGMOD Record，1997，26(2)：255-264.

[6] 陈江平，傅仲良，徐志红. 一种 Apriori 的改进算法[J]. 武汉大学学报(信息科学版)，2003，28(1)：94-99.

[7] 黄建明，赵文静，王星星. 基于十字链表的 Apriori 改进算法[J]. 计算机工程，2009，35(2)：37-38，40.

[8] 刘维晓，陈俊丽，屈世富，等. 一种改进的 Apriori 算法[J]. 计算机工程与应用，2011，47(11)：149-159.

[9] AL-MAOLEGI M，ARKOK B. An improved apriori algorithm for association rules[J]. International Journal on Natural Language Computing，2014，3(1)：21-29.

[10] SINGH J，RAM H，SODHI D J S. Improving efficiency of apriori algorithm using transaction reduction[J]. International Journal of Scientific and Research Publications，2013，3(1)：1-4.

[11] 纪怀猛. 基于频繁 2 项集支持矩阵的 Apriori 改进算法[J]. 计算机工程，2013，39(11)：183-186.

[12] 胡吉明，鲜学丰. 挖掘关联规则中 Apriori 算法的研究与改进[J]. 计算机技术与发展，2006，16(4)：99-101.

(责任编辑：陈志贤　　英文审校：吴逢铁)