文章编号:1001 - 9081(2006)11 - 2654 - 03

一种有效的基于图的关联规则挖掘算法

陈 明[1,2],史忠植[2],王文杰[1]
(1.1 中国科学院研究生院 信息科学与工程学院,北京 100049;
2. 中国科学院计算技术研究所 智能信息处理重点实验室,北京 100080)
(chenm@ics.ict.ac.cn)

摘 要:基于图的关联规则挖掘算法是一种通过构建关联图并直接生成候选频繁项集,进而验证得到所有频繁项集的算法。在该算法中,对候选项集的验证操作占用了大量的时间,为此提出了改进算法。改进主要体现在两个方面:按支持度降序对频繁 1 项重新编号再构建关联图;利用 Apriori 性质删减用来生成候选项集的冗余扩展项节点。实验结果表明,在最小支持度阈值较小时,改进算法有效减少了冗余的候选频繁项集,提高了算法的性能。

关键词:数据挖掘;关联规则;关联图;频繁项集
中图分类号: TP311.13 **文献标识码**:A

An efficient graph-based algorithm for discovering association rules

CHEN Ming[1,2], SHI Zhong-zhi[2], WANG Wen-jie[1]
(1. *School of Information Science and Engineering, Graduate University of Chinese Academy of Sciences, Beijing* 100049, *China*;
2. *Key Laboratory of Intelligent Information Process, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: The algorithm for discovering association rules based on graph only scans the database once to construct an association graph and then traverses the graph to generate all frequent itemsets. It costs too much time in proving the candidate frequent itemsets to be really frequent. An improved algorithm was proposed. The improvements were renumbering the frequent 1 item in the support degree descending order and utilizing the Apriori property to prune the redundant extended items which were used to generate the candidate frequent itemsets. Experiment results show that the improved algorithm prune the redundant candidate frequent itemsets when the minimum support degree is small, and the performance is improved.

Key words: data mining; association rules; association graph; frequent itemsets

0 引言

自从 1993 年关联规则挖掘问题首先由 Agrawal 等人提出以来,经过十几年的发展,关联规则相关问题的研究从内涵到外延都得到了不断地扩展[1]。关联规则最初定义在布尔型事务数据库之上,每个事务包含若干事务项,事务对应的记录描述了事务项出现与否的信息。关联规则挖掘基本问题就是研究事务数据库中事务项出现之间的关系,找出具有用户给定的最小支持度和最小可信度的关联规则,其中频繁项集的挖掘又是最基本的问题[2]。

关联规则挖掘问题的一个经典算法是 Agrawal 等人提出的 Apriori 算法[2],该算法基于 Apriori 性质,即一个频繁项集的任一非空子集必定是频繁项集。Apriori 算法大大减少了候选频繁项集的数量,但是它需要自连接生成候选项集,以及多次扫描数据库。近年来,许多研究人员对该算法加以改进,提出了基于哈希表[3]、数据分块[4]、采样[5]、动态项集[6]等技术的改进算法。FP-growth 算法[7]是另一个典型算法,它使用了模式增长的方法。除了以上两类算法,还有其他一些有效的算法,如 Show-Jane Yen 和 Arbee L. P. Chen 提出的基于图的关联规则挖掘算法[8,9],该算法仅扫描一次数据库,然后构建关联图,并根据该图来生成候选频繁项集,最后通过验证得到所有的频繁项集。实验表明该算法是一种有效的算法[9]。在基于图的关联规则挖掘算法中,对候选频繁项集的验证操作占用了大量的时间。如果能减少验证不必要的候选频繁项集,将对算法效率的提高有较大帮助。基于这样的分析,我们提出了一种改进算法,并通过实验验证了改进算法有效减少了冗余的候选频繁项集,提高了算法的性能。

1 基于图的关联规则挖掘算法

文献[8,9]中提出了基于图的关联规则挖掘算法(Direct Large itemset Generation,DLG),其基本思想是:采用垂直数据库布局,将数据库的信息映像到位向量矩阵,因此只需扫描一遍数据库;构建关联图,可以通过该图直接生成候选频繁项集;对候选频繁项集中各项的位向量做相交操作(即逻辑“与”操作)来验证该候选项集是否是频繁项集。

表 1 一个事务数据库 T

TID	Itemset	TID	Itemset
1	A B C D	3	A C E
2	E C B	4	E B

收稿日期:2006 - 05 - 24;**修订日期**:2006 - 06 - 28 **基金项目**:国家自然科学基金资助项目(60435010, 90604017,60675010);国家 973 项目(2003CB317004);北京市自然科学基金资助项目(4052025)

作者简介:陈明(1984 -),男,安徽淮北人,硕士研究生,主要研究方向:数据挖掘; 史忠植(1941 -),男,江苏宜兴人,研究员,博士生导师,主要研究方向:人工智能、机器学习、神经计算、认知科学; 王文杰(1964 -),男,北京人,副教授,主要研究方向:人工智能、多 Agent 系统.

下面结合示例具体介绍 DLG 算法的各个步骤。

例如表 1 所示的一个事务数据库 T,每个记录是一个 <TID,Itemset> 对,TID 标识事务编号,Itemset 记录了事务中出现的项。不妨设最小支持度阈值为 2(表示事务数,按百分比即为 50%)。要求找出 T 中所有的频繁项集。

DLG 算法的第 1 步扫描一遍事务数据库 T,为每个项赋予一个唯一的整数编号,并且为每个项建立一个位向量。项的位向量的长度等于事务数,若项在第 n 个事务中出现,则该项的位向量的第 n 位置 1,否则置 0。项的位向量中 1 的数量表示该项在数据库中出现的次数,即表示支持度的大小。项 i 的位向量以 $\boldsymbol{BV}_i$ 表示。

对上述示例,第 1 步后,项 A,B,C,D,E 分别被编号为 1,2,3,4,5,对每个项都建立了相应的位向量,其中不小于最小支持度的频繁 1 项为 1,2,3 和 5。将它们加入频繁 1 项集的集合 L_1,其中各项对应的位向量分别是 $\boldsymbol{BV}_1 = \{1010\}$,$\boldsymbol{BV}_2 = \{1101\}$,$\boldsymbol{BV}_3 = \{1110\}$,$\boldsymbol{BV}_5 = \{0111\}$。

第 2 步,构建关联图并生成频繁 2 项集。关联图是一种有向图,图中每个节点表示一个频繁 1 项(节点称为项节点)。关联图表示了频繁 1 项之间的一种关联关系,其构建方法为:对 L_1 中任意两个频繁 1 项 i 和 j(不妨设 $i < j$)的位向量做相交操作 $\boldsymbol{BV}_i \wedge \boldsymbol{BV}_j$,若结果位向量中 1 的个数不小于最小支持度,那么在关联图中就创建一条从 i 到 j 的有向边(注意有向边都是编号小的项指向编号大的项);同时将 $\{i, j\}$ 加入频繁 2 项集的集合 L_2。

对上述示例,第 2 步后,关联图如图 1 所示。频繁 2 项集有 {1,3},{2,3},{2,5},{3,5}。

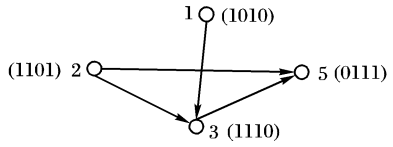

图 1　构建的关联图及频繁 1 项节点的位向量

第 3 步,得到所有的频繁 $k(k > 2)$ 项集。第 2 步后得到了关联图和所有的频繁 2 项集,频繁 $k+1(k \geqslant 2)$ 项集的生成方法这样的(令 k 初值为 2):令频繁 $k+1$ 项集的集合 L_{k+1} 为空。对每一个频繁 k 项集,其最后一个项被用来扩展得到候选频繁 $k+1$ 项集。扩展方法为:设 $\{i_1, i_2, \cdots, i_k\}$ 表示一个频繁 k 项集,i_k 是最后一项。设关联图中从 i_k 发出的有向边指向的所有项节点为 $\{u_1, u_2, \cdots, u_n\}$,则 $\{i_1, i_2, \cdots, i_k, u_1\}$,$\{i_1, i_2, \cdots, i_k, u_2\}$,$\cdots$,$\{i_1, i_2, \cdots, i_k, u_n\}$ 就是扩展得到的候选频繁 $k+1$ 项集。然后将每个候选频繁 $k+1$ 项集中的 $k+1$ 个项的位向量进行相交操作,若结果位向量中 1 的个数不小于最小支持度,则该候选项集是频繁的,将其加入 L_{k+1}。令 $k = k+1$,若频繁 k 项集 L_k 不为空,循环执行上述操作;否则,算法终止。最终所有的频繁项集为 $L_1 \cup L_2 \cup \cdots \cup L_{k-1}$。

上例中,频繁 2 项集是 {1,3},{2,3},{2,5},{3,5}。对 {1,3},由于关联图中项节点 3 发出的有向边指向的项节点为 {5},则扩展得到候选频繁 3 项集 {1,3,5},但 $\boldsymbol{BV}_1 \wedge \boldsymbol{BV}_3 \wedge \boldsymbol{BV}_5$ 结果中 1 的个数为 1,小于最小支持度,所以 {1,3,5} 不是频繁的 3 项集;对 {2,3},扩展得到候选频繁 3 项集 {2,3,5},且 $\boldsymbol{BV}_2 \wedge \boldsymbol{BV}_3 \wedge \boldsymbol{BV}_5$ 结果中 1 的个数为 2,不小于最小支持度,所以 {2,3,5} 是频繁的 3 项集;由于关联图中项节点 5 没有发出有向边,所以 {2,5},{3,5} 就不能扩展。接下来由于频繁 3 项集 {2,3,5} 不能扩展,则频繁 4 项集为空,算法终止。最终所有的频繁项集的集合为 { {1},{2},{3},{5},{1,3},{2,3},{2,5},{3,5},{2,3,5} }。

分析 DLG 算法可以看出,该算法仅需扫描一次数据库,相比多次扫描数据库的算法减少了访问数据库的操作时间;通过构造关联图,提供了候选频繁 k 项集($k > 2$)的生成方法。考察 Apriori 算法,其候选频繁 $k+1$ 项集的生成,是通过频繁 k 项集两两自连接,然后再进行剪枝得到。DLG 算法相比 Apriori 算法,候选项集的生成方法更加简单,并且仅扫描一次数据库,算法效率较高。

2　对 DLG 算法的改进

对 DLG 算法进行分析可以看到,从已知频繁 k 项集得到候选频繁 $k+1$ 项集的方法是利用关联图直接扩展的,但其中存在着一定的冗余。可以从以下方面改进。

1)按支持度降序对频繁 1 项重新编号再构建关联图

频繁 1 项得到之后,按照支持度大小降序对频繁 1 项的各项从 1 开始递增重新编号,即支持度最大的频繁项重编号为 1,次之重编号为 2,依次类推。然后再构建关联图。

根据 DLG 算法,关联图的边都是由编号小的项节点指向编号大的项节点。这样在关联图中,重新编号后的项 1 发出的有向边指向的节点最多;项 2 次之,依次类推。而频繁 k 项集中的各项是按编号由小到大排列的。因此,按照改进方法生成的关联图由频繁 k 项集扩展得到候选频繁 $k+1$ 项集时,扩展的节点与频繁 k 项集的前 $k-1$ 个节点在关联图中相邻的几率增大了,这样就去掉了原来算法可能产生的一部分冗余候选频繁 $k+1$ 项集,相当于做了一次剪枝。

2)利用 Apriori 性质删减用来生成候选项集的冗余扩展项节点

考察从频繁 k 项集扩展得到候选频繁 $k+1$ 项集的方法,原算法是根据关联图对频繁 k 项集直接加入扩展项节点;我们的改进算法对扩展项节点增加了验证,即验证扩展的项节点在关联图中是否与频繁 k 项集的前 $k-1$ 个项节点都相邻,如满足验证条件,则扩展该项节点生成候选频繁 $k+1$ 项集;否则,不应扩展该项节点。

我们进行这种操作基于如下的原因:若扩展的项节点在关联图中与频繁 k 项集的前 $k-1$ 个项节点中的任意一个不相邻,则这两个项组成的项集就是非频繁项集,进而包含这两个项的候选频繁 $k+1$ 项集显然也是非频繁项集,不用再验证。因此,这样的项节点就不应被扩展。这样做本质上是根据 Apriori 性质删减了一部分冗余候选频繁 $k+1$ 项集,相当于做了第二次剪枝。

给定一事务数据库,当最小支持度阈值越小时,频繁项集的数量越多,频繁项集的最大长度也越大;相应地,原算法产生的冗余候选项集的数量也越多,改进算法较原算法的性能提升也越大。后文的实验验证了这一点。而在实际应用中,最小支持度阈值较小的情况也很常见。因此,我们的改进是有意义的。

基于上述的改进思想,我们提出了改进算法 DLG2。

算法 1　DLG2

输入　事务数据库 D,给定最小支持度 $minsup$

输出　所有的频繁项集

\\频繁 1 项集生成和频繁 1 项重新编号阶段

扫描数据库 D,为每个项编号并得到相应的位向量;

```
for (each item i in D) {                    \\ L1 为频繁 1 项集的集合
  if (number of 1's in BV_i >= minsup)
    按照支持度大小从高到低,将{ i }加入到 L1 中适当位置;}
for( each itemset { i } in L1 ) {
  按项 i 在 L1 中的位置对 i 重新编号;}
\\关联图生成及频繁 2 项集生成阶段
for ( every two frequent itemset { i },{ j } ( i < j ) in L1 ) {
  if (number of 1's in BV_i ∧ BV_j ) >= minsup ){
    将{ i, j }加入 L2 ;                    \\ L2 为频繁 2 项集的集合
    在关联图中构建从 i 到 j 的有向边;
    \\若 i 和 j 之间存在有向边,则称它们相邻 }}
\\频繁 k ( k >2) 项集生成阶段
k = 2;
while( Lk ≠∅ ){                            \\ Lk 为频繁 k 项集的集合
  Lk+1 = ∅;                                 \\ Lk+1 为频繁 k + 1 项集的集合
  for (each itemset( i1, i2, ..., ik ) in Lk )
    for ( ik 在关联图中发出的有向边指向的每个项节点,
      设为 u ) {
      if ( u 在关联图中与频繁 k 项集的前 k - 1 项对应的项节点
        都相邻)
        if (number of 1's in BV_i1 ∧ ... ∧ BV_ik ∧ BV_u ) > =
          minsup )
          将{ i1, i2, ..., ik, u }加入 Lk+1 ; }
  k = k + 1 ;}
输出 L1, L2, ..., Lk-1 中所有的频繁项集;
```

3 实验验证及讨论

本节将 DLG 算法(记为 DLG1)与改进后的 DLG2 算法进行性能比较。我们用 C ++ 语言分别实现了 DLG1 和 DLG2 算法的程序,编译环境为 VC ++6.0,测试机器环境为 Pentium IV 2.8 GHz,内存 512MB,运行 Windows XP SP2 操作系统。

实验所用的 3 个测试数据集是根据文献[2]中给出的人造数据生成算法生成的数据。如数据集 T25.I10.D10k 表示事务数据库每个事务的平均大小为 25(项),频繁项集的平均长度为 10(项),事务的个数为 10000(条)。数据集中不同的项个数都为 1000。我们分别在 3 个数据集上测试了两个算法在不同最小支持度阈值下(0.6% ~1.5%)的运行时间相对比值。测试结果如图 2 所示。

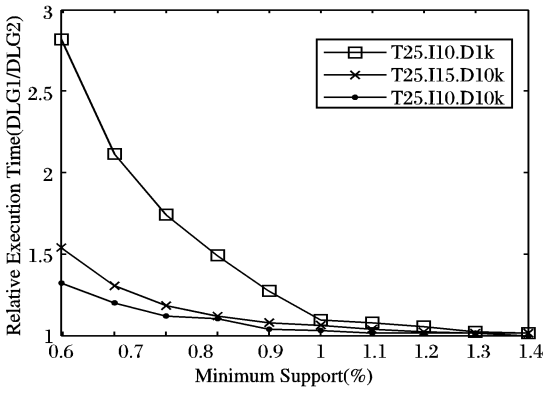

图 2 原算法 DLG1 与改进算法 DLG2 相对运行时间比较

图 2 给出了 DLG1 算法与 DLG2 算法在不同数据集和不同最小支持度阈值下的运行时间相对比值。运行时间指从读数据开始,一直到求出所有频繁项集的时间,但不包括输出结果的时间。两个算法在同一数据集和相同最小支持度阈值下得到的结果集完全相同。图 2 表明 DLG2 算法比原 DLG1 算法有显著的性能改进,并且性能改进随着最小支持度阈值的减小而增加。这是因为随着最小支持度阈值减小,频繁项集的数量和频繁项集的最大长度会增加,原 DLG1 算法要验证的冗余候选频繁项集就会增多,而改进算法 DLG2 删减的冗余候选频繁项集也增多,从而节省了更多的候选项集验证操作时间。随着支持度阈值的增加,改进算法与原算法的性能趋于持平(相对运行时间比值趋于 1)。这是因为改进算法的性能提升主要体现在对冗余候选频繁 k 项集($k > 2$)的删减;随着最小支持度阈值增加,频繁项集的数量和频繁项集的最大长度会减小,冗余候选项集的数量也相应减少,直至为零,因此改进算法的性能提升也会相应减少。如图所示支持度阈值为 1.4% 时,数据集 T25.I10.D10k 的频繁项集的最大长度为 2,即仅有频繁 1 项集与 2 项集,在这种情况下改进算法没有冗余的候选项集可以删减,性能与原算法持平。

4 结语

本文对文献[9]基于图的关联规则挖掘算法提出了一种改进算法。改进主要体现在两个方面:一是按支持度降序对频繁 1 项重新编号再构建关联图;二是利用 Apriori 性质删减用来生成候选项集的冗余扩展项节点。这两点改进目的是删减冗余的候选频繁项集。通过实验比较验证,在最小支持度阈值较小时,改进算法比原算法更加高效。

致谢:感谢中科院计算所智能信息处理实验室张素兰和石川对本文提出的修改意见。

参考文献:

[1] 贾彩燕, 倪现君. 关联规则挖掘研究述评[J]. 计算机科学. 2003, 30(4):145.

[2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[A]. Proceedings of 1994 International Conference on Very Large Data Bases(VLDB'94) [C]. Santiago, Chile, 1994. 487 - 499.

[3] PARK JS, CHEN MS, YU PS. An Effective Hash-based Algorithm for Mining Association Rules[A]. Proceedings of 1995 ACM - SIGMOD International Conference on Management of Data (SIGMOD'95) [C]. San Jose, CA, 1995. 175 - 186.

[4] SAVASERE A, OMIECINSKI E, NAVATHE S. An Efficient Algorithm for Mining Association Rules in Large Databases[A]. Proceedings of 1995 International Conference on Very Large Data Bases (VLDB'95) [C], 1995.432 - 443.

[5] TOIVONEN H . Sampling Large Databases for Association Rules [A]. Proceedings of 1996 International Conference on Very Large Data Bases (VLDB'96) [C]. Bombay, India, 1996. 134 - 145.

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

[7] HAN J, PEI J, YIN Y. Mining Frequent Patterns without Candidate Generation[A]. Proceedings of 2000 ACM-SIGMOD International Conference on Management of Data[C]. Dallas, TX, 2000. 1 - 12.

[8] YEN SJ, CHEN ALP. An Efficient Approach to Discovering Knowledge from Large Database[A]. Parallel and Distributed Information Systems - Proceedings of the International Conference[C], 1996. 8 - 18.

[9] YEN SJ, CHEN ALP. A graph-based approach for discovering various types of association rules[J]. IEEE Transactions on Knowledge and Data Engineering, 2001, 13(5): 839 - 845.