数据挖掘中关联规则 Apriori 算法①

亓文娟[1], 晏 杰[2]

[1](武夷学院 数学与计算机系, 武夷山 354300)
[2](武夷学院 团委, 武夷山 354300)

摘 要: 深入研究关联规则算法, 针对 Apriori 算法瓶颈问题提出了一种改进算法, 该算法在构建向量矩阵的基础上, 只需要扫描一次事务数据库, 通过优化连接和剪枝, 提高了算法的运行效率. 研究和实验表明, 改进后的算法在大规模的事务数据库中, 较 Apriori 算法有明显的优势.

关键词: 关联规则; Apriori 算法; 向量矩阵

Apriori Algorithm of Association Rules in Data Mining

QI Wen-Juan[1], YAN Jie[2]

[1](Mathematics and Computer Science department, Wuyi University, Wuyishan 354300, China)
[2](Youth League Committee, Wuyi University, Wuyishan 354300, China)

Abstract: In this paper, study of the association rules algorithm, aiming at the bottleneck problem of Apriori algorithm an improved algorithm is proposed. The algorithm on the basis of building a vector matrix, only need to scan a transaction database by optimizing the connections and pruning to improve the operating efficiency of the algorithm. Research and experiments show that the improved algorithm in a large-scale transaction database than Apriori algorithm has obvious advantages.

Key words: Association rules; Apriori algorithm; vector matrix

数据挖掘(Data Mining)就是从数据库中发现知识(KDD)、数据分析、数据融合(Data Fusion)以及决策支持等. 关联规则的概念和模型是首先由 R.Agrawal 等人在 1993 年提出来的, 是对一个事物和其它事物的相互依存和相互关联的一种描述. 针对数据而言是发现数据中项集之间潜在的关联或依赖联系. 关联规则挖掘算法最经典的算法 Apriori 算法使用频繁项集性质的先验知识, 通过逐层搜索的迭代方法来查找频繁项集, 需要多次扫描事务数据库, 运算效率不高.

本文针对 Apriori 算法的不足提出了改进, 将事务数据布尔化, 并在其基础上通过优化连接和剪枝, 尽可能减少候选项集, 并根据判断相应的布尔向量“与”运算的结果, 快速查找出频繁项集[1].

1 相关概念和理论

1.1 关联规则概念

(1) 设 I=[i1,i2,,i3,… in]项的集合. 设任务相关的数据集 D 是事务数据库的集合, 其中每个事务 T 是项目的集合, 使得 T⊆I. 每一个事务有一个表示符, 称作 TID. 事务 T 包含一个项目集 A 当且仅当 A⊆T, 一个关联规则就是形如 A==>B 的逻辑蕴涵式, 其中 A⊂I, B⊂I 并且 A∩B=∅[2].

(2) 支持度: 即 A 和 B 这两个项集的并集 A∪B 在所有事务 D 中出现的概率. Support(A==>B)=P(A∪B)＝Support(A∪B)＝S.

(3) 置信度: 即在出现了项集 A 的事务 D 中, 项集 B 也同时出现的概率. Confidence(A==>B)=P(B|A)＝Support(A∪B)/Support(A)＝C.

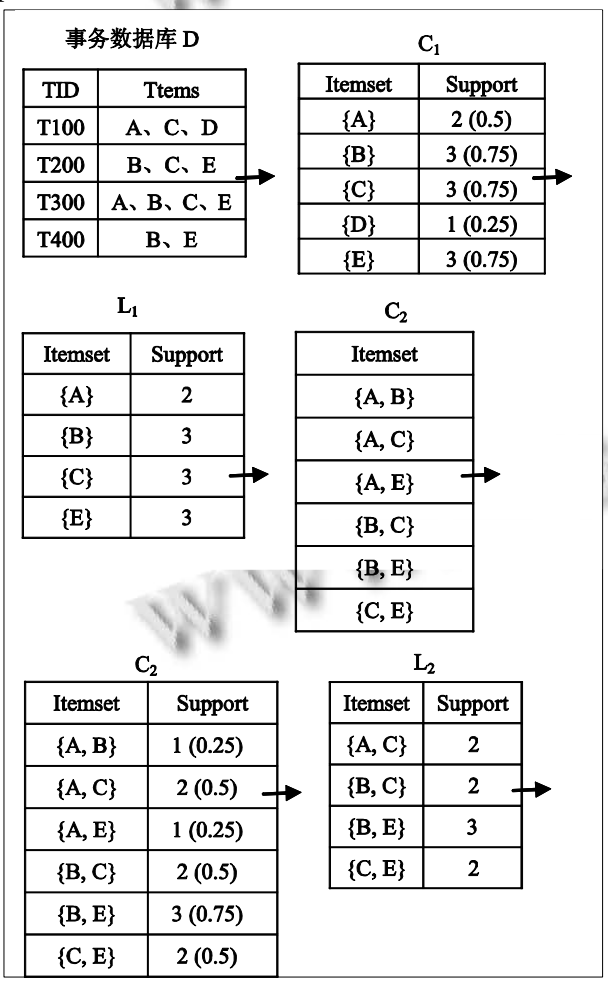
① 收稿时间:2012-09-21;收到修改稿时间:2012-10-18

强规则即同时满足最小支持度阈值和最小置信度阈值的规则. 给定一个事务集 D, 挖掘关联规则问题就是产生强规则的问题.

1.2 经典算法 Apriori

Apriori 算法是布尔关联规则挖掘频繁项集的原创性算法. 该算法利用逐层搜索的迭代方法找出数据库中项集的关系, 以形成规则, 其过程由连接与剪枝组成. Apriori 算法寻找最大项目集的基本思想是:第一步, 统计所有含一个元素项目集出现的频率,并找出不小于 minsup 的一维最大项目集.从第 k 步(k≥2)开始根据第 k-1 步生成的(k-1)维最大项目集产生 k 维侯选项目集,搜索数据库得到侯选项目集的项集支持度, 与 minsup 比较, 直到没有候选项集为止, 最终找到 k 维最大项目集. 整个过程需要多次遍历事务数据库,产生大量的候选集,这需要很大的 I/O 负载.

已知事务数据库 D, 设最小支持度是 2, 图 1 是 Apriori 算法寻找 D 中频繁项集的过程.



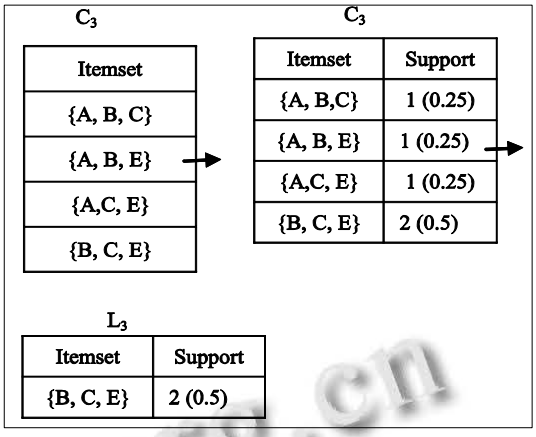


图 1　Apriori 算法

2　Apriori算法的改进

Apriori 算法主要缺点有两个: 可能产生大量的候选集和可能需要重复扫描数据库, 为了提高 Apriori 算法的效率, 国内外的许多文献提出了 Apriori 算法的优化, 如划分、抽样、事务压缩、散列技术、动态项集计数等[3].

2.1 改进算法思路

扫描一次数据库,将事务数据库 D 映射为布尔向量矩阵. 各个项目的向量长度等于事务数, 若项在第 n 个事务中出现, 则该项的布尔向量的第 n 位设置为“1”, 否则为“0”, 对各个项目中“1”的数目进行统计,如果大于或等于最小支持度,则属于频繁 1 项集. 对频繁 1 项集中任意 2 个向量, 根据候选项集对应的布尔向量进行逻辑“与”运算,布尔向量中相同位置都为“1”时结果为“1”, ,如果结果中“1”的数目大于或等于最小支持度,则它们的组合为频繁 2 项集. 对已存在的频繁 2 项集中任意 2 个符合连接条件的项集进行连接,在连接生成的 3 项集中判断最后 2 位项目的“与”运算, 结果不满足最小支持度阈值的要求, 则不用进一步的剪枝, 相应的减少候选项集, 结果大于或等于最小支持度的方可继续进行剪枝. 通过优化连接和剪枝, 使候选项集数较大幅度减少. 如果剪枝生成的所有 3 项集的子集都存在于频繁项集 L2 中,那么只需要对此 3 项集的各个项进行“与”操作, 所得结果满足支持度阀值要求的, 该 3 项集为频繁 3 项集. 以此类推,最终生成所有频繁项集. 改进流程图如图 2 所示.

2.2 算法示例

为更清晰的说明算法的功能, 以图 1 中的事务数据库为例, 最小支持度为 2.

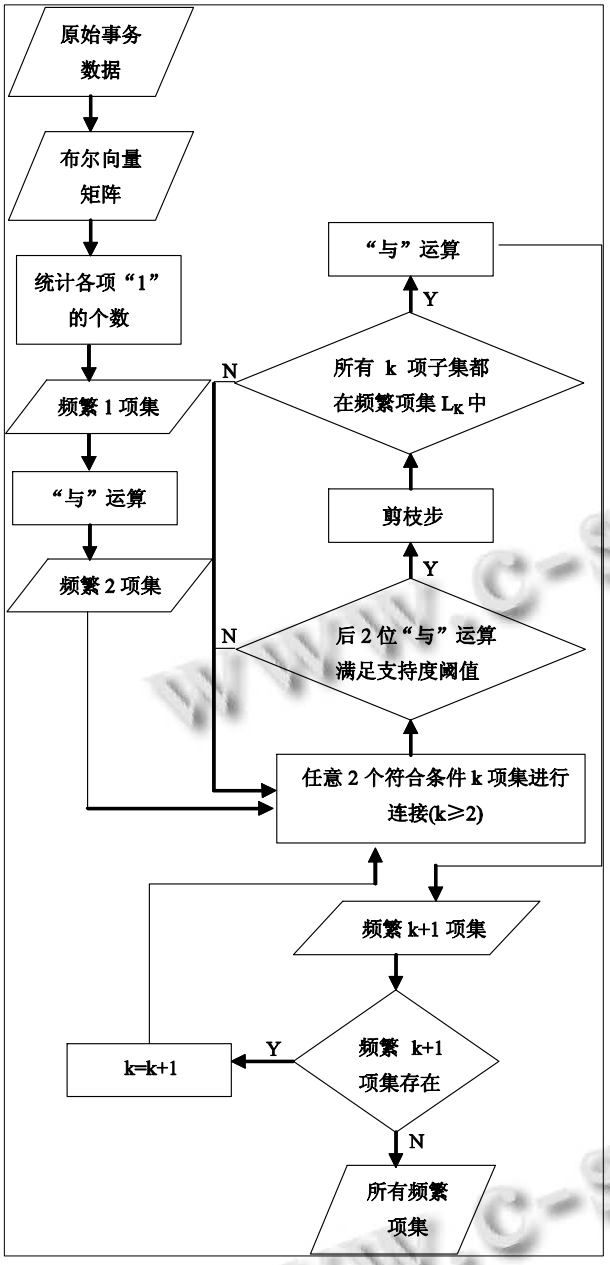


图 2 改进算法流程图

(1) 扫描一次数据库,将事务数据库 D 映射为布尔向量矩阵[4]. 如表 1 所示:

表 1 事务数据库 D 对应的布尔向量矩阵

	A	B	C	D	E
T100	1	0	1	1	0
T200	0	1	1	0	1
T300	1	1	1	0	1
T400	0	1	0	0	1

(2) 频繁 1 项集. 统计每个项中 “1”的数目, 删除不满足最小支持度 2 的项. A=1010, B=0111, C=1110, D=1000, E=0111, 项目 A, B, C, E 均符合满足最小支持度 2 的要求, 故频繁 1 项集 L1={{A}, {B}, {C}, {E}}.

(3) 频繁 2 项集. 对频繁 1 项集 L1 中任意 2 个向量进行逻辑 “与”运算,其结果中如果 “1”的数目满足支持度阀值,则这 2 个向量的组合属于频繁 2 项集. A∧B=0010, A∧C=1010, A∧E=0000, B∧C=0110, B∧E=0111, C∧E=0110, 只有A∧C, B∧C, B∧E, C∧E满足要求, 故频繁2项集L2={{A, C}, {B, C}, {B, E}, {C, E}}.

(4) 频繁 3 项集. 对频繁 2 项集 L2 中任意 2 个符合连接条件的项集进行连接生成 3 项集{A, B, C }, {A, C, E }, {B, C, E }, 判断最后 2 位项目的 “与”运算, A∧B, A∧E 不符合最小支持度阈值的要求, 故 3 项集{A, B, C }, {A, C, E }不用考虑, 而{B, C, E }的所有 2 项子集都满足要求, B∧C∧E=0110, 故频繁 3 项集 L3={{B, C, E }}. 根据算法 L4 不存在, 推导结束.

故事务数据库 D 中的频繁项集 L={{A}, {B}, {C}, {E}, {A, C}, {B, C}, {B, E}, {C, E}, {B, C, E }}.

3 改进算法实验分析

为了验证改进后算法的效率, 采用 VC++6.0 分别实现 Apriori 算法和改进后的 Apriori 算法. 在 CPU 为 2GHZ, 内存为 1G, 操作系统为 Windows XP 的计算机上测试. 测试数据随机筛选于武夷学院图书馆管理系统数据库, 共 3 个事务集, 分别包含事务数为 500,1000, 2000, 每个事务包含 10 个项目. 分别对支持度为 0.2,0.3,0.5 时进行测试. 测试结果如表 2、表 3 所示.

表 2 Apriori 算法测试结果(耗时单位:s)

事务数 \ 支持度	0.2	0.3	0.5
500	0.029	0.018	0.009
1000	0.039	0.021	0.019
2000	0.089	0.068	0.048

表 3 改进后的 Apriori 算法测试结果(耗时单位:s)

事务数 \ 支持度	0.2	0.3	0.5
500	0.018	0.015	0.008
1000	0.027	0.016	0.015
2000	0.043	0.036	0.030

根据测试结果我们对事务数为 1000 的 Apriori 算法和改进后的 Apriori 算法数据进行分析，如图 3、图 4 所示.

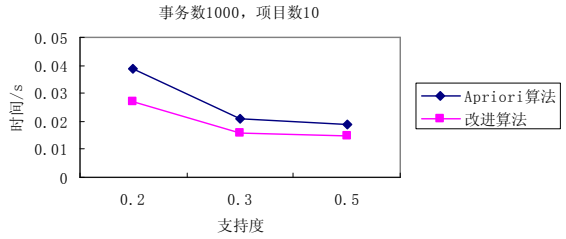


图 3　不同支持度下两种算法执行时间

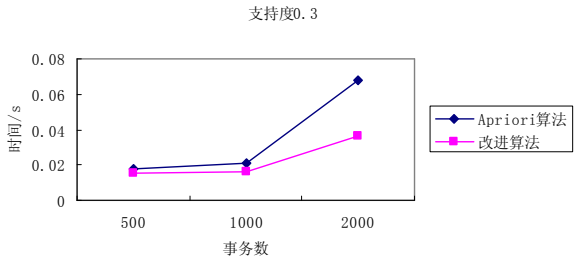


图 4　不同事务数下两种算法执行时间

从上图可以看出，当最小支持度比较小时，改进的算法所需的时间比 Apriori 算法少很多，当最小支持度逐渐增大时，两种算法执行时间比较接近；当事务数量不断增加时，改进的算法所需的时间比 Apriori 算法少很多，但在事务数量比较少时，改进的算法不具备明显优势. 这是因为改进的算法主要是针对减少候选项的数量和跳过对应频繁项集产生无贡献的记录的考虑而获得性能改进的.

4　结语

本文在深入研究关联规则算法的基础上，针对 Apriori 算法可能产生大量的候选集和可能需要重复扫描数据库的问题，提出了一种基于布尔矩阵的改进算法，实验表明，改进算法适合在大规模的事务数据库中挖掘关联规则，对于产生大量候选项集的情况下具有较高的挖掘效率.

参考文献

1 郭云峰,张集祥.对关联规则挖掘中 Apriori 算法的一种改进.杭州电子科技大学学报,2009,4:60–63.

2 Han JW, Kamber Mi. 范明,孟小峰译.数据挖掘概念与技术. 北京:机械工业出版社,2006.

3 周怡,王世伟.医学数据挖掘－SQL Server 2005 案例分析. 北京:中国铁道出版社,2008.

4 袁剑,王文海.挖掘关联规则 Apriori 算法的一种改进.青岛科技大学学报,2008,10:448–451.

(上接第 141 页)

2 Chen L, Ozsu MT, Oria V. Robust and fast similarity search for moving object trajectories. Proc. of ACM SIGMOD Int. Conf. on Management of Data. 2005.

3 Cranor CD, Johnson T, spatscheck O, Gigascope: A stream database for network applications. Proc. of ACM SIGMOD Int. Conf. on Management of Data. 2003.

4 Xue W, Luo Q, Chen L, Liu Y. Contour map matching for event detection in sensor networks. Proc. of ACM SIGMOD Int. Conf. on Management of Data. 2006.

5 Srikant R, Agrawal R. Mining generalized association rules. Proc. of the 21st VLDB Conf. 1995: 409-419.

6 Lian X, Chen L. Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Database. Proc. ACM SIGMOD Int. Conf. on Management of Data. Vancouver, Canada. 2008. 213–226.

7 Agrawal R, Faloutsos C, Swami AN. Efficient similarity search in sequence databases. FODO, 1993.

8 Morinaka Y, Yoshikawa M, Amagasa T, Uemura S. The L-index: An indexing structure for efficient subsequence matching in time sequence databases. PAKDD. 2001.

9 Cai Y, Ng R. Indexing spatio-temporal trajectories with Chebyshev polynomials. SIGMOD. 2004.