第18卷 第5期
2008年5月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.18 No.5
May 2008

一种改进的 Apriori 算法

袁万莲[1,2],郑 诚[1],翟明清[2]
(1. 安徽大学 计算机科学与技术学院,安徽 合肥 230039;
2. 滁州学院 数学系,安徽 滁州 239012)

摘 要:数据挖掘中的关联规则挖掘能够发现大量数据中项集之间有趣的关联或相关联系,特别是随着大量数据不停地收集和存储,从数据库中挖掘关联规则就越来越有其必要性。通过对关联规则挖掘技术及其相关算法 Apriori 进行分析,发现该技术存在的问题。Apriori 算法是关联规则挖掘中的经典算法。对 Apriori 算法做了改进。借助 0-1 矩阵给出了计算项集的支持度计数的更快方法,同时还简化了 Apriori 算法中的连接和剪枝操作,从而在时间和空间上提高了 Apriori 算法的效率。

关键词:数据挖掘;关联规则;Apriori 算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)05-0051-03

An Improvement on Apriori Algorithm

YUAN Wan-lian[1,2], ZHENG Cheng[1], ZHAI Ming-qing[2]
(1. School of Computer Science and Technology, Anhui University, Hefei 230039, China;
2. Department of Mathematics, Chuzhou University, Chuzhou 239012, China)

Abstract: The many interesting relations and relevance between the different items of sets can be located by means of the association rules mining technique of data mining. The mining of association rules from databases becomes especially necessary with the data in collection and storation becomes even larger. However, an analysis of the association rules mining technique and its relevant algorithm of Apriori has indicated that there exist problems in the technique in discussion. Apriori algorithm is classical in association rules mining. Provides an improvement on Apriori algorithm. It can compute the count of support of itemsets more quickly by 0-1 matrix. It can also simplify the join step and the prune step in Apriori algorithm. Thus the efficiency of Apriori algorithm get improvement both in time and in space.

Key words: data mining; association rule; Apriori algorithm

0 引 言

Apriori 算法[1]是 R. Agrawal 和 R. Srikant 于 1994 年提出的为布尔关联规则挖掘频繁项集的原创性算法。Apriori 算法使用一种称作逐层搜索的迭代方法,k 项集用于搜索($k+1$)项集。算法的核心思想是基于频繁理论的一种递推方法,目的是从数据库中挖掘出那些支持度和置信度都不低于给定的最小支持度阈值和最小置信度阈值的关联规则。

目前国内外学者提出了许多有效算法,但大多数都是基于经典的 Apriori 算法所提出的自底向上的思想,不足之处在于要对数据库进行多次扫描,确定候选频繁项集和计算候选频繁项集的支持度的算法复杂,工作量大,效率低。文中依据 Apriori 算法的思路加以改进,将事务数据库转换成 0-1 矩阵[2,3],通过 0-1 矩阵可以很快计算出各个候选项集的支持度计数,这样就避免了传统 Apriori 算法频繁扫描数据库的操作。另外,还简化了传统 Aprior 算法中的连接和剪枝操作,可以较快地找出频繁项集。

收稿日期:2007-08-23
基金项目:安徽省自然科学项目(KJ2007B124)
作者简介:袁万莲(1977-),女,安徽滁州人,讲师,硕士研究生,研究方向为数据挖掘、数据库及其应用;郑 诚,副教授,博士,研究方向为数据挖掘与知识发现、人工智能及其应用、数据库及其应用。

1 Apriori 算法原理

Apriori 算法是最经典的关联规则挖掘算法,它利用已知的高频数据项集推导其它高频数据项集,是一种宽度优先算法。Apriori 算法步骤如下[4]:

(1) 连接步。

为找 L_k,通过将 L_{k-1} 与自身连接产生候选 k 项集的集合。该候选项集合记做 C_k。设 l_1 和 l_2 是 L_{k-1} 中的项集。记号 $l_i[j]$ 表示 l_i 中的第 j 项。为方便起见,Apriori 假定事务或项集中的项按字典次序排序。对于

$(k-1)$ 项集 l_i，意味将项排序，使 $l_i[1] < l_i[2] < \cdots < l_i[k-1]$。执行连接 $L_{k-1} \infty L_{k-1}$，其中 L_{k-1} 的元素是可连接的，如果它们的前 $(k-2)$ 个项相同。即 L_{k-1} 的元素 l_1 和 l_2 可连接，如果 $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$。条件 $l_1[k-1] < l_2[k-1]$ 仅是保证不产生重复。连接 l_1 和 l_2 产生的结果项集是 $l_1[1], l_1[2], \cdots, l_1[k-1], l_2[k-1]$。

(2) 剪枝步。

C_k 是 L_k 的超集，即 C_k 的成员可以是也可以不是频繁的，但所有频繁 k 项集都包含在 C_k 中。扫描数据库，确定 C_k 中每个候选的计数，从而确定 L_k（即根据定义，计数值不小于最小支持度计数的所有候选是频繁的，从而属于 L_k）。然而，C_k 可能很大，这样所涉及的计算量就很大。为压缩 C_k，可用以下办法使用 Apriori 性质。任何非频繁的 $(k-1)$ 项集都不是频繁 k 项集的子集。因此，如果候选 k 项集的 $(k-1)$ 项子集不在 L_{k-1} 中，则该候选也不可能是频繁的，从而可以从 C_k 中删除。

2 Apriori 算法改进

2.1 事务数据库的 0-1 矩阵

定义 1：设事务数据库 D 中有 n 个事务 $T_i(i = 1, 2, 3, \cdots, n)$，$m$ 个项目 $I_j(j = 1, 2, 3, \cdots, m)$。定义一个 $n \times m$ 阶矩阵 $\boldsymbol{A}(D) = (a_{ij})_{n \times m}$ 如下：

$$a_{ij} = \begin{cases} 1 & I_j \in T_i \\ 0 & I_j \notin T_i \end{cases} (i = 1, 2, 3, \cdots, n; j = 1, 2, 3, \cdots, m)$$

矩阵 $\boldsymbol{A}(D)$ 称为事务数据库 D 对应的 0-1 矩阵。例如，图 1 所示的事务数据库对应的 0-1 矩阵如图 2 所示。

2.2 算法改进

2.2.1 Apriori 算法存在的问题

在 Apriori 算法中候选集是逐层产生，而产生此层的频繁项集必须要扫描整个数据库一次，然后再结合频繁项集产生下一层级的候选项集合，直到频繁项集无法结合产生候选集。基于上述原因，Apriori 算法存在下述问题[5]：

(1) 所挖掘的规则存在大量冗余；

(2) 因计算项过多而造成执行性能缓慢。

2.2.2 算法改进

按照定义 1，矩阵 $\boldsymbol{A}(D)$ 的第 i 行反映了事务 T_i 所含项目情况，第 j 列反映了项目 I_j 所属于的事务情况，由于项目与矩阵 $\boldsymbol{A}(D)$ 的列向量一一对应，一个项集也对应了一个由其所含项目相应列向量构成的子矩阵。同时由定义 1 可以得出下面的结论：

结论 1：对 $j = 1, 2, 3, \cdots, m$，单项集 $\{I_j\}$ 在事务数据库 D 中的支持度计数为 $\boldsymbol{A}(D)$ 的第 j 列 1 的个数；而一般项集的支持度计数，为该项集对应的子矩阵中全 1 行向量的个数。

显然借助矩阵 $\boldsymbol{A}(D)$ 可以很快计算出项集的支持度计数。例如图 1 所示事务数据库 D，项目 I_1 相应的列向量为 $\boldsymbol{A}(D)$ 的第 1 列，该列有 6 个 1，于是项集 $\{I_1\}$ 的支持度计数为 6；项集 $\{I_1, I_2, I_5\}$ 相应的子矩阵由矩阵 $\boldsymbol{A}(D)$ 的第 1,2,5 列构成，如图 3 所示，全 1 行向量有 3 个，于是项集 $\{I_1, I_2, I_5\}$ 的支持度计数为 3。

TID	项目列表
$T100$	I_1, I_2, I_5
$T200$	I_2, I_4
$T300$	I_2, I_3
$T400$	I_1, I_2, I_4, I_5
$T500$	I_1, I_3
$T600$	I_2, I_3
$T700$	I_1, I_3
$T800$	I_1, I_2, I_3, I_5
$T900$	I_1, I_2, I_3

图 1 事务数据库 D

$$\boldsymbol{A}(D) = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

图 2 D 对应的 0-1 矩阵

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

图 3 项集 $\{I_1, I_2, I_5\}$ 对应的子矩阵

另外，频繁项集肯定是事务数据库中某个或多个事务的子集。由 Apriori 算法知，算法的结束标志是当经过最后一次连接后，不再产生新的候选项集。因此寻

找候选项集可以从事务数据库中项目数最多的事务开始,将其对应的项集作为候选项集,引出算法。

算法1:

(1) 初始时,设事务数据库 D 中有 m 个项目 $I_j(j=1,2,\cdots,m)$,D 对应的0-1矩阵为 $A(D)$。在 D 中找出项目数最多的事务(对应于 $A(D)$ 的含1最多的行向量,可能不止一个),将它们对应的项集作为候选项集,这些候选项集的集合记为 T。

(2) 记 T 中每个候选项集所含项目数为 d,若 $d=0$,说明找不到频繁项集,退出程序;否则,当 $d>0$ 时,执行(3)。

(3) 利用结论1计算出 T 中每个候选项集的支持度计数。找出所有满足支持度计数大于等于 min_sup 的项集,若存在,它(们)就是频繁项集,算法到此结束;否则执行(4)。

(4) 将 T 中每个候选项集分成 C_d^{d-1} 个子项集(这里每个子项集所含项目数为 $d-1$),作为候选项集;再找出 D 中所含项目数为 $d-1$ 的事务对应的项集(相应于 $A(D)$ 的含 $d-1$ 个1的行向量),也作为候选项集。将 T 更新为这些候选项集构成的集合,返回(2)。

3 实例分析

针对图1的事务数据库 D,它的0-1矩阵 $A(D)$ 见图2,假设最小支持度计数 min_sup = 2。根据算法1,矩阵 A 的行向量中1个数最多的为第4、8行(有4个1),对应于事务 T_4 和 T_8,相应项集为 $\{I_1,I_2,I_4,I_5\}$ 和 $\{I_1,I_2,I_3,I_5\}$,即 $T=\{\{I_1,I_2,I_4,I_5\},\{I_1,I_2,I_3,I_5\}\}$,$T$ 中每个候选项集含 $d=4$ 个项目。根据结论1,$\{I_1,I_2,I_4,I_5\}$ 和 $\{I_1,I_2,I_3,I_5\}$ 的支持度计数均为1 < min_sup,因此它们均不是频繁项集。将项集 $\{I_1,I_2,I_4,I_5\}$ 和 $\{I_1,I_2,I_3,I_5\}$ 分成含3个项目的子项集:$\{I_1,I_2,I_4\}$,$\{I_1,I_2,I_5\}$,$\{I_2,I_4,I_5\}$,$\{I_1,I_4,I_5\}$,$\{I_1,I_2,I_3\}$,$\{I_1,I_3,I_5\}$,$\{I_2,I_3,I_5\}$,作为候选项集;此外,D 中含3个项目的事务对应的项集有 $\{I_1,I_2,I_5\}$,$\{I_1,I_2,I_3\}$,也作为候选项集。根据结论1,这些候选项集的支持度计数如图4所示。

项集	支持度计数
I_1, I_2, I_4	1
I_1, I_2, I_5	3
I_2, I_4, I_5	1
I_1, I_4, I_5	1
I_1, I_2, I_3	2
I_1, I_3, I_5	1
I_2, I_3, I_5	1

图4　候选项集的支持度计数

由图4可以看出,只有两个候选项集的支持度计数满足 min_sup 条件,分别为 $\{I_1,I_2,I_5\}$ 和 $\{I_1,I_2,I_3\}$,则此事务数据库的频繁项集为:$\{I_1,I_2,I_3\}$ 和 $\{I_1,I_2,I_5\}$。

4 结束语

关联规则挖掘是当前数据挖掘领域的主要研究课题[6]。文中主要在Apriori算法的基础上,改进了计算支持度计数和寻找频繁项集的方法。改进后的算法较原Apriori算法在时间和空间上有了明显的提高。

参考文献:

[1] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范明,孟小峰译. 北京:机械工业出版社,2007.

[2] 刘以安,羊　斌. 关联规则挖掘中对Apriori算法的一种改进研究[J]. 计算机应用,2007,27(2):418-420.

[3] 王柏懿,刘寒冰,靳　书,等. 基于矩阵的关联规则挖掘算法[J]. 微计算机信息,2007,24(5-3):143-145.

[4] Agrawal R, Limielinski T, Swami A N. Mining Association Rules Between Sets of Items in Large Database[C]// Proceedings of ACM SIGOD Conference on Management of Data. Washinton DC:[s.n.],1993:207-216.

[5] 张梅峰,张建伟. 基于Apriori的有效关联规则算法的研究[J]. 计算机工程与应用,2003,39(19):196-198.

[6] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules in Large Database[C]// Proceeding of the 20th International Conference on Very Large Databases. Santiago, Chile:[s.n.],1994:487-499.

(上接第50页)

江工业大学学报,2007,35(2):159-162.

[3] 吴志军,马　兰,沈笑云. Visual C++ 视频会议开发技术与实例[M]. 北京:人民邮电出版社,2006.

[4] 计算机基础教程网. SIPSAP及SDP协议组合应用的研究[J/OL]. 2006-10-20. http://www.itwen.com/03office/06file/file20061020/66563.html.

[5] 陈华林,盛翊智. SIP协议中的媒体协商[J]. 技术交流,2005,25(4):71-79.

[6] 曾庆珩,胡瑞敏,边学工. 基于SIP的集中式会议控制模型及实现[J]. 计算机工程,2005,31(3):198-200.

[7] 张友波,张焕强,孙利民. 基于SIP的视频会议系统设计与实现[J]. Computer Engineering,2005,31(21):167-169.