

关联规则挖掘 Apriori 算法的研究与改进

付 沙, 周航军

(湖南财政经济学院, 湖南 长沙 410205)

摘 要: 针对传统 Apriori 算法的不足之处, 提出两种 Apriori 改进算法, 分别基于构造辅助表和项集求交集策略。改进算法大幅度减少扫描数据库的次数, 缩减对不必要事务的扫描时间, 显著提高频繁项集的生成效率, 从而使算法达到更高的运算效率。实验结果表明, 两种改进算法都是有效的关联规则挖掘方法, 且比 Apriori 算法性能更优, 亦为关联规则挖掘研究提供了一些科学可行的新思路。

关键词: 数据挖掘; 关联规则; Apriori; 辅助表; 交集策略; 频繁项集

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2013)09-0110-05

The Research and Improvement of Apriori Algorithm for Mining Association Rules

FU Sha, ZHOU Hang-jun

(Hunan University of Finance and Economics, Changsha 410205, China)

Abstract: This paper proposed two Apriori improved algorithm with the inadequacies of the traditional Apriori algorithm, respectively based on the constructed auxiliary table and itemsets intersection strategies. Improved algorithm significantly reduced the number of times of scanning the database, and reduced unnecessary transaction scan time; improve the generation efficiency of frequent itemsets significantly, so that the algorithm can achieve higher operation efficiency. The experimental results show that the two improved algorithms are effective association rule mining method, and is better than the Apriori algorithm performance, and also provides a feasible new ideas to the association rule mining research.

Key words: data mining; association rules; Apriori; aid table; intersection strategy; frequent itemsets

1 引言

关联规则挖掘是数据挖掘中最活跃的研究方向之一, 其本质是要找出隐藏在数据间的相互关系。文献[1]提出了挖掘关联规则的经典算法——Apriori 算法。关联规则挖掘算法是关联规则挖掘研究的主要内容, 诸多研究人员对关联规则的挖掘问题进行了大量研究, 已提出许多关联规则挖掘算法, 并不断有学者提出对这些算法的改进方案^[2], 以提高算法的性能, 但各种方法都存在不同程度的缺陷。

Apriori 算法广泛应用于各种领域, 通过对数据的关联性进行分析和挖掘, 获得的信息在决策制定

过程中具有重要的参考价值^[3]。然而, 对于大型数据库挖掘, 该算法存在着两个致命的性能瓶颈: 一是多次扫描事务数据库, 需要很大的 I/O 负载; 二是可能产生大量的候选项集, 需耗费大量的时间处理^[4]。基于此, 对 Apriori 算法予以改进具有较高的理论与较强的现实意义。

2 关联规则的基本概念和原理

关联规则挖掘问题的形式化描述如下: 设 $I = (I_1, I_2, \dots, I_m)$ 是项目的集合, 任务相关的数据 D 是数据库事务的集合, 其中每一个事务 T 都是项的集合, 使得 $T \subseteq I$ 。关联规则是形如 $X \Rightarrow Y$ 的蕴含式, 其

收稿日期: 2012-11-08; 修回日期: 2012-12-14

基金项目: 湖南省高等学校科学研究项目(12C0559)

中 X 和 Y 是项集,且 $X \subset I, Y \subset I, X \cap Y = \emptyset$. 定义支持度(support)为 D 中包含 $X \cup Y$ 的事务占全部事务的百分比,记作 $\text{support}(X \Rightarrow Y) = P(X \cup Y)$;置信度(confidence)为 D 中包含 $X \cup Y$ 的事务数与包含 X 的事务数的比值,记作 $\text{confidence}(X \Rightarrow Y) = P(X|Y)$. 多个项组成的集合称为项集. 包含 k 个项的项集称为 k -项集. 如果某个项集的支持度大于等于设定的最小支持度阈值 min_sup ,则称这个项集为频繁项集,所有的频繁 k -项集组成的集合记为 $L_k^{[5]}$.

Apriori 算法是发现关联规则领域的经典算法. 该算法将发现关联规则的过程分为两个阶段:第一阶段通过迭代检索出事务数据库中的所有频繁项集,即支持度不低于用户设定的阈值的项集;第二阶段利用频繁项集构造出满足用户最小置信度的规则^[6]. 即包含两个步骤:①生成频繁项集;②根据频繁项集产生关联规则. 其中,步骤①是核心,决定了关联规则挖掘的总体性能. 因此,该研究主要针对步骤①.

3 改进算法 Improve_Apriori_1

3.1 Improve_Apriori_1 算法的基本思想

当一个事务中不包含长度为 k 的频繁项集时,则必然不包含长度为 $(k+1)$ 的频繁项集;而任意一个 k -项集的支持度与规模小于它的事务无关. 因此,在生成候选 k -项集时,则无需再扫描字段长度小于 k 的记录,以大幅减少扫描数据量. 依此构建一张辅助表存储这些信息,包含该记录的编号及其字段长度. 在后续过程中及时删除表中不可能出现在候选项集中的记录,即字段长度不大于已生成的频繁 k -项集的 k 值,而且也不被包含在频繁项集中的记录. 在每次扫描时,只扫描辅助表中存在的记录,无需对所有记录进行扫描.

3.2 Improve_Apriori_1 算法实例

某事务数据库中有 9 个事务, $D = \{T1, T2, T3, T4, T5, T6, T7, T8, T9\}$, 对应的项目集 $I = \{I_1, I_2, I_3, I_4, I_5\}$, 如表 1 所示. 假设最小支持度 $\text{min_sup} = 40\%$, 则最小支持度计数为 $\text{min_sup} \times |D| = 40\% \times 9 \approx 4$.

(1)扫描事务数据库 D ,根据事务中所含项目情况生成数据库表,如表 2 所示.

将转换后的数据库表装入内存中,并计算每一条记录的规模,生成辅助表 $F1$,如表 3 所示.

表 1 事务数据库

TID	项 ID 的列表
T1	I_3
T2	I_1, I_3
T3	I_4, I_5
T4	I_1, I_2, I_3, I_4, I_5
T5	I_1, I_3, I_4, I_5
T6	$\emptyset$
T7	I_1
T8	I_1, I_2, I_3, I_4, I_5
T9	I_1, I_2, I_3, I_5

表 2 转换后的数据库表

TID	I_1	I_2	I_3	I_4	I_5
T1	0	0	1	0	0
T2	1	0	1	0	0
T3	0	0	0	1	1
T4	1	1	1	1	1
T5	1	0	1	1	1
T6	0	0	0	0	0
T7	1	0	0	0	0
T8	1	1	1	1	1
T9	1	1	1	0	1

表 3 辅助表 F1

TID	字段长度	TID	字段长度
T1	1	T6	0
T2	2	T7	1
T3	2	T8	5
T4	5	T9	4
T5	4		

根据给定的最小支持度阈值 min_sup ,得出频繁 1-项集 L_1 .

项集	支持度计数
$\{I_1\}$	6
$\{I_3\}$	6
$\{I_4\}$	4
$\{I_5\}$	5

(2)删除 $F1$ 中字段长度不大于 1 且不包含在频繁项集中的记录,得到辅助表 $F2$,如表 4 所示.

表 4 辅助表 F2

TID	字段长度	TID	字段长度
T2	2	T5	4
T3	2	T8	5
T4	5	T9	4

L_1 进行自连接产生候选 2-项集 C_2 .

项集	支持度计数
$\{I_1, I_3\}$	5
$\{I_1, I_4\}$	3
$\{I_1, I_5\}$	4
$\{I_3, I_4\}$	3
$\{I_3, I_5\}$	4
$\{I_4, I_5\}$	4

根据给定的最小支持度阈值 $\min_sup$, 得出频繁 2-项集 L_2 .

项集	支持度计数
$\{I_1, I_3\}$	5
$\{I_1, I_5\}$	4
$\{I_3, I_5\}$	4
$\{I_4, I_5\}$	4

(3) 通过扫描辅助表 F2, 删除 F2 中字段长度不大于 2 的记录, 得到辅助表 F3, 如表 5 所示.

表 5 辅助表 F3

TID	字段长度	TID	字段长度
T4	5	T8	5
T5	4	T9	4

L_2 进行自连接产生候选 3-项集 C_3 .

项集	支持度计数
$\{I_1, I_3, I_5\}$	4
$\{I_1, I_4, I_5\}$	3
$\{I_3, I_4, I_5\}$	3

根据给定的最小支持度阈值 $\min_sup$, 得出频繁 3-项集 L_3 .

项集	支持度计数
$\{I_1, I_3, I_5\}$	4

(4) 算法终止, 得到所有频繁项集, 如表 6 所示.

表 6 频繁项集

项集	支持度计数
$\{I_1, I_3\}$	5
$\{I_1, I_5\}$	4
$\{I_3, I_5\}$	4
$\{I_4, I_5\}$	4
$\{I_1, I_3, I_5\}$	4

4 改进算法 Improve_Apriori_2

4.1 Improve_Apriori_2 算法的前提

定理 1 包含一个项集的事务集合等于包含该项集的元素的事务的交集, 只需知道所有包含项的事务, 就可以求出包含该项集的事务, 进而得出该项集的支持度计数^[7].

性质 1 若频繁 k -项集还能生成频繁 $(k+1)$ -项集, 则频繁 k -项集中的项集个数必须大于 k .

4.2 Improve_Apriori_2 算法的基本思想

扫描事务数据库 D , 生成候选 1-项集 C_1 . 在扫描所有事务时, 记录每个候选项的事务标识符列表, 并统计各项集数目, C_1 的结构如下: 项集、事务列表、支持度计数. 对 C_1 中各项集支持度计数小于最小支持度计数的项集予以删除, 得到频繁 1-项集 L_1 . 然后, 在由频繁 $(k-1)$ -项集 L_{k-1} ($k \geq 2$) 进行自连接产生候选 k -项集 C_k ($k \geq 2$) 时, 对 C_k 的计数不再扫描其事务数据库, 而是根据定理 1, 对项集的事务标识符列表进行交集操作以生成新的事务列表, 并通过统计新生成事务列表中的事务个数获得关于 k -项集的支持度计数, 最终生成频繁 k -项集 L_k . 如此反复, 直至得到所有频繁项集^[8].

与 Apriori 算法相比, 该改进算法只在生成 C_1 时扫描一次数据库, 此后无需再扫描, 不必像 Apriori 算法计算候选 k -项集 C_k ($k \geq 1$) 的支持度时需要反复地扫描数据库, 进而节省访问数据库的时间. 此外, Apriori 算法运行过程中模式匹配需耗费大量时间, 而该改进算法则大大减少事务与频繁项集、候选项集的模式匹配操作, 从而提高算法的效率.

4.3 Improve_Apriori_2 算法实例

实例同上文 3.2 节.

(1) 扫描事务数据库 D , 产生 C_1 , 如表 7 所示.

表 7 候选 1-项集 C_1

项集	事务列表	支持度计数
$\{I_1\}$	$\{T2, T4, T5, T7, T8, T9\}$	6
$\{I_2\}$	$\{T4, T8, T9\}$	3
$\{I_3\}$	$\{T1, T2, T4, T5, T8, T9\}$	6
$\{I_4\}$	$\{T3, T4, T5, T8\}$	4
$\{I_5\}$	$\{T3, T4, T5, T8, T9\}$	5

(2) 根据 C_1 生成频繁 1-项集 $L_1 = \{I_1:6, I_3:6, I_4:4, I_5:5\}$, 如表 8 所示.

表 8 频繁 1-项集 L_1

项集	事务列表	支持度计数
$\{I_1\}$	$\{T2, T4, T5, T7, T8, T9\}$	6
$\{I_3\}$	$\{T1, T2, T4, T5, T8, T9\}$	6
$\{I_4\}$	$\{T3, T4, T5, T8\}$	4
$\{I_5\}$	$\{T3, T4, T5, T8, T9\}$	5

(3) 由 L_1 进行自连接产生候选 2-项集 $C_2 = \{\{I_1, I_3\}, \{I_1, I_4\}, \{I_1, I_5\}, \{I_3, I_4\}, \{I_3, I_5\}, \{I_4, I_5\}\}$, 然后计算每个项集的支持度计数. 其计算过程如下: 包含 $\{I_1, I_3\}$ 的事务集合为: $\{T2, T4, T5, T7,$

$T8, T9\} \cap \{T1, T2, T4, T5, T8, T9\} = \{T2, T4, T5, T8, T9\}$, 支持度计数为 5. 同理可得其它项集的事务列表及支持度计数, 候选 2-项集 C_2 如表 9 所示.

表 9 候选 2-项集 C_2

项集	事务列表	支持度计数
$\{I_1, I_3\}$	$\{T2, T4, T5, T8, T9\}$	5
$\{I_1, I_4\}$	$\{T4, T5, T8\}$	3
$\{I_1, I_5\}$	$\{T4, T5, T8, T9\}$	4
$\{I_3, I_4\}$	$\{T4, T5, T8\}$	3
$\{I_3, I_5\}$	$\{T4, T5, T8, T9\}$	4
$\{I_4, I_5\}$	$\{T3, T4, T5, T8\}$	4

(4) 生成频繁 2-项集 $L_2 = \{\{I_1, I_3\}:5, \{I_1, I_5\}:4, \{I_3, I_5\}:4, \{I_4, I_5\}:4\}$, 如表 10 所示.

表 10 频繁 2-项集 L_2

项集	事务列表	支持度计数
$\{I_1, I_3\}$	$\{T2, T4, T5, T8, T9\}$	5
$\{I_1, I_5\}$	$\{T4, T5, T8, T9\}$	4
$\{I_3, I_5\}$	$\{T4, T5, T8, T9\}$	4
$\{I_4, I_5\}$	$\{T3, T4, T5, T8\}$	4

由 L_2 进行自连接产生候选 3-项集 $C_3 = \{\{I_1, I_3, I_5\}, \{I_1, I_4, I_5\}, \{I_3, I_4, I_5\}\}$, 其中 $\{I_1, I_4, I_5\}, \{I_3, I_4, I_5\}$ 从 C_3 中删除, 剩下 $\{I_1, I_3, I_5\}$. 其计算过程如下: 包含 $\{I_1, I_3, I_5\}$ 的事务集合为: $\{T2, T4, T5, T8, T9\} \cap \{T4, T5, T8, T9\} = \{T4, T5, T8, T9\}$, 支持度计数为 4. 同理可得其它项集的事务列表及支持度计数, 候选 3-项集 C_3 如表 11 所示.

表 11 候选 3-项集 C_3

项集	事务列表	支持度计数
$\{I_1, I_3, I_5\}$	$\{T4, T5, T8, T9\}$	4
$\{I_1, I_4, I_5\}$	$\{T4, T5, T8\}$	3
$\{I_3, I_4, I_5\}$	$\{T4, T5, T8\}$	3

(5) 生成频繁 3-项集 $L_3 = \{I_1, I_3, I_5\}:4$.

项集	事务列表	支持度计数
$\{I_1, I_3, I_5\}$	$\{T4, T5, T8, T9\}$	4

算法终止, 得到所有频繁项集.

5 算法实验与分析

为检验两种改进算法的性能, 将其与 Apriori 算法进行比较. 实验环境: CPU 为 Intel Pentium IV 2.66GHz、内存 1GB 和 Windows XP 专业版操作系统. 算法采用 C++ 语言编写, 在 Visual C++ 6.0 环境下编译, 分别实现了 Apriori、Improve_Apriori_1 和 Improve_Apriori_2 三种算法, 测试数据库为 Microsoft SQL Server 2005 所带的 foodmart 2005.

mdb 数据库, 实验结果如图 1 所示.

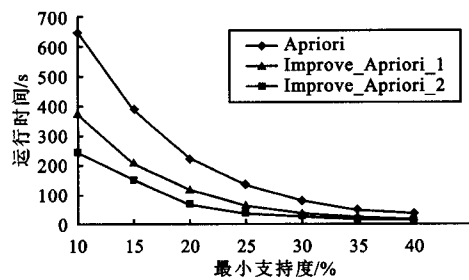


图 1 不同支持度下三种算法的运行时间对比

图 1 比较了最小支持度不同时各算法的性能, 在相同的最小支持度下两种改进算法的时间曲线均低于 Apriori 算法. 当最小支持度较小时, 三种算法之间的时间开销差距较大, 改进算法的优势非常明显, 其中改进算法 Improve_Apriori_2 的执行效率比 Apriori 算法高 2~3 倍; 而随着最小支持度的增加, 迭代次数逐渐减少, 生成的候选项集与频繁项集的数量均大幅减少, 各算法时间开销相当, 其运行时间都较短, 然而两种改进算法的运行时间依然小于 Apriori 算法. 这是由于 Apriori 算法含有 C_k 中连接、剪枝的时间开销, 而改进算法则对此予以了简化.

改进算法 Improve_Apriori_1 通过构造辅助表减少访问表中的无效记录来大幅降低访问数据库的次数, 从而提升运算效率. 另外, 将由事务中包含的项目情况生成的数据库表装入内存中, 之后的扫描过程无需再访问数据库, 而是直接访问内存以减少 I/O 开销, 提高访问速度.

改进算法 Improve_Apriori_2 采用对项集事务列表求交集的策略减少扫描数据库的次数, 使算法达到较高效率. 该算法全过程只扫描一次事务数据库, 而 Apriori 算法则反复扫描数据库致使 I/O 开销较大; 计算候选 k -项集 $C_k (k \geq 2)$ 各项集的支持度时, 改进算法 Improve_Apriori_2 只需访问频繁 $(k-1)$ -项集 $L_{k-1} (k \geq 2)$ 即可, 而且随着 k 值的增加, 所生成 $L_{k-1} (k \geq 2)$ 项集总数会越来越小, 其访问的数据量也相应减少. 此外, 根据性质 1, 对频繁 $(k-1)$ -项集 $L_{k-1} (k \geq 2)$ 求候选 k -项集 C_k 前, 先判断 L_{k-1} 中项集的个数, 若项集个数小于 k , 则 L_{k-1} 不能产生 L_k , 返回 $C_k = \emptyset$. 此时不再需要进行自连接操作来产生 C_k , 这样不仅节省了产生 C_k 的时间, 也节省了判断 C_k 中的项集是否为频繁项集的时间.

6 结束语

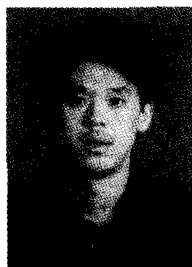
本文在 Apriori 算法的基础上, 针对其存在的

不足之处予以改进,提出了较为优化的算法.实验结果表明,所提出的两种改进算法能有效压缩搜索空间,减少扫描数据库的次数,缩减对不必要事务的扫描时间,提高频繁项集的生成效率,其性能较传统的 Apriori 算法有显著提高.关联规则挖掘的应用非常广泛,如何进一步优化算法并使其适合于不同条件下应用等将是作者下一步的主要研究目标.

参考文献:

- [1] Agrawal R, Srikant R. Fast algorithm for Mining association rules in large databases[C]//Proceedings of the 20th VLDB Conference, Santiago, Chile: IEEE, 1994: 487-499.
- [2] 李晓虹,尚晋.一种改进的新 Apriori 算法[J].计算机科学,2007,34(4):196-198.
- [3] 洪梓璇,边馥苓.基于支持度矩阵的 Apriori 改进算法[J].武汉大学学报:信息科学版,2008,33(12):1246-1249.
- [4] 崔贯勋,李梁,王柯柯,等.关联规则挖掘中 Apriori 算法的研究与改进[J].计算机应用,2010,30(11):2952-2955.
- [5] 付沙,廖明华,宋丹.基于压缩矩阵方式的 Apriori 改进算法[J].微电子学与计算机,2012,29(6):28-32.
- [6] 熊平.数据挖掘算法与 Clementine 实践[M].北京:清华大学出版社,2011.
- [7] 赵松,孙立铸. Apriori 算法的改进[J].哈尔滨理工大学学报,2007,12(5):94-97.
- [8] 崔贯勋,李梁,王柯柯,等.基于改进 Apriori 算法的入侵检测系统研究[J].计算机工程与科学,2011,33(4):40-44.

作者简介:



付沙男,(1980—),硕士,讲师.研究方向为信息系统安全、数据挖掘.

周航军男,(1979—),博士,讲师.研究方向为并行与分布式计算.

(上接第 109 页)

- [4] 魏宁.无线传感器网络的三维空间的覆盖与目标定位问题研究[D].西安:西安电子科技大学,2009.
- [5] 刘维亭,范洲远.基于混沌粒子群的无线传感器网络覆盖控制优化[J].计算机应用,2011,31(2):338-340.
- [6] Lu M, Wu J, Cardi M, et al. Energy-efficient connected coverage of discrete targets in wireless sensor networks [C]//Proc of the International Conference on Computer Net - works and Mobile Computing . Zhang - jiajie, China: IEEE, 2005: 43-52.
- [7] 王开郁.遗传算法在无线传感器网络中的应用[J].信息技术,2008,32(7):81-84.

作者简介:

薛文涛男,(1974—),博士,副教授,硕士生导师.研究方向为人工智能、无线传感器网络.

王丁章男,(1987—),硕士研究生.研究方向为先进控制理论及应用.