DOI:10.13657/j.cnki.gxkxyxb.2013.01.001

广西科学院学报
Journal of Guangxi Academy of Sciences
2013,29(1):1～3
Vol. 29,No. 1 February 2013

一种改进的 Apriori 算法
An Improved Apriori Algorithm

符丽锦,覃 华,邓 海,孙 欣
FU Li-jin,QIN Hua,DENG Hai,SUN Xin

(广西大学计算机与电子信息学院,广西南宁 530004)
(School of Computer and Electronics Information,Guangxi University,Nanning,Guangxi,530004,China)

摘要:分析 Apriori 算法的不知,提出从先删减后连接的新角度来生成频繁项集,达到减少无用连接,进而减少剪枝步骤候选项集判断数量来改进 Apriori 算法。改进后的 Apriori 算法在时间效率上优于传统的算法,而且所获得的关联规则质量与传统算法相当。

关键词:Apriori 算法 关联规则 频繁项集 删减
中图法分类号:TP311 **文献标识码**:A **文章编号**:1002-7378(2013)01-0001-03

Abstract:Generating frequent itemsets from a new view of the connection after the deletion can reduce a lot of useless connections,thereby it can reduce the number of the candidate set to be determined in the pruning steps,and improve Apriori algorithm. Experimental results confirm that the improved Apriori algorithm is more time-efficient than traditional algorithms while the quality of association rules is the same as that of traditional algorithms.

Key words:Apriori algorithm,association rules,frequent itemsets,deletion

Apriori 算法是发现关联规则领域的经典算法,但是对于大规模数据库,如果用经典的 Apriori 算法(以下简称 Apriori 算法)进行关联规则挖掘,将是一项非常耗时的工作。所以很多学者都想从不同角度来优化和改进 Apriori 算法。文献[1,2]提出先约简候选项目集的数目,达到减少 Apriori 扫描数据库次数的目的。文献[3]把 Apriori 算法与 FP-Tree 的结构相结合,提出一种增量式的 Apriori 算法。该算法不需要递归计算产生候选模式集和子模式树,计算速度比的 Apriori 算法快,与 FP-Growth 算法相当。文献[4]把粗糙集理论与 Apriori 算法相结合,用粗糙属性约简算法提取特征集,再使用 Apriori 算法在特征集上挖掘出关联规则。文献[5]使用 Hash 结构来产生频繁项集 L_2,并使用一种高效的数据存储与表示方法来提高 Apriori 算法的速度。文献[6]提出一种只扫描一次数据库的 Apriori 算法,避免了计算机重复多次遍历数据库。文献[7]提出一种分割数据库的思想,即在分割的数据库上设计 Apriori 算法,获得比传统算法更好的效果。文献[8]提出一种新的剪枝方法来提高 Apriori 计算效率。文献[9]通过在小规模的兴趣集上挖掘关联规则来提高规则产生的效率。

本文分析了 Apriori 算法的不足,并提出从先删减后连接的新角度来生成频繁项集,达到减少无用连接,进而减少了剪枝步骤候选项集判断数量来优化 Apriori 算法。实验结果表明,改进的 Apriori 算法可以有效地减少连接步骤产生的大量无用项集,进而减少了剪枝步骤候选项集判断的数量,比 Apriori 算法在时间效率上的优势更明显。

1 Apriori 算法分析

Apriori 算法使用逐层搜索的迭代方法寻找频繁项集,而且频繁项集的寻找是由连接和剪枝这两个关键步骤来完成。算法第一步是连接:通过 L_{k-1} 与自身连接产生候选 k-项集的集合 C_k。连接的规则是,如果 L_{k-1} 中有两个项集 l_1 和 l_2,它们的前 $k-2$ 项相同,且 l_2 的最后一项大于 l_1 的最后一项(目的是确保不产生重复项集),则 l_1 和 l_2 可以连接到一

收稿日期:2012-12-10
修回日期:2012-12-20
作者简介:符丽锦(1988-),女,硕士研究生,主要从事电子商务系统应用技术研究。

起。由于 C_k 中的候选项集中既有频繁项集也有非频繁项集，而且所有的频繁项集都包含在 C_k 中。为确定 C_k 中的所有频繁项集，必须扫描数据库并计算每个候选集的支持度，从而确定不小于最小支持度的频繁项集 L_k。第二步是剪枝：由 Apriori 性质可知，任何频繁 k - 项集的子集肯定都是频繁项集。所以，只要检测每个候选 k - 项集的所有 $(k-1)$ - 项子集是否都在 L_{k-1} 中，就可以判断候选项集是否是频繁的，从而可以通过减少 C_k 的数量，压缩搜索空间，提高挖掘效率。分析 Apriori 算法可以发现，为了生成候选 k - 项集 C_k，在连接步骤中需要进行大量的比较，而连接产生的部分项集可以经过关联规则性质确定其是不是候选项集，但是在确定之前即进行剪枝步骤，仍然需要判断其子项集是否都在频繁 $(k-1)$ - 项集 L_{k-1} 中。这些步骤需要花费大量的时间，如果由连接生成的项集都能保证是候选项集，那么就可以省去不必要的连接比较和剪枝步骤。

2 改进的 Apriori 算法

频繁 $(k-1)$ - 项集 L_{k-1} 生成候选项集 C_k 前，对 L_{k-1} 中的每项进行扫描，记下项集：

$\{l_1[1]\},\{l_1[1],l_1[2]\},\cdots,\{l_1[1],l_1[2],\cdots,l_1[k-2]\},\{l_2[1]\},\{l_2[1],l_2[2]\},\cdots,\{l_2[1],l_2[2],\cdots,l_2[k-2]\},\cdots$。

由 Apriori 性质可知，如果 l_i 属于 C_k，那么 C_k 的所有 $(k-1)$ - 项集就必须都出现在 L_{k-1} 中，所以 $\{l_i[1]\}$ 至少要出现 $(k-1)$ 次，$\{l_i[1],l_i[2]\}$ 至少要出现 $(k-2)$ 次，以此类推，$\{l_i[1],l_i[2],\cdots,l_i[k-1]\}$ 至少要出现 1 次。

设扫描得到的 $\{l_i[1]\}$ 出现的次数为 a_1，如果 $a_1<(k-1)$，则删除 L_{k-1} 中所有以 $l_i[1]$ 开头的项集，如果 $a_1\geqslant(k-1)$，那么比较 $\{l_i[1],l_i[2]\}$ 出现的次数 a_2，如果 $a_2<(k-2)$，则删除 L_{k-1} 中所有以 $\{l_i[1],l_i[2]\}$ 开头的项集，如果 $a_2\geqslant(k-2)$，则继续比较下一项。通过简单的数字比较，可以从 L_{k-1} 中删减大量的项集，最后得到 L_{k-1}'，这样可以大大减少不必要的项集连接。再将 L_{k-1}' 自连接获得 k - 项集，只需要比较一次就可以判断他是否属于候选项集 C_k。

改进 Apriori 算法的描述：

输入：事务数据库 D 和最小支持阈值 min_sup；

输出：D 中的频繁项集 L。

步骤：

(1) L_1 = find_frequent_1 - itemset(D)；//生成频繁 1 - 项集；

(2) L_2 = find_frequent_2 - itemset (L_1)；//由频繁 1 - 项集生成频繁 2 - 项集；

(3)for ($k=3;L_{k-1}\neq\varnothing;k++$) do begin；

(4) L_{k-1}' = optimize_apriori_gen (L_{k-1})；//

(5) C_k = apriori_gen (L_{k-1}')；// 由 L_{k-1}' 生成候选 k - 项集；

(6)for all transaction $t\in$ D do begin//扫描数据库，以确定每个候选项集的支持频度；

(7) C_t = subset (C_k, t)；

(8)for all candidate $c\in C_t$ do

(9)c. count++；

(10)end；

(11) $L_k=\{c\in C_k\mid$c. count>=min_sup$\}$；

(12)end；

调用 optimize_apriori_gen (L_{k-1}) 方法，对 L_{k-1} 中每一项 l_i 的 $\{l_i[1]\}$、$\{l_i[1],l_i[2]\}$、…、$\{l_i[1],l_i[2],\cdots,l_i[k-2]\}$ 累计其各出现的次数，再根据改进方法进行数字比较，删剪不必要的连接项集，最后得到 L_{k-1}'。

3 实验验证

使用 Java 语言在 Eclipse 开发软件下编写实现算法，实验平台选 PC(Intel Core 2，CPU 2.13GHz，内存 2GB)，操作系统 Windows7。实验数据集使用 UCI 公共数据库(http://kdd.ics.uci.edu/)提供的 MONK's Problems 和 Balance Scale 数据集。

MONK's Problems：MONK 问题数据集，是不同学习算法进行比较结果的数据集。该数据集共有 423 条数据实例，每个实例有 6 个属性，每个属性有 2～4 个取值，将每个不同取值的属性作为一个事务项，展开数据集，从而形成 17 项布尔数据集。

Balance Scale：天平实验数据集，是有关天平物理实验结果的数据集。该数据集共有 624 条数据实例，每个实例有 4 个属性，每个属性有 5 个取值，将每个不同取值的属性作为一个事务项，展开数据集，从而形成 20 项布尔数据集。

所有实验结果都取 10 次实验的平均值，详见表 1、表 2、表 3 和表 4。从表 1 结果可以看出，当支持度为 0.1 时，Apriori 改进算法比传统 Apriori 算法挖掘时间效率提高 74.28%；当支持度为 0.08 时，Apriori 改进算法比 Apriori 算法挖掘时间效率提高 42.88%；在支持度为 0.06 和 0.04 时，Apriori 改进算法比 Apriori 算法挖掘时间效率分别提高了

42.22%和 41.66%。

表 1 MONK's Problems 数据集的实验结果

支持度	运行时间(s)		时间相对提高百分比(%)
	Apriori 算法	改进的 Apriori 算法	
0.10	0.817	0.497	74.28
0.08	1.453	0.830	42.88
0.06	1.530	0.884	42.22
0.04	3.308	1.930	41.66

表 2 Balance Scale 数据集的实验结果

支持度	运行时间(s)		时间相对提高百分比(%)
	Apriori 算法	改进的 Apriori 算法	
0.100	0.452	0.280	38.05
0.060	0.471	0.305	35.24
0.020	1.951	1.114	42.90
0.008	6.587	3.918	40.52

表 3 基于 MONK's Problems 数据集的频繁项集挖掘质量

支持度	频繁项集	频繁集数量		相同度(%)
		Apriori 算法	改进的 Apriori 算法	
0.10	L1	19	19	100
	L2	110	110	100
	L3	29	29	100
0.08	L1	19	19	100
	L2	150	150	100
	L3	140	140	100
0.06	L1	19	19	100
	L2	150	150	100
	L3	178	178	100
	L4	16	16	100
0.04	L1	19	19	100
	L2	151	151	100
	L3	476	476	100
	L4	204	204	100

表 4 基于 Balance Scale 数据集的频繁项集挖掘质量

支持度	频繁项集	频繁集数量		相同度(%)
		Apriori 算法	改进的 Apriori 算法	
0.100	L1	22	22	100
	L2	20	20	100
0.060	L1	23	23	100
	L2	32	32	100
0.020	L1	23	23	100
	L2	190	190	100
	L3	126	126	100
0.008	L1	23	23	100
	L2	210	210	100
	L3	742	742	100
	L4	180	180	100

从表 2 结果也可以看出,当支持度分别为 0.1、0.06、0.02、0.008 时,改进的 Apriori 算法比 Apriori 算法挖掘时间效率分别提高了 38.05%、35.24%、42.90%、40.52%。表 3 和表 4 的结果表明,Apriori 算法和改进算法在 MONK's Problems 和 Balance Scale 这两个数据集上挖掘频繁项集得到的数量和质量是一致的。

上述结果充分说明本文的改进算法可以在较短的时间里挖掘出数据库中的所有频繁项集,而且所获得的关联规则质量与传统算法相同。由于改进算法是从先删减后连接的新角度来生成频繁项集,可以减少大量的无用连接,进而减少了剪枝步骤候选项集判断的数量,节省了大量的比较判断时间,明显地提高了频繁项集的挖掘效率。但是该改进算法仍然不能对 Apriori 算法产生大量候选项集和需要多次重复扫描数据库两大缺陷进行改进,为进一步提高 Apriori 算法的计算效率,还需要更深入的研究。

参考文献:

[1] Chang Rui, Liu Zhiyi. An improved apriori algorithm [C]. 2011 International Conference on Electronics and Optoelectronics (ICEOE), 1: 476-478.

[2] Liu Jing, Lu Yongquan, Wang Jintao, et al. An improved Apriori algorithm for early warning of equipment failue[C]. 2nd IEEE International Conference on Computer Science and Information Technology, 2009: 450-452.

[3] Wu Bo, Zhang Defu, Lan Qihua, et al. An efficient frequent patterns mining algorithm based on Apriori algorithm and the FP-Tree structure[C]. Third International Conference on Convergence and Hybrid Information Technology, 2008, 1: 1099-1102.

[4] Chen Chuxiang, Shen Jianjing, Chen Bing, et al. An improvement Apriori arithmetic based on rough set theory[C]. 2011 Third Pacific-Asia Conference on Circuits, Communications and System, 2011: 1-3.

[5] Sun Dongme, Teng Shaohua, Zhang Wei, et al. An algorithm to improve the effectiveness of Apriori[C]. 6th IEEE International Conference on Cognitive Informatics, 2007: 385-390.

[6] Zhang Yan, Chen Jing. VI: Based on the vertical and intersection operation of the improved Apriori algorithm [C]. 2nd International Conference on Future Computer and Communication, 2010, 2: 718-721.

[7] Wang Guofeng, Yu Xiu, Peng Dongbiao, et al. Research of data mining based on Apriori algorithm in cutting database [C]. International Conference on Mechanic Automation and Control Engineering, 2010: 3765-3768.

[8] Chen Zhuang, Cai Shibang, Song Qiulin, et al. An improved Apriori algorithm based on pruning optimization and transaction reduction [C]. 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, 2011: 1908-1911.

[9] Libing Wu, Gong K, Guo F, et al. Research on improving Apriori algorithm based on interested table [C]. 3rd IEEE International Conference on Computer Science and Information Technology, 2010, 3: 422-426.

(责任编辑:尹 闯)