一种基于动态散列和事务压缩的关联规则挖掘算法

崔 亮 郭 静 吴玲达
(装备学院复杂电子系统仿真实验室 北京 101416)

摘 要 关联规则挖掘搜索给定数据集中反复出现的数据模式,找到它们之间的相关性。分析了经典Apriori算法存在的时空效率低的缺点和数据形式对算法效率的影响。提出一种基于动态散列和事务压缩技术的改进,动态应用散列技术减小候选频繁项集的规模和数据库扫描次数,应用事务压缩技术缩小数据库中事务量的长度和总数,从而提高了算法的时间空间效率。与Apriori算法进行的比较验证了新算法的正确性与效率。

关键词 关联规则,频繁模式,动态散列,事务压缩
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.009

Algorithm for Mining Association Roles Based on Dynamic Hashing and Transaction Reduction

CUI Liang GUO Jing WU Ling-da
(Science and Technology on Complex Electronic System Simulation Laboratory, Equipment Academy, Beijing 101416, China)

Abstract Mining association rules search frequent data patterns in given dataset, and find out the correlation between them. This paper analyzed the shortcomings of the classical Apriori algorithm's efficiency in time and space, and the effect of data form on algorithm's efficiency. An effective algorithm based on dynamic hashing and transaction reduction technique was proposed, and it was compared with Apriori algorithm. Experimental results verify the correctness and effectiveness of the algorithm.

Keywords Association roles, Frequent patterns, Dynamic hashing, Transaction reduction

1 引言

随着大数据时代的来临,数据挖掘对信息产业界的重要性和影响力越来越大。将数据转换成有用的信息和知识的需求更加迫切。关联规则挖掘作为数据挖掘的重要部分,能为我们揭示数据之间的关联、相关性和许多其他的联系。此外,关联规则挖掘对数据分类、聚类和其它数据挖掘任务也有很大促进与影响,因此一直是数据挖掘研究领域的一个重要课题。最基本的关联规则挖掘算法Apriori算法[1]是布尔关联规则挖掘频繁项集的原创性算法,通过限制候选产生发现频繁项集,使用了逐层搜索的迭代方法[2]。但是其时空效率并不出色。另一类基于频繁模式树的算法也取得了很大进展[3]。近来的研究主要针对于这两大类算法进行改进。对Apriori算法的改进基本可以分为以下几大方向:(1)基于散列的技术[4];(2)基于事务压缩技术[5];(3)基于划分的技术[6];(4)基于选样的技术[7];(5)基于动态项集的技术[8]。此外,另一些研究致力于如何减少规则数量和表示规则,使得挖掘出的规则便于理解和运用[9]。

2 问题分析

2.1 基本概念

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是数据项的集合,数据记录 T_i 也是数据项的集合,且 $T_i\subset X$,再设 X 是 I 的子集,记录 T_i 包含项集 X。关联规则 $X\Rightarrow Y$ 的表示形式为:$X\rightarrow Y$,支持度为 s,置信度为 c。其中 X,Y 是同一记录内的数据项,且 $X\cap Y=\emptyset$,s 和 c 是规则的支持度和置信度。支持度和置信度取值都在0和1之间。其中 s 是包含 $X\cup Y$ 项的事务记录占数据集的百分数,记作 $Support(X\rightarrow Y)=P(X\cup Y)$;$c$ 是在一条记录包含 X 的情况下,同时也包含 Y 的可能性,记作 $Confidence(X\rightarrow Y)=P(Y|X)$。同时满足最小支持度阈值和最小置信度阈值的规则称为强关联规则。符合最小支持度阈值限制的项集称为频繁项集。关联规则挖掘中决定性能最重要的部分是挖掘频繁项集[2]。

2.2 Apriori算法

Apriori算法是一种布尔关联规则挖掘频繁项集的原创性算法。算法使用逐层搜索的迭代算法,k-1-项集用于搜索 k-项集。首先,通过扫描数据库,统计每个项的计数,得到满足最小支持度的项,它们的集合中项长度为1,记为 L_1。然后使用它找出频繁2-项集的集合 L_2。如此迭代下去,直到不能再找到频繁 k-项集为止。可以使用先验性质压缩搜索空间。在使用 L_{k-1} 找出 L_k 时,具体分为两步[2]:

1)连接步:将 L_{k-1} 的项集两两互相连接产生候选k-项集的集合 C_k。

到稿日期:2014-06-13 返修日期:2014-08-11

崔 亮(1990-),男,硕士生,主要研究方向为多媒体信息系统、数据挖掘,E-mail:icuiliang@gmail.com;郭 静(1977-),女,助理研究员,主要研究方向为系统科学,E-mail:gquiet@163.com;吴玲达(1962-),女,博士,教授,博士生导师,CCF会员,主要研究方向为多媒体信息系统、虚拟现实,E-mail:wld@nudt.edu.cn。

2)剪枝步：C_k 是 L_k 的超集，对于其中符合先验性质的，扫描数据库，确定每个候选的计数，从而确定 L_k。

Apriori 算法的缺点：可能需要产生大量候选项集；可能需要重复地扫描数据库，通过模式匹配检查一个很大的候选集。例如，若有 10^4 个频繁 1-项集，则 Apriori 算法需要产生多达 5×10^7 个候选 2-项集，在对如此大的候选集合进行匹配检查时需要重复扫描数据库，算法复杂度很高。

3 散列与事务压缩技术的改进

为了提高 Apriori 算法的性能，很多学者以散列和事务压缩技术为基础进行了大量的研究工作，提出了很多改进的算法。

3.1 基于散列的技术的改进

可以用基于散列的技术[4]来压缩候 k-项集 C_k 所占的空间。当遍历数据库中的每个事物，由候选 1-项集 C_1 产生项集 L_1 时，我们可以对每个事务生成其所有的 2-项集，将它们散列到表结构的不同桶中，并增加对应的统计数。在散列表中，对应的桶计数低于支持度阈值的 2-项集不可能是频繁 2-项集，因而应当从候选项集中删除。这种基于散列的技术可以大大压缩要考察的 k-项集。事务压缩技术压缩进一步迭代扫描遍历的事务数：不包含任何 k-项集的事务不可能包含任何 $(k+1)$-项集。这样，在其后考虑这种事务时，其可以加上标记或删除，因为产生 j-项集（$j>k$），扫描数据库时不再需要它们。

散列技术在产生 k-候选项集时，把数据库中的每一条事务分解为 k-项集，然后放入桶中。在产生 2-项集时，效率较高。然而，随着 k 的增大，需要放入桶中的 k-项集也在逐渐增大。对于数据库中一条包含 m 个项的事务，它需要产生 C_m^2 个 2-项集及 C_m^3 个 3-项集。假如最终一直产生到 m-项集，那么它一共需要产生 2^m 个候选项集。

例如，对于包含 30 个项的事务，可以得到 $C_{30}^2=435$ 个 2-项集、$C_{30}^3=4060$ 个 3-项集，$C_{30}^4=27405$ 个 4-项集。这种候选集在一定范围内呈指数增长。所以尽管在桶中会排除一部分，但是得到候选项集的开销也非常大，从而严重影响效率；反之，仅仅使用连接的方法进行候选项集的生成，1 项集的大小为 n，2 项集的大小为 $n^2/2$，此后 k 项集的大小绝对不会大于 $k-1$-项集，时间上的开销是收敛并且逐渐减小的。

综合这两种方法可以看出，随着 k 的增大，用散列的方法产生候选项集的开销可能会大于 Apriori 方法。如果动态地选择两种产生候选项集的方法，最终效果是优于只使用其中任何一个的。并且，Apriori 方法产生的候选项集与具体数据库中事务的特征无关，效率并不依存于事务的平均长度（其包含的项的数量）。散列的方法要分解每一条事务，其效率与数据库中事务的特征息息相关，如果平均长度很短，那么效率就很高；如果遇到的事务长度不一并且相差很大，那么效果就可能很差。

3.2 基于事务压缩技术的改进

事务压缩技术可以动态删减数据的大小，和数据库中每一条事务的长度。例如，对于一个项集大小为 p 的数据库，得到的频繁 1-项集大小为 q。那么就可以从数据库的每一条事务中把剩余 $p-q$ 个事务删去，从而显著减小事务长度。每一次连接剪枝结束后，得到了频繁 k 项集。可以扫描数据库，任意一条不包括频繁 k 项集的事务显然也不会包含频繁 $k+1$ 项集，把它们删去，也会减小数据的大小。

基于散列技术的改进着手于减少候选项集的大小，并更进一步减少数据库扫描的次数。而事务压缩技术能够逐渐减小数据库中事务的总量和长度。这两种方法有着天然的联系，能够作用于关联规则挖掘过程的不同部分，并且会促进对方的效果，而不产生负面影响。所以，将这两种方法结合起来能够得到一种新的更优秀的方法。

4 DHTR 算法

综合以上的分析，我们提出了一种动态散列事务压缩频繁规则挖掘算法（Dynamic Hashing and Transaction Reduction，DHTR）。

4.1 算法思想

使用逐层搜索的迭代方法，先扫描数据库，产生频繁 1-项集。然后在迭代时分别估计散列数据库事务和 Apriori 算法中连接 $k-1$-项集产生的候选 k-项集大小，选择时空效率更高的方法来产生候选项集。然后再进行剪枝步，验证先验性质，统计它们在数据库中的计数，得到频繁 k-项集。并对数据库中的事务进行标记压缩，删去没有包含任何频繁 k-项集的事务。

在产生候选项集时，选择连接的方法是通过散列技术，其判断依据是两种方法产生候选项集的大小。散列技术产生的候选项集初始时较小，但随着项长度的增加逐渐增大。连接方法产生候选项集初始时大，然后逐渐减小，每一步产生的候选 k-项集的大小不会大于 $k-1$-项集。DHTR 方法在动态选择时对这两种方法可能产生的候选项集大小进行估计。图 1 所示为 DHTR 算法流程。

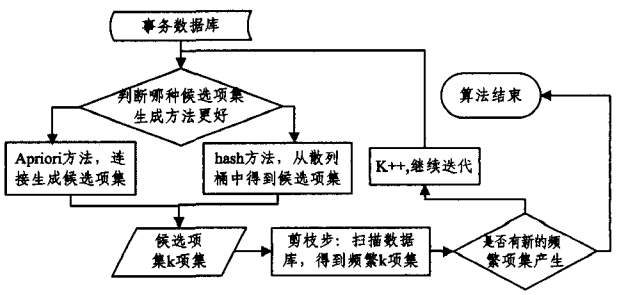

图 1 DHTR 算法流程

4.2 算法描述

输入：事务数据库 D，最小支持度计数 min_sup

输出：D 中的频繁项集 L

```
1. L1 = find_frequent_1_itemsets(D);
2. FOR(k=2;Lk-1≠∅;k++) {
3. IF hash_is_better(k)
4. Ck= hash _gen(Lk-1,min_sup);
5. Ck= apriori_gen(Lk-1,min_sup);
6. FOREACH transaction t∈D {//扫描数据库进行计数
7. IF Ct. lable<k-1 { delete t from D ;BREAK;} // 事务压缩
8. Ct=subset(Ck,t); //得到 t 在候选项集中的子集
9. FOREACH candidate c∈Ct{
10. c. count++;//计数
11. IF c. count >= min_sup c. lable=k;
```

12. }
13. }
14. $L_k=\{c\in C_k \mid c.count \geqslant min\_sup\}$
15. }
16. RETURN $L=\cup_k L_k$;

函数说明：

procedure apriori_gen(L_{k-1}:frequent($k-1$)itemset,min_sup:support) 用 apriori 方法得到产生候选 k-项集。

procedure hash_gen(L_{k-1}:frequent($k-1$)-itemset;min_sup:support) 用 hash 方法产生候选 k-项集。

procedure has_infrequent_subset(c: candidate-itemset; L_{k-1}:frequent($k-1$)-itemset)验证先验性质，是否有非频繁子集。

procedure hash_is_better(k:length of k-itemset;L_{k-1} frequent($k-1$)-itemset)估计两种方法哪个效率更高。

5 算法运行举例

设有表 1 所列的事务数据库 D，最小支持度为 40%，求频繁项集。

表 1 事务数据库

Tid	项集
100	1,2,3,4,5,6
200	6,7,8,9,10,11,12,
300	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
400	1,2,3,8
500	2,4,5,9
600	2,5,10

扫描数据得到频繁项集{1,2,3,4,5,6,7,8,9,10}，它们出现的次数都大于 $min\_sup * |D| = 2.4$。然后采用 DHTR 算法，判断应用 Hash 方法产生候选 2-项集的效率更高。把数据中的事务散列到桶中进行计数，然后进行剪枝，{(1,2),(1,3),(2,3),(2,4),(2,5)}。然后迭代，求候选 3-项集。判断应用连接的方法产生候选 3-项集的效率更高。对频繁 2-项集用连接算法之后，得到的候选 3-项集为 {(1,2,3),(2,3,4),(2,3,5),(2,4,5)}。进行剪枝，扫描数据库，满足最小支持度限制的有{(1,2,3),(2,4,5)}，即频繁 3-项集。继续迭代，求候选 4-项集，判断用 Apriori 连接的方法的效率更高，连接步得到候选项集为空，算法结束。数据库的频繁项集集合为：{1,2,3,4,5,6,7,8,9,10,(1,2),(1,3),(2,3),(2,4),(2,5),(1,2,3),(2,4,5)}。

DHTR 算法在求 $k=3$ 时的候选项集时，如果选择 Hash 方法，由事务{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17}将得到 $C_{17}^3=680$ 条候选项集，$k=4$ 时达到 2380 条，彻底掩盖 Hash 方法的优点。

6 实验与验证

为了进一步验证 DHTR 算法的性能，用 C# 在内存为 2GB、CPU 为 inter core i3 2.0GHz、操作系统为 windows7 的计算机上实现了 Apriori 算法和 DHTR 算法。实验数据参照 UCI 机器学习库中的数据集数据。由于该数据集中的数据噪声较多，数据集大小较小不利于对比，因此本实验参照其数据特征进行了模拟，得到了平均长度不等、数量从 10000 到 40000 不等的数据集来进行试验。

如图 2 所示，最小支持度为 0.05，几组数据随着事务量增长，DHTR 算法的运行时间都明显小于 Apriori 算法。而且可以看出事务平均长度增加之后，DHTR 算法效率的提高更加明显。

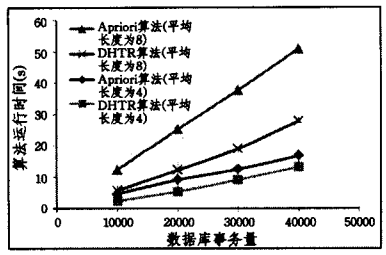

图 2 事务量和平均长度变化时算法效率对比

图 3 中，数据库事务量为 10000 条，平均长度为 8。可以看出，随着最小支持度的增加，两种算法的运行时间都在减小，而且 DHTR 算法优于 Apriori 算法。由于项集大小为 100，平均长度为 12，当最小支持度减小到 0.05 时，几乎所有事务都属于频繁项集，频繁项集规模迅速变大，算法运行时间呈指数增长，效率比较失去意义。

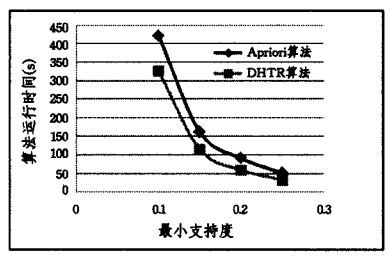

图 3 最小支持度变化时算法效率对比

由实验和分析可以得出如下结论：

(1)DHTR 算法极大改善了 Apriori 算法产生大量候选项集时的效率问题，显著地减少了候选 2-项集的规模，提高了产生候选集的速度。

(2)相对于 Apriori 算法的 Hash 技术改进，DHTR 能够极大地克服 k 增长时，因为数据库中少数事务长度过长而导致的候选项集爆发问题，并且其效率不依赖于桶规模的大小。

(3)DHTR 算法集合了事务压缩技术的优点，对数据库中的事务进行标记或者修改，能够一定程度上缩减 Apriori 算法需要频繁扫描数据库的时间开支。

结束语 挖掘频繁项集是进行关联规则挖掘的关键问题。本文充分分析了制约 Apriori 算法和基于 Hash 技术算法性能的原因。另外，提出了一种动态结合 Hash 方法和 Apriori 算法，同时采用事务压缩技术的改进 DHTR 算法。该算法能充分利用几种方法的特点，减小候选项集的规模与数据库扫描次数，并且克服了数据库中事务长度过长对 Hash 方法的影响，在长度变化时依然有效。理论和实验结果表明，DHTR 算法有更优越的性能，能有效提高频繁项集挖掘的效率。

参 考 文 献

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//Proc. 20th Int. Conf. Very Large Data Bases, VLDB. 1994,1215:487-499

[2] 韩家炜,坎伯. 数据挖掘:概念与技术[M]. 北京:机械工业出版社,2001:130-160
Han Jia-wei, Kamber M. Data Mining: Concepts and Techniques [M]. Beijing: China Machine Press, 2001: 130-160

[3] Yen S J, Wang C K, Ouyang L Y. A Search Space Reduced Algorithm for Mining Frequent Patterns[J]. Journal of Information Science and Engineering, 2012, 28(1): 177-191

[4] Park J S, Chen M S, Yu P S. An effective hash-based algorithm for mining association rules[J]. ACM SIGMOD Record, 1995, 24(2): 175-186

[5] Singh J, Ram H, Sodhi D J S. Improving Efficiency of Apriori Algorithm Using Transaction Reduction[J]. International Journal of Scientific and Research Publications, 2013, 3(1): 1-4

[6] Savasere A, Omiecinski E R, Navathe S B. An efficient algorithm for mining association rules in large databases[J]. ACM SIGMOD Record, 2000, 29(2): 1-12

[7] Toivonen H. Sampling large databases for association rules [C]// VLDB. 1996, 96: 134-145

[8] Brin S, Motwani R, Ullman J D, et al. Dynamic itemset counting and implication rules for market basket data[J]. ACM SIGMOD Record. ACM, 1997, 26(2): 255-264

[9] 孟军,王蓬,张静,等. 基于项集依赖的最小关联规则挖掘[J]. 计算机科学,2013,40(1):183-186,217
Meng Jun, Wang Peng, Zhang Jing, et al. Minimal Association Rules Mining Based on Itemset Dependency[J]. Computer Science, 2013, 40(1): 183-186, 217

(上接第 36 页)

5.2.4 推荐结果可信度的验证

基于以上推荐实验结果和原因,依旧根据大众点评网的数据来进行推荐结果的对比验证。由于大众点评网中的数据不一定是最优的结果,因此对于推荐结果的验证,采用可信度来衡量。由于商铺选址的复杂性较高,准确性较低,我们仅考虑了 4 个会对最终结果造成误差的影响因素,因此经过训练之后,当系统的整体可信度达到 50%时,我们认为推荐结果可信。根据推荐过程中可能遇到的情况,做如下情况定义。

情况一 如果某地点的原有商铺类型为 A,系统最优推荐商铺类型也为 A,实际评分 P_1 与最优推荐评分 P_2 的差距小于 10%,即如式(6)所示,则认为推荐结果可信。

$$\frac{|P_1-P_2|}{\min\{P_1,P_2\}}<10\% \tag{6}$$

情况二 令某地点的原有商铺类型为 A,系统最优推荐商铺类型为 B,实际评分为 P_1,最优推荐评分为 P_2,以及系统计算 A 类型的评分为 P_3,若任意两个评分差距都小于 10%,即如式(7)所示,则认为推荐结果可信。

$$\begin{cases}\dfrac{|P_1-P_2|}{\min\{P_1,P_2\}}<10\%\\ \dfrac{|P_1-P_3|}{\min\{P_1,P_3\}}<10\%\\ \dfrac{|P_2-P_3|}{\min\{P_2,P_3\}}<10\%\end{cases} \tag{7}$$

根据以上定义,随机选取 50 个地点,统计验证结果,如表 2 所列。

表 2 推荐结果验证

	可信结果数量		不可信结果数量		可信度	
	情况一	情况二	情况一	情况二	情况一	情况二
	9	23	10	8	47%	74%
合计	32		18		64%	

根据表 2 的数据,满足情况一的只有 17 个地点,而这 17 个地点的最优推荐结果是西餐类或者娱乐类,但是其可信度只有 47%;而情况二的可信度要比情况一的高,这也反映了最优推荐结果分布不均衡,并且各个推荐结果差距不是很大。将两种情况综合起来,可信度达到了 64%。因此根据多样性、竞争性、相关性、客流性 4 种商铺选址因素而实现的商铺选址推荐系统的推荐结果的可信度是符合设计期望的。

结束语 本文根据用户在 LBSN 上的签到数据,提出商铺选址的影响因素,并以此为基础实现了商铺选址推荐系统。

下一步,我们将更细粒度地挖掘用户签到特征和选址影响因素,包括签到用户的行为、情感特征以及用户的签到内容、评论等方面。

参 考 文 献

[1] 於志文,周兴社,郭斌. 移动社交网络中的感知计算模型、平台与实践[J]. 中国计算机学会通讯,2012,8(5):15-20
Yu Zhi-wen, Zhou Xing-she, Guo Bin. Aware computing model, platform and practice of Mobile social networks[J]. Communications of China Computer Federation, 2012, 8(5): 15-20

[2] 於志文,於志勇,周兴社. 社会感知计算:概念、问题及其研究进展[J]. 计算机学报,2012,35(1):16-26
Yu Zhi-wen, Yu Zhi-yong, Zhou Xing-she. Socially aware computing[J]. Chinese Journal of Computers, 2012, 35(1): 16-26

[3] Zheng Y. Location-based social networks: Users[M]// Computing with Spatial Trajectories. Springer, New York, 2011: 243-276

[4] Karamshuk D, Noulas A, Scellato S, et al. Geo-spotting: mining online location-based services for optimal retail store placement [C]// Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013: 793-801

[5] Jensen P. Analyzing the localization of retail stores with complex systems tools[M]// Advances in Intelligent Data Analysis VIII. Springer Berlin Heidelberg, 2009: 10-20

[6] Yu Z, Zhang D, Yang D. Where is the Largest Market: Ranking Areas by Popularity from Location Based Social Networks[C]// Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC). IEEE, 2013: 157-162

[7] http://dev.jiepang.com

[8] http://developer.baidu.com/map