

基于 Apriori 的快速剪枝和连接的新算法

李 雷, 黄 蓉

(南京邮电大学 自动化学院, 江苏 南京 210000)

摘 要:挖掘关联规则是目前数据挖掘领域热点研究话题之一。它的目的在于在数据库中挖掘有趣的关联规则。在关联规则分析及 Apriori 算法分析上,针对 Apriori 算法的瓶颈问题,许多有效的改进算法被提出。文中提出了 QPCA 算法。该算法利用矩阵分析的方法,仅需要扫描数据库一次,同时此算法优化了连接和剪枝操作,通过快速的剪枝和连接可以很快地获取最少的候选项集,避免了频繁项集之间的重复判断连接,因此大大提高了算法的效率。实验结果表明,该算法在挖掘时间上有很大提高。

关键词:关联规则;Apriori;QPCA;数据挖掘

中图分类号:TP301.6

文献标识码:A

文章编号:1673-629X(2014)05-0031-05

doi:10.3969/j.issn.1673-629X.2014.05.008

A New Quick Pruning and Connection Algorithm Based on Apriori

LI Lei, HUANG Rong

(College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210000, China)

Abstract: Mining of association rules is an important research topic in data mining field. Its purpose is to mine interesting associations in transaction database. For the analysis of association rules and Apriori algorithm principle, in view of the bottlenecks of Apriori algorithm, lots of improved algorithms are proposed. In this paper, put forward the QPCA. The algorithm uses the method of matrix analysis, only needs to scan the database once. At the same time, the algorithm optimizes the pruning and connection operation, which can quickly obtain less candidate itemsets by quick pruning and connection, avoiding duplication of judgment and connection between frequent itemsets. Thus it's greatly improving the efficiency of the algorithm. The experimental results show that the algorithm has great improvement in mining speed.

Key words: association rules; Apriori; QPCA; data mining

0 Introduction

With the arrival of the information age, the database information storage volume increases rapidly. In the face of huge data resources, people need a powerful tool to mine useful knowledge. Data mining is a new technique proposed in this background. Data mining^[1] is a multi-disciplinary research field, it is to extract unknown, potential, useful knowledge from large amounts of data. Judging from the current situation, the research of data mining is still in the extensive stage of exploration.

Association rule mining^[2] is one of the most active research methods in data mining. It is mainly to find out the correlation or related links from a large number of re-

cords in database. The most classic association rule mining algorithm proposed by R. Agralwal et al. is Apriori^[3] algorithm. The algorithm uses the method of layer-by-layer iterative search method to mine the correlation between the data. However, the algorithm repeatedly scans the source data, and results a large number of intermediate data so that the efficiency is greatly reduced. Since then, many improved algorithms had been proposed. For example, the DHP algorithm^[4] proposed by Park et al. using hash technique effectively improves the generation process of candidate sets; Partition algorithm^[5] based on partition principle proposed by Savasere et al. only needs to scan the database two times to mine frequent itemsets;

收稿日期:2013-07-09

修回日期:2013-10-16

网络出版时间:2014-02-11

基金项目:国家自然科学基金资助项目(61070234)

作者简介:李 雷(1958-),男,安徽砀山人,教授,研究方向为模糊数学和智能系统、信号处理、非线性分析;黄 蓉(1989-),女,江苏南通人,硕士研究生,研究方向为数据挖掘。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20140211.1452.020.html>

The frequent itemsets discovery algorithm proposed by Toivonen^[6] is based on the sampling, reduces the consumption of I/O; Literature [7] proposed a transaction reduction technique; Literature [8] proposed FP – Growth algorithm; Literature [9] proposed weighted algorithm based on probability.

This paper also starts from the defects of Apriori algorithm, reduces the times of scanning database, and saves a lot of candidate set generation process, so as to improve the efficiency of the algorithm.

1 The Concepts of Association Rules

Let T be a transaction data set, denoted as: $T = \{t_1, t_2, \dots, t_n\}$, $t_i (1 \leq i \leq n)$ is recorded data, let $I = \{i_1, i_2, \dots, i_n\}$ is the collection of data items, $i_j (1 \leq j \leq n)$ is a data item in T , each t_i is a subset of I , each record corresponds with a identifier TID. Set $X, Y \subseteq I$ and $X \cap Y = \emptyset$, says $X \Rightarrow Y$ is an association rule. If the rules are established in T , S is called the support of rules $X \Rightarrow Y$ in T , C is called the confidence of the rule, S and C are defined as follows:

$$\text{Support}(X \Rightarrow Y) = \frac{|\{t \mid t \text{ contains } X, Y\}|}{|T|} = P(X \cup Y)$$

$$\text{Confidence}(X \Rightarrow Y) = \frac{|\{t \mid t \text{ contains } X, Y\}|}{|\{t \mid t \text{ contains } x\}|} = P(Y \mid X)$$

Frequent Itemsets: Refers to the degree of support is not less than the minimum support threshold (Minsup) of the set.

Strong and weak rules: $\text{Support}(X \Rightarrow Y) \geq \text{Minsup}$, and $\text{Confidence}(X \Rightarrow Y) \geq \text{Minconf}$, said the association rules $X \Rightarrow Y$ is a strong rule, otherwise called $X \Rightarrow Y$ is a weak rule.

2 Analysis of Apriori Algorithm

Apriori algorithm is one of the most classic and influential association rule mining algorithms. The basic idea as follows:

(1) First scan the database D , get 1-candidate sets C_1 , and then calculate the items' support of C_1 in the database.

(2) Remove the item whose support is less than the Minsup, get 1-frequent sets L_1 .

(3) Get 2-candidate set C_2 by self-connected with L_1 , and then scan the database D again, calculate the item-

sets' support of C_2 , thus get L_2 . Repeat the steps of (2), (3) until not find frequent itemsets, and then get K -frequent itemsets L_K through connection and pruning.

From the above analysis can be found, Apriori algorithm uses the iterative method of layer-by-layer search, resulting in high dimensional frequent item sets through the low dimensional frequent item sets. In this way, the Apriori algorithm has two fatal performance bottlenecks:

(1) Scan the transaction database many times.

Every calculation sets needs to scan and compare all the records of database to determine whether to join L_K or not. For large databases, this scan will greatly increase the I/O overhead, and this overhead will show the geometric progression increase as the increase of records.

(2) May generate huge candidate sets, so time and space complexity of Apriori algorithm is higher.

3 QPCA Algorithm

3.1 The Relevant Theorems and Inferences

Theorem 1: Any nonempty subset of frequent itemsets is also frequent itemsets, infrequent itemsets superset is infrequent itemsets.

Corollary 1: If the number of itemsets is less than or equal to K frequent itemsets in L_K itemsets, then the set is the set of maximum frequent itemsets.

Corollary 2: For all the candidates of the $c \in C_K$, if they are generated with $L_{K-1} \propto L_{K-1}$, and $\exists t \in L_{K-1}$, so it can be generated by $L_{K-1} \propto L_1^{[10]}$.

Corollary 3: $\forall c \in C_K, S(c)$ can be got by the two elements of $S(x), S(y) (x \neq y)$ and $S_{(c)} = S_{(x)} \cup S_{(y)}^{[10]}$.

Corollary 4: If there is $j \in X$ making $|L_{K-1}(j)| < k - 1$ in K dimension data itemsets $X = \{i_1, i_2, \dots, i_k\}$, then X is not frequent itemsets. Where $|L_{K-1}(j)|$ represents the j th $K - 1$ dimension data itemsets^[11].

Corollary 5: If the L_{K-1} has elements of e includes the project P , which makes $|L_{K-1}(j)| < k - 1$, then all different from the elements of e and e connected to the K candidate dimensional itemsets are not be frequent item sets^[12].

3.2 The Main Idea of QPCA Algorithm

QPCA algorithm is improved in the pruning and the connection based on Apriori algorithm. Unlike Apriori algorithm, QPCA stores the data in the form of boolean matrix, and can obtain the support degree by directly op-

eration in the ranks of matrix, thus it can avoid the repeated scanning the database each time through the loop, greatly reducing the time overhead to scan database. In order to illustrate the superiority of the algorithm, give an example. For example: a transaction database as shown in Table 1, the minimum support degree is $2/8$.

Table 1 Transaction table TID

TID	Item
T_0	i_1, i_3, i_4, i_7
T_1	i_2, i_3, i_5
T_2	i_2, i_4, i_7
T_3	i_1, i_3, i_5, i_6
T_4	i_2, i_5
T_5	i_1, i_4, i_5
T_6	i_1, i_3, i_5
T_7	i_3, i_4, i_6

For a given transaction database, construct a matrix, its rows and columns represent the data item and transaction respectively. Then scan the database, if the items of i_1, i_2, \dots, i_k in j th transaction, then $A[i_1][j], A[i_2][j], \dots, A[i_k][j]$ are initialized to 1, otherwise, they are initialized to 0. Then count the number of “1” in each line, so can get the support of each item, delete the whole row that the items’ support is less then minsup, and then draw the 1-frequent sets as shown in Table 2.

Table 2 Storage structure of 1-frequent itemsets (L_1)

TID Item	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
i_1	1	0	0	1	0	1	1	0
i_2	0	1	1	0	1	0	0	0
i_3	1	1	0	1	0	0	1	0
i_4	1	0	1	0	0	1	0	1
i_5	0	1	0	1	1	1	1	0
i_6	0	0	0	1	0	0	0	1
i_7	1	0	1	0	0	0	0	0

Second, get 2-candidate sets by $L_1 \propto L_1$, for any two rows of the boolean matrix, bitwise AND operation, the number of “1” in result is the support of K -itemsets, then delete the itemsets that the support transaction number is less than the minimum support, then can reach the 2-frequent itemsets (L_2). The storage structure of 2-frequent itemsets (L_2) is shown as Table 3.

Table 3 Storage structure of 2-frequent itemsets (L_2)

TID Item	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$i_1 i_3$	1	0	0	1	0	0	1	0
$i_1 i_4$	1	0	0	0	1	0	0	0
$i_1 i_5$	0	0	0	1	0	1	1	0
$i_2 i_5$	0	1	0	0	1	0	0	0
$i_3 i_4$	1	0	0	0	0	0	0	1
$i_3 i_5$	0	1	0	1	0	0	1	0
$i_3 i_6$	0	0	0	1	0	0	0	1
$i_4 i_7$	1	0	1	0	0	0	0	0

And so on, remove the item sets which can’t be frequent K itemsets to the $K-1$ -frequent itemsets by corollary 4 and corollary 5. In this table, because of i_2, i_6, i_7 is appearing only once less than 2, so delete the itemsets which containing i_2, i_6 or i_7 , this can avoid many unnecessary candidate set. So can get Table 4 as follows.

Table 4 Storage structure of 2-itemsets after pruning itemsets (L_2')

TID Item	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$i_1 i_3$	1	0	0	1	0	0	1	0
$i_1 i_4$	1	0	0	0	1	0	0	0
$i_1 i_5$	0	0	0	1	0	1	0	0
$i_3 i_4$	1	0	0	0	0	0	0	1
$i_3 i_5$	0	1	0	1	0	0	0	0

Similarly, should delete the items correspondingly in L_1 . Here, also remove the items i_2, i_6, i_7 in 1-frequent itemsets, then can get L_1' as shown in Table 5.

Table 5 The remaining items after pruning in 1-frequent itemsets (L_1')

TID Item	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
i_1	1	0	0	1	0	1	1	0
i_3	1	1	0	1	0	0	1	0
i_4	1	0	1	0	0	1	0	1
i_5	0	1	0	1	1	1	1	0

Then, get K candidate itemsets using the corollary 4. Here, when connect L_{K-1} with L_1 , should connect the itemsets in L_{K-1}' with the item in L_1' which is larger than the last item of itemsets in L_{K-1}' , this can avoid duplicate items. Here connect $i_1 i_3$ with $i_4, i_1 i_3$ with $i_5, i_1 i_4$ with i_5 , and $i_3 i_4$ with i_5 , so get candidate sets $\{i_1 i_3 i_4\}, \{i_1 i_3 i_5\}, \{i_1 i_4 i_5\}, \{i_3 i_4 i_5\}$. Then by calculating the support of candidate sets delete the candidate sets which do not meet

the conditions, so get the 3-frequent sets $\{i_1 i_3 i_5\}$. This is to avoid big problems in the time overhead caused by pattern matching in Apriori algorithm.

As can be seen from the above example, QPCA algorithm follows the following three steps:

- (1) Give a transaction database, generate the Boolean matrix by scanning database.
- (2) For frequent itemsets L_{k-1}, L_1 , prune and generate L_{k-1}', L_1' . Then generate C_k by connecting L_{k-1}' with L_1' . Through counting the support of candidate sets, get k -frequent itemsets L_k .
- (3) Repeat step (2) until cannot produce frequent itemsets.

3.3 Algorithm Pseudo Code

Input: transaction database D ; minimum support threshold minsup.

Output: the database of all the frequent.

```

for ( i = 1; i < n + 1; i ++ ) //Build a Boolean matrix
{
    for ( j = 1; j < m; j ++ )
    if ( I_i in T_j ) set A[ij] = 1;
    else set A[ij] = 0;
}

L_1 = find_L_1(D, minsup);
for ( k = 2; L_{k-1} ≠ ∅; k ++ )
{
    L_{k-1}' = L_{k-1}
    L_1' = L_1 //Count the number of each item that appears in L_{k-1}

    for each item x ∈ L_{k-1}
    for each field f contains x
    { f.count ++
    if f.count ++ < k - 1
    delete ( L_{k-1}', x ) //Delete items that will not generate frequent itemsets in L_{k-1}', L_1'
    delete ( L_1', x )
    }

    for each item x ∈ L_{k-1}'
    for each item y ∈ L_1'
    {
        C_k = x ∞ y //L_{k-1}' connected with L_1' that the item of L_1' are larger than the last item of each itemset of L_{k-1}'

        for each candidates c ∈ C_k
        c.sup = | S(C) | ;
        if ( c.sup < minsup ) then delete c
    }
    return L_k
    S(L_k) = S(L_{k-1}') ∩ S(L_1')
}

```

```

}
return L = ∪_k L_k
Procedure find_L_1(D, minsup); // Obtain frequent 1-itemsets
C_1 = get_item(D) //Obtain the transaction database project
for each transaction t ∈ D
{
    for each item x ∈ t
    { x.count ++
    if x.count ++ < minsup
    delete x ;
    }
}
L_1 = { x | x ∈ C_1 ∧ x.count ≥ minsup }
return L_1

```

4 Simulation

In order to evaluate the QPCA algorithm, this paper uses the standard Mushroom data sets to test. The Mushroom is characterized by the frequent itemsets in dense distribution. In the case of large support will always produce a large number of frequent itemsets, which comprises 8 124 records, 23 attributes, the size of Mushroom is 465 kB. The experimental platform using Intel P4 2.5 G/512 MB 700 GB, the operating system is Windows7. The simulation environment is Matlab R2012a.

First select a different minimum support to test the algorithm execution time, and compared with the Apriori algorithm, the experimental results as shown in Fig. 1.

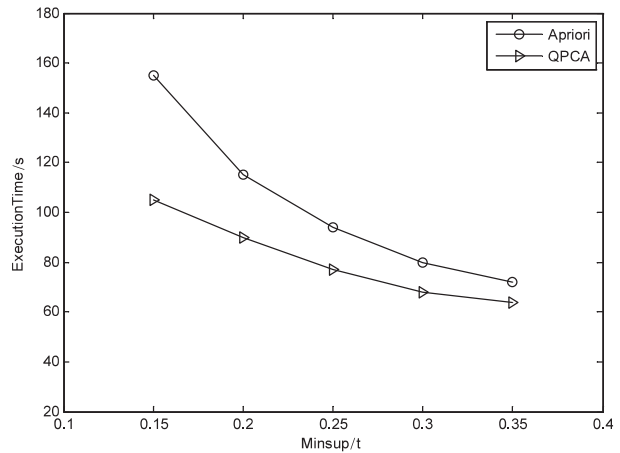


Figure 1 Comparison of execution time of two algorithms in different supports

From Fig. 1 can see the execution time of QPCA algorithm is obviously better than Apriori algorithm, and the support degree is small, the advantage is more obvious, which indicates that the improved algorithm is better than the Apriori algorithm.

Then consider the execution time of the two algorithms in different transaction data records, support degree is set to 0.3, then select 2 000 to 8 000 data records, test the execution time of two algorithms in different data size, the experimental results as shown in Fig. 2.

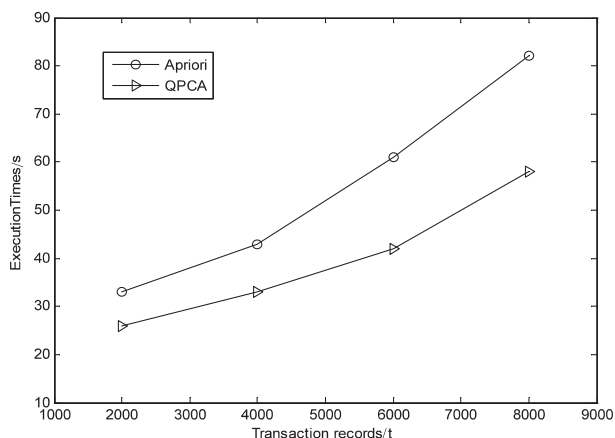


Figure 2 Comparison of execution time in different number of transaction records

As you can see from Fig. 2, with the data records increasing, the efficiency of the QPCA algorithm is superior to Apriori algorithm.

5 Conclusion

In this paper, based on the study of Apriori algorithm, point out the limitation of Apriori algorithm, and put forward QPCA algorithm, the algorithm only needs to scan the database once, and obtain candidate itemsets from fast pruning and connection to frequent itemsets by $L_{K-1} \propto L_1$, it also avoids repeated judgment to produce a large number of candidate itemsets, which greatly improves the efficiency of the algorithm. The simulation results show that, the algorithm has high efficiency.

参考文献:


[1] Chen M S, Han J, Yu P S. Data mining: An overview from a database perspective [J]. IEEE Transaction on Knowledge and Data Engineering, 1996, 8(6): 866-883.

[2] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases [C]//Proceedings of ACM SIGMOD international conference on

management and data. [s. l.]: [s. n.], 1993.

- [3] Agrawal R, Srikant R. Fast algorithms for mining association rules in large database [C]//Proceedings of the 20th international conference on very large data bases. San Francisco: Morgan Kaufmann Publishers, 1994: 487-499.
- [4] Park J S, Chen M S, Yu P S. An effective Hash based algorithm for mining association rules [C]//Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD-95). California, San Jose: [s. n.], 1995: 175-186.
- [5] Savasere A, Onieciniski E, Navathe S. An efficient algorithm for mining association rules in large databases [C]//Proceedings of the 21th international conference on very large databases (VLDB-95). Zurich, Switzerland: [s. n.], 1995: 432-443.
- [6] Toivonen H. Sampling large databases for association rules [C]//Proceedings of the 22th international conference of very large databases (VLDB-96). Bombay, India: [s. n.], 1996: 134-145.
- [7] Han J, Fu Y. Discovery of multiple-level association rules from large databases [C]//Proc of 1995 international conference on very large data bases. Zurich, Switzerland: [s. n.], 1995: 420-431.
- [8] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [C]//Proc of the 2000 ACM-SIGMOD international conference on management of data. Dallas, TX: ACM Press, 2001: 1-12.
- [9] Yin Qun, Wang Lizhen, Tian Qiming. A weighted association rules mining algorithm based on probability [J]. Computer Application, 2005, 25(4): 805-807.
- [10] Liu H T, Guo R X, Jang H. Research and improvement of Apriori algorithm of association rule mining [J]. Computer Applications and Software, 2009, 26(1): 146-149.
- [11] Liu Yongchang, Zhang Ping, Yan Weidong. Research the changed information extraction of TM image based on KL transform [J]. Computer Engineering and Application, 2002, 38(4): 69-71.
- [12] Cui Guanxun, Li Liang, Wang Keke, et al. Research and improvement of Apriori algorithm of association rule mining [J]. Computer Applications, 2010, 30(11): 2952-2955.

基于Apriori的快速剪枝和连接的新算法

作者: [李雷](#), [黄蓉](#), [LI Lei](#), [HUANG Rong](#)
作者单位: [南京邮电大学 自动化学院, 江苏 南京, 210000](#)
刊名: [计算机技术与发展](#) 
英文刊名: [Computer Technology and Development](#)
年, 卷(期): 2014(5)

本文链接: http://d.wanfangdata.com.cn/Periodical_wjfz201405008.aspx