Association Rules Mining Algorithm based on Matrix

Lei MA[1, a]

[1]Department of Media, Linyi University, Linyi, 276000, China

[a]email: malei1999@163.com

Keywords: Data Mining; Association Rules; Frequent Itemsets; Matrix;

Abstract. Association rules mining is an important model in data mining. The classic Apriori algorithm for Association rules need scan database many times and generate a great deal of candidate itemsets. This paper proposes a new algorithm for mining association rules based on matrix. The algorithm scans database only once to generate matrix of transaction and item, which generates candidate itemsets by using matrix. The experimental results of the algorithm show that the new algorithm is more efficient than traditional algorithm.

Introduction

Association rules was proposed by Agrawal in 1993[1], it is an important research field in data mining. Mining association rule reflects interesting association or correlation between data items. It is widely used in many industries [2-3] recent years. Mining frequent itemsets is the key problem in mining association rules, the most famous algorithm of mining frequent itemsets is Apriori and FP-Tree[4], Apriori algorithm generate frequent itemsets by candidate itemsets, FP-Tree algorithm generate frequent itemsets without candidate itemsets. But these two algorithms are inefficient in mining association rules. Many improved algorithms[5-8] have been proposed in recent years. These algorithms improve the efficiency of mining association rules, but there are also shortcomings.

Literature [9] proposed a sampling method to solve the problem of mining association rules, this algorithm mines local frequent itemsets in the subset of original database randomly, then calculates the global support according to local frequent itemsets support. The algorithm also proposes a mechanism to ensure that all global frequent itemsets are mined. The algorithm in Literature [10] reduces the times of scanning database, if a transaction record does not contain frequent k-itemset, then it does not contain frequent k+1-itemset, so we can delete the transaction after scanning. The algorithm in Literature [11] mines two frequent patterns and maximum frequent itemsets by building two support matrices, but its time and space complexity are very high. DFS algorithm in Literature [12] can mine frequent itemsets efficiently, but the algorithm generates errorous frequent itemsets. The improved algorithm in Literature [13] scans database and maps data to binary matrix based on compression transaction matrix multiplication, which generates frequent 1-itemset by binary matrix and corresponding auxiliary matrix multiplication, and so on other frequent itemset. The efficiency of the algorithm is lower because matrix multiplication spent more time. Apriori_M algorithm in Literature [14] need still scan database two times. The algorithm in Literature [15] adopts item array to mine frequent itemset, which generate many worthless candidate set. The algorithm in Literature [16] reduces candidate sets and improves the efficiency of the algorithm by interest items, but it is very difficult to find interest item.

This paper puts forward an improved algorithm based on matrix of transaction and item, which need scan database only once, which can reduce space overhead. The algorithm produces frequent 2-itemsets by support matrix, which solves the bottleneck problem of generating frequent 2-itemsets; new algorithm produces frequent itemsets by the value of row logic operation, which mines association rules efficiently.

Relevant Concepts and Theories

Definition 1 Let D = $\{T_1, T_2, \ldots, T_n\}$ be transaction database, item sets $I=\{I_1,I_2,\ldots,I_m\}$,which is all items of transaction database, where T_i $(1\leq i\leq n)$ is a transaction which contains a set of items in I, each transaction exists a flag, denoted as TID. Let A be an itemset, if $A\subseteq T_j$, then T_j contains A.

Definition 2 Let X be an itemset, the support of X is the proportion of transactions in D which contain X, denoted as sup(X); if sup(X) ≥min_sup, min_sup is given by user, then X is a frequent itemset, otherwise X is a non-frequent itemset. Itemset which contains k itemsets is called k-itemset. If the support of X which is a k-itemset is greater than or equal min_sup, then X is called frequent k-itemset. The set of all frequent k-itemsets is denoted by L_k.

Definition 3 If T_j contains item I_i, then the value of row j and column i is 1, else 0 in transaction and item matrix.

Definition 4 Let X be an itemset, if there exists sup(X)≥min_sup and $X\subset Y$ for item set Y, sup(Y)<min_sup, then X is called the maximum frequent itemset.

Definition 5 Let X,Y be itemset, there exists $X\subset I$, $Y\subset I$, and $X\cap Y=\varnothing$.The implication of the form X=>Y is called an association rules.

Definition 6 Let I_i be an item set, binary array of I_i is denoted by A_i. All item sets are sorted by lexicographic order.

Algorithm Description

We construct transaction and item matrix by scanning database. If T_j contains item I_i, then the value of row j and column I is 1, else 0 in the matrix.

Then we construct an upper triangular matrix when we generate matrix, the upper triangular matrix is defined as follows:

1) Rows represent item set $I=\{I_1,I_2,\ldots,I_m\}$, columns also represent I.

2) The value of UTM [i,j] in matrix represent the count of transactions which contains item set $\{I_i, I_j\}(i\leq j)$ in D.

Algorithm 1: generate transaction and item matrix and UTM:

Input: database D;

Output: transaction and item matrix M, UTM U

for each transaction T_j in D

{

if (T_j contains item I_i)

M[j,i]=1;

else

M[j,i]=0;

for ($\forall\{I_k, I_l\}\subseteq T_j$)

{

U[k, l] plus 1;

}

}

For example, transaction database D is shown in Table 1, minimum support is 2, D = $\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9\}$, items $I=\{I_1,I_2,I_3,I_4,I_5,I_6\}$.

Table 1. Transaction Database D

TID	Item
T_1	I_1,I_3,I_5
T_2	I_2,I_4,I_6
T_3	I_1,I_2,I_3,I_5
T_4	I_3,I_5,I_6
T_5	I_1,I_3,I_6
T_6	I_1,I_3
T_7	I_2,I_4,I_5
T_8	I_1,I_3,I_5
T_9	I_2,I_3,I_4, I_5,I_6

According to the definition of transaction and item matrix and UTM, we construct transaction and item matrix and UTM, matrix as shown in Table 2 and Table 3:

Table 2 Transaction and Item matrix M on D

	I_1	I_2	I_3	I_4	I_5	I_6
T_1	1	0	1	0	1	0
T_2	0	1	0	1	0	1
T_3	1	1	1	0	1	0
T_4	0	0	1	0	1	1
T_5	1	0	1	0	0	1
T_6	1	0	1	0	0	0
T_7	0	1	0	1	1	0
T_8	1	0	1	0	1	0
T_9	0	1	1	1	1	1

Table 3 Upper Triangular Matrix U on D

	I_1	I_2	I_3	I_4	I_5	I_6
I_1	5	1	5	0	3	1
I_2		4	2	2	3	2
I_3			7	1	5	3
I_4				3	2	2
I_5					6	2
I_6						4

In U, if the value of $[i,j](i\leq j)$ is greater than or equal to min_sup, item $\{I_i,I_j\}$ is frequent 2-itemset.We can easily get frequent 2-itemset from U. $L_2=\{\{I_1,I_3\},\{I_1,I_5\},\{I_2,I_3\},\{I_2,I_4\},\{I_2,I_5\},\{I_2,I_6\},\{I_3,I_5\},\{I_3,I_6\},\{I_4,I_5\},\{I_4,I_6\},\{I_5,I_6\}\}$.

There exist $\{I_i,I_j\}$ and $\{I_i,I_k\}$ that they are frequent 2-itemset, if the value of [i,k] is greater than or equal min_sup, then we can generate the set of$\{I_i,I_j,I_k\}$ and add the set $\{I_i,I_j,I_k\}$ to candidate 3-itemset C_3. We continue to find connected items until the last item in L_2 is judged. The process of constructing candidate 3-itemsets C_3 is end.

Then we judge whether each candidate 3-itemset is frequent 3-itemset according to matrix M. If itemset $\{I_i,I_j,I_k\}$ is a candidate itemset, then we do AND operation by row i, j and k. If the count of 1 is greater than or equal to min_sup, the itemset $\{I_i,I_j,I_k\}$ is a frequent itemset, otherwise it is not frequent itemset.

Followed by analogy, generating candidate k+1-itemset by frequent k-itemset according to U, then judge each candidate itemset.

According to above description, we get candidate 3-itemsets $C_3=\{\{I_1,I_3,I_5\},\{I_2,I_3,I_5\},\{I_2,I_3,I_6\},\{I_2,I_4,I_6\},\{I_2,I_5,I_6\}\}$. We judge each candidate itemset according to matrix M. We get frequent 3-itemsets $L_3=\{\{I_1,I_3,I_5\},\{I_2,I_3,I_5\}\}$. Now no candidate 4-itemset will be produced, the algorithm is over. Finally we get frequent itemset $FS_D=\{\{I_1,I_3\},\{I_1,I_5\},\{I_2,I_3\},\{I_2,I_4\},\{I_2,I_5\},\{I_2,I_6\},\{I_3,I_5\},\{I_3,I_6\},\{I_4,I_5\},\{I_4,I_6\},\{I_5,I_6\},\{I_1,I_3,I_5\},\{I_2,I_3,I_5\}\}$.

Algorithm 2: generate candidate k+1 itemset according to frequent k itemset

Input:the upper triangular matrix U,frequent k-itemset L_k, min_sup;

Output:candidate k+1 itemset C_{k+1};

$C_{k+1}=\varnothing$;
foreach($\forall X \in L_k$)
{
  foreach($\forall Y \in L_k$,Y after X in L_k)
  {
    if(front k-1 items of X and Y are same)
    {
      r= subscript of the last item I_r in X;
      s= subscript of the last item I_s in Y;
      number=the value of the coordinates [r,s] in U
      if (number≥min_sup)
        $C_{k+1}=C_{k+1} \cup XUI_s$;
    }
  }
}

Algorithm 3: candidate k itemsets ,generate frequent k itemsets
Input: candidate k itemsets
Output: frequent k itemsets L_k
$L_k=\varnothing$;
if ($C_i \neq \varnothing$)
  foreach($\forall X \in C_i$)
  {
    Result = All rows of X AND operation
    count=the count of 1 in result;
    if (count≥min_sup)
      $L_k = L_k \cup X$;
  }

Final frequent itemset set contains all frequent itemsets, that is $FS_D=L_1 \cup L_2 \cup \cdots \cup L_k$.

Test results

Compared to classic Apriori algorithm, the improved algorithm has the following advantages:

The algorithm need scan transaction database only once, other algorithms can't achieve this goal. The algorithm constructs association and item matrix and upper triangular matrix when scanning database, candidate itemsets are executed in the memory,which greatly reduces the time-consuming.

The improved algorithm generates candidate itemset by using connected items. Frequent k-1 itemsets are used to generate candidate k itemset.The improved algorithm can easily produce frequent 2-itemsets by traversing the upper triangular matrix U, if the value of coordinate is greater than or equal to min_sup, then the item [I_i, I_j] is frequent 2-itemset. So the improved algorithm solves the bottleneck problem of generating frequent 2-itemsets.

In order to verify the performance of the improved algorithm, this paper realized Apriori algorithm and improved algorithm. Experimental environment: Memory 2G, CPU for Intel Pentium M 1.73GHZ, Windows 7 operating system, using C#, and test database is mushroom database which is provided by Literature [18], the database has a total of 8124 records and 22 attributes. The experimental results obtained in different support are shown in **Fig. 1**.

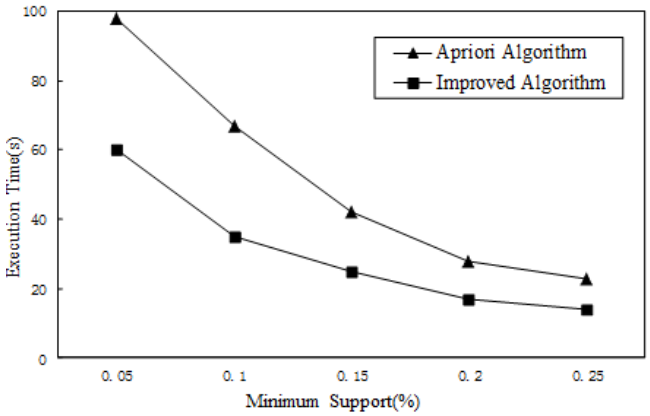

Fig. 1. Performance comparison of two algorithms in different minimum support

In order to further test the performance of the improved algorithm, because of too few records in mushroom database, this paper copy records from mushroom database, test records are 2,4,6,8,10(10^4). This paper realized Apriori algorithm and improved algorithm with the minimum support degree of 20%, the experimental results obtained in different database scale are shown in **Fig. 2**.

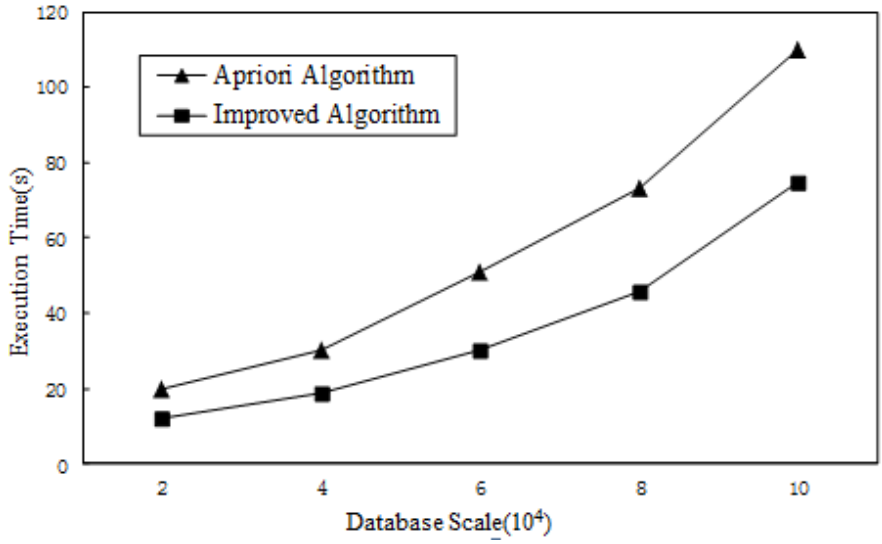

Fig. 2. Performance comparison of two algorithms in different scale database

From above experiment results, the improved algorithm need scan database only once, but Apriori algorithm need scan database many times, more advantages of the improved algorithm are obvious when records are more and more. The improved algorithm generates frequent itemset more quickly than Apriori algorithm when minimum support is lower. The improved algorithm need less time than Apriori algorithm when database scale is more and more large. From the above experimental results, whether it is on different scale of database, or in the case of different minimum support, time consumption of the improved algorithm is less than Apriori algorithm, and this advantage of the improved algorithm will be more obvious with the increasing scale database. So the improved algorithm is more suitable for large-scale database.

Conclusion

Mining association rules is very important in data mining. Mining frequent itemsets is key problem in mining association rules, this paper proposed an improved algorithm, the core idea of the improved algorithm is how to use transaction and item matrix and upper triangular matrix to generate frequent itemsets. The experimental results show that the improved algorithm generates frequent itemset more quickly than Apriori algorithm when minimum support is lower. The improved algorithm need less time than Apriori algorithm when database scale is more and more

larger. So the improved algorithm is very excellent.

Although the improved algorithm has many advantages, but we also found that the improved algorithm has some problems which are not solved. For example, the improved algorithm can't process incremental update mining. We will continue to study the problem.

Acknowledgement

In this paper, the research was sponsored by the Major Projects of Innovation and Transformation of Achievements of Shandong Province (Project No. 2014ZZCX02702).

References

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Buneman P, Jajodia S,eds. Proc. of the ACM SIGMOD Conf. on Management of Data (SIGMOD'93). New York: ACM Press, 1993. 207-216.

[2] Tsai CF, Lin YC, Chen CP. A new fast algorithm for mining association rules in large databases. In: Kamel AE, Mellouli K, Borne P, eds. Proc. of the 2002 IEEE Int'l Conf. on Systems, Man and Cybernetics (SMC 2002). IEEE Computer Society Press, 2002.251-256.

[3] Jan M. Zytkow and Willi Klosgen. Hand Book of Data Mining and Knowledge Discovery. Oxford University Press, 2002.

[4] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Chen WD, Naughton J, Bernstein PA, eds. Proc. Of the 2000 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2000). New York: ACM Press, 2000. 1~12.

[5] Aly HH, Taha Y, Amr AA. Fast mining of association rules in large-scale problems. In: Abdel-Wahab H, Jeffay K, eds. Proc. of the 6th IEEE Symp. on Computers and Communications (ISCC 2001),New York: IEEE Computer Society Press,2001.

[6] Gouda K,Zaki M J,Eficiently mining maximal frequent itemsets,Proceedings of the 1 st IEEE International Conference on Data Mining(ICDM),San Jose,2001.

[7] PARK J S,CHENM S,YU P S. An effective Hash based algorithm for mining association rules. Proceedings of International Conference on the Special Interest Group on Management of Data.New York:ACM, 1995.

[8]WANG Ming,SONG Shunli. Algorithm for discovering frequent item sets based on optimized and regrouped item sets[J]. Journal of Computer Application,vol.30.no.9,pp.2332-2334,2010.

[9]TOLVONEN H. Sampling large databases for association roles. Proceedings of the 22nd Intemational Conference on Very Large Database.Bombay,India[s.n.],1996.

[10] YOU Lei,LAN Yang,XIONG Yan. An Optimized Apriori Implementation Based on Relational Algebra[J]. Journa l of Xinyang Normal University, vol.23,no.1,pp.156-160,2010.

[11]He Jian-zhong,Lv Zhen-jun. Optimized Algorithm for Mining Association Rule Based on Two Matrixes[J]. Computer Engineering,vol.34,no.17,pp.56-61,2008.

[12]Huang Long-Jun,Duan Long-zhen. An Frequent Mining Algorithm Based on UTM[J]. Application Research of Computers,vol.11,no.25,pp.25-40,2006.

[13]YANG Zhi-gang,HE Yue-shun. An improved Apriori algorithm based on compression transaction matrix[J]. China New Technologies and Products, vol.30,no.6,pp.57-58,2010.

[14] ZHANG Yuntao,YU Zhilou,ZHANG Huaxiang. Research on high efficiency mining frequent itemsets on association rules[J]. Computer Engineering and Applications,vol.47,no.3,pp.139-141,2011.

[15]Xuan Shili,Yang Qiuye,Liang Chaoxiang. Improvement and Research for Array-based Association Rule Mining Algorithm[J]. Computer& Digital Engineering,Vo1.39 No.8,pp.1-3,2011.

[16] LIU Weixiao， CHEN Junli， Qu Shi ， WAN Wanggen. An improved Apriori algorithm[J].Computer Engineering and Applications,vol.47,no.11,pp.149-151,2011.

[17] Ma Li-sheng,Yao Guang-shun.Mining algorithm for maximal frequent itemsets based on improved FP-Tree.Journal of Computer Application,2012,32(2):326-329.

[18] SCHLIMMER J.Mushroom data set[DB/OL].[2010-4-30].http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data.