AN ASSOCIATION RULE MINING ALGORITHM BASED ON A BOOLEAN MATRIX

Hanbing Liu[1]* and Baisheng Wang[2]

**[1]Department of Information & Electrical Engineering, Hebei University of Engineering, 056038 Handan Hebei, China*

Email: hbliu_2004@126.com

[2]Department of Information & Electric Engineering, Hebei University of Engineering, 056038 Handan Hebei, China

Email: bswang45@yahoo.com.cn

ABSTRACT

Association rule mining is a very important research topic in the field of data mining. Discovering frequent itemsets is the key process in association rule mining. Traditional association rule algorithms adopt an iterative method to discovery, which requires very large calculations and a complicated transaction process. Because of this, a new association rule algorithm called ABBM is proposed in this paper. This new algorithm adopts a Boolean vector "relational calculus" method to discovering frequent itemsets. Experimental results show that this algorithm can quickly discover frequent itemsets and effectively mine potential association rules.

Keywords: Data mining, Association rule, Frequent itemsets, Boolean matrix, Relational calculus

1 INTRODUCTION

Data mining is the key step in the knowledge discovery process, and association rule mining is a very important research topic in the data mining field (Agrawal, Imielinski, & Swami, 1993). The original problem addressed by association rule mining was to find a correlation among sales of different products from the analysis of a large set of supermarket data. At present, research work on association rules is motivated by an extensive range of application areas, such as banking, manufacturing, health care, and telecommunications. The discovery of association rules is typically done in two steps: discovery of frequent itemsets and the generation of association rules. The second step is rather straightforward, and the first step dominates the processing time, so we explicitly focus this paper on the first step.

A number of efficient association rule mining algorithms have been proposed in the last few years. Among these, the Apriori algorithm (Agrawal & Srikant, 1994) has been very influential. Since its inception, many scholars have improved and optimized the Apriori algorithm and have presented new Apriori-like algorithms, Klemetinen, Mannila, & Ronkainen (1994), Park, Chen, & Yu (1995), Toivonen (1996), Kotásek & Zendulka (2000), and Han, Pei, & Yin (2000). The Apriori-like algorithms adopt an iterative method to discover frequent itemsets. The algorithm starts from frequent 1-itemsets until all maximum frequent itemsets are discovered. The Apriori-like algorithms consist of two major procedures: the join procedure and the prune procedure. The join procedure combines two frequent k-itemsets, which have the same (k-1)-prefix, to generate a (k+1)-itemset as a new preliminary candidate. Following the join procedure, the prune procedure is used to remove from the preliminary candidate set all itemsets whose k-subset is not a frequent itemset. A huge calculation and a complicated transaction process are required during the two procedures. Therefore, the mining efficiency of the Apriori-like algorithms is very unsatisfactory when transaction database is very large.

In this paper, a new algorithm called ABBM is proposed. This algorithm transforms a transaction database into a

Boolean matrix stored in bits. Meanwhile it uses the Boolean vector "relational calculus" method to discover frequent itemsets. We use the fast and simple "and calculus" in the Boolean matrix to replace the calculations and complicated transactions that deal with large numbers of itemsets. Experimental results show that this algorithm is more effective than the Apriori-like algorithms.

2 AN ALGORITHM BASED ON BOOLEAN MATRIX (ABBM)

In this section, we propose a new association algorithm. The section is organized as follows: the correlative definition and proposition, an introduction to the ABBM algorithm details, and a description of a sample execution of the ABBM algorithm.

2.1 Definition and proposition

Definition 1: Let $I= \{i_1,i_2,\ldots,i_n\}$ be a set of literals, called items. Let D be an attribute and Dom(D) be the domain of D. A transaction database is a database containing transactions in the form of (d, E), where $d \in$ Dom(D) and $E \subseteq I$.

Definition 2: Let D be a transaction database, m be the number of transactions in D, and *minsup* be the minimum support of D. The minimum support number *minsupth* is defined below:

$minsupth = minsup \times m$.

Definition 3: The Boolean matrix is a matrix with element values of '1' or '0.'

Definition 4: The Boolean 'and calculus' is carried out to an arbitrary k columns vector of the Boolean matrix; the sum of '1' of the operation result is called k- support of the k columns vector.

Proposition 1: If the sum of '1' in a row vector A_i is smaller than k, it is not necessary for A_i attending calculus of the k- supports.

> **Rationale:** According to the principle of the Boolean 'and calculus,' the result is '1' when the value of all vector elements is '1.' If the sum of '1' in a row vector A_i is smaller than k, there is at least one '0' element in $A_{i,}$.

Proposition 2: Itemset X is a k-itemsets; $|L_{K-1}(j)|$ presents the number of items 'j' in all frequent (k-1)-itemsets of the frequent set L_{K-1}. There is an item j in X. If $|L_{K-1}(j)|$ is smaller than k-1, itemset X is not a frequent itemset (Xu & Zhang, 2003).

Proposition 3: $|L_K|$ presents the number of k-itemsets in the frequent set L_K. If $|L_K|$ is smaller then k+1, the maximum length frequent itemsets is k.

> **Rationale:** Frequent (k+1)-itemsets X have k+1 frequent k-subsets. If the number of frequent k-itemsets in the frequent set L_K is smaller than k+1, there are no frequent (k+1)-itemsets in the mined transaction database.

2.2 Algorithm Details

In this section, we will first present the ABBM algorithm step by step. In general, the ABBM algorithm consists of four phases as follows:

1. Transforming the transaction database into the Boolean matrix
2. Generating the set of frequent 1-itemsets L_1
3. Pruning the Boolean matrix
4. Generating the set of frequent k-itemsets L_k(k>1)

The detailed method, phase by phase, is presented below.

2.2.1 Transforming the transaction database into the Boolean matrix

The mined transaction database is D, with D having m transactions and n items. Let $T=\{T_1,T_2,\ldots,T_m\}$ be the set of transactions and $I=\{I_1,I_2,\ldots,I_n\}$be the set of items. We set up a Boolean matrix A_{m*n}, which has m rows and n columns. Scanning the transaction database D, if item I_j is in transaction T_i, where $1\leq j\leq n, 1\leq i\leq m$, the element value of A_{ij} is '1,' otherwise the value of A_{ij} is '0.'

2.2.2 Generating the set of frequent 1-itemset L_1

The Boolean matrix A_{m*n} is scanned and support numbers of all items are computed. The support number I_j.supth of item I_j is the number of '1s' in the jth column of the Boolean matrix A_{m*n}. If I_j.supth is smaller than the minimum support number *minsupth*, itemset $\{I_j\}$ is not a frequent 1-itemset and the jth column of the Boolean matrix A_{m*n} will be deleted from A_{m*n}. Otherwise itemset $\{I_j\}$ is the frequent 1-itemset and is added to the set of frequent 1-itemset L_1 .

The sum of the element values of each row is recomputed, and according to Proposition 1, the rows whose sum of element values is smaller than 2 are deleted from this matrix.

2.2.3 Pruning the Boolean matrix

Pruning the Boolean matrix means deleting some rows and columns from it. First, the column of the Boolean matrix is pruned according to Proposition 2. This is described in detail as: Let I' be the set of all items in the frequent set L_{K-1}, where k>2. Compute all $|L_{K-1}(j)|$ where $j \in I'$, and delete the column of correspondence item j if $|L_{K-1}(j)|$ is smaller than k-1. Second, recompute the sum of the element values in each row in the Boolean matrix. According to Proposition 1, the rows of the Boolean matrix whose sum of element values is smaller than k are deleted from this matrix.

2.2.4 Generating the set of frequent k-itemsets L_k

Frequent k-itemsets are discovered only by "and" relational calculus, which is carried out for the k-vectors combination. If the Boolean matrix A_{p*q} has q columns where $2<q\leq n$ and $minsupth\leq p\leq m$, C_q^k, combinations of k-vectors will be produced. The 'and' relational calculus is for each combination of k-vectors. If the sum of element values in the "and" calculation result is not smaller than the minimum support number *minsupth*, the k-itemsets corresponding to this combination of k-vectors are the frequent k-itemsets and are added to the set of frequent k-itemsets L_k.

A detailed description of the ABBM algorithm is given in Figure1.

```
Input: The transaction database D, the minimum support number minsupth
Output: the set of frequent itemsets L
1.   Transform the transaction database D into the Boolean matrix A;
2.   For each column Ai of A
3.     If sum(Ai)>=minsupth   // sum(Ai) is the sum of the element value of Ai
4.       L1←Ii;
5.     Else delete Ai from A;
6.   For each row Am of A
7.     If sum(Am)<2
8.       Delete Am from A;
9.   For (k=2;|Lk-1|>k-1;k++)
10.   {
11.    Produce k-vectors combination for all columns of A;
12.    For each k-vectors combination {Ai1,Ai2,…,Aik}
13.    {
14.      B= Ai1∩Ai2∩…∩Aik;
15.      If sum(B)>=minsupth
16.        Lk←{Ii1,Ii2,…,Iik}; //{Ii1,Ii2,…,Iik} is the itemsets according to {Ai1,Ai2,…,Aik}
17.      }
18.    For each item Ii in Lk
19.      If |LK(Ii)|<k
20.        Delete the column Ai according to item Ii from A;
21.    For each row Am of A
22.      If sum(Am)<k+1
23.        Delete Am from A;
24.    k=k+1
25.   }
26.  Return L=L1∪L2∪…∪LK;
```

Figure 1. ABBM Algorithm

2.3 Example

This section describes a sample execution of the ABBM algorithm. The transaction data of the transaction database D are given in Table 1; the minimum support is 0.4; n=5 is the number of items, and m=5 is the number of transactions. Therefore, the minimum support number minsupsh=2.

The transaction database D is transformed into the Boolean matrix A_{5*5}:

Table 1. Transaction data of the transaction database D

TID	Itemsets
T1	I1,I4
T2	I2,I3,I5
T3	I1,I2,I3,I5
T4	I2,I5
T5	I1,I2,I3

	I1	I2	I3	I4	I5
T1	1	0	0	1	0
T2	0	1	1	0	1
T3	1	1	1	0	1
T4	0	1	0	0	1
T5	1	1	1	0	0

Figure 2. The Boolean matrix A_{5*5}

We compute the sum of the element values of each column in the Boolean matrix A_{5*5} and the set of frequent 1-itemset is:

L_1= {{I1},{I2},{I3},{I4}}

The fourth column of the Boolean matrix A_{5*5} is deleted because the support number of item I4 is smaller than the minimum support number 2. We then compute the sum of the element values of each row in the Boolean matrix and delete all rows where the sum of the element values is smaller than 2. Finally, the Boolean matrix A_{4*4} is generated.

	I1	I2	I3	I5
T2	0	1	1	1
T3	1	1	1	1
T4	0	1	0	1
T5	1	1	1	0

Figure 3. The Boolean matrix A_{4*4}

The operation of 2-supports is executed for the all columns of the Boolean matrix A_{4*4}, and the set of frequent 2-itemset is:

L_2={{I1,I2},{I1,I3},{I2,I3},{I2,I5},{I3,I5}}

In pruning the Boolean matrix A_{4*4} by the set of frequent 2-itemsets L_2 , the third row of the Boolean matrix A_{4*4} is deleted because sum of its element values is smaller than 3. Finally, the Boolean matrix A_{3*4} is generated.

	I1	I2	I3	I5
T2	0	1	1	1
T3	1	1	1	1
T5	1	1	1	0

Figure 4. The Boolean matrix A_{3*4}

The operation of 3-supports is executed for all columns of the Boolean matrix A_{3*4}, and the set of frequent 3-itemset is:

L_3= {{I1,I2,I3},{I2,I3,I5}}

According to Proposition 3, the ABBM algorithm is terminated because there are two frequent 3-itemsets in the set of frequent 3-itemset L_3.

3 EXPERIMENT

In order to appraise the performance of the ABBM algorithm, we conducted an experiment using the Apriori algorithm and the ABBM algorithm. The algorithms were implemented in Visual C^{++}6.0 and tested on a WindowsXP Professional platform. The test database T20I4D100K was generated synthetically by an algorithm designed by the IBM Quest project. The synthetic data generation procedure can be found in detail in Agrawal & Srikant (1994), whose parameter settings we followed: The number of items N is set to 1000; |D| is the number of transactions; |T| is the averages size of transactions, and |I| is the average size of the maximum frequent itemsets.

Figure 4 presents the experimental results for different numbers of minimum supports. The results show that the performance of the ABBM algorithm is much better than that of the Apriori algorithm. Moreover, the better the performance efficiency of ABBM algorithm is, the smaller the minimum support is. This is because the smaller

the minimum support, the more candidate itemsets the Apriori algorithm has to determine, and also the Apriori algorithm's join and pruning processes take more time to execute. However, the ABBM algorithm does not produce candidate itemsets, and it spends less time calculating k-supports with the Boolean matrix pruned.

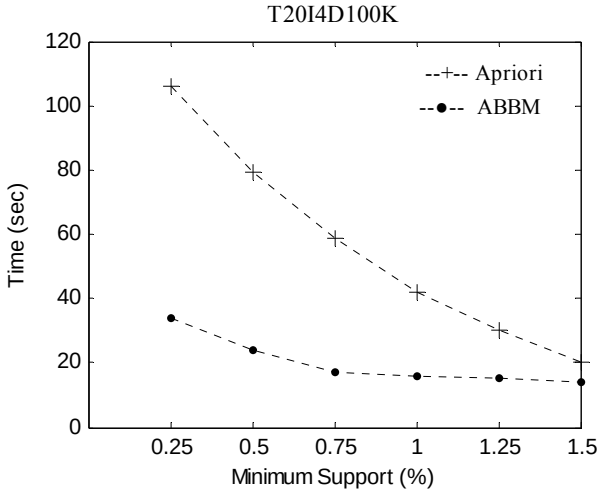

Figure 5. Performances of Apriori and ABBM

4 CONCLUSION

In this paper, an association rule mining algorithm based on the Boolean matrix (ABBM) is proposed. The main features of this algorithm are that it only scans the transaction database once, it does not produce candidate itemsets, and it adopts the Boolean vector "relational calculus" to discover frequent itemsets. In addition, it stores all transaction data in bits, so it needs less memory space and can be applied to mining large databases.

5 ACKNOWLEDGEMENTS

Project 05457205D-8 supported by Research on Science Technique and Development Planning of Hebei Province.

6 REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993) Mining association rules between sets of items in large databases. Proceedings of the *ACM SICMOD conference on management of data* pp. 207-216. Washington, D.C.

Agrawal, R. & Srikant, R. (1994) Fast Algorithms for Mining Association Rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases* pp. 487-499. Santiago, Chile.

Han, J., Pei, J., & Yin, Y (2000) Mining frequent patterns Candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Management of Data (SIGMOD'00)*, Dallas, TX.

Klemetinen, L., Mannila, H., Ronkainen, P., et al. (1994) Finding interesting rules from large sets of discovered association rules. *Third International Conference on Information and Knowledge Management* pp. 401-407.Gaithersburg, USA.

Kotásek, P. & Zendulka J. (2000) Comparison of Three Mining Algorithms for Association Rules. *Proc. of 34th Spring Int. Conf. on Modelling and Simulation of Systems (MOSIS'2000)*, Workshop Proceedings Information Systems Modelling (ISM'2000), pp. 85-90. Rožnov pod Radhoštěm, CZ, MARQ.

Liu, D. & Kedem, Z. (2002) An Efficient Algorithm for Discovering The Maximum Frequent Set. *IEEE Transaction on Knowledge and Data Engineering 14(3)*, 553-566.

Park, J., Chen, M., & Yu, P. (1995) An effective hash-based algorithm for mining association rules. *Proc 1995 ACM-SIGMOD Int. Conf Management of Data* pp. 175-186. San Jose: ACM Press.

Toivonen, H. (1996) Sampling large databases for association rules. *22nd International Conference on Very Large Data Bases* pp. 134–145. Morgan Kaufmann.

Tung, A., Lu, H., Han, J., & Feng, L. (2003) Efficient Mining of Intertransaction Association Rules. *IEEE Transaction on Knowledge and Data Engineering 15(1)*, 43-56.

Xu, Z. & Zhang, S. (2003) An Optimization Algorithm Base on Apriori for Association Rules. *Computer Engineering 29(19)*, 83-84.

.