**TELKOMNIKA**, Vol. 11, No. 10, October 2013, pp. 5711 ~ 5717
ISSN: 2302-4046

# An Efficient Association Rules Algorithm Based on Compressed Matrix

**Zhiyong Wang**
Shandong Youth University of Political Science, Jinan 250103, Shandong, China
e-mail: 83161982@qq.com

***Abstract***

*This paper analyses the classic Apriori algorithm as well as some disadvantages of the improved algorithms, based on which the paper improves the Boolean matrix. A row and a column are added on the former Boolean matrix to store the row vector of weight and account of the column vector. According to the quality of Apriori algorithm, Boolean matrix is largely compressed, which greatly reduces the complexity of space. At the same time, we adopt the method of weighting vector inner-product to find frequent K-itemsets so as to get the association rules. The complexity of space and time is developed to a large extent by the improved algorithm. In the end, the paper gives the computing procedure of the improved algorithm and by experiments, it proves that the algorithm is effective.*

***Keywords****: Apriori Algorithm, Association Rules, Compressed Boolean Matrix, Frequent Itemsets*

## 1. Introduction

Apriori algorithm is the most classic among the association rules algorithm. This algorithm mines frequent itemsets through Boolean association rules, which is divided into two steps: connection and pruning. In the process of solving algorithm of frequent itemsets, it needs to scan the database many times, which may produce a large number of candidate itemsets.

In view of this, people have made a lot of improvements to the Apriori algorithm. Requirements for the improved algorithm in the paper [1] itemsets need to be orderly arrangement according to the dictionary, and compression on Boolean matrix is not complete and still needs a lot of space in the calculation process. The improved algorithm based on literature [2] only converts the database to a Boolean matrix, and is calculated with the method of vector inner-product frequent itemsets. There is no compression on Boolean matrix and no introduction about the weight concept. Improved algorithm of paper [3, 4] adds a new column in the Boolean matrix, but only the row vector compression and no compression on the columns. The improved algorithm of literature [7] is a kind of algorithm based on the sort of matrix algorithm and this improved algorithm has certain advantages in generating frequent itemsets, but there isn't much improvement for data compression, and Boolean matrix sorting process also costs more time. Improved algorithm in reference [8] is generated by 2-itemsets support matrix, which avoids the invalid 2-itemsets and solves its bottleneck problem, but during generating frequent itemsets, it still needs repeating the scanning of matrix, and the only solving efficiency of 2-itemsets is more obvious.

As mentioned above, based on the current study, this paper has made the improvements in the Apriori algorithm. On one hand, it compresses the row vector and column vector of Boolean matrix in two directions; in addition, it introduces the weighted vector inner product and the algorithm of frequent itemsets.

## 2. Compressing Boolean Matrix

### 2.1. Boolean Matrix Natures and Formalization

Apriori algorithm has the following properties [5, 6]:

Quality 1 All the subset of the frequent itemsets are also frequent.

Quality 2 All superset of infrequent itemsets are infrequent.

*Received April 28, 2013; Revised June 30, 2013; Accepted July 14, 2013*

Quality 3 Noting $|L_K|$ for the number of frequent $K-$itemsets, if $|L_K| \geq K+1$, then there is frequent $(K+1)-$itemsets in the transaction database, and vice versa, frequent $(K+1)-$itemsets is not existing [9].

Through the form of a matrix, Boolean matrix represents the transaction in the database; each value in the matrix is a Boolean value (0 or 1). In the matrix, each row represents a transaction $T_i$, each column represents one item $I_j$. Formal description is as follows:

$M = (T_{ij})_{m\times n}$, $T_{ij} = \begin{cases} 1 & I_j \subseteq T_i \\ 0 & I_j \not\subset T_i \end{cases}$, among them $i = 1,2,3,\cdots,m$ ; $j = 1,2,3,\cdots,n$ .

In the matrix $M$ , if the transaction $T_i$ includes $I_j$ , then $T_{ij} = 1$, otherwise $T_{ij} = 0$. Such as transaction set T is as follows. This is shown in Table 1.

Table 1. The Transaction Set T

| TID | Item Sets |
|---|---|
| $T_1$ | $I_1,I_3,I_5,I_6$ |
| $T_2$ | $I_1,I_2,I_3$ |
| $T_3$ | $I_5$ |
| $T_4$ | $I_1,I_2,I_3$ |
| $T_5$ | $I_6$ |
| $T_6$ | $I_2,I_5,I_6$ |
| $T_7$ | $I_1,I_2,I_3$ |
| $T_8$ | $I_2,I_3,I_4$ |

By the Boolean matrix, the representation is as follows:

$$M = \left(\begin{array}{ccccccc} & I_1 & I_2 & I_3 & I_4 & I_5 & I_6 \\ T_1 & 1 & 0 & 1 & 0 & 1 & 1 \\ T_2 & 1 & 1 & 1 & 0 & 0 & 0 \\ T_3 & 0 & 0 & 0 & 0 & 1 & 0 \\ T_4 & 1 & 1 & 1 & 0 & 0 & 0 \\ T_5 & 0 & 0 & 0 & 0 & 0 & 1 \\ T_6 & 0 & 1 & 0 & 0 & 1 & 1 \\ T_7 & 1 & 1 & 1 & 0 & 0 & 0 \\ T_8 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}\right)$$

## 2.2. Processing of Boolean Matrix Compression

The compression of Boolean matrix is divided into two aspects: row vector compression and column vector compression.

Boolean matrix compression algorithm is as follows [3, 5]:

Step 1: Improve representation method of the Boolean matrix. Add a column $(w_i)$ on the right side of the Boolean matrix, and add a line (the sum) on the bottom of Boolean matrix. Each value of new column $w_i$ stores the corresponding number of repeating row vector. The value in the new row sum stores the corresponding number of nonzero elements in the column. For example, the improved matrix $M$ is as follows:

$$M = \left\{ \begin{array}{ccccccccc} & I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & W_i \\ T_1 & 1 & 0 & 1 & 0 & 1 & 1 & \\ T_2 & 1 & 1 & 1 & 0 & 0 & 0 & \\ T_3 & 0 & 0 & 0 & 0 & 1 & 0 & \\ T_4 & 1 & 1 & 1 & 0 & 0 & 0 & \\ T_5 & 0 & 0 & 0 & 0 & 0 & 1 & \\ T_6 & 0 & 1 & 0 & 0 & 1 & 1 & \\ T_7 & 1 & 1 & 1 & 0 & 0 & 0 & \\ T_8 & 0 & 1 & 1 & 1 & 0 & 0 & \\ sum & 4 & 5 & 5 & 1 & 3 & 3 & \end{array} \right\}$$

We sum nonzero elements of each column, putting into the corresponding sum with none value on $w_i$.

Step 2: In Boolean matrix, the summation of nonzero elements in each column [5], that is, the supporting count of item $I_j$, which means the value of the sum is the corresponding supporting count of item $I_j$. According to the natures of 1 and 2, set for the minimum support count as $min\_sup\,port\_count$ ,when item supporting count is less than $min\_sup\,port\_count$, all itemsets containing the $I_j$ are infrequent itemsets. So the column can be removed directly, which will not affect the solution of frequent itemsets. If set $min\_sup\,port\_count = 2$, according to the sum of the values in the line, the support count of $I_4$ is 1. Thus it can be deleted, which will not affect the solution of frequent itemsets. After deleting, the Matrix is expressed as follows:

$$M = \left\{ \begin{array}{ccccccc} & I_1 & I_2 & I_3 & I_5 & I_6 & W_i \\ T_1 & 1 & 0 & 1 & 1 & 1 & \\ T_2 & 1 & 1 & 1 & 0 & 0 & \\ T_3 & 0 & 0 & 0 & 1 & 0 & \\ T_4 & 1 & 1 & 1 & 0 & 0 & \\ T_5 & 0 & 0 & 0 & 0 & 1 & \\ T_6 & 0 & 1 & 0 & 1 & 1 & \\ T_7 & 1 & 1 & 1 & 0 & 0 & \\ T_8 & 0 & 1 & 1 & 0 & 0 & \\ sum & 4 & 5 & 5 & 3 & 3 & \end{array} \right\}$$

Step 3: Calculate the repeating number of rows in the matrix vector, put the results of calculation in the corresponding value of the column, delete unnecessary repeating row vector, and retain only one line, making the row vector in the matrix without repeating such as matrix $M$ ,$T_2$, $T_4$ and $T_7$ regarded as the repeating row vector. The number of repeating row vector is 3, deleting $T_4$ and $T_7$, keeping $T_2$ only. After arrangement, the Matrix can be expressed as follows:

$$M' = \left\{ \begin{array}{ccccccc} & I_1 & I_2 & I_3 & I_5 & I_6 & w_i \\ T_1 & 1 & 0 & 1 & 1 & 1 & 1 \\ T_2 & 1 & 1 & 1 & 0 & 0 & 3 \\ T_3 & 0 & 0 & 0 & 1 & 0 & 1 \\ T_5 & 0 & 0 & 0 & 0 & 1 & 1 \\ T_6 & 0 & 1 & 0 & 1 & 1 & 1 \\ T_8 & 0 & 1 & 1 & 0 & 0 & 1 \\ sum & 4 & 5 & 5 & 3 & 3 & \end{array} \right\}$$

After the above three-step compression, matrix $M$ is improved into matrix $M'$, and the space for the matrix will reduce further.

## 3. Inner-Product of Weight Vector for Frequent Itemsets

The vector quantity of Boolean matrix $M$ is defined as $M = (I_1, I_2, \cdots, I_j, \cdots, I_n)$, in which the vector quantity of item $I_j$ is defined as $I_j = (T_{1j}, T_{2j}, \cdots, T_{nj})^T$ [10].

Definition 1 the inner-product of vector $I_i$ and $I_j$ is defined as:

$$< I_i, I_j >= I_i \wedge I_j = \{T_{1i} \wedge T_{1j}, T_{2i} \wedge T_{2j}, \cdots, T_{ni} \wedge T_{nj}\} \quad (1)$$

Definition 2 the inner-product of vector $I_i, I_j, \ldots, I_k$ is defined as:

$$< I_i, I_j, \cdots, I_k >= I_i \wedge I_j \wedge \ldots \wedge I_k = \{T_{1i} \wedge T_{1j} \wedge \ldots \wedge T_{1k}, T_{2i} \wedge T_{2j} \wedge \ldots \wedge T_k, \ldots, T_{ni} \wedge T_{nj} \wedge \ldots \wedge T_{nk}\} \quad (2)$$

The $\wedge$ presents logic and arithmetic in the above.

The computing method of using inner-product of weight vector for frequent itemsets is as follows [1, 6].

Quality 4 The supporting count of 2-itemsets $\{I_i, I_j\}$ $(i \neq j)$ can be got by the vector inner-product of $I_i$ and $I_j$, that is

$$support\_count(I_{ij}) = I_i \wedge I_j = T_{1i} \wedge T_{1j} + T_{2i} \wedge T_{2j} + \ldots + T_{ni} \wedge T_{nj} \quad (3)$$

The supporting count of K-itemsets $\{I_i, I_j, \ldots, I_k\}$ $(i \neq j \neq k)$ can be got by the vector inner-product of $I_i, I_j, \ldots, I_k$, that is

$$support\_count(I_{ij\ldots k}) = I_i \wedge I_j \wedge \ldots \wedge I_k = T_{1i} \wedge T_{1j} \wedge \ldots \wedge T_{1k} + T_{2i} \wedge T_{2j} \wedge \ldots \wedge T_k, + \ldots + T_{ni} \wedge T_{nj} \wedge \ldots \wedge T_{nk} \quad (4)$$

For the compression matrix $M'$, quality 4 needs to be improved and to introduce weights, which means to compress the last column $w_i$ in the compressed matrix $M'$.

Quality 5 The weighted vector inner-product of $I_i$ and $I_j$ is defined as:

$$< I_i, I_j >= I_i \wedge I_j = \{w_1 \times (T_{1i} \wedge T_{1j}), w_2 \times (T_{2i} \wedge T_{2j}), \cdots, w_n \times (T_{ni} \wedge T_{nj})\} \quad (5)$$

The weighted vector inner-product of $I_i$ and $I_j$ is defined as:

$$< I_i, I_j, \cdots, I_k >= I_i \wedge I_j \wedge \ldots \wedge I_k = \{w_1 \times (T_{1i} \wedge T_{1j} \wedge \ldots \wedge T_{1k}), w_2 \times (T_{2i} \wedge T_{2j} \wedge \ldots \wedge T_k), \ldots, w_n \times (T_{ni} \wedge T_{nj} \wedge \ldots \wedge T_{nk})\}. \quad (6)$$

The supporting count of 2-itemsets $\{I_i, I_j\}$ $(i \neq j)$:

$$support\_count(I_{ij}) = I_i \wedge I_j = w_1 \times (T_{1i} \wedge T_{1j}) + w_2 \times (T_{2i} \wedge T_{2j}) + \cdots + w_n \times (T_{ni} \wedge T_{nj}) \quad (7)$$

The supporting count of K-itemsets $\{I_i, I_j, \ldots, I_k\}$ $(i \neq j \neq k)$:

$$support\_count(I_{ij\ldots k}) = I_i \wedge I_j \wedge \ldots \wedge I_k = w_1 \times (T_{1i} \wedge T_{1j} \wedge \ldots \wedge T_{1k}) + w_2 \times (T_{2i} \wedge T_{2j} \wedge \ldots \wedge T_k) + \ldots + w_n \times (T_{ni} \wedge T_{nj} \wedge \ldots \wedge T_{nk}). \quad (8)$$

## 4. Algorithm Thought

(1) Scanning the database, transaction sets can be converted to Boolean matrix, marked $M$ .

(2) According to the above matrix compression algorithm, compress matrix $M$ , rearrange, and get matrix $M'$ .

(3) Put combination $C_n^2$ onto the column vector of the rearranging matrix $M'$, and according to the quality 5 of supporting count, calculate support count of 2-itemsets, the itemsets of which is greater than $min\_sup\,port\_count$ is collection of frequent 2-itemsets, $L_2$ .

(4) According to the quality of 1, 2, 3, if there is a frequent 3-itemsets, certainly it is a superset of frequent 2-itemsets. So the recounted frequent 2-itemsets contains frequent item; delete other infrequent items, recount the number of the same row vector, and rearrange the compression matrix $M'$ .

(5) Put combination $C_n^3$ onto the column vector of the rearranging matrix $M'$ , and according to the quality 5, calculate support count of 2-itemsets, the support count of which is greater than $min\_sup\,port\_count$ is collection of frequent 3-itemsets, $L_3$ .

(6) Similar to 4 and 5, continue statistics frequent items in the generated frequent (k-1)-itemsets, delete the other items, recount the number of the same row vector in the matrix $M'$ , and compress matrix $M'$ . After arrangement of the matrix composite, calculate support count of k-itemsets according to the quality 5, the one of which more than $min\_sup\,port\_count$ is frequent k–itemsets, $L_k$ . So repeatedly, until the concentration of $L_k < K+1$, that is $|L_K| < k+1$ , the $L_k$ is the maximum frequent itemsets.

## 5. Case Analyses

(1) Scanning the database, convert the transaction sets T into Boolean matrix $M$ .

(2) According to the above compression algorithm, compressing matrix $M$ , matrix is obtained

$$M' = \left\{ \begin{array}{lcccccc} & I_1 & I_2 & I_3 & I_5 & I_6 & w_i \\ T_1 & 1 & 0 & 1 & 1 & 1 & 1 \\ T_2 & 1 & 1 & 1 & 0 & 0 & 3 \\ T_3 & 0 & 0 & 0 & 1 & 0 & 1 \\ T_5 & 0 & 0 & 0 & 0 & 1 & 1 \\ T_6 & 0 & 1 & 0 & 1 & 1 & 1 \\ T_8 & 0 & 1 & 1 & 0 & 0 & 1 \\ sum & 4 & 5 & 5 & 3 & 3 & \end{array} \right\}$$

(3) Putting combination $C_n^2$ onto the column vector of the rearranging matrix $M'$and according to the quality 5, calculate the support count of each 2-itemsets.

$\text{support\_count}(I_{12}) = 1\times(1\wedge 0) + 3\times(1\wedge 1) + 1\times(0\wedge 0) + 1\times(0\wedge 0) + 1\times(0\wedge 1) + 1\times(0\wedge 1) = 3$ ;

For others, in a similar way there are $\text{support\_count}(I_{13}) = 4$; $\text{support\_count}(I_{15}) = 1$; $\text{support\_count}(I_{16}) = 1$; $\text{support\_count}(I_{23}) = 3$; $\text{support\_count}(I_{25}) = 1$; $\text{support\_count}(I_{26}) = 1$; $\text{support\_count}(I_{35}) = 1$; $\text{support\_count}(I_{36}) = 1$; $\text{support\_count}(I_{56}) = 1$. If setting $min\_sup\,port\_count = 2$ , then $I_{12}, I_{13}, I_{23}$ will be collection of frequent 2-itemsets.

(4) According to the quality of 1, 2, 3, if there is a frequent 3-itemsets, certainly it is a superset of frequent 2–itemsets, and contains $I_1, I_2, I_3$ only. So make compression processing on matrix $M$ , and delete the item $I_5, I_6$ in the matrix. After adding up the number of repeating row vector, we can get

$$M'' = \left\{ \begin{array}{ccccc} & I_1 & I_2 & I_3 & W_i \\ T_1 & 1 & 0 & 1 & 1 \\ T_2 & 1 & 1 & 1 & 3 \\ T_3 & 0 & 0 & 0 & 2 \\ T_6 & 0 & 1 & 0 & 1 \\ T_8 & 0 & 1 & 1 & 1 \\ sum & 4 & 5 & 5 & \end{array} \right\}$$

(5) Making $C_n^3$ combination in three columns of matrix, and according to the quality 5, calculate the support count of each 3-itemsets,

$\text{support\_count}(I_{123}) = 1\times(1\wedge 0\wedge 1)+3\times(1\wedge 1\wedge 1)+2\times(0\wedge 0\wedge 0)+1\times(0\wedge 1\wedge 0)+1\times(0\wedge 1\wedge 1)=3$,

That means $I_{123}$ is the collection of frequent 3-itemsets. According to quality 3, for $|L_3| < 4$, the collection of frequent 4-itemsets does not exist.

## 6. Algorithm Analyses

Space complexity analyses: The data of Apriori algorithm stored is the item value, and in the process of solving the frequent itemsets, it takes up a large amount of space; the improved Apriori algorithm with stored data is Boolean value, with compression processing made in the Boolean matrix including two directions: row and column; Boolean matrix compression processing is also constantly made in the process of solving the frequent itemsets, avoiding the connection and pruning operation, greatly reducing the space complexity.

Time complexity analyses: The improved Apriori algorithm only needs to scan the database for one time, through the column vector to solve Boolean matrix frequent k-itemsets, avoiding the connection and pruning operation.

In order to test the effectiveness of the algorithm, a test environment has been set up as follow: For 2G memory, CPU for the Intel (R) Core (TM) i5 2.67 GHz, the operating system on a Windows XP. In this kind of computer, Apriori algorithm and the improved Apriori algorithm realized. Experimental data are collected from the candidates' seven-year application information of one university students for an examination, nearly 32476 records, and 25 items. The experimental results are shown in Figure 1. It can be seen that improved Apriori algorithm is far superior to the Apriori algorithm.

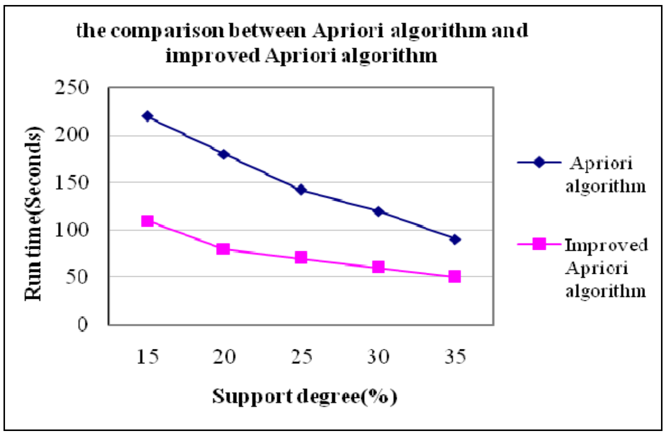

Figure 1. The comparison between the Apriori algorithm and improved Apriori algorithm

## 7. Conclusion

This paper analyses the classic Apriori algorithm and the shortages of some improved algorithm. And based on this, it retreats them from two directions of rows and columns to compress the Boolean matrix; in the process of solving the frequent itemsets it also constantly does a great of compression, which greatly reduces the space complexity of the algorithm; in the process of solving the frequent itemsets, there is the introduction about the association rules algorithm for the inner-product method of weighted vector to evaluate frequent k-itemsets. This improved algorithm compresses the data, reduces the number of database access, avoids the connection and pruning process, greatly reduces the time complexity and space complexity of the algorithm, and also improves the execution efficiency of the algorithm a great deal.

## References

[1] QIAN Guangchao, JIA Ruiyu, ZHANG Ran, LI Longshu. One Optimized Method of Apriori Algorithm. *Computer Engineering*. 2008; 34(23): 196-198.
[2] WANG Chengliang, WU Yanjuan. Research and Application of Efficient Association Rule Discovery Algorithm of Chinese Medicine. *Computer Engineering and Applications.* 2010; 46(34): 119-122.
[3] ZENG Wandan, ZHOU Xubo, DAI Bo, CHANG Guiran, LI Chunping. An Association Mining Algorithm Based on Matrix. *Computer Engineering*. 2006; 32(2): 45-47.
[4] ZHANG Yueqin. Research of Frequent Itemsets Mining Algorithm Based on 0-1 Matrix. *Computer Engineering and Design*. 2009; 30(20): 4662-4664.
[5] PEI Guying. A Fast Algorithm for Mining of Association Rules Based on Boolean Matrix. *Automation & Instrumentation.* 2009; 5: 16-18.
[6] ZHANG Wendong, YIN Jinhuan, JIA Xiaofei, HUANG Chao, YUAN Yanmei. Research of A Frequent Itemsets Mining Algorithm Based on Vector. *Journal of Shandong University (Natural Science)*. 2011; 46(3): 31-34.
[7] LV Taoxia, LIU Peiyu. Algorithm for Generating Strong Association Rules Based on Matrix. *Application Research of Computers.* 2011; 28(4): 1301-1303.
[8] ZHANG Yuntao, YU Zhilou, ZHANG Huaxiang. Research on High Efficiency Mining Frequent Itemsets on Association Rules. *Computer Engineering and Applications*. 2011; 47(3): 139-141.
[9] ZHANG Zhongping, LI Yan, YANG Jing. Frequent Itemsets Mining Algorithm Based on Matrix. *Computer Engineering.* 2009; 35(1): 84-85.
[10] Wang Lifeng. An Efficient Association Rule Algorithm Based on Boolean Matrix. *International Review on Computers and Software*. 2012; 7(2): 695-700.