Sensors & Transducers, Vol. 153, Issue 6, June 2013, pp. 84-91

Sensors & Transducers

Research of Improved Apriori Algorithm Based on Itemset Array

*** Naili Liu, Lei Ma**
Linyi University, Linyi City, 276005, China
* Tel.: 13754706905, fax: 0539-8766320
* E-mail: lnl1999@163.com

Received: 2013 /Accepted: 14 June 2013 /Published: 25 June 2013

Abstract: Mining frequent item sets is a major key process in data mining research. Apriori and many improved algorithms are lowly efficient because they need scan database many times and storage transaction ID in memory, so time and space overhead is very high. Especially, they are lower efficient when they process large scale database. The main task of the improved algorithm is to reduce time and space overhead for mining frequent item sets. Because, it scans database only once to generate binary item set array, it adopts binary instead of transaction ID when it storages transaction flag, it adopts logic AND operation to judge whether an item set is frequent item set. Moreover, the improved algorithm is more suitable for large scale database. Experimental results show that the improved algorithm has better efficiency than classic Apriori algorithm.

Keywords: Data mining, Association Rules, Frequent item sets, Binary item set array, Logic AND operation

1. Introduction

Mining association rule is one of the key problems in data mining research. Association rules can show the relationships between data items. So, it is widely used in commercial, organizational, administrative and other domains [1-2]. Mining association rule must firstly discover frequent item set, so, discovering frequent item sets has been studied popularly. In recent times, with the development of electronic information, size and number of available databases are explosive growth [3]. The most challenging in database mining is developing fast and efficient algorithms, these algorithms can deal with large scale database.

1.1. Apriori Algorithm

The Apriori [4] algorithm is the most well known association rule algorithm and it is used in most commercial product. The basic idea of this algorithm is to scan database to generate candidate item sets and then scan database again to determine whether candidate item set is frequent item set. So, the problem of Apriori algorithm is scanning database many times. It is too inefficient when database scale is large. Therefore, many improved algorithms were proposed after Apriori algorithm for improving efficiency and scalability.

1.2. Literature Review on Mining Frequent Itemset

Since Agrawal put forward Apriori [4] algorithm in 1993, many improved algorithms [5-7] have been proposed. But the main limitation of these algorithms is the need to scan database many times to generate frequent item sets and to generate a huge amount of candidate item sets.

Article number P_1223

FP-growth algorithm [8] mines frequent patterns without candidate generation. The algorithm must recursively mine conditional patterns and conditional FP- tree to generate frequent item sets, and other problem is no common prefixed within the data items.

Sample algorithm [9] reduces the scanning database time, because it scans database one time, but wastes amount of considerable time on candidate itemsets.

Hash table algorithm [10] adopts a hash table technology, it determines C'_{k+1} according to C_k and stores C'_{k+1} in appropriate hash.

The Partition algorithm [11] is to reduce scanning database times and improve efficiency, database is divided into many sub-libraries into memory, respectively mining local frequent patterns of each sub-library, and then all local frequent patterns are candidate item sets. Finally it determines whether a candidate item set is a frequent item set by scanning database again. Partition algorithm may exacerbate the problem of combinatorial explosion because of too many candidate item sets.

Compression transaction algorithm [12] scans database and maps data to binary matrix, 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.

Literature [13] mines two frequent itemsets and maximum frequent itemsets by building two support matrices, but its time complexity and its space complexity are very high.

Taxonomy superimposed graphs is frequent patterns in this new graph model, there may be many patterns that are implied by the specialization and generalization hierarchy of the associated node label taxonomy [14].

Any kind of information is able to be representing as graphs of graph databases, where changes in data can be possible for naturally accommodation [15, 16].

Literature [17] mines two frequent itemsets and maximum frequent itemsets by building two support matrices, but its time complexity and its space complexity are very high.

DFS algorithm in Literature [18] can generate errorous frequent itemsets.

Literature [19] proposes Apriori_M algorithm, which still scans database two times.

Patition and reduction is adopted by many algorithms [20-22].

According to the above algorithms, this paper puts forward an improved algorithm to overcome the above said limitations. The improved algorithm need scan database only once, which can reduce the space overhead. The improved algorithm generates binary item sets array to find frequent item set, which saves memory space. The improved algorithm produces frequent itemsets by logic AND operation, which improves the efficiency of mining frequent itemsets.

2. Relevant Definition and Property

Association rules' concept was first proposed by Agrawal. Mining association rule is an important research in the data mining. The concept of association rules as follows:

Definition 1 D is a transaction database, let $D = \{T_1, T_2, \ldots, Tn\}$, where T_i is an identifier which be associated with each transaction, I is a set of items in D, let $I = \{I_1, I_2, I_m\}$, where T_i $(1 \le i \le n)$ 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 X is an item set, 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 the minimum support threshold, then X is a frequent item set, otherwise X is a infrequent item set.

Definition 3 An item set contains k items, so it is called k-itemset, if X is a k-itemset and sup(X) ≥min_sup, then X is called frequent k-itemset. All frequent k-itemsets set is called frequent k itemset, denoted as L_k.

Definition 4 I_i $(1 \le i \le m)$ is an itemset, the binary array of I_i is denoted as A_i. All itemsets are sorted by lexicographic order.

Definition 5. All items of frequent k itemset are sorted by lexicographic order, if the former k-1 items are same in two k itemsets, these two k-itemsets are called connected item sets.

Definition 6 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.

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

Property 1 If X is a frequent item set, then any subset of X must be a frequent item set; if X is not a frequent item set, then any super set of X must be not a frequent item set. This property can be inferred from the property of Apriori.

Generally most of the algorithms are executed in two steps. First step, mining frequent itemsets; second step, generating association rules. In the two steps, mining frequent item sets is more difficult than generating rules.

3. The Improved Algorithm Description

3.1. Algorithm Relation Concept

It is a chance to overcome the shortcomings of the previous algorithms; the improved algorithm scans the database only once. The improved algorithm establish that generating all frequent itemsets is available by producing binary item set array which is used to simplify the process of generating frequent itemsets.

The binary item set array and matrix important in the improved algorithm, matrix is constructed in sequential numbers where the rows represent items and the columns represent items, so we only need storage upper matrix. The improved algorithm is to reduce time for scans database only once to generate matrix as a two dimensional array.

First, scanning the database to construct binary item set arrays and upper triangular matrix. According to the definition of 1, 2 and 3, we generate binary array for each item, for item I_i ($1 \le i \le m$), if T_j($1 \le j \le n$) contains I_i, then $A_i[j]$ will be set to 1, otherwise will be set to 0. So there have m binary item set arrays, each array' length is the number of transactions in D, the improved algorithm adopts binary to storage 1 and 0 to reduce space overhead.

Second, constructing the upper triangular matrix at the same time generating binary item set arrays; the upper triangular matrix for D is defined as follows:

1) The rows represent the item set of database $I=\{I_1, I2,\dots, Im\}$, the columns also represent items.

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

The pseudo code for constructing binary item set array and upper triangular matrix is given below:

```
for each transaction Tj in D
begin
        if item Ii is present in Tj then
         Ai[j]=1;
else
Ai[j]=0;
sort all item sets in Tj by lexicographic order
        for (∀{Ik, Il}⊆Tj)
UTM [k, l] plus 1;
end
```

When scanning the database, all items of each transaction are sorted by lexicographic order, each item set $\{I_k, I_l\}$($k \le l$) is contained by transaction, the coordinate value UTM (k,l) in the upper triangular matrix plus 1.

Finally, we count the number of 1s in array A_i ($1 \le i \le m$), if the number of 1s in A_i is greater than or equal the minimum support threshold, then modify the value of coordinate UTM[i,i], otherwise, delete row i and column i in UTM and delete binary item set array A_i in binary item set arrays. Finally we get new binary item set arrays and new UTM.

The structure of the upper triangular matrix UTM is shown in Table 1.

Table 1. Upper Triangular Matrix UTM.

	I_1	I_2	I_3	…	I_m
I_1	I_{11}	I_{12}	I_{13}	…	I_{1m}
I_2		I_{22}	I_{23}	…	I_{2m}
…			…	…	…
I_m					I_{mm}

The structure of binary item set array is shown in Table 2.

Table 2. Binary Item set Array.

Array Name	Array Value
A_1	The number of 1 or 0,the count of numbers is n
A_2	The number of 1 or 0,the count of numbers is n
…	…
A_m	The number of 1 or 0,the count of numbers is n

3.3. Algorithm Idea

How to generate candidate itemsets by the upper triangular matrix UTM and binary item set array. All items of the rows in UTM are frequent 1-itemset, but we can easily get frequent 2-itemset from UTM, so all frequent 1-itemset are useless. The value of coordinate UTM[i,j]($i \le j$) is the support of item set $\{I_i,I_j\}$, if the value of coordinate[i,j]($i \le j$) is not less than the minimum support threshold, then item set $\{I_i,I_j\}$ is frequent itemset, because item set $\{I_i,I_j\}$($i \le j$) in UTM has two items, so all item sets that it's coordinate value is greater than or equal the minimum support threshold are frequent 2-itemset. So we can easily get frequent 2-itemset L_2 by traversing the UTM. The pseudo code for generating frequent 2-itemset L_2 is given below:

```
for (int i=0;i<m;i++)
for (int j=i;j<m;j++)
if (the value of the coordinates [i,j] in UTM≥the
            minimum support threshold)
add { Ii,Ij } to L2;
```

We continually generate frequent k-itemset by using frequent 2-itemset L_2. All items in each itemset of L_2 are sorted by lexicographic order. Firstly, generating candidate itemset C_3 according to frequent 2-itemset L_2, all itemset in L_2, if there exists $\{I_i,I_j\}$ and $\{I_i,I_k\}$,according to definition 5, they are connected items, if the value of the coordinates [j,k] in UTM is not less than min_sup, then we get item set $\{I_i,I_j,I_k\}$ by connecting $\{I_i,I_j\}$ and $\{I_i,I_k\}$, we will add item set $\{I_i,I_j,I_k\}$ to candidate 3-itemset C_3, otherwise, $\{I_i,I_j\}$ and $\{I_i,I_k\}$ can not be connected. We will continue to find candidate itemset by using connected items and UTM until no-connected item can be found.

Followed by analogy, generating candidate k+1-itemset by candidate k-itemset, each candidate k-itemset X in C_k, from the position X in the C_k after codes in order of X k-1-item start other candidate k-itemsets find such a candidate k-itemset Y ,connect the last label I_r of X and the last label I_l of Y to form a matrix coordinates [r, s] and find the value of coordinates [r,s] in the matrix of the UTM, if the value is not less than the minimum support, then connect the X and Y to generate candidate k +1 itemset, otherwise no connection, we continue to find other connected items until settle the last k-itemset in C_k. Continue indefinitely, until it can no longer generate the candidate projects set. We can generate candidate k+1-itemset C_{k+1} from candidate k-itemset C_k by using the same method. K is not greater than the max length of transaction in D.

How to verify that a candidate item set is a frequent item set. All candidate itemsets $\{C_3,\ldots, Ck\}(k\geq 3)$, we judge each candidate itemset in accordance with the order from C_k to C_3 in order to reduce time overhead. First any item set in C_k, $C_k=\{I_i,I_j,...I_l\}(i\leq j\leq ...\leq l)$, binary item set array corresponding to each item logic AND operation, that is $A_i\&A_j\&...\&A_l$, if the number of 1s in the result is greater than or equal the minimum threshold, then the candidate itemset is frequent itemset, add it to frequent itemset, according to property 1, we will add all subsets of the frequent itemset in all candidate itemsets to frequent item set and delete these subsets from all candidate itemsets; otherwise, delete the candidate itemset from candidate itemsets. We will continue to verify candidate itemset until candidate itemset is empty.

3.4. Algorithm Pseudo Code

The improved algorithm has two important steps in finding frequent itemset:generating candidate k+1-itemset C_{k+1} from candidate k-itemset C_k and verifying that a candidate item is a frequent itemset. So we introduce these two algorithms.

Firstly introduce generating candidate itemsets algorithm, generate candidate k+1 itemset C_{k+1} from candidate k itemset C_k by using connected item concept.

```
Input:UTM, Ck, minimum support threshold min_sup;
Output:candidate k+1 itemset Ck+1;
Ck+1=∅;
if (k+1 is not greater than the max length of transaction in D)
{
  Sort all itemsets in Ck by lexicographic order;
  Sort all items in each candidate itemset by lexicographic order;
  for (each itemset X of Ck)
  {
   for (all Y after X in Ck)
   {
     if(X and Y are connected items according
     to definition)
   {
     l= subscript of section k item Il in X;
     m= subscript of section k item Im in Y;
     t=coordinate [l,m];
     count=the value of t in UTM
        if (count≥min_sup)
   {
        A=XUIm;
        Ck+1=Ck+1 ∪A;
              //add A to candidate k+1 itemset
        }
      }
     }
}
  }
```

Secondly introduce generating frequent itemset algorithm, verify each candidate itemset of candidate itemsets$\{C_3,\ldots,C_k\}(k\geq 3)$ by using binary item set array logic AND operation.

```
Input: {C3,…,Ck}(k≥3), Ai(1≤i≤m)
Output:frequent itemset FSD
FSD=∅;
 FSD=FSD∗L2; //add L2 to FSD
//verify each candidate itemset from Ck to C3 in order
int i=k;
while (i≥3)
{
    while (Ci is not null, ∀X∈Ci)
{
      find all items of X={Il,…,Is}(s≥l)
      corresponding binary array from binray
      item set arrays
   B=Al&…&As;
   count= counting the number of 1s in B;
   if (count≥min_sup)
   {
   FSD=FSD ∪X;
   //according to property 1
      for(any subset of X in candidate
      itemset Cj◘{C3,…,Ck}(k≥3))
for (each itemset Y in Cj)
   if (Y is a subset of X)
{
   add Y to FSD;
   delete Y from Cj;
            }
}
else
delete X from Ci;
     }
     i--;
}
```

4. Algorithm Example

Here, considering an example the suspect dataset, as to predict task whether a suspect is robbery, that suspect's data attributes are name, sex, age, education, work, race, marrital-status, law-case, crime-mode, which are mainly taken for analysis.

Initially an original transaction database consists of four tables, the data structure of the tables is suspect (name, sex, age, education, work, race, marrital-status, law-case, crime-mode)

education(ID,Name)
work(ID,Name)
law-case(ID,Name)

Before preprocess suspect dataset, select twenty transaction items, this paper deletes name, sex, race, marrital-status, crime-mode column in order to simple analysis, only remains age, education, work and low-case columns, analyzing their relations. Initial suspect data is shown in Table 3.

Table 3. Initial suspect data.

TID	Age	Education	Work	Law-case
T_1	49	1	1	1
T_2	22	3	2	2
T_3	17	2	1	1
T_4	29	1	2	3
T_5	17	2	1	3
T_6	39	1	1	3
T_7	28	2	2	1
T_8	41	1	1	1
T_9	27	1	2	3
T_{10}	13	1	1	3
T_{11}	30	3	2	2
T_{12}	15	2	1	1
T_{13}	16	2	1	1
T_{14}	42	1	2	1
T_{15}	33	1	1	3
T_{16}	17	2	1	2
T_{17}	46	1	1	1
T_{18}	25	3	2	1
T_{19}	12	2	1	3
T_{20}	19	1	1	1

In Table 3, the value of education column is corresponding to ID in education table, the value of work column is corresponding to ID in work table, the value of low-case column is corresponding to ID in low-case table. In order to analysis, process each attribute, for example, the value of age column is number, which is not analyzed, so discrete age attribute, other attributes are respectively processed. Relational database attribute value and transaction data concentration project correspondence is shown in Table 4.

According to Table 4, transfer each transaction of suspect dataset in Table 3, for example, the value of age column is 49 in T_1,49>18, satisfies I_3, so T_1 contains I_3. All values of suspect data are are processed in this method. Transferred suspect data is shown in Table 5.

According to the improved algorithm, we construct UTM and binary item set arrays, suppose the minimum support threshold is 3, that is, min_sup is 3, by counting the number of 1s in each array is determined the support for each 1 itemset, as follows: support($\{I_1\}$)=2,support($\{I_2\}$)=5, support($\{I_3\}$)=13, support($\{I_4\}$)=10, support($\{I_5\}$)=7, support($\{I_6\}$)=3, support($\{I_7\}$)=13, support($\{I_8\}$)=7, support($\{I_9\}$)=10, support($\{I_{10}\}$)=3, support($\{I_{11}\}$)=7, according to the definition of frequent itemset, thus the frequent 1 itemsets are $\{\{I_2\},\{I_3\},\{I_4\},\{I_5\},\{I_6\},\{I_7\},\{I_8\},\{I_9\},\{I_{10}\},\{I_{11}\}\}$ as their supports are not less than the minimum support threshold. But $\{I_1\}$ is not frequent 1 itemset as its support is less than the minimum support threshold. So delete row I_1 and column I_1 from UTM and delete A_1 from binary item set arrays at the same time in order to save time overhead. UTM has 11 rows and 11 columns. There have 11 binary item set arrays. Finally, UTM for suspect dataset is shown in Table 6, binary item set arrays for suspect dataset is shown in Table 7.

Table 4. Attribute and transaction item project.

ID	Name	Item
	age	
1	≤14	I_1
2	14<Age≤18	I_2
3	>18	I_3
	education	
1	illiteracy	I_4
2	primary and middle	I_5
3	high and over	6
	work	
1	unemployed	I_7
2	employed	I_8
	law-case	
1	robbery	I_9
2	kill	I_{10}
3	steal	I_{11}
…	…	…

Table 5. Transferred suspect data.

TID	Item
T_1	I_3, I_4, I_7, I_9
T_2	I_3, I_6, I_8, I_{10}
T_3	I_2, I_5, I_7, I_9
T_4	I_3, I_4, I_8, I_{11}
T_5	I_2, I_5, I_7, I_{11}
T_6	I_3, I_4, I_7, I_{11}
T_7	I_3, I_5, I_8, I_9
T_8	I_3, I_4, I_7, I_9
T_9	I_3, I_4, I_8, I_{11}
T_{10}	I_1, I_4, I_7, I_{11}
T_{11}	I_3, I_6, I_8, I_{10}
T_{12}	I_2, I_5, I_7, I_9
T_{13}	I_2, I_5, I_7, I_9
T_{14}	I_3, I_4, I_8, I_9
T_{15}	I_3, I_4, I_7, I_{11}
T_{16}	I_2, I_5, I_7, I_{10}
T_{17}	I_3, I_4, I_7, I_9
T_{18}	I_3, I_6, I_8, I_9
T_{19}	I_1, I_5, I_7, I_{11}
T_{20}	I_3, I_4, I_7, I_9

Table 6. UTM for suspect dataset.

Item	I_2	**I_3**	**I_4**	**I_5**	**I_6**	**I_7**	**I_8**	**I_9**	**I_{10}**	**I_{11}**
I_2	5	0	0	5	0	5	0	3	1	1
I_3		13	9	1	3	6	7	7	2	4
I_4			10	0	0	7	3	5	0	5
I_5				7	0	6	1	4	1	2
I_6					3	0	3	1	2	0
I_7						13	0	7	1	5
I_8							7	3	2	2
I_9								10	0	0
I_{10}									3	0
I_{11}										7

Table 7. Binary item set arrays for suspect dataset.

Array	**Value**
A_2	{0,0,1,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0}
A_3	{1,1,0,1,0,1,1,1,1, 0,1,0,0,1,1,0,1,1,0,1}
A_4	{1,0,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,0,0,1}
A_5	{0,0,1,0,1,0,1,0,0,0,0,1,1,0,0,1,0,0,1,0}
A_6	{0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0}
A_7	{1,0,1,0,1,1,0,1,0,1,0,1,1,0,1,1,1,0,1,1}
A_8	{0,1,0,1,0,0,1,0,1,0,1,0,0,1,0,0,0,1,0,0}
A_9	{1,0,1,0,0,0,1,1,0,0,0,1,1,1,0,0,1,1,0,1}
A_{10}	{0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0}
A_{11}	{0,0,0,1,1,1,0,0,1,1,0,0,0,0,1,0,0,0,1,0}

We can easily get frequent 2 itemsets from UTM, if the value of coordinates UTM [i,j] is not less than min_sup, then $\{I_i, I_j\}$ is frequent 2 itemset. Traversal matrix, the frequent 2 itemsets L_2 will be: $\{\{I_2,I_5\},\{I_2,I_7\},\{I_2,I_9\},\{I_3,I_4\},\{I_3,I_6\},\{I_3,I_7\},\{I_3,I_8\},\{I_3,I_9\},\{I_3,I_{11}\},\{I_4,I_7\},\{I_4,I_8\},\{I_4,I_9\},\{I_4,I_{11}\},\{I_5,I_7\},\{I_5,I_9\},\{I_6,I_8\},\{I_7,I_9\},\{I_7,I_{11}\},\{I_8,I_9\}\}$, their supports are equal and above 3.

Next generating candidate 3 itemsets from L_2, the max length of transaction is 4, 3<4, so generate candidate 3 itemsets according to algorithm which is generating candidate k+1 itemsets from k itemsets. After executing algorithm, we get candidate 3 itemsets, as follows: $C_3=\{\{I_2,I_5,I_7\},\{I_2,I_5,I_9\},\{I_2,I_7,I_9\},\{I_3,I_4,I_7\},\{I_3,I_4,I_8\},\{I_3,I_4,I_9\},\{I_3,I_4,I_{11}\},\{I_3,I_6,I_8\},\{I_3,I_7,I_9\},\{I_3,I_7,I_{11}\},\{I_3,I_8,I_9\},\{I_4,I_7,I_9\},\{I_4,I_7,I_{11}\},\{I_4,I_8,I_9\},\{I_5,I_7,I_9\}\}$. K+1=4, 4 is not greater than the max length of transaction 4, so we continually generate candidate 4 itemsets from candidate 3 itemsets, according algorithm, generating candidate 4 itemsets as follows: $C_4=\{\{I_2,I_5,I_7,I_9\},\{I_3,I_4,I_7,I_9\},\{I_3,I_4,I_7,I_{11}\},\{I_3,I_4,I_8,I_9\}\}$. Generating candidate itemsets is end because k+1=5>4.

Then generate frequent itemsets by candidate itemsets. Firstly, verify each candidate itemset in C_4 .For example, candidate itemset $\{I_2,I_5,I_7,I_9\}$, the item I_2,I_5,I_7,I_9 corresponding to A_2,A_5,A_7,A_9 logic AND operation. That is, $B= A_2\&A_5\&A_7\&A_9$,then B={0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0}, counting the number of 1s in array B, the support of B is 3,because its support is not less than the minimum support threshold, so $\{I_2,I_5,I_7,I_9\}$ is frequent 4 itemset, add $\{I_2,I_5,I_7,I_9\}$ to FS_D. Then add all subsets of $\{I_2,I_5,I_7,I_9\}$ in C_3 and delete it from C_3, through this process, $FS_D=\{\{I_2,I_5,I_7,I_9\},\{I_2,I_5,I_7\},\{I_2,I_5,I_9\},\{I_2,I_7,I_9\}\{I_5,I_7,I_9\}\}$, $C_3=\{\{I_3,I_4,I_7\},\{I_3,I_4,I_8\},\{I_3,I_4,I_9\},\{I_3,I_4,I_{11}\},\{I_3,I_6,I_8\},\{I_3,I_7,I_9\},\{I_3,I_7,I_{11}\},\{I_3,I_8,I_9\},\{I_4,I_7,I_9\},\{I_4,I_7,I_{11}\},\{I_4,I_8,I_9\}\}$, $C_4=\{\{I_3,I_4,I_7,I_9\},\{I_3,I_4,I_7,I_{11}\},\{I_3,I_4,I_8,I_9\}\}$.

Then verify candidate itemset $\{I_3,I_4,I_7,I_9\}$, $B=A_3\&A_4\&A_7\&A_9$={1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0}, the number of 1s is 3,which is not less than min_sup, so $\{I_3,I_4,I_7,I_9\}$ is a frequent itemset, Then add all subsets of $\{I_3,I_4,I_7,I_9\}$ in C_3 and delete it from C_3, through this process, $FS_D=\{\{I_2,I_5,I_7,I_9\},\{I_2,I_5,I_7\},\{I_2,I_5,I_9\},\{I_2,I_7,I_9\},\{I_5,I_7,I_9\},\{I_3,I_4,I_7\},\{I_3,I_4,I_9\},\{I_3,I_7,I_9\},\{I_4,I_7,I_9\},\{I_3,I_4,I_7,I_9\}\}$, $C_3=\{\{I_3,I_4,I_8\},\{I_3,I_4,I_{11}\},\{I_3,I_6,I_8\},\{I_3,I_7,I_{11}\},\{I_3,I_8,I_9\},\{I_4,I_7,I_{11}\},\{I_4,I_8,I_9\}\}$, $C_4=\{\{I_3,I_4,I_7,I_{11}\},\{I_3,I_4,I_8,I_9\}\}$.

Candidate itemset $\{I_3,I_4,I_7,I_{11}\}$ is not a frequent itemset because $B= A_3\&A_4\&A_7\&A_{11}=$ ={0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0}, in which the number of 1s is 2. Candidate itemset $\{I_3,I_4,I_8,I_9\}$ is also not a frequent itemset because its support is less than min_sup. At this time, C_4 is null. Then verify the remaining items in C_3, by using the same method to verify each candidate itemset. Finally, $FS_D=\{\{I_2,I_5\},\{I_2,I_7\},\{I_2,I_9\},\{I_3,I_4\},\{I_3,I_6\},\{I_3,I_7\},\{I_3,I_8\},\{I_3,I_9\},\{I_3,I_{11}\},\{I_4,I_7\},\{I_4,I_8\},\{I_4,I_9\},\{I_4,I_{11}\},\{I_5,I_7\},\{I_5,I_9\},\{I_6,I_8\},\{I_7,I_9\},\{I_7,I_{11}\},\{I_8,I_9\},\{I_2,I_5,I_7\},\{I_2,I_5,I_9\},\{I_2,I_7,I_9\},\{I_3,I_4,I_7\},\{I_3,I_4,I_8\},\{I_3,I_4,I_9\},\{I_3,I_4,I_{11}\},\{I_3,I_6,I_8\},\{I_3,I_7,I_9\},\{I_3,I_8,I_9\},\{I_4,I_7,I_9\},\{I_4,I_7,I_{11}\},\{I_5,I_7,I_9\},\{I_2,I_5,I_7,I_9\},\{I_3,I_4,I_7,I_9\}\}$.

So we can generate association rules according to generated frequent itemsets and generating association rules algorithm. It is simple; this paper does not study this problem.

4. Performance Analysis and Testing

From the above analysis, the improved algorithm can generate all frequent itemsets, and the improved algorithm has many advantages compared to classic Apriori algorithm and other improved algorithms in generating frequent itemsets, as follows:

(1) The improved algorithm need scan database only once, which greatly reduces time overhead, Apriori algorithm need scan database many times, Apriori_M algorithm also need scan database two times, DFS algorithm also need scan database two times.

(2) Construct the upper triangular matrix when scanning database, in the upper triangular matrix the rows represent the item set of database and the columns also represent the item set of database. So the upper and the lower are same, then we only need storage the upper of matrix, which saves a lot of space.

(3) The improved algorithm can easily generate frequent 2 itemset by traversing UTM, if the value of the coordinates [i,j] is not less than the minimum support threshold, then the item set $\{I_i, I_j\}$ is frequent 2 itemset. So the improved algorithm solves the bottleneck problem of generating frequent 2 itemset.

(4) The improved algorithm adopts binary 1 and 0 to storage transaction information instead of storing transaction number TID, if transaction number TID need k bytes, then DFS algorithm need k*8*n*m bits storage space, but the improved algorithm only need n*m bits storage space, n is the number of transactions, m is the number of items for database. So k, m and n are more, the improved algorithm can save more space.

(5) Although the improved algorithm need generate candidate item set, but it uses connected items and UTM, so it is efficient to generate candidate item set. There have n candidate k itemsets, Apriori algorithm need perform join operation m^2 times, but the improved algorithm only needs perform join operation m times.

(6) The improved algorithm verifies whether a candidate item set is a frequent item set by using logic AND operation, which greatly improve the speed of generating frequent itemsets. In addition, the algorithm can generate all frequent itemsets, which solves the problem that some algorithm [18] can not generate all frequent itemsets.

In order to further verify the performance of the improved algorithm, this paper implemented Apriori algorithm, DFS algorithm and the improved algorithm. Experimental environment: Memory is 1 G, CPU is Intel Pentium M 1.73 GHZ, using C# language, and test database is mushroom database which is provided by Literature [23]. Experiment results obtained in different minimum support are shown in Fig. 1.

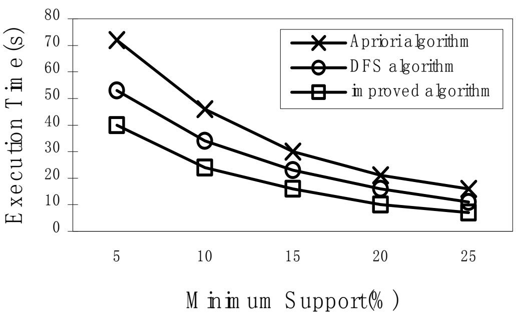

Fig. 1. Performance comparison of three algorithms in different minimum support.

From the Fig. 1, we can see clearly that the improved algorithm is better than Apriori algorithm and DFS algorithms in different minimum support. The improved algorithm is more obvious advantage when the minimum support is lower.

How is the improved algorithm when database scale is more and more large? We still in mushroom database for example, because records in mushroom database are very little, so copy records. Experiment results obtained in different database scale are shown in Fig. 2.

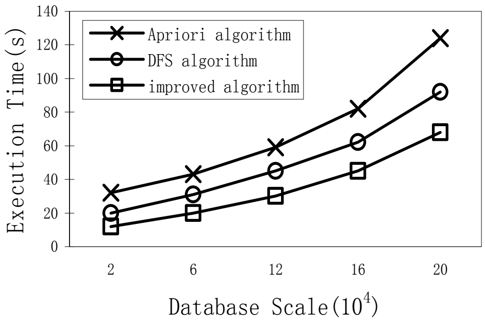

Fig. 2. Performance comparison of three algorithms in different scale database.

From the Fig. 2, we can see clearly that the improved algorithm is better than Apriori algorithm and DFS algorithms in different scale database. The improved algorithm is more obvious advantage when database scale is very large. So the improved algorithm is very suitable for large scale database.

Experimental results show that the improved algorithm has more advantages than Apriori algorithm and DFS algorithm in different minimum support or different database scale. So the improved algorithm is useful to mine frequent itemsets, and the improved algorithm has lower time and space overhead.

5. Conclusion

This paper proposes the new generating frequent itemset method for database, the algorithm reduces the number of scans in the database and improves efficiency and reduces the computing time by taking the advantage of connected item and UTM and logic AND operation technique and reduces the space overhead by using binary 1 and 0. Experimental results show that the improved algorithm has better efficiency than other algorithms, and the improved algorithm is more suitable for large scale database.

Although the improved algorithm has many advantages, we also found the improved algorithm has some problems to be studied further. For example, the improved algorithm can not process incremental update mining. We will give improved research on this.

References

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

[2]. Balaji Raja N. and Balakrishnan G., Evaluation of Rule Likeness Measures for a Learning Environment, in *Proceedings of the ICOREM International Conf.*, 2009, pp. 1682-1690.

[3]. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, 2001.

[4]. Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases, in *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, 1993, pp. 207-216.

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

[6]. Tsai C. F., Lin Y. C., Chen C. P., A new fast algorithms for mining association rules in large databases, in *Proceedings of the 2002 IEEE Int'l Conf. on Systems, Man and Cybernetics (SMC'02)*, 2002, pp. 251-256.

[7]. Aly H. H., Taha Y., Amr A. A., Fast mining of association rules in large-scale problems, in *Proceedings of the 6th IEEE Symp. on Computers and Communications (ISCC'01)*, 2001, pp. 107-113.

[8]. Han J, Pei J, Yin Y, Mining frequent patterns without candidate generation, in *Proceedings of the ACM SIGMOD*, 2000, pp. 1-12.

[9]. F. Berzal, J. C. Cubero, N. Marin, J. M. Serrano, TBAR, An Efficient Method for Association Rule Mining in Relational Databases, *Elserier Data and Knowledge Engineering*, 2001, pp. 47-64.

[10]. Park J. S., Chenm S., Yu P. S., An effective Hash based algorithm for mining association rules, in *Proceedings of International Conference on the Special Interest Group on Management of Data*, 1995, pp. 175-186.

[11]. You Lei, Lan Yang, Xiong Yan, An Optimized Apriori Implementation Based on Relational Algebra, *Journal of Xinyang Normal University*, Vol. 23, Issue 1, 2010, pp. 156-160.

[12]. Tolvonen H, Sampling large databases for association roles, in *Proceedings of the 22nd International Conference on Very Large Database,* Bombay, India [s.n.], 1996, pp. 134-145.

[13]. Brin S., Dynamic itemset counting and implication rules for market basket analysis, in *Proceedings of International Conference on the Special Interest Group on Management of Data*, 1997, pp. 255-264.

[14]. Cakmak A., Ozsoyoglu G., Taxonomy superimposed graph mining, in *Proceedings of the 11th International Conf. on Extending Database Technology, Advances in Database Technology, ACM Int'l Conf. Proceeding Series*, Vol. 261, 2008, pp. 217-228.

[15]. Silvescu A., Caragea D., Anna Atramentov, Graph Databases Artificial Intelligence Research Laboratory, *Department of Computer Science Iowa State University Ames*, Iowa, 2007.

[16]. Sadhana Priyadarshini and Debahuti Mishra, A Hybridized Graph Mining Approach, in *Proceedings of the ICT2010, CCIS 101*, 2010, pp. 356-361.

[17]. He Jian-Zhong, Lv Zhen-Jun, Optimized Algorithm for Mining Association Rule Based on Two Matrixes, *Computer Engineering*, Vol. 34, Issue 17, 2008, pp. 56-61.

[18]. Huang Long-Jun, Duan Long-Zhen, An Frequent Mining Algorithm Based on UTM, *Application Research of Computers*, Vol. 11, Issue 25, 2006, pp. 25-40.

[19]. Zhang Yuntao, Yu Zhilou, Zhang Huaxiang, Research on high efficiency mining frequent itemsets on association rules, *Computer Engineering and Applications*, Vol. 47, Issue 3, 2011, pp. 139-141.

[20]. Wang Ming, Song Shunli, Algorithm for discovering frequent item sets based on optimized and regrouped item sets, *Journal of Computer Application*, 9, 2010, pp. 2332-2334.

[21]. D. Kerana Hanirex, Dr. A. Kumaravel, An efficient partition and two dimensional approach for mining frequent itemsets, *International Journal of Technological Synthesis and Analysis*, 12, 2012, pp. 14-17.

[22]. Jaishree Singh, Hari Ram, Dr. J. S. Sodhi, Improving efficiency of Apriori algorithm using transaction reduction, *International Journal of Scientific and Research Publications*, Vol. 3, Issue, 1, 2013, pp. 1-4.

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