The Research of Association Rules Mining Algorithm Based on Binary

Gang FANG[1], Zu-Kuan WEI[1*], Qian YIN[2]
1 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, P.R. China
2 College of Information Science and Technology, Beijing Normal University, Beijing 100875, P.R. China
E-MAIL: fg_7@163.com, anlexwee@uestc.edu.cn*, yinqian@bnu.edu.cn

***Abstract*—An algorithm of association rules mining based on binary has been introduced to solve two problems that how to easily generate candidate frequent itemsets and fast compute support. However the basic notion of presented algorithms in generating candidate itemsets is still similar to Apriori. In some degree the efficiency of these algorithms is very confined, and so this paper proposes two different searching strategies of association rules mining algorithms based on binary, which are suitable for mining corresponding characteristic database. One is that the notion of generating candidate frequent itemsets is similar to up-down searching of traditional association rules mining algorithm, which uses the method of forming subset to generate candidate frequent itemsets from long to short and is suitable for mining long frequent itemsets. The other is that the method of increasing value is used to generate candidate frequent itemsets, which is more efficient than Apriori based on binary and is more suitable for mining short frequent itemsets, in this mining course length of candidate frequent itemsets crossways varies from short to long. The both algorithms use digital character to reduce the number of scanned transactions. The experiment based on above three algorithms indicates that the efficiency of two presented algorithms is fast and efficient when mining corresponding characteristic database.**

***Keywords*—data mining, association rules, binary, increasing search, digital transaction**

I. INTRODUCTION

Traditional association rules mining algorithm Apriori [1] mainly wants to solve two problems: one is that how to reduce the number of candidate frequent itemsets and the times of browsing database, the other is that how to generate candidate frequent itemsets and compute the support of candidate frequent itemsets. In order to solve the first problem, scholars presented many algorithms, such as Max-Miner [2], Pincer-Search [3], DMFI [6] and DMFIA [7]. And then, in order to solve the second, some algorithms based on binary [8, 9, 11] were presented, such as B-Apriori [8]. The kind of algorithm calculates support by binary logic "and" operation to indeed improve efficiency. However, at present, some of algorithms based on binary generate candidate frequent itemsets still according to the notion of Apriori. When mining some character of database, the efficiency of these algorithms is badly influenced, therefor this paper proposes two kinds of association rules mining algorithm based on binary. An algorithm denoted by B_UDMA uses strategy of up-down searching to generate candidate frequent itemsets of binary. The algorithm uses binary logical "and" operation to compute support of itemsets and uses the method of forming subset to generate candidate frequent itemsets from long to short, which is suitable for mining long frequent itemsets. The other denoted by B_IVMA uses the method of increasing value to generate candidate frequent itemsets of digital, which is more efficient than Apriori and is suitable for mining short frequent itemsets. In this mining course length of candidate frequent itemsets crossways varies from short to long. The both algorithms use digital character to reduce the number of scanned transactions. The experiment based on above three algorithms indicates that the efficiency of two presented algorithms is fast and efficient at the time of mining corresponding characteristic database.

II. BASIC NOTIONS AND PROPERTIES

Let $I=\{i_1, i_2 \ldots i_m\}$ be a set of items, if i_k ($i_k \in I$), let $T=\{i_1 \wedge i_2 \wedge \ldots \wedge i_m\}$ ($T_k \subseteq I$) be a subset of items, called a Transactions. For example, let $T_k=\{i_1, i_2, i_3\}$ be a subset of items, called a transaction. And then let $D=\{T_1, T_2 \ldots T_n\}$, let $T_k \subseteq I$, (k=1…n) be a set of transactions, called Transaction Database (*TD*).

Definition 1 Binary Transaction (*BT*), a transaction is expressed as binary, binary transaction of transaction T is expressed as $BT=(b_1\, b_2 \ldots b_m)$, $b_k \in [0, 1]$, k=1…m.

If $i_k \in T_i$, and then b_k=1, otherwise b_k=0, the order of binary digits is as same as items which have been fixed.

Example, let I={1,2,3,4,5} be a set of items, if a transaction is expressed as T_i={2,3,5}, and then BT_i=(01101).

Definition 2 Digital Transaction (*DT*), which is an integer, the value of which would be obtained by turning binary of transaction into algorism.

Example, if *BT*=01101, and then *DT*=13.

Definition 3 Digital Transaction Length (*DTL*), which is an integer, the value of which is equal to the number of digital "1" that is contained in digital transaction.

Example, if *DT*=13, and then *DTL*=3.

* corresponding author
This work was fully supported by a grant from the S&T Foundation of Chengdu Sci.&Tech. Bureau. (Project No. 06GGYB801GX-032)

Definition 4 Relation of digital transaction accord with relation of transaction set.

Example, suppose digital transaction of a transaction T_1 is denoted by DT_1, digital transaction of a transaction T_2 is denoted by DT_2. If $T_1 \subseteq T_2$, and then $DT_1 \subseteq DT_2$, DT_1 is regarded as subset of DT_2, DT_2 is regarded as superset of DT_1.

Definition 5 Frequent Digital Transaction (*FDT*), which is a digital transaction, the support of which excess minimal support given by users.

Definition 6 Candidate Digital Transaction Section (*CDTS*), this is an integral section from 3 to max, which includes all the candidate frequent digital transaction that may generate frequent digital transaction except for power of 2. Here 3 is regarded as initialization because 3 is the least digital transaction expressing dualistic relation, and max is regarded as the maximal digital transaction in database.

Example, if the maximal digital transaction in database is denoted by max=63, and so *CDTS* = [3, 63].

Definition 7 Candidate Frequent Digital Transaction (*CFDT*), which is a digital transaction that may become frequent digital transaction from *CDTS*.

Property 1 The given binary transaction uniquely corresponds to a digital transaction, vice versa.

Property 2 Let p and q be binary transactions with m bits, let T_p be transaction about p, let T_q be transaction about q, then $T_p \subseteq T_q \Leftrightarrow p \wedge q=p$.

Proof is expressed as follows:

Suppose digit 1 locates each bit of binary p from i_{j1} to i_{jk} ($j_k \leq m$), digit 0 locates other ones. If p and $q=p$, then digit 1 locates each bit of binary q from i_{j1} to i_{jk} ($j_k \leq$m) (otherwise these bits must occur digit 0 with logic "and" operation), other ones will be either 0 or 1, so $T_p \subseteq T_q$ according to definition 1.

And then via the hypothesis about p as before, since $T_p \subseteq T_q$, then digit 1 must locate each bits of binary q from i_{j1} to i_{jk} ($j_k \leq m$) (otherwise, $\exists\ i_k$, so $i_k \in T_p$, $i_k \notin T_q$, the conclusion is contrary to premises as $T_p \subseteq T_q$), other ones will be either 0 or 1, so $p \wedge q=p$.

Via theorems as before, there are two conclusions obviously deduced.

Conclusion 1 Let p and q be binary transactions, let DT_p be digital transaction about p, let DT_q be digital transaction about q. If $p \wedge q=p$, and then $DT_p \leq DT_q$.

Conclusion 2 Let p and q be binary transactions, let T_p be transaction about p, let DT_p be digital transaction about p, let T_q be transaction about q, let DT_q be digital transaction about q. If $DT_p \leq DT_q$, then $T_q \not\subset T_p$.

Property 3 Let p and q be binary transactions with m bits, let T_p be transaction about p, let T_q be transaction about q, and $p \wedge q=p$, let F be frequent itemsets.

①If $T_q \subseteq F$, then $T_p \subseteq F$.

②If $T_p \not\subset F$, then $T_q \not\subset F$

Proof is expressed as follows:

Since $p \wedge q=p$, via **property 2**, then $T_p \subseteq T_q$, and $T_q \subseteq F$, based on set theory, then $T_p \subseteq F$. If $T_p \not\subset F$, then $T_q \not\subset F$ as the same theory.

III. THE UP-DOWN SEARCHING STRATEGY MINING ALGORITHM BASED ON BINARY

A. The key technologies of mining algorithm

As we all know, it is the most key technology for association rules mining algorithm to search frequent itemsets which includes generating candidate frequent itemsets and pruning redundancy candidate frequent items. According to up-down [6] searching strategy of traditional association rules, the process of up-down searching frequent digital transaction based on binary is expressed as follows:

Let I= $\{i_1, i_2 \ldots i_m\}$ be a set of items, digital transactions of database are sorted on descending. If candidate frequent digital transaction is denoted by $CFDT_k$, where k is equal to the number of items in *CFDT*, and is also equal to digital transaction length.

Step1: Generating the first C_k= $\{i_1, i_2 \ldots i_m\}$ (k=m) which is denoted by $CFDT_k$.

Step2: At first, if $CFDT_k$ can't belong to frequent digital transaction denoted by *FDT*, and then calculating support of $CFDT_k$ which is denoted by S_k. if $S_k \geq$minimal support, then $CFDT_k$ is became into FDT_k, and adding FDT_k to *FDT*, otherwise the algorithm would go on generating candidate frequent digital transaction $CFDT^i_{k-1}$, which is the (k-1) subsets of $CFDT_k$, and if $CFDT^i_{k-1}$ is subsets of *FDT*, it will be pruned via property 3.

Step3: Aiming to each $CFDT^i_{k-1}$, we regard $CFDT^i_{k-1}$ as $CFDT_k$ in step 1, and then repeatedly execute from step 2 to 3 until k=1.

The algorithm of generating candidate digital transaction from $CFDT_k$ to $CFDT^i_{k-1}$ is expressed as follows:

We regarded $\{i_k\}$ (k=1…m) as a transaction, digital transaction of which is denoted by DT^k.

The algorithm of *Creating-Candidate* (int [] *Array*) will return all $CFDT^i_{k-1}$ after inputting $CFDT_k$.

```
(1) Array [0] =CFDT_k;
(2) c=0;
(3) For (i=1; i≤m; i++)
(4) {
(5)   If (DT^i ⊂ Array [0])
(6)    Array [++c] =Array [0] & (~ DT^i);
(7) }
(8) Array [0] =c;
(9) Return Array;
```

B. The algorithm of up-down generating frequent digital transaction based on binary

Firstly, we define following signs:

DT_0: expressing digital transaction of $\{i_1, i_2 \ldots i_m\}$.

DB: expressing database with N digital transactions in a descending order.

DT_i: expressing digital transaction in *DB*, which includes value and count.

F: saving frequent digital transaction.

NF: saving non-frequent digital transaction.

C: saving candidate frequent digital transaction.

The algorithm is expressed as follows:

```
(1) Write DT_0 to C;
```

(2) While (exit=false) {
(3) If($C!=\phi$) {
(4) $NF=\phi$;
(5) For(all $CFDT^i_k \in C$){
(6) S_count=*Count-Support* ($CFDT^i_k$);
(7) If (S_count≥*minisup*)
(8) Write $CFDT^i_k$ to *F*;
(9) Else
(10) Write $CFDT^i_k$ to *NF*;
(11) }//separating candidate itemsets
(12) $C=\phi$;
(13) For(all $DT^i_k \in NF$) {
(14) *Array* [0]= DT^i_k;
(15) *Creating-Candidate* (*Array*);
(16) Write *Array*[*i*] (i≠0) to *C* and pruning all repeated digital transactions in *C*;
(17) }
(18) Deleting all subset of *F* from *C*;
(19) }
(20) Else
(21) exit=true;
(22) }

Count-Support (int *DT*)

(1) While ($i \le N \&\& DT \le DB_i$.value) {
(2) If ($DT \subset DB_i$.value) support+=DB_i.count;
(3) *i*++;
(4) }
(5) Return support;

IV. THE INCREASING VALUE SEARCHING STRATEGY MINING ALGORITHM BASED ON BINARY

A. The process of generating frequent digital transaction

According to basic notion of algorithm Apriori, we further propose an increasing value searching strategy of mining algorithm based on binary, but the notion of generating candidate frequent itemsets is different from the other algorithm. The process of generating frequent digital transaction is expressed as follows:

Step1: Creating candidate digital transaction section (*CDTS*) according to database and **definition 5**.

Step2: Generating candidate frequent digital transaction (*CFDT*) from minimum digital to maximum digital by increasing value.

Setp3: Computing support of *CFDT* when it isn't superset of non frequent digital transaction denoted by *NFDT*. If support≥ minimal support, and then writing *CFDT* to frequent digital transaction denoted by *FDT* after deleting all subsets of *CFDT* from *FDT*, otherwise, writing *CFDT* to *NFDT*.

Setp4: Generating the next *CFDT*, and then repeatedly execute Step3 until maximum of *CDTS*.

B. The algorithm of increasing-value generating frequent digital transaction

Let [3, max] be a *CDTS*, and there are *N* digital transaction saved in *DB* on descending, where data are different from each other, and we define following signs:

DB: expressing database with *N* digital transactions in a descending order.

D_i: expressing digital transaction in *DB*, which includes value and count.

FDT: saving frequent digital transaction.

NFDT: saving non-frequent digital transaction.

The algorithm is expressed as follows:

(1) While ($DT \in$ [3, max]) {
(2) *DT*=3;
(3) If (all $NFDT_j \not\subset DT$) {
(4) While ($i \le N \&\& DT \le D_i$.value) {// $D_i \in DB$
(5) If ($DT \subseteq D_i$.value)
(6) s_count+= D_i.count;
(7) *i*++;
(8) }//calculating support of items
(9) If (s_count/*N*≥**support**) {
(10) Delete all FDT_k ($FDT_k \subset DT$) from *FDT*;
(11) Write *DT* to *FDT*;
(12) }
(13) Else
(14) Write *DT* to *NFDT*;
(15) }
(16) *DT*++;
(17) }

V. COMPARING CAPABILITY OF ALGORITHM

Aiming to the algorithm of B_Apriori presented by reference[8], this paper proposes two different searching strategies of mining algorithm based on binary, which are B_UDMA and B_IVMA. Firstly, we compare capability of these algorithms, which are expressed as follows:

B_Apriori: The algorithm uses combination of items to generate candidate frequent items based on binary. The number of items contained by candidate frequent itemsets varies from fewness to many, namely, generating (k+1)-candidate itemsets after inputting some k-frequent itemsets. The algorithm uses the method of ② in **property 3** to prune redundant candidate frequent itemsets. And the algorithm uses binary logical "and" operation to calculate support of itemsets. However, if the number of item contained by frequent itemsets is too many, the algorithm will generate a lot of candidate frequent itemsets and repeatedly scan database, and because of these the efficiency of B_Apriori is too bad. Hence, the algorithm is suitable for mining relative short frequent itemsets.

B_UDMA: The algorithm uses forming subset of non frequent itemsets to generate candidate frequent items based on binary. The number of items contained by candidate frequent itemsets varies from many to fewness, namely, generating *k*-candidate itemsets after inputting (k+1) non-frequent itemsets. The algorithm uses the method of ① in **property 3** to prune redundant frequent itemsets. And the algorithm also uses binary logical "and" operation to calculate support of itemsets, and at the time it is the most key to use **conclusion 2** of **property 2**, namely digital character, to reduce the number of scanned transactions in database, which is different from B_Apriori. However, if the number of item

contained by frequent itemsets is too few and the number of frequent itemsets is too many, the algorithm will generate a lot of candidate frequent itemsets and repeatedly scan database, and so since these the efficiency of B_UDMA is also too bad. Hence, the algorithm is suitable for mining long frequent itemsets.

B_IVMA: The algorithm generates candidate frequent itemsets by automatically increasing value. *DTL* of candidate frequent digital transaction generated across varies, but trend of variation is from few to many. For example:

If *CDTS*= [3, 15], the sequence of generating candidate frequent itemsets is expressed as follows:

DT_1=3, DTL_1=2.
DT_*=4, DTL_*=1. (4 have been pruned via **definition 6**)
DT_2=5, DTL_2=2.
DT_3=6, DTL_3=2.
DT_4=7, DTL_4=3.
DT_*=8, DTL_*=1. (8 have been pruned via **definition 6**)
DT_5=9, DTL_5=2.
DT_6=10, DTL_6=2.
DT_7=11, DTL_7=3.
DT_8=12, DTL_8=2.
DT_9=13, DTL_9=3.
DT_{10}=14, DTL_{10}=3.
DT_{11}=15, DTL_{11}=4.

As before their lengths crossways vary from 2 to 4, candidate frequent itemsets are easily generated. The method of the algorithm is different from the other algorithm. The algorithm uses the method of ① and ② in **property 3** to prune redundant candidate frequent itemsets. The algorithm also uses binary logical "and" operation to calculate support of itemsets, and at the time it is the most key to use **conclusion 2** of **property 2**, namely digital character, to reduce the number of scanned transactions in database. However, if the number of item contained by frequent digital transaction is too many, the algorithm will generate a lot of candidate frequent digital transaction, and so since these the efficiency of B_IVMA is too bad. But it is easier to create candidate frequent items than B_Apriori and B_UDMA. Hence, the algorithm is more efficient than B_Apriori, which is more suitable for mining short frequent itemsets.

Now we use result of experiment to testify above analyses. Three mining algorithms are used to generate frequent itemsets from these digital transactions, which are expresses as digital from 3 to 4095, m=12, *N*=4083. Our experimental circumstances are expressed as follow: AMD Athlon (tm) 64X2 Dual Core Processor 3800+ 2.00 GHz, 512MB, language of the procedure is Visual C# 2005.NET, OS is Windows XP Professional.

One of experiment results is expressed as figure 1 and figure 2, here is absolute support. As the relative support of frequent itemsets changes, the executed time of algorithms is expressed as figure 3. As the length of frequent itemsets changes, the executed time of algorithms is expressed as figure 4 and 5.

According to these result, we can draw a conclusion as follows: B_UDMA is suitable for mining long frequent itemsets. B_IVMA is more efficient than B_Apriori and is more suitable for mining relative short frequent itemsets.

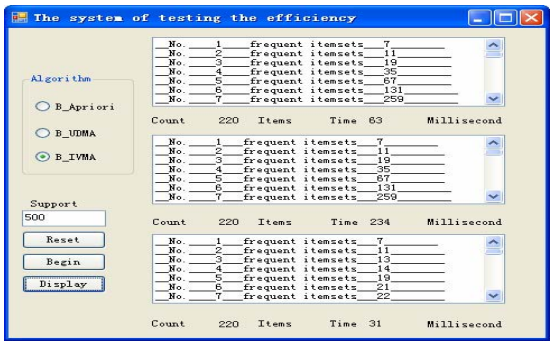

Figure 1. The result of experiment Length=3. B_IVMA< B_Apriori< B_UDMA

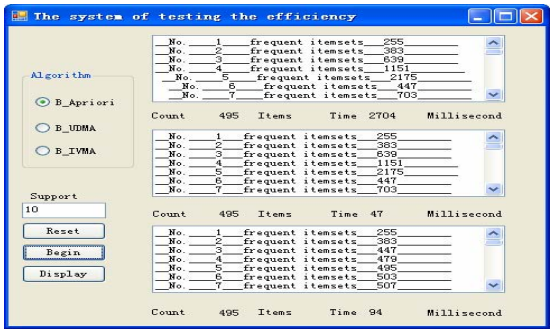

Figure 2. The result of experiment Length=8. B_UDMA< B_IVMA< B_Apriori

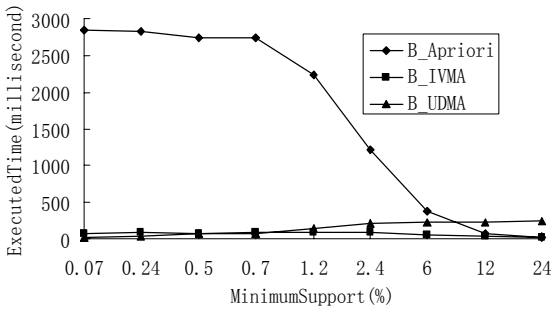

Figure 3. The executed time of three algorithms as changing of support

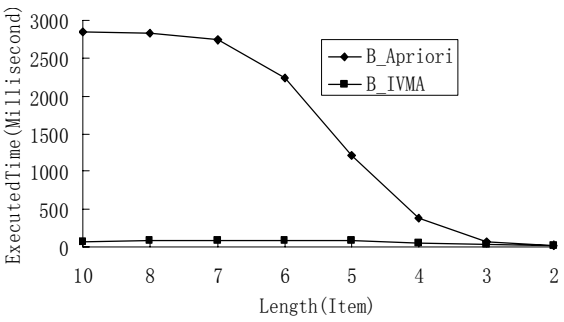

Figure 4. The executed time as changing of length

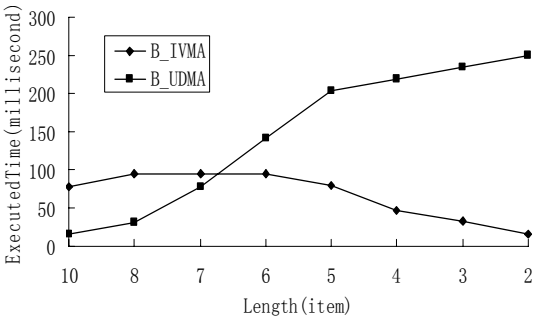

Figure 5. The executed time as changing of length

VI. CONCLUSIONS

The algorithms of association rules mining based on binary have been presented in order to easily generate candidate frequent itemsets and fast compute support of itemsets. This paper proposes an algorithm of up-down association rules mining based on the notion of Apriori denoted by B_UDMA, which is suitable for mining long frequent. Another algorithm of increasing value association rules mining denoted by B_IVMA is different from the basic notion of Apriori, which is more efficient than B_Apriori and is more suitable for mining relative short frequent itemsets. It is two key ways of innovation to use the method of increasing value to generate candidate frequent itemsets and use digital character to reduce the number of scanned transactions in database. The experiment indicates two kinds of algorithms are fast and efficient for mining corresponding to character database.

ACKNOWLEDGMENT

This work was fully supported by a grant from the S&T Foundation of Chengdu Sci.&Tech. Bureau. (Project No. 06GGYB801GX-032).

REFERENCES

[1] R.Agrawal, T.Imielinski, A.Swami1. Mining association rules between sets of items in large databases. ACM SIGMOD Int'l Conf. Management of Data, Washington, D. C., 1993.

[2] R.Bayardo. Efficiently mining long patterns from databases. In: L.M.Haas, A.Tiwary, eds. Proc. of the ACM SIGMOD Int'l Conf. Management of Data. New York: ACM Press, 1998. 85~93.

[3] Lin D, Kedem ZM. Pincer-Search: A new algorithm for discovering the maximum frequent set. In: H.J.Schek, F.Saltor,I. Ramos et al. eds. Proc. of the 6th European Conf. Extending Database Technology. Berlin: Springer-Veriag, 1998. 105~119.

[4] Agrawal, R Srikant. Fast algorithms for mining association rules. In: Proc. Of the 20th Int'l Conf. Very Large Data Bases (VLDB'94).1994.487-499.

[5] Feng Yu-Cai, Feng Jian-Lin. Incremental updating algorithms for mining association rules. Journal of Software, 1998,9(4):301-306.

[6] Lu SF, Lu ZD. Fast mining maximum frequent itemsets. Journal of Software, 2001, 12(2):293−297.

[7] Song YQ, Zhu YQ, Sun ZH, Chen G. An algorithm an its updating algorithm based on FP-Tree for mining maximum frequent itemsets. Journal of Software, 2003,14(9):1586−1592.

[8] Chen Geng, Zhu Yuquan, Yang Hebiao, Study of Some Key Techniques in Mining Association Rule, Journal of Computer Research and Development 2005,42 (10) : 1785～1789.

[9] Wang Lizhen, Zhou Lihua, A Greedy Algorithm for Discovering Frequent Itemsets, Journal of Computer Engineering and Applications 2001,13 86-88.

[10] JI Gen-Lin, YANG Ming. SONG Yu-Qing. SUN Zhi-Hui. Fast Updating Maximum Frequent Itemsets. Chinese Journal of Computers, 2005,28(1):128−135.

[11] FAN Ping, LIANG Jia-rong, LI Tian-zhi, GONG Jian-min, Association rules mining algorithm based on binary. Journal of Application Research of Computers 2007,24(8): 79-81.