http://ansinet.com/itj

ITJ

ISSN 1812-5638

# INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (8): 1235-1241, 2009
ISSN 1812-5638


# A Fast Association Rules Mining Algorithm for Dynamic Updated Databases

[1]Ni Tian-quan, [1]Wang Jian-dong, [2]Peng Xiao-bing and [3]LiuYi-an
[1]Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China
[2]Jiangsu University, Zhenjiang, 212013, China
[3]Information Engineering School of Southern Yangtze University, Wuxi, 214122, China

**Abstract:** To overcome the difficulty of updating frequent item sets in the dynamic database, this study proposes a new algorithm for efficiently mining association rules in dynamic updated databases. The algorithm constructs the corresponding vector subspace according to the number of nonempty subsets in the item sets which is based on the concept of the Apriori algorithm that the maximal frequent item sets are definitely the subsets of database's item set. After the construction of the vector subspace, the dynamic tuples additions and deletions of the database, as well as the updated solutions to the frequent item sets when the minimum support is changed, are determined efficiently by the vector inner computing. Studies show that the algorithm is not only simple in that it needs only to scan the database once, but also capable of processing super database.

**Key words:** Association rules, data updating, frequent item set

## INTRODUCTION

As a highly interdisciplinary area that includes databases, artificial intelligence, machine learning, statistical theory and other fields of technology, Data Mining has become a hot topic in the research of database technology and applications in recent years. Apriori (Agrawal *et al*., 1993) algorithm is the most famous and basic among many of the association ruled mining algorithms. The core idea of its algorithm is based on a recursive method of the frequent set theory and its purpose is to dig out the association rules whose support and confidence are not less than the given minimum support threshold and minimum confidence threshold, respectively. However, Apriori algorithm needs to repeatedly scan the database to find frequent item sets so that it has to process a large number of candidate sets and results in inefficient memory utilization. Many researchers have put forward a number of ways to improve or expand the Apriori algorithm recent years. In order to decrease scanning data-base the PARTITION (Savasere *et al*., 1995) algorithm does split-horizon on a data-base which making it is possible to parallel processing each data set after segmentation in memory. But this way has less flexible as is needs rescan the database when finding news rules after supports is changed or new data is added in database. The DHP (Park *et al*., 1997) using Hash technique and curtail of services in data-base to improve Apriori though is it unpractical in some actual applications (Zaki, 1999). Han *et al*. (2000) proposed FP-growth method which puts frequent set into a FP-tree after the first scanning of the database and then it split FP-tree into condition-bases that correlative to some frequent sets that has unit length and digging these condition-bases, respectively at last. The FP-growth has good adaptability to rules of different length and has better efficiency than Apriori but it also, has to scan the whole database and the efficiency is not high to deal with large data-base. All mentioned algorithms above are non- increment methods so they need search the whole data-base to update rules when the data-base has little changes and this being low efficiency. To update, maintenance and manage association rules efficiently lots of work have been done in dynamic updating databases. Such algorithms as FUP (Cheung *et al*., 1996), FUP2 (Cheung *et al*., 1997), IUA (Feng and Feng, 1998), SWF (Lee and LIN, 2001), NEWIUA (Zhou, 1999), etc. FUP is a typically dynamic update method. Its basic framework is formed by a lot of iteration just like Apriori and DHP. FUP do pruning to primary candidate sets while scanning new added part of the database but to deal with large candidate sets is a time consuming work and FUP has disadvantage which is scanning the database repeatedly. FUP2 studies how digging goes when new services are added in database or some old services are deleted. It has to scan the database in every iteration and not suitable for minimal supports changes situation. SWF using strategy to decrease scan times of the database but the candidate sets will increase rapidly. NEWIUA studies increment digging when minimal supports changes and never

**Corresponding Author:** Ni Tian-quan, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

consider how to do when database updating. So it is important to study how to dig frequent sets in static or dynamic databases efficiency.

This study proposes a simple, practical method to find frequent item sets for data mining in both static and dynamic databases. First of all, in view of the method which using Apriori algorithm to look for frequent item sets in a static database, a static database is mapped into augmented Boolean matrix and then the algorithm constructs a vector subspace according to the number of items of nonempty subsets in item sets and quickly identifies the target frequent item sets according to the products of vectors in the vector subspace and row vectors in the augmented Boolean matrix; finally the algorithm gives the answer to the frequent item sets of dynamic database according to the products of vectors in vector subspace and vectors in additions/deletions tuple when the dynamic tuple additions and deletions in database occur and minimum support is changed. Present studies shows to convert the problem of digging frequent sets in database D into searching vectors in constructed subspace $W_K$ is not only need scan D only once and produce no middle candidate sets but also can cut searching space dynamically according to change of supports. This method also can provide more memory when we deal with very large database or update database incrementally via release memory that hold by augmented Boolean matrix R.

## DESCRIPTION OF ASSOCIATION RULE MINING

Association rule is one of the most important techniques in data mining and it was put forward firstly by Agrawal *et al.* (1993). It is used to mine the rules of potential relationships between item sets in customer database. At present, association rules mining has been an important direction in database mining. The Apriori algorithm could be divided into two steps:

- Based on the degree of support, generate frequent item sets
- Based on credibility, generate strong association rules. But the core of which is generated by frequent item sets in

That is to identify all item sets whose support is not less than a given min sup. The basic principle is the use of a layer by layer called the iterative search method, which uses k- item sets to explore (k+1)-item sets.

Let $I = \{I_1, I_2,..., I_m\}$ be a set of m different items of database D and each $I_i$(i = 1, 2,..., m) is equivalent to a commodity. $W = \{T_1, T_2,..., T_n\}$ is a set of service sets and each service $T_i$(i = 1, 2,..., n) is a set of commodities, $T_i \subseteq I$. Each service T has the unique identifier TID. And we call T a k-itemset if the cardinality of itemset T is k. For a given database D, its frequent itemsets generated as follows (SHI, 2002):

- Computing all the 1-itemsets (record as $C_1$) and finding all the commonly used 1-itemsets whose support ($C_1$)≥min sup, record as $L_1$
- Finding all candidate 2-itemsets (record as $C_2$) according to commonly used 1-itemsets and find all the commonly used 2-itemsets whose support ($C_2$)≥min sup, record as $L_2$
- Finding all candidate 3-itemsets (record as $C_3$) according to commonly used 2-itemsets and finding all the commonly used 3-itemsets whose support ($C_3$)≥min sup, record as $L_3$

Going on these steps until the higher-dimensional frequent itemsets cannot be found.

Clearly, Apriori algorithm required to scan the database many times and generate a few more candidate item sets. In particular, when the number of items is large, the complexity of the algorithm show exponential growth and lead to a serious impact on its operating efficiency.

## AUGMENTED BOOLEAN MATRIX

Let $I = \{I_1, I_2,..., I_m\}$ be a set of m different items in database D, each item $I_K$ (k = 1, 2, ..., m) equivalent to a commodity. $W = \{T_1, T_2, ..., T_n\}$ is a services set and each $T_i$ (i = 1, 2, ..., n) is a group of services, $T_i \subseteq I$. Each service T has a unique identifier TID. The number of items in item sets is called Dimension or length. If the length is k, then it is called k-itemsets. For any given database, make:

$$f: D \rightarrow R \quad (1)$$

Where:

$R = f(D) = (r_{ij})_{n\times(m+1)}$

where,

$$r_{ij} = \begin{cases} 1, & I_j \in T_i \\ 0, & I_j \notin T_i \vee j = m+1 \end{cases}$$

where, (i = 1,2, ..., n) (j = 1,2, ..., m+1). Thus, Database D can be mapped into an augmented Boolean matrix by use of f after only one scan. For example, the database D proposed by Professor Jiawei Han (Han and Kambr, 2001) can be mapped into an augmented Boolean matrix R, as the following (Fig. 1a, b).

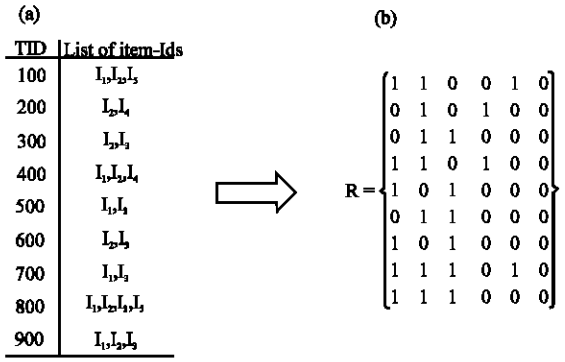


Fig. 1: (a) Transaction database and (b) Augmented boolean matrix

**Expression of transaction database:** It can be known from the above analysis that a corresponding augmented Boolean matrix will be determined after only one scan. Therefore, the issues of database mining can be translated into the analysis of augmented Boolean matrix. The Boolean matrix R is described in the form of augmented matrix to facilitate computation in the vector subspace, as discussed next. Each row in R in Fig. 1b corresponding to a relevant line in Fig. 1a, just like [1 1 0 0 1 0] corresponding to [$I_1$, $I_2$, $I_5$]. The other similar figure in this study has the same expression.

## CONSTRUCTION OF VECTOR SUBSPACE

It is not difficult to find that the frequent item sets of database D must be elements of power set $2^I$ corresponding to its item sets I = {$I_1$,$I_2$, ..., $I_m$}, which means, power set $2^I$ is the solution space of frequent item sets of database D. In order to determine the solution to frequent item sets of a dynamic database, it defines

$$e_i = (0,...,0,1,0...,0),\ i = 1,2,...,\ m$$

which is a (m+1)-dimensional vector. Here, the I-th place of $e_i$ is 1, others are 0. The (m+1)-th place of $e_i$ is to store the statistics support of vector of database.

For the solution space to power set $2^I$ excluding null elements, m (m+1)-dimensional vector-subspace can be built according to the number of items contained in element:

$$\begin{cases} W_1 = \{e_i \mid i = 1,2,\cdots,m\} \\ W_2 = \{e_{i_1} + e_{i_2} \mid 1 \le i_1 < i_2 \le m; i_1, i_2 \in Z\} \\ \cdots\cdots \\ W_k = \{e_{i_1} + e_{i_2} + \cdots + e_{i_k} \mid 1 \le i_1 < i_2 < \cdots < i_k \le m; i_1, i_2, \cdots, i_k \in Z\} \\ \cdots\cdots \\ W_m = \{e_1 + e_2 + \cdots + e_m\} \end{cases} \quad (2)$$

Apparently, there must exist a m+1-dimensional vector corresponding to the maximum frequent itemsets in some vector subspace $W_K$, k = 1,2,..., m, based on the core idea of the Apriori algorithm.

## SEARCHING ALGORITHM FOR FREQUENT ITEM SETS

Assume that the augmented Boolean matrix corresponding to database D is R, R = $(\alpha_1, \alpha_2,..., \alpha_n)^T$ where, T means transpose, $\alpha_i$ is m+1-dimensional row vector in real domain, I = 1,2,..., n. Assume that I = {$I_1$,$I_2$..., $I_m$} is a set of different items in D and the subset $T_p \subseteq I$ is the maximum k-frequent item sets of D, then m+1-dimensional vector β must satisfy $\beta \in W_K$ but $\beta \notin W_j$, $j \in \{1,2,..., k-1, k+1,..., m\}$ and:

$$<\beta, \alpha_i> \le k,\ i = 1,2,...,\ n \quad (3)$$

Here, $<\beta,\alpha_i>$ is the inner product of β and $\alpha_i$. It is clear that the sum of the inner product will not exceed k. Therefore, to find the maximum k-frequent item sets of database D which users need, we calculate the inner product of vector $\beta_{kj}$,, $j = 1,2,\cdots,C_m^k$ ($\beta_{kj}$ is vector of $W_K$ in Eq. 2, k = 1,2,..., m) and each row vector of augmented Boolean matrix R, here, $<\beta_{kj},\alpha_i> \ge k$, i = 1,2,..., n, then get the statistical supports $sum_k(\beta_{kj})$ and save it on the m+1-th place in vector $\beta_{kj}$. Select the vector $\beta_{kj}$ corresponding to the maximum k satisfying $sum_k(\beta_{kj}) \ge$ min sup and the item set of its k items on non-zero places is the maximum k-frequent item set that we need. The detailed procedure is as follows:

- Building m m+1-dimensional vector subspace $W_K$, k = 1,2,..., m, as described in Eq .2, according to the number of items in database D, m
- Scanning database D, construct the corresponding augmented Boolean matrix R = $(\alpha_1,\alpha_2,..., \alpha_n)^T$ and give the minimum support number expected by users (min sup)
- Calculating the inner product of vector $\beta_{kj}$, $j = 1,2,\cdots,C_m^k$ ($\beta_{kj}$ is vector of $W_K$ in Eq. 2, k = 1,2,..., m) nd each row vector of augmented Boolean matrix R, here, $<\beta_{kj},\alpha> \ge k$, i = 1,2,...,n, then get the statistics of support number $sum_k$ ($\beta_{kj}$) and save it on the m+1-th place in vector $\beta_{kj}$
- Selecting the vector $\beta_{kj}$ corresponding to the maximum k satisfying $sum_k$ ($\beta_{kj}$) ≥ min sup and the item set of its k items on non-zero places is the maximum

Obviously, this algorithm has the following advantages:

- Algorithm is not only simple but also able to handle large item sets dimension
- One scan of the database is sufficient and there are no middle-designate issues involved in searching frequent item sets
- Frequent item sets mining of database D could be turned into searching vector in vector subspace $W_K$, so larger memory space could be provided when handling super large database or incremented updating database by releasing the memory space of augmented Boolean matrix R
- The algorithm can be applied in situations where users require variable min sup
- For static database mining, this algorithm could be simplified when searching frequent item sets. Operating the vector from $W_{k+1}$ to $W_m$ is not necessary and the needed frequent item sets are determined directly from $W_{k-1}$ when statistics of support corresponding to all the vectors of $W_k$ do not satisfy the condition $sum_j \geq$min sup. This simplification can greatly improve the searching speed

## DATA MINING ALGORITHM FOR DATABASES WITH DYNAMIC UPDATE

Nowadays, most of the data in the databases are dynamic over time, such as the business of the supermarket, the bank's credit card records, government statistics and so on. Yet the traditional rules mined from static database commonly reflect only the current and static status of the database. The modification of the database can lead to some serious problems for data mining, such as the adjusted given minimum threshold, generation of new rules and the need of eliminating certain existing rules. Consequently, studying the dynamic data updating of frequent item sets is of great practical significance for ensuring the effectiveness and reliability of the rules dug from dynamic databases.

**Description of the problem of dynamic data updating:** Dynamic data updating means that when the dynamic changes occur the data in the database doesn't need to re-scan the entire database, but the association rules update only arising from the new situation based on the original data mining results. This association rules mining technology which reflects the dynamic changes of data, can be made with better characteristics of time and dynamic study. Among them, the more typical algorithm is the FUP algorithm made by D. W. Cheng and others, which mainly includes two situations:

- The minimum support and minimum confidence level remained unchanged, a database evolving over time, that is, when adding or deleting a data set d in the original database DB, the frequent item sets and association rules arising from the generated new database D∪ {d} orD-{d} changes
- The database remains unchanged, adjust the minimum support min sup and minimum confidence min conf, causing frequent item sets and association rules change

However, due to the existence of reading the original database content before changing many times and producing a large number of candidate item sets and so on, the FUP algorithm has not a good using efficiency. In recent years, for the shortcoming of FUP algorithm, a number of ways has been filed to solve the problem of association rules updating when tuple number and minimum support or minimum confidence change such as the IUAR (Zhu *et al*., 2005) algorithm which is through amendments to the candidate item sets for mining association rules, the IUA algorithm which focuses on the association rules mining when the minimum support and the smallest credibility changes and so on. However, these algorithms still have very low operating efficiency when the dimension of items in database is large. Based on the idea of constructing the vector subspace to find the frequent item sets proposed by this study, the answer to the frequent item sets of dynamic database could be given for both situations raised by D.W. Cheng and others according to the of vectors' products in vector subspace and vectors in additions/deletions tuple when the dynamic tuple additions and deletions in database occur and minimum support is changed.

**Dynamic data updating algorithm:** Assume that the original database is D, when adding or deleting some service tuple dynamic, the frequent item sets can be obtained by simple supplement of the algorithm in the last section. The procedure is as follows:

- Calculating the inner product of vectors in $W_k$ and vectors in augmented Boolean matrix R corresponding to the original database D according to the steps in the last section, then the support degree of each item set in database D could be obtained and at the same time the memory space occupied by R is released
- When some business tuple d is added to or deleted from the original database D, mapping d into m+1-dimensional vector $\alpha$, then operating the inner product $<\beta_{kj},\alpha>$, $\beta \in W_k$. When adding tuple d, update

the data on m+1-th place of $\beta_{kj}$ after adding 1 to the support number of $\beta_{kj}$ which satisfies the condition $<\beta_{kj}\geq k$. Here, k = 1,2,..., m, $j=1,2,\cdots,C_m^k$. When deleting tuple d, update the data on m+1-th place of $\beta_{kj}$ after subtracting 1 from the support number of $\beta_{kj}$ which satisfies the condition $<\beta_{kj},\alpha\geq k$. Here, k = 1,2,..., m, $j=1,2,\cdots,C_m^k$

- Releasing the memory space occupied by tuple d and finding the vector $\beta_{kj}$ corresponding to the maximum k in $W_k$ which satisfies the condition $sum_k(\beta_{kj})\geq$min sup, then constructing an item set of corresponding k non-zero items and this item set is the maximum frequent item set after updating the database D

From the above we can see that the algorithm does not need to repeat a static mining process when the original data in the database has dynamically changed. Instead, it utilizes the vectors for the construction of the vector subspace to store the information that has been retrieved and uses this stored information to update the modified frequent item sets simply and easily. At the same time, users can also adjust previously given minimum support threshold for frequent item sets updating and mining according to the changes of tuple in database.

**The dynamic data updating of very large database:** The updating algorithm can extend to adapt for dynamic updating of very large database D. First splits D into $D_1$, $D_2$, ..., $D_N$ according to memory or user requirement (Liu an Yang, 2007). Then steps of creating of frequent sets of D can described as follows:

- Set N and the size of $D_i$, i = 1
- Read in $D_i$ and map it into augmented Boolean matrix $R_i$
- Do statistic of support by doing inner product of each vector in $W_k$and $D_i$'s attached $R_i$. Release memory hold by $R_i$, k = 1,2,..., m
- i = i+1. If i≤N, turn to 2), else turn to 5)
- Input minimum support that required by user minsup, find vectors in $W_k$ which satisfy $sum_k(\beta_{kj})\geq$min sup and the biggest k corresponding to vector $\beta_{kj}$. The k nonzero terms are formed most frequent term set of D
- When service group d is added in or deleted from D, mapping d into m+1 dimension vector α. Do inner product of α and every vector in $W_k$ viz., $<\beta_{kj},\alpha>$. When d is an added group in D, add 1 to $\beta_{kj}$ which satisfy $<\beta_{kj},\alpha\geq k$ and then update the number locate on m+1 place of $\beta_{kj}$. When d is deleted from D, minus 1 from $\beta_{kj}$ which satisfy $<\beta_{kj},\alpha\geq k$ and then update the number locate on m+1 place of k = 1,2,...,m, $j=1,2,\cdots,C_m^k$
- Release memory hold by d, turn to 5

We can dynamic updating large database by mapping d into α and do inner product of α and the vector which hold dig information.

## EXAMPLE

Where the database shown in Fig. 1 is used as an example to illustrate the process of the above method. First, map the database D onto augmented Boolean matrix R in terms of the order $I_1,I_2,I_3,I_4,I_5$, as shown in Fig. 1b. Assume that the minimum support min sup = 2, that is, $R = (\alpha_1,\alpha_2,...,\alpha_9)^T$, then the following vector subspace can be constructed following Eq. 2 according to the number of items contained in the database:

$W_1 = \{\beta_{11} = (1,0,0,0,0,0), \beta_{12} = (0,1,0,0,0,0), \beta_{13} = (0,0,1,0,0,0),$
$\beta_{14} = (0,0,0,1,0,0), \beta_{15} = (0,0,0,0,1,0)\}$
$W_2 = \{\beta_{21} = (1,1,0,0,0,0), \beta_{22} = (1,0,1,0,0,0), \beta_{23} = (1,0,0,1,0,0),$
$\beta_{24} = (1,0,0,0,1,0), \beta_{25} = (0,1,1,0,0,0), \beta_{26} = (0,1,0,1,0,0),$
$\beta_{27} = (0,1,0,0,1,0), \beta_{28} = (0,0,1,1,0,0), \beta_{29} = (0,0,1,0,1,0),$
$\beta_{210} = (0,0,0,1,1,0)\}$
$W_3 = \{\beta_{31} = (1,1,1,0,0,0), \beta_{32} = (1,1,0,1,0,0), \beta_{33} = (1,1,0,0,1,0),$
$\beta_{34} = (1,0,1,1,0,0), \beta_{35} = (1,0,1,0,1,0), \beta_{36} = (1,0,0,1,1,0),$
$\beta_{37} = (0,1,1,1,0,0), \beta_{38} = (0,1,1,0,1,0), \beta_{39} = (0,1,0,1,1,0),$
$\beta_{310} = (0,0,1,1,1,0)\}$
$W_4 = \{\beta_{41} = (1,1,1,1,0,0), \beta_{42} = (1,1,1,0,1,0), \beta_{43} = (1,0,1,1,1,0),$
$\beta_{44} = (1,1,0,1,1,0), \beta_{45} = (0,1,1,1,1,0)\}$
$W_5 = \{\beta_{51} = (1,1,1,1,1,0)\}$

Then, calculate the inner product $<\beta_{kj},\alpha>$, $\alpha_1$ R, i = 1,2,...,9 and store the number of $\beta_{kj}$, that is $sum_k$ $\beta_{kj}$, on the sixth place of $\beta_{kj}$, here $\beta_{kj}$ satisfies the condition $<\beta_{kj},\alpha\geq k$. Then, the vectors from to $W_1$ to $W_5$ are:

$W_1 = \{\beta_{11} = (1,0,0,0,0,6), \beta_{12} = (0,1,0,0,0,7), \beta_{13} = (0,0,1,0,0,6),$
$\beta_{14} = (0,0,0,1,0,2), \beta_{15} = (0,0,0,0,1,2)\}$
$W_2 = \{\beta_{21} = (1,1,0,0,0,4), \beta_{22} = (1,0,1,0,0,4), \beta_{23} = (1,0,0,1,0,1),$
$\beta_{24} = (1,0,0,0,1,2), \beta_{25} = (0,1,1,0,0,4), \beta_{26} = (0,1,0,1,0,2),$
$\beta_{27} = (0,1,0,0,1,2), \beta_{28} = (0,0,1,1,0,0), \beta_{29} = (0,0,1,0,1,1),$
$\beta_{210} = (0,0,0,1,1,0)\}$
$W_3 = \{\beta_{31} = (1,1,1,0,0,2), \beta_{32} = (1,1,0,1,0,1), \beta_{33} = (1,1,0,0,1,2),$
$\beta_{34} = (1,0,1,1,0,0), \beta_{35} = (1,0,1,0,1,1), \beta_{36} = (1,0,0,1,1,0),$
$\beta_{37} = (0,1,1,1,0,0), \beta_{38} = (0,1,1,0,1,1), \beta_{39} = (0,1,0,1,1,0),$
$\beta_{310} = (0,0,1,1,1,0)\}$
$W_4 = \{\beta_{41} = (1,1,1,1,0,0), \beta_{42} = (1,1,1,0,1,1), \beta_{43} = (1,0,1,1,1,0),$
$\beta_{44} = (1,1,0,1,1,0), \beta_{45} = (0,1,1,1,1,0)\}$
$W_5 = \{\beta_{51} = (1,1,1,1,1,0)\}$

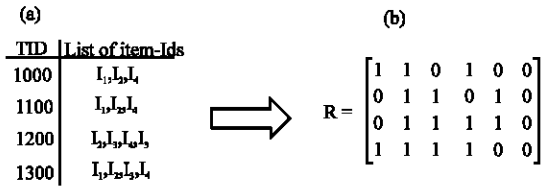


Fig. 2: (a) Added business data and (b) Corresponding augmented boolean matrix

Obviously, the maximum frequent item set whose statistics of support ≥min sup = 2 is in the vector subspace $W_3$ and the corresponding vectors are $\beta_{31}$ and $\beta_{33}$. So, the maximum frequent item set of database D is composed of the items on the non-zero places of $\beta_{31}$ and $\beta_{33}$ which are $\{I_1,I_2,I_3\}$ and $\{I_1,I_2,I_5\}$, respectively. The result is the same as the result determined by the traditional Apriori algorithm.

When the 4 business data sets shown in Fig. 2a are added to the database in Fig. 1a, the minimum support users set changes to min sup = 3, then the corresponding Boolean matrix is as shown in Fig. 2b. We can get the maximum frequent item set which satisfies the condition statistical support ≥min sup = 3 from the vector subspace $W_3$, that is:

$$W_3 = \{\beta_{31} = (1,1,1,0,0,3), \beta_{32} = (1,1,0,1,0,3), \beta_{33} = (1,1,0,0,1,2),$$
$$\beta_{34} = (1,0,1,1,0,1), \beta_{35} = (1,0,1,0,1,1), \beta_{36} = (1,0,0,1,1,0),$$
$$\beta_{37} = (0,1,1,1,0,2), \beta_{38} = (0,1,1,0,1,3), \beta_{39} = (0,1,0,1,1,1),$$
$$\beta_{310} = (0,0,1,1,1,1)\}$$

where, the corresponding vectors in $W_3$ should be $\beta_{31}$, $\beta_{32}$ and $\beta_{33}$ and the corresponding maximum frequent item sets after the change in the database should be $\{I_1,I_2,I_3\}$, $\{I_1,I_2,I_4\}$ and $\{I_1,I_2,I_5\}$, respectively. From this we can see, some of the old rules will be eliminated and some new ones will be generated when the minimum support threshold adjusted with tuples in database changed.

## CONCLUSION

Association rules mining can help decision-makers in different areas identify potential association among data in large databases, which has attracted significant attention in recent years. This study proposes an idea of searching vectors in vector subspace by using vectors inner product calculation, to overcome searching problems of frequent item sets in dynamic databases, which not only applies to the static database frequent item sets mining, but also applies to the dynamic database frequent item sets mining, updating and maintenance. This new algorithm may be useful for addressing the current information explosion and knowledge depletion problems.

## REFERENCES

Agrawal, R., T. Imielinski and A. Swami, 1993. Mining association rules between sets of items in large database. Proceedings of ACM SIGMOD International Conference on Management of Data, Jun. 1993, Washington DC, USA, pp: 207-216.

Cheung, D.W., J. Han, V.T. Ng and C.Y. Wong, 1996. Maintenance of discovered association rules in large databases: An incremental updating technique. Proceedings of International Conference on Data Engineering, Feb. 26-Mar. 1, New Orleans, Louisiana, pp: 106-114.

Cheung, D.W., S.D. Lee and B. Kao, 1997. A general incremental technique for maintaining discovered association rules. Proceedings of the 5th International Conference on Database Systems for Advanced Applications, (ICDSAA'1997), Melbourne, Australia, pp: 185-194.

Feng, Y.C. and J.L. Feng, 1998. Incremental updating algorithms for mining association rules. J. Software, 9: 301-306.

Han, J., J. PEI and Y. YIN, 2000. Mining frequent patterns without candidate generation. Proceedings of ACM SIGMOD Internal Conferenceon Management of Data, May 16-18, Dallas, Texas, pp: 1-12.

Han, J. and M. Kambr, 2001. Data Mining Concepts and Techniques. Higher Education Press, Beijing.

Lee, C.H. and C.R. LIN, 2001. Ming-Syan Chen. Sliding window filtering: An efficient algorithm for incremental mining. Proceedings of the ACM 10th International Conference on Information and Knowledge Management, Nov. 5-10, New York, USA., pp: 263-270.

Liu, Y.A. and B. Yang, 2007. Research of an improved Apriori algorithm in mining association rules. J. Comput. Appl., 27: 418-420.

Park, J.S., M.S. Chen and P.S. Yu, 1997. Using a hash-based method with transaction trimming for mining association rules. IEEE Trans. Knowl. Data Eng., 9: 813-825.

SHI, Z.Z., 2002. Knowledge Discovery. Tsinghua University Press, China.

Savasere, A., E. Omiecinski and S. Navathe, 1995. An efficient algorithm for mining association rules in large database. Proceedings of the 21th Intertional Conference on Very Large Database, (ICLD'1995), Zurich, Swizerland, pp: 432-443.

Zaki, M.J., 1999. Parallel and distributed association mining: A survey. IEEE Concurrency, Special Issue On Parallel Mechanisms for Data Mining, 7: 14-25.

Zhou, H.Y., 1999. Data mining and incremental updating on association rules. J. Software, 10: 1078-1084.

Zhu, Y.Q., Y.Q. Song and G. Chen, 2005. Research on incremental updating algorithms for mining association rules. J. Comput. Eng. Appl., 15: 186-187.