BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **13**, Special Issue

Sofia • 2013

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2013-0033

Research of the Optimization of a Data Mining Algorithm Based on an Embedded Data Mining System

Xindi Wang, Mengfei Chen*, Li Chen***

** Information Management Department, Beijing Jiaotong University, Beijing, CO 100044 China*
*** Logistics Management Department,* Beijing Jiaotong *University, Beijing, CO 100044 China*
Emails: xdwang@bjtu.edu.cn 12120649@bjtu.edu.cn lchen1@bjtu.edu.cn

Abstract: *At present most of the data mining systems are independent with respect to the database system, and data loading and conversion take much time. The running time of the algorithms in a data mining process is also long. Although some optimized algorithms have improved it in different aspects, they could not improve the efficiency to a large extent when many duplicate records are available in a database. Solving the problem of improving the efficiency of data mining in the presence of many coinciding records in a database, an Apriori optimized algorithm is proposed. Firstly, a new concept of duplication and use is suggested to remove and count the same records, in order to generate a new database of a small size. Secondly, the original database is compressed according to the users' requirements. At last, finding the frequent item sets based on binary coding, strong association rules are obtained. The structure of the data mining system based on an embedded database has also been designed in this paper. The theoretical analysis and experimental verification prove that the optimized algorithm is appropriate and the algorithm application in an embedded data mining system can further improve the mining efficiency.*

Keywords: *Embedded database, data mining, association rules, Apriori algorithm, duplication, frequent item sets.*

1. Introduction

Data mining technology has been widely used in all fields concerning data analysis and knowledge discovery. The implementation of a series of data mining algorithms is the core of a data mining system. Implementing the algorithms will treat a large amount of data in the implementation process, so they need a database to manage these data. Since the data mining systems use only some of the most basic functions of a database, they can use an embedded database for data management. The two current mainstream types of an embedded data mining system are based on embedded applications and embedded data source modes. This paper mainly studies the data source embedded mode. It is integration of a data mining platform and a database system. It is called a data mining system based on an embedded database.

At the same time, it is noticed apriori that the algorithms in data mining are of low efficiency due to their database scanning every time and producing a large number of candidate item sets. In order to improve the computational speed, Z h a n g has proposed in [1] other algorithms optimized in different aspects respectively, such as reducing the data volume, reducing the times of scanning the database, reducing the number of candidate item sets, and so on. But these algorithms have a common shortcoming that they do not consider the actual application background. They are all operated directly on the transaction records in the database, and even when the algorithm efficiency is improved, they cannot avoid lots of duplicate records participating in the operation, so that they could not improve the efficiency to a large extent, when there are many duplicate records in a database. The mining efficiency will be further improved if the duplicate records in the transaction database are reasonably removed.

Therefore, a different approach is used in this paper, considering the application background of the credit cards business, introducing the concept of record duplication. The transaction database is first scanned to remove the duplicate records and then the remaining different records are stored in another new database in the form of a two-dimensional array, a new duplication parameter is added to each record, so it is convenient to calculate the support and confidence of the association rules finally. An optimized algorithm is put forward in this paper. Records are removed according to the duplication to generate a new database, the database is compressed for the next stage data mining, binary codes conversion is used to find frequent item sets and then strong association rules are obtained. This gradually reduces the database searching time and the conversed codes length, and finally further improves the efficiency in data mining.

2. Structure of an embedded data mining system

In [2] D i n g has explained the advantage of an embedded database. It is of a small volume, open free, and applying it to a particular data mining system it does not only ensure the system’s perfect function, but also guarantees system’s good portability, so that the data will be better managed.

Data mining system must import source data, pre-process data and convert data. It must also mine specific data, show data mining results (visualize data mining) and assess the model. In [3] N a v e e n has investigated the implementation of a lean manufacturing system. The structure framework of the embedded data mining system includes five parts: a visible GUI, a data mining module, a storage management module, a data conversion module and a file configuration module. The idea of embedding a database into a data mining system to constitute the storage management module is innovatively proposed in this paper. The specific frame diagram is shown in Fig. 1.

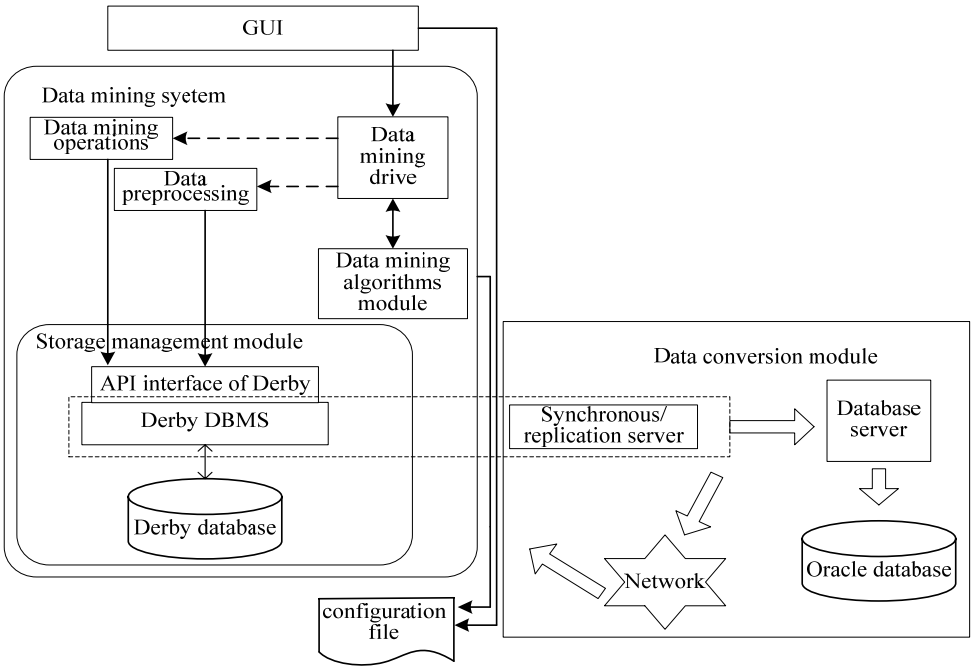

Fig. 1. Frame diagram of a data mining system based on an embedded database

In the improved system, the users can control the whole data mining platform through a GUI interface, the data needed for the algorithm module is extracted from a memory management module in the embedded database, and the data in Derby database is transmitted by a synchronizing/copy server from the external database server. The memory management module provides database interface plug-in, the plug-ins' main function being to establish correlation between the data mining algorithm and Derby database, to store the mining algorithms' data. The configuration file provides and maintains mainly basic information of data mining, and in [4] L i u has investigated the application of an embedded database in a data mining system. The data conversion module is a part of the application database, it can download a subset to the embedded Derby database. The new data mining system will save a lot of time in data conversion and further improve the efficiency of data mining.

3. Apriori algorithm optimization

3.1. Apriori algorithm

3.1.1. The concept of an apriori algorithm

Apriori algorithm is one of the most influential classic algorithms of mining the Boolean association rules. The algorithm's main job is to find frequent item sets. It takes advantage of the fact that a subset of frequent item sets must be frequent item sets.

Set $I = \{i_1, i_2, \dots, i_m\}$ as a set of items, D as the transaction database, transaction T as the set of items, and $T \subseteq I$. The k-item set contains k 1-items. The frequency of the item sets is the number of transaction records that contain this item, and it is the support of the item set for short. The formula $X \Rightarrow Y$ is called an Association rule, $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. Set support $(X \Rightarrow Y)$ as the support of rule $X \Rightarrow Y$, given in

(1) $$\text{Support}\,(X \Rightarrow Y) = \\ = \frac{|\{T: X \cup Y \subseteq T,\ T \in D\}|}{|D|} \times 100\% = S\% .$$

The confidence of rule $X \Rightarrow Y$ is confident ($X \Rightarrow Y$), as expressed in

(2) $$\text{confident}(X \Rightarrow Y) = \\ = \frac{|\{T: X \cup Y \Rightarrow T,\ T \in D\}|}{|\{T: X \Rightarrow T,\ T \in D\}|} \times 100\% = C\% .$$

The value of min_sup is responsible for a set of items meeting the minimum set in a statistical sense, and the value of min_conf is responsible for the minimum reliability of the association rules. The rules that satisfy min_sup and min_conf at the same time are called strong association rules. Therefore, two threshold values are set in the process of association rules data mining. The task of the association rules data mining is to find out strong association rules with min_sup and min_conf in the transaction database D.

In [5] L u has explained association rule data mining, the theoretical foundations of which are as follows:

Given

(3) $$\text{confident}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X) \times D},$$

so that if the first nonempty subset s of the frequent item set 1 satisfies

(4) $$\frac{\text{Support}(1) \times D}{\text{Support}(s) \times D} \geq \min_{\text{conf}},$$

then obtain a strong associate rule: $s \Rightarrow 1 - s$.

3.1.2. The principle and shortcomings of the Apriori algorithm

The main content of the Apriori algorithm is as follows:

Step 1. Use the iteration method of searching one by one. Calculate the support of all frequent 1-item sets by scanning the database, find the set L_1 of all frequent 1-items and get frequent 1-item set C_1.

Step 2. Connection. Realize an "interact" operation to get the candidate frequent 2-item sets C_2 with two frequent 1-item sets, in which only one item is different.

Step 3. Pruning. Get frequent sets L_2 after pruning C_2.

Step 4. Scanning the database, calculate the support of each item set, delete the item sets which do not meet the support, then repeat Steps 2-4 through an iterative loop. Finally, find the maximum frequent item sets, and the algorithm stops. In [6] L u o has investigated the improved Apriori algorithm.

As the data increases, the Apriori algorithm shows two shortcomings. It needs to scan the database many times in order to determine whether each element in the candidate item sets can be an element of the frequent items. It produces a large number of candidate item sets, which not only occupies a lot of the main memory space, but also costs a long running time of the algorithms.

3.2. The optimized algorithm

3.2.1. Calculating the duplication

The times of record occurrence in a database is called duplication of the record, the value scope of duplication are all natural numbers. The use of the records occurrence repeats in a database is suitable to measure the records duplication, the more frequently the record appears, the higher the duplication of the record is. There are many duplicate records in a large-scale database when the record contains fewer attributions of the item. Removing the duplicate records and keeping only the duplication parameter can save lots of operations for the next stage data mining.

Data mining uses transaction records in a huge database, only the un-duplicate records can help people find out the item sets that constitute the association rules, the duplicate records only determine the confidence level of the association rules. When the amount of the duplicate records is very large in a database, the reasonable removing of the duplicate records is of great significance to improve the efficiency of data mining. Therefore, we can quantify the record duplication attributes according to certain rules.

Firstly, scan the database and remove the duplicate records. Secondly, save the remaining different records in another new database in the form of a two-dimensional array. For example, a record is t_1= {TID, Gender, Marriage, Check account, Credit history}, then it will be stored in the new database in the form of t[1][0]= TID, t[1][1]= Gender, t[1][2]= Marriage, t[1][3]= Check account, t[1][4]= Credit history. At last, add a duplication parameter to every record in the new database. Let Duplicate (Ti) calculates the duplication of record Ti. The duplication computation algorithm is as follows:

Algorithm: Apriori_duplicate

I n p u t: Traction database T

O u t p u t: Database *P* with duplicate records removed

A l g o r i t h m d e s c r i p t i o n:

Procedure Apriori_duplicate (T: traction-all-records; P: traction-removeDup-records)

(1){int k=traction-all-records.length, n= record-item.length;

//Set k as the number of records in the transaction database D, and set n as the number of the attribute item of every record.

(2) $t[i][j]$ is j-th item of record $t[i]$;

(3) $t[i][0]=t$ID;

(4) for (int i=0;i<k;i++) // Loop all records

(5) for (int j=i+1;j<k;j++) //Loop all attribute items of every record

(6) {if(($t[i][1]=t[j][1]$)&&($t[i][2]=t[j][2]$)...&&($t[i][n]=t[j][n]$))

//The other attribute values in addition to TID of record i and record j are equal

(7) {$t[i]$.count++; // The duplication of record i plus 1}

(8) else {$t[i]$.count=1; //If the two records are not duplicate, the duplication of $t[i]$=1}

(9) add $t[i]$ to P }

(10) return P }

(11) }

3.2.2. Compressing the new database

Scanning the new database to compress it according to the condition that records without k-item sets cannot contain any k+1 item sets, then delete the records, in which the number of items is less than the default minimum number of frequent item sets k, so that the unqualified item sets shall not be entitled to participate in coding. Recombine the records containing 1-item sets, in order to save the encoding and calculating time. The new database compression algorithm is as follows:

Algorithm: Compress_database (P)

I n p u t: Remove duplicate records database P

O u t p u t: New Database P

A l g o r i t h m d e s c r i p t i o n:

Procedure Apriori_compress_database (P: traction-removeDup-records)

(1) { int ε=min_sup, c=min_conf, k=min_item;

(2) for(i=0; i<P.length; i++)

(3) {n= $p[i]$.length;// Set $p[i]$ as the number of items in record i in the database P

(4) if (n<k)

(5) Remove $P[i]$; }

(6) return P; }

3.2.3. Find out frequent item sets based on a binary code

After calculating the duplication of the records and compressing the new database, we must encode the records, calculate the codes, find out all frequent item sets and finally get the strong associate rules.

(1) The principle of coding

Item sets coding plays a very important role in the algorithm, the coding principle is: first of all, decide the length of codes according to the number of 1-items existing in the database, then order the 1-item sets, each item corresponds to a position in the item codes, if a 1-item exists in the item sets, then the corresponding position is set as “1”, otherwise set as “0”. For example, if a database

has 6 kinds of 1-item sets (I_1, I_2, I_3, I_4, I_5, I_6), the record is I1I3I4I6, according to the encoding principle, the transaction code is (101101).

(2) Coding operation and calculation of the item sets support

"Interact" operation is similar to the "JOIN" operation of common sets, it can find out frequent item sets rapidly without any candidate item sets in the process. In [7] Y e has investigated a kind of a searching frequent item sets algorithm based on the binary code, each record corresponding to an item set. We will get the records' public item sets after operation, and then translate the public item sets. In order to avoid many item set codes being repeatedly executed, the algorithm will begin with the 1-item sets of minimum support. Delete the code from every item set after operation.

The number, indicating how many items are contained in a database is called records' support. The calculation principle of the support is: get the public item set after operation of few item sets, let the number of the item sets multiplied by the duplication, and set the result as support of this public item set. Thus finding out the support of several different item sets by an "interact" operation is done quickly and precisely.

3.2.4. Finding out strong association rules

The final step is to get strong association rules after finding out all frequent item sets according to (3)-(4). The steps of generating association rules are two.

Step 1. Produce all nonempty subsets for each frequent item sets.

Step 2. For each nonempty subset s, if $\frac{\text{Support\_count}(1)}{\text{Support\_count}(s)} \geq \min\_\text{conf}$, then output $s \Rightarrow 1 - s$.

3.3. Implementation steps of the optimized algorithm

3.3.1. Mining process of the algorithm

The related knowledge of the optimized algorithm has expatiated, the implementation steps of the optimized algorithm are seven.

Step 1. Set k as the minimum number of frequent item sets and ε as the support threshold of frequent item sets.

Step 2. Compress the transaction databases and delete all items sets, in which the number of 1-item is less than k in the database. Set i as the number of 1-item in the current database, i being greater than k.

Step 3. Encode each item set in the database after pruning, compute the support of each 1-item sets and rank the supports from small to big.

Step 4. Delete the 1-item set whose support is less than ε. Set n as the number of deleted 1-item, then $i = i - n$. Record all codes of the item containing the remaining 1 -item set which has the smallest sets support.

Step 5. Do "interact" operation with all item sets codes produced by Step 4, and get the public item set. Set the number of the item sets which have the same public item set as support of the public item set. Set the item set as a frequent item set when the threshold is greater than the corresponding threshold and record the

support, otherwise delete the item set. Delete the 1-item after operation directly from all item sets.

Step 6. Check if the length of the left codes *i* is equal to *k*, if *i* = *k*, record all item codes of 1-item sets directly, and turn to Step 7. If *i* > *k*, compress the transaction database and return to Step 4.

Step 7. Output all frequent item sets, and the algorithm is ended.

The above algorithm steps do not include the use of the candidate item sets and a threshold, the threshold setting is only associated with frequent item sets, the users can set it according to the specific circumstances. The optimized algorithm efficiency has been greatly improved. It has saved the computation and quantity of the data input and output of the association rule data mining compared to the Apriori algorithm.

The optimized algorithm running flow is shown in Fig. 2.

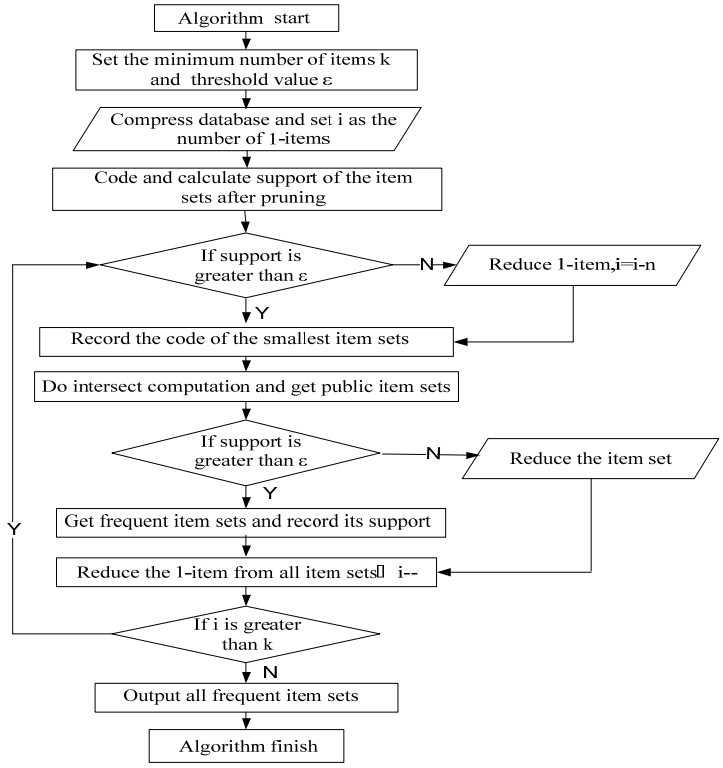

Fig. 2. Running flow of the optimized algorithm

3.3.2. Algorithm description

Algorithm. Find_frequent_itemsets(*P*);//find out all frequent item sets in database *P*

I n p u t: Remove duplicate records database *P*

O u t p u t: Strong association rules

Algorithm description:
Procedure Apriori_encode_records()
(1) {for(j=0; j<P.length; j++)
(2) for(p=1; p<P[j].length; p++)
(3) {if P[j][p] not null
(4) P[j][p]=1;
(5) else P[j][p]=0; }
(6) return P; }
Procedure Apriori_ interact _operation(P:public_itemsets)
(1) {for(q=1; q< P.length; q++)
(2) if(R[q][1]=P)
(3) Return P; }

3.4. Optimized algorithm analysis

3.4.1. Qualitative analysis of time complexity

Compared with Apriori algorithm, the optimized algorithm scans the database just twice. Scanning the database to remove the duplicate records in the transaction database and getting a new database R at the first time. The more duplicate records in the transaction database, the greater time is saved when encoding each item set in database R, since the duplicate records have been removed.

The second scanning database is to encode each item set in database R. The size of database R is smaller than T because many duplicate records have been removed, so the time of finding frequent item sets has been also reduced. Therefore, reducing the running time of the algorithm is feasible.

3.4.2. Qualitative analysis of space complexity

The traditional Apriori algorithm mainly stores the trading records data, these data take up a lot of memory space. While the optimized algorithm stores the records only after removing duplicate records, and these records are translated into codes before storing in memory space, the codes are composed of a Boolean value, so the optimized algorithm can save a lot of storage space. Therefore, reducing the storage space is feasible for the optimized algorithm.

4. Example analysis

4.1. Data mining in the management of a credit card fraud

At present many commercial banks use data mining techniques for customer credit rating, fraud detection of customer's trading actions in order to reduce the credit risk. We use the personal trading information data of bad cardholders collected by a domestic commercial bank to analyze the specific features of a cardholder, then explain the association rules in simple applications of the credit card.

4.2. The modelling process

4.2.1. Data cleaning and selection

Data cleaning. Data cleaning is eliminating noise and inconsistent data. Given the original data involves privacy and only certain attribute values that can affect the results of data mining, so it is necessary to delete the noise and inconsistent attributes of the records, such as name, telephone, address and so on. We only use the resulting data after deleting irrelevant variables or discrete appropriate data. In [8] S u n has done an experiment of association rules analysis in a credit card fraud. This experiment adopts a single data source, so we just need to put the data in a table of Derby database.

Data selection. Select 20 000 sample records from the transaction database of the bank credit card center, each sample record has 7 attributes, corresponding to the I_1-I_7 1-item, each specific data attributes can be described by the corresponding item sets as shown in Table 1.

Table 1. Attributes and corresponding item sets of a credit card

No	Gender I_1	Marriage I_2	Check account I_3	Bank deposit I_4	Bank loan I_5	Bondsman I_6	Credit history I_7
1	Male	Married	Small	Small	Small	No	Timely
2	Female	Unmarried	Large	Large	Large	Have	Delayed

4.2.2. Association rules data mining

Data mining system is running in Lenovo PC Y330 series, Intel core duo, 2.13 GHz CPU, 2 GB memory environment. Data mining tool is SAS Enterprise Miner.

With the personal trading information data of bad cardholders, we use the traditional Apriori algorithm and the optimized Apriori algorithm to mine frequent item sets in the transaction database. There are 6 item sets, in order not to lose significant association rules and get all frequent item sets, we initialize the number of the frequent item sets to $k = 3$, the support threshold of the frequent item sets $\varepsilon = 2$.

Firstly, delete the duplicate records and calculate the duplication, then every record has a duplication parameter, for example, if there are 8 records I1I2I3I5I7 in a database, the record will be described as I1I2I3I5I7 (8) in the new database. Secondly, compress the database and delete all item sets, in which the number of 1-item is less than k in the database. At last, encode each set in the database after pruning, compute the support of each 1-item and order the supports from small to big, the process being shown in Fig. 3.

In Fig. 3, after database compression, the 6 item sets becomes 4, so you can see the importance of the database compression. Do “interact” operation with all item codes, the running process of the optimized algorithm is shown in Fig. 4. In [9] Z h o u has proposed a new algorithm with no candidate sets of mining frequent item sets.

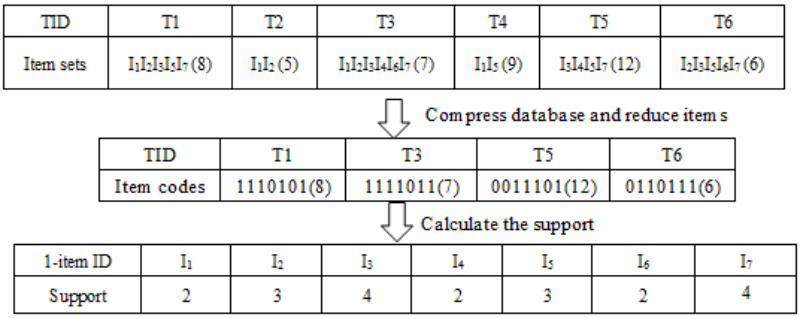

Fig. 3. Diagram of compressing a database and calculating codes and support

Fig. 4 shows the process of operation of finding out frequent item sets, it describes the steps of the optimized algorithm explicitly through a series of charts. Thus making the optimized Apriori algorithm more specific, the people understand the various steps more clearly.

Finally, find out all strong associate rules according to formulas (3)-(4).

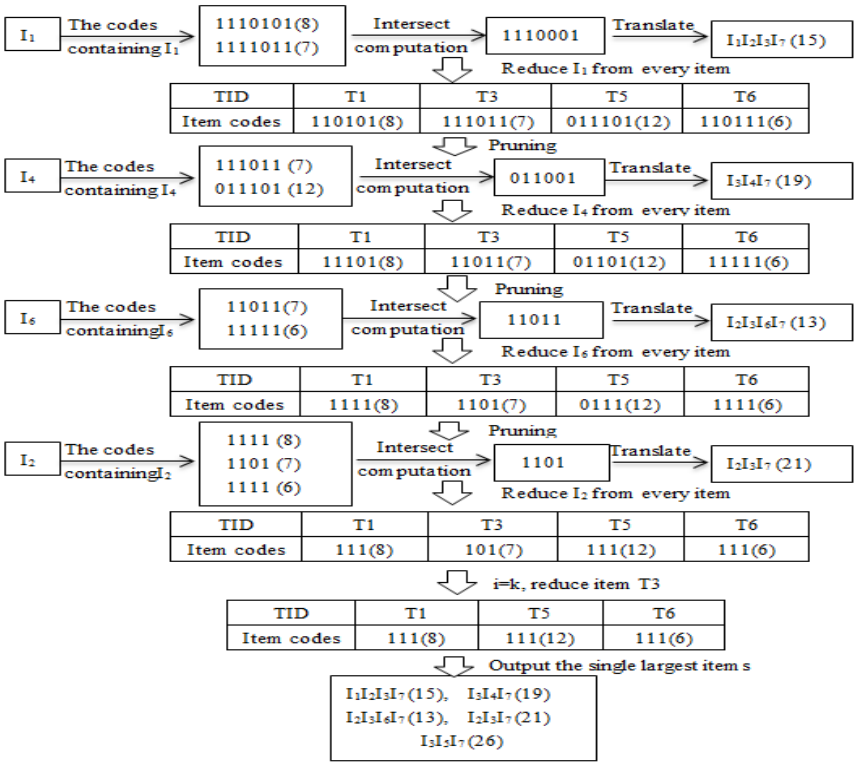

Fig. 4. Running process of the optimized Apriori algorithm

4.3. Model evaluation and presentation

Under the condition of the same min_sup threshold, we use the traditional Apriori algorithm and the optimized Apriori algorithm in an embedded and un-embedded data mining system for association rules data mining respectively, and then

compare the running time of both algorithms, the efficiency diagram of the two algorithms in un-embedded system being shown in Fig. 5:

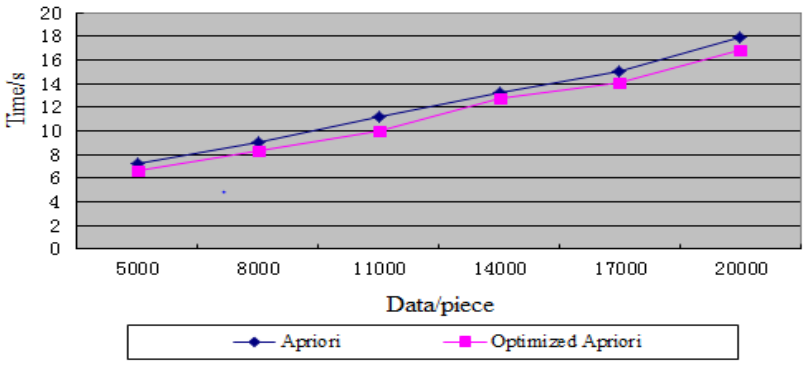

Fig. 5. The efficiency diagram of the two algorithms in un-embedded system

The efficiency diagram of the two algorithms in an embedded system is shown in Fig. 6.

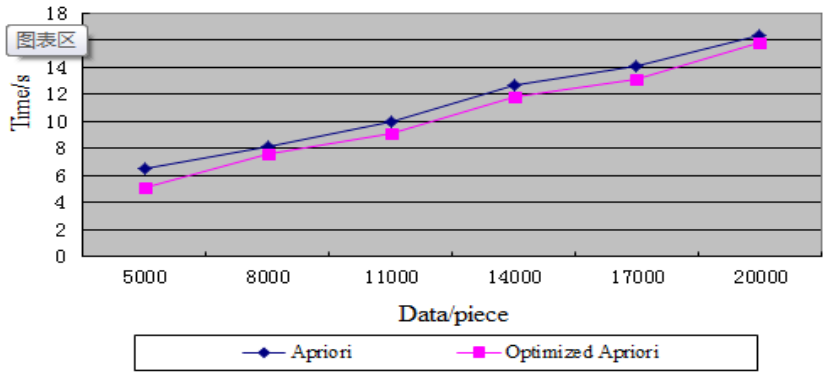

Fig. 6. The efficiency diagram of the two algorithms in an embedded system

From the above comparison we can see that with the optimized algorithm in association rules data mining, for two different data mining systems, the mining efficiency has been significantly improved. As seen, the efficiency has increased one to two times. From the experiment we can also see that increasing the amount of the business data, the optimization efficiency of the algorithm can become larger, and the embedded data mining system can further save time. Therefore, the optimized algorithm is very efficient.

5. Conclusion

Although many other optimized algorithms have been proposed with respect to different aspects, these algorithms have a common shortcoming that they do not consider the actual application background and cannot improve the efficiency to a large extent when there are many duplicate records in the database. Therefore, a

new optimized algorithm has been suggested with a new concept of duplication proposed. The optimized algorithm generates a new database of a small size and compresses the database according to users' requirements. At last, it finds frequent item sets based on binary coding, and then gets strong association rules.

The view of embedding a database into a data mining system has also been proposed and the embedded data mining system has been introduced. The specific conditions of management of a credit card fraud confirm the efficiency of the optimized algorithm by using two kinds of a data mining system in association rules data mining. The theoretical analysis and experimental results show that the algorithm has obviously improved the efficiency and that the efficiency can be better improved in an embedded data mining system.

Acknowledgment: This work has been supported by basic scientific research funds No B11JB00500.

References

1. Z h a n g, G. L., J. S. L e i, X. H. W u. An Improved Apriori Algorithm for Mining Association Rules. – Computer Technology and Development, Vol. **20**, 2010, No 6, 84-89.
2. D i n g, R. Embedded Database Technology. Xi'an, Northwest Industry University Press, 2001, 65-91.
3. N a v e e n K u m a r, S a n j a y K u m a r, A b i d H a l e e m, P a r d e e p G a h l o t. Implementing Lean Manufacturing System: ISM Approach. – Journal of Industrial Engineering and Management, Vol. **6**, 2013, No 4, 996-1012.
4. L i u, Y., C. Y. Y u, X. J. Z h a n g. The Application of Embedded Database in Data Mining System. – Journal of Liaoning University of Petroleum and Chemical, Vol. **30**, 2010, No 4, 63-65.
5. L u, Q. C., P. Z o u. Research and Application Development of Data Mining. – Journal of Kunming University of Science and Technology, Vol. **27**, 2002, No 5, 62-66.
6. L u o, X. L. Research of Improved Apriori Algorithm – Journal of Yangtze University (Natural Science Edition), Vol. **8**, 2011, No 3, 75-77.
7. Y e, X. B. A Kind of Searching Frequent Item Sets Algorithm Based on Binary Code. – Journal of Chuxiong Normal University, Vol. **24**, 2009, No 3, 13-19.
8. S u n, D. L. Association Rules Analysis and its Application in Credit Card Fraud. – China's Credit Card, Vol. **11**, 2007, 36-37.
9. Z h o u, H. Y., Y. Z h a n g, P. L i n. A New Algorithm With no Candidate Sets of Mining Frequent Item Sets. – Computer Engineering and Applications, Vol. **40**, 2004, No 15,182-185.