一种改进的 Apriori 关联规则挖掘算法

张广路[1],雷景生[2],吴兴惠[1]
(1.海南师范大学 数学与统计学院,海南 海口 571158;
2.南京邮电大学 信息与技术学院,江苏 南京 211815)

摘 要:关联规则挖掘是数据挖掘中的一个重要研究内容。为了高效、快速地从事务数据库中挖掘出频繁项集,针对数据挖掘的经典关联规则 Apriori 算法的瓶颈问题提出了改进的方法。算法将事物数据库映射到布尔型数组中,然后所有的操作都针对数组元素值展开。这样大大减少了数据库的扫描次数。算法利用数组的随机访问特性及布尔型数据的简单"与"操作,直接产生频繁项集,而不产生大量的候选项集。经理论分析和实验结果显示该算法在效率上明显优于 Apriori 算法。

关键词:数据挖掘;关联规则;Apriori 算法;频繁项集
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2010)06-0084-05

An Improved Apriori Algorithm for Mining Association Rules

ZHANG Guang-lu[1], LEI Jing-sheng[2], WU Xing-hui[1]
(1. School of Mathematics and Statistics, Hainan Normal University, Haikou 571158, China;
2. School of Information Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 211815, China)

Abstract: Association rule mining is an important part of research content in data mining. In order to efficiently and quickly mine all frequent itemset from the transaction database, an improved algorithm of mining association rules is presented for the bottleneck problem of the classic Apriori algorithm. The transaction database is mapped to Bool array, then all the operations are carried out based on array elements value, thereby reducing the database scanning frequency. Then use bitwise "AND" operation and random access characteristics of array, a direct consequence of frequent itemsets, rather than have a large number of candidate sets, thereby improving the efficiency of the algorithm.

Key words: data mining; association rules; Apriori algorithm; frequent itemset

0 Introduction

Mining Association Rules is a research field by proposed earlier, as an important part of data mining, that has experienced the development of a longer period of time. Association rule mining aims to find rules in the transactions database D with user given minimum support and minimum confidence, that can be seen as two processes: First, finding all frequent itemsets, i.e., each of these itemsets will be occurred at least as frequently as a predetermined minimum support count, min_sup. Second, generating interesting association rules from the frequent itemsets, i.e., these rules must satisfy minimum support and minimum confidence. Because the second step is much less costly than the first step, the overall performance of mining association rules is determined by the first step, so mining association rules is usually converted to mining frequent itemsets. Nowadays, there have many works[1~5] focus on frequent itemsets mining. Frequent pattern mining often generates a very large number of patterns and rules, which reduces not only the efficiency but also the effectiveness of mining[6,7]. To efficiently solve the problem, traditional algorithms like Apriori[8] and its variants[9~11] focus on reducing the number of database scans as well as cutting down the enumeration space of candidate itemsets (CI's). By the downward closure

收稿日期:2009-10-26;**修回日期**:2010-01-22
基金项目:海南省自然科学基金资助项目(808155,109002);海南师范大学青年教师科研启动资助项目(QN0923);海南师范大学教改项目(HSJG0924)
作者简介:张广路(1978-),女,山东泰安人,讲师,硕士,研究方向为数据挖掘技术、数据库系统。

property of FI's – allsubsets of a frequent itemset must also be frequent – they can iteratively enumerate, prune, and test 1 – extension supersets of existing FI's and end up with much reduced number of CI's. The major problem with these algorithms is that their repeated search of prof use itemsets against the huge amount of transactions turns out to be very time – consuming and wasteful. For dense databases with enormous long FI's, the breadth – first search of the itemset lattice and subset testing to prune CI's by existing FI's are both space and time consuming. Another innovative algorithm targeting dense databases is FP – growth[12]. It compacts the repetitive transactions into some concise FP – tree. Itemsets are organized in that frequency – ordered prefix tree so that they share common prefix part as much as possible. This approach can cut down the database sizes and reduce repeated computation for dense databases. In this paper, an effective mining algorithm is proposed based on two – dimensional Bool array. Algorithm uses bitwise "and" operator and a random access characteristics of array, mine frequent k – itemsets L_k, but does not have a generation of candidate itemsets, which more efficient generate frequent k – itemsets.

1 Problem Description

Let D, be a set of transaction database; let $I=\{I_0, I_1, I_2, \cdots, I_m\}$ be a set of items, I_i is the first i item, Let $T=\{T_0, T_1, T_2, \cdots, T_n\}$ be a set of transactions in D, T_i is the first ith transcation, which by I the n – components that can be expressed as $T_i=\{tid, x_1, x_2, x_3, \cdots, x_n\}$, of which $x_i \in I(1 \leqslant i \leqslant n)$, n is the size of transactions, *tid* is a transaction identifier. A set of items is referred to be as itemset. An itemset that contains k items is a k – itemset. The occurrence frequency of an itemset is the number of transactionts that contain the itemset, that is also known, simply, as the frequency, support count of the itemset, if the support count of an itemset I satisfies a prespecified minimum support threshold, then I is a frequency itemset, the set of frequency k – itemset is commonly denoted as L_k. e. g. $X=\{I_1, I_2, I_3, \cdots, I_k\}$, of which sup($X$) expressed the transaction number of X, if sup(X) $\geqslant$ min_sup, then itemset X is frequency k – item set which be known L_k. In this paper, the task is to design an algorithm in a given user – defined minimum support and minimum conditions, only by scanning of database D once, to map the database to the two – dimensional Bool array, then mine all frequent itemsets from the array.

2 Description of Improved Apriori Algorithm

Algorithm uses the vertical express format of databases, thereby reducing the time and space cost. Database, there are two express format: the horizontal data format and vertical data format (P. Shenoy[5] and others have verified the vertical data format that is better than the horizontal data format). Table 1 and table 2 give two express formats. Based on the vertical data format algorithm is proposed in this paper: Each item is expressed with a binary array elements. Let Tdarray[m][n] is the Bool two – dimensional array that express the entire database, that is:

$$\text{TDarray}[u][v]=\begin{cases}1 & I_u \in T_v \\ 0 & I_u \notin T_v\end{cases}$$

In which $u=0,1,2,\cdots,m, v=0,1,2,\cdots,n$.

The Application of Bool array format aims to save time of calculating the support count, and reduces the number of scaning array elements. Because the Bool array have a random access features and can perform a simply bitwise "and" operator. Therefore get better time and space efficiency.

Table 1　Horizontal data format

TID	Items
T_0	I_0, I_1, I_2, I_4, I_5
T_1	I_1, I_3, I_4
T_2	I_0, I_2
T_3	I_1, I_2, I_4
T_4	I_0, I_1, I_2
T_5	I_1
T_6	I_3, I_4, I_5
T_7	I_1, I_2, I_3, I_4

Table 2　Vertical data format

Item	Tidset
I_0	T_0, T_2, T_4
I_1	$T_0, T_1, T_3, T_4, T_5, T_7$
I_2	T_0, T_2, T_3, T_4, T_7
I_3	T_1, T_6, T_7
I_4	$T_0, T_1, T_2, T_3, T_6, T_7$
I_5	T_0, T_7

2.1 Description of Algorithms – Relevant Fundamental Property

In order to better explain algorithms, now relates to

algorithms – relevant fundamental properties and characteristics described as follows:

Property 1: All nonempty subsets of a frequent itemset must also be frequent, and all superesets of a nonfrequentcy itemset must also be not frequent.

Property 2: if the item l_i and l_j of the frequent $(k-1-)$itemsets are not satisfied with the joining conditional, then the item l_i and all items after item l_j are not satisfied with the joining conditional.

This property is a kind of optimization of the join conditions for Apriori algorithm, because all items of itemset about Apriori algorithm are sorted in lexicographic order. Now let $L_{k-1}=\{l_1,l_2,l_3,\cdots,l_{k-1}\}$, each $l_i\in L_{k-1}$ and each $l_j\in L_{k-1}(1\leqslant i<j\leqslant k-1)$, if there is not have the equation of $l_i[1]=l_j[1]\wedge l_i[2]=l_j[2]\wedge\cdots\wedge l_i[k-2]=l_j[k-2]\wedge l_i[k-1]<l_j[k-1]$, then the equation of $l_i[1]=l_{j+1}[1]\wedge l_i[2]=l_{j+1}[2]\wedge\cdots\wedge l_i[k-2]=l_{j+1}[k-2]\wedge l_[k-1]<l_{j+1}[k-1]$ is not satisfied. Therefore, it is not need to determine the possibility of joining of the item l_i and all items after the item of l_j. So make use of itemsets characteristics between the orderly operation to reduce the number of connections in order to improve algorithm efficiency.

Property 3: If let L_{k-1} is a set of frequent $k-1$ itemset where each itemset c_i is a set of item and $\forall I_j\in c_i$ such that $|L_{k-1}(I_j)|<k-1$ (of which $|L_{k-1}(I_j)|$ express the number of the occurrence in frequent itemsets L_{k-1}), then all candidate k itemsets generated by All elements of non – connect to c in L_{k-1} are not frequent itemsets.

Use the property to reduce the size of frequent $(k-1)$ – itemsets so as to reduce the number of comparison of generating frequent k – itemsets. So before generating L_k, L_{k-1} can be performed the pruning treatment. Because if the items I_j in the collection appear in L_{k-1} is less than the number of $K-1$, then after the join, $L_{k-1}\infty L_{k-1}$, support counts of all itemsets which include item I_j are less then k, so itemsets which include item I_j will be removed from itemsets L_{k-1}, Thereby improving the efficiency of algorithm.

2.2 Algorithm Implementation Steps and Algorithm Design

The improved Apriori algorithm of mining frequent itemsets will be divided into three performing steps:

① The initialization step.

The database is mapped to the two – dimensional Bool array by scanning the database once. At the same time, define a one – dimensional array which will stat the number of transaction contains one item in database. Then in accordance with the result of the stating to generate the frequent 1 – itemset L_1, and let candidate frequent1 – itemset $C_1=L_1$.

② The join step.

The join, $C_{k-1}\infty C_{k-1}$ (k from 2 to start) is performed, generating the set of frequent k – itemsets without the Candidate k – itemset, where members of C_{k-1} are joinable if their first$(k-2)$items are in common and the property 2 is satisfied by the first $k-1$ items. At the same time, calculate the support count of itemset c generated. For examble let $c=\{I_0,I_1,I_2,\cdots,I_k\}$, support$(c)$ = sum(TDarray[0] ∧ TDarray[1] ∧ TDarray[2] ∧ ⋯ ∧ TDarray[k]) = TDarray[0][0] ∧ TDarray[1][0] ∧ ⋯ ∧ TDarray [k][0] + TDarray[0][1] ∧ TDarray[1][1] ∧ ⋯ ∧ TDarray[k][1] + ⋯ + TDarray[0][n] ∧ TDarray[1][n] ∧ ⋯ ∧ TDarray[k][n]. If support$(c)\geqslant$ min_sup, then $L_k=L_k\cup c$ is performed.

③ The prune step.

Based on the property 3, counting the number that $I_u(I_u\in I)$ occurrence in L_k and recording the result in the array of $A_k[u]$ (here $A_k[u]$ refers to the number that $I_u(I_u\in I)$ occurrence in L_k). If $A_k[u]<k$, then $C_k=L_k-c$ is performed, of which $c\in L_k$ and $I_u\in c$. Repeat this process until the candidate frequently generate k – itemsets denoted C_k. The purpose of performing prune step is to reduce the size of k – itemsets and to improve the efficiency of algorithms. And then $k++$ be performed and back to steps 2 until $C_k<2$.

Algorithm design: Improved Apriori Algorithm

Input:

* D: a database of transactions;

* min_sup: the minimum support count threshold.

Output: L, frequent item_ sets in D.

Method:

(1) for(i=0;i<=m;i++) // initialization

for(j=0;j<=n;j++)

(2)if($I_i\in T_j$) { TDarray[i][j]=1;

(3) B[i]++; //count the transaction number which include item I_i|

(4) else TDarray[i][j]=0;

(5) for(j=0;j<m;j++) //generate frequency 1

- itemset L_1

(6) If(B[j] >= min_sup) $L_1 = L_1 \cup I_j$;

(7) else delete TDarray[i];//remove the row value of unfruitful item, packed array

(8)free(B); // saving space

(9)$C_1 = L_1$;

(10) for(k=2;|C_k|>2;k++){

(11) L_k = find_frequent_k_itemsets(C_{k-1}, TDarray[m][n], min_sup);// generate frequent k - itemset

(12) for each itemset $c \in L_k$

(13) for each item $I_j \in c$

(14) $A_k[j] = I_j$.count++; // Statistics the times that I_j($I_j \in I$) appear in L_k and records the result in the array of $A_k[u]$

(15) C_k = TDarray_gen (L_k, A_k[m]) // prune setp: call porcedure TDarray_gen

(16) If(|C_k|<2)

(17) Return L_{k-1};

(18) }

procedure find_frequent_k_itemsets (C_{k-1}, TDarray[m][n], min_sup)

(1) for each itemset $l_i \in C_{k-1}$

(2) for each itemset $l_j \in C_{k-1}${

(3) if($l_i[1] = l_j[1] \wedge l_i[2] = l_j[2] \wedge l_i[3] = l_j[3] \cdots l_i[k-2] = l_j[k-2] \wedge l_i[k-1] < l_j[k-1]$) //based on property 2

(4) { $c = l_i \infty l_j$/ /join setp:

(5) for(u=0;u<=n;u++)

(6) if($I_u \in c$) {C[n] = TDarray[u]; //copy the value of the first item to array C[n]

(7) break;}

(8) for(j=u+1;j<=n;j++)

(9) if($I_j \in c$)

(10) C[n] = C[n] ∧ Vmatrix[j];

(11) d = support(C[n]);

(12) if(d>min_sup)

(13) $L_k = c \cup L_k$; / /generate frequent k - itemset

(14) }

(15) else break; // stop Comparison and join

(16) } }

procedure TDarray_gen (C_k, L_k, A_k[m])

(1) for(j=0;j<A_k.legth;j++)

(2) { If($A_k[j] < k$)

(3) for each $c \in L_k$ and $I_j \in c$

(4) $C_k = L_k - c$;

(5) ruturn C_k;}

3 Algorithm Analysis

3.1 Examples Analysis

In order to better explain the algorithm implementation process, now takes 8 records from the database to show. Transaction database as shown in table 2. The preforming process is as follows:

①The transactions in D are scaned in order to initialization array TDarray$^{[4,10]}$, that corresponds to the value of the array elements such as shown in table 3, at the same time counts the number of occurrences in database of each item and records the result in the array of $B^{[4]}$ in order to generates the frequent 1 - itemset.

Table 3 Array element value list

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
I_0	1	0	1	0	1	0	0	0
I_1	1	1	0	1	1	1	0	1
I_2	1	0	1	1	1	0	0	1
I_3	0	1	0	0	0	0	1	1
I_4	1	1	0	1	0	0	1	1
I_5	1	0	0	0	0	0	0	1

②Suppose that the minimum support count threshold required 2, that is , min_sup = 2 (Here the corresponding relative support is 2/8 = 25%). The set of frequent 1 - itemsets is denoted L_1, that can be determined by scaning array $B^{[4]}$. $L_1 = \{I_0, I_1, I_2, I_3, I_4, I_5\}$. Let $C_1 = L_1$. free $B^{[4]}$.

③ The join, $C_1 \infty C_1$ is performed in order to generate the set offrequent 2 - itemsets which is denoted L_2. $L_2 = \{\{I_0, I_1\}, \{I_0, I_2\}, \{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_4\}, \{I_2, I_4\}, \{I_3, I_4\}, \{I_4, I_5\}\}$

④ Based on property 3, prunes L_2, then generetes the set of candidate frequent 2 - itemsets that is denoted C_2, because $A_2^{[5]} < 2$, then the itemset $\{I_4, I_5\}$ that include item I_2 will be removed from L_2. So $C_2 = \{\{I_0, I_1\}, \{I_0, I_2\}, \{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_4\}, \{I_2, I_4\}, \{I_3, I_4\}\}$

⑤ The join, $C_2 \infty C_2$ is performed in order to generate the set of frequent 3 - itemsets which is denoted L_3.

Before performing the $C_2 \infty C_2$, the items in the set of C_2 are checked in $l_1, l_2, l_3, l_4, l_5, l_6, l_7$, we find l_1 and l_3 not satisfied the connection conditions of algorithm

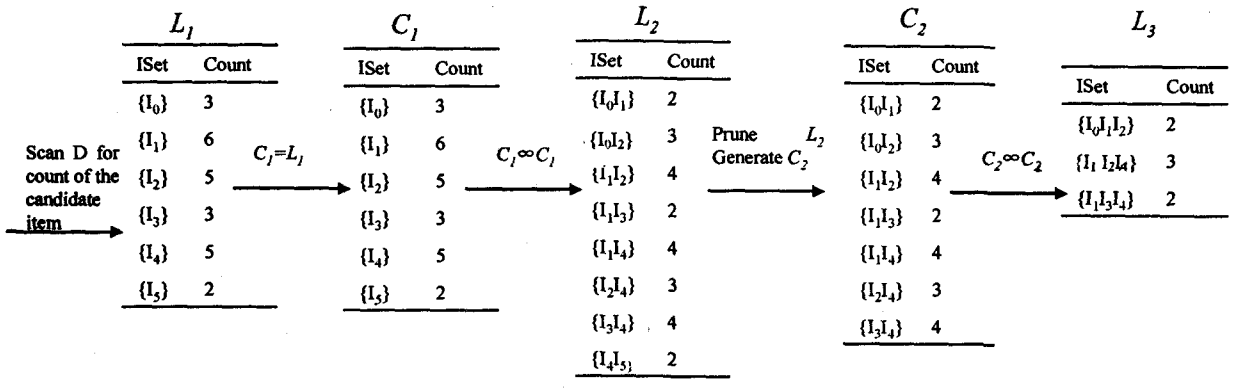

Fig 1 Generation of frequent itemset and candidate itemset, where min_ sup is 2

find_ frequent_ k_ itemsets(), then the join operation of l_1 and l_4, l_5, l_6, l_7 not be performed, too. So we can break in it , perform $l_2 \infty l_3$ and others. In the end $L_3 = \{\{I_0, I_1, I_2\}, \{I_1, I_2, I_4\}, \{I_1, I_3, I_4\}\}$.

⑥ By setp ④, can see the set of candidate frequent 3 – itemsets is empty, so the process is stop.

We use figure 1 to illustrate the improved Apriori algorithm for finding frequent itemsets in D .

From the algorithm design and algorithm perform – ing process analysis, improved Apriori algorithm (be denoted ATMC1), mainly from the following points better than the classic Apriori algorithm (be denoted ATMC2): the number of scanning database; the number of generating a set of candidate itemsets; the number of join and comparing and space utilization. In this paper, the implementation of the example described in table 4 contrast. Judging from the contrast, improved Apriori algorithm significantly better than the classical Apriori algorithm.

Table 4 Comparison of the algorithm before and after improvement

K	the number of scaning database ATMC1/ ATMC2	the number of join and comparing ATMC1/ ATMC2	the number of generating candidate itemsets ATMC1/ATMC2
1	1/8	0/0	6/6
2	0/8	11/28	7/15
3	0/8	3/3	0/10
4	0/8	0/0	0/2

3.2 Experimental Analysis

In this paper, test a large number of experimental data. Test environment: J2EE, SQL Server2000, Pentium 4 processors, 512M memory, 120G hard drive. Experimental data from an information library of a particular City Library in 2006 that recorded books borrowing information of this year. 10000 data records are tested, in order to identify the demand for all types of books of readers that are from the wide variety of industries, a variety of age groups and the academic background. Comparison of experimental data has been shown in figure 2. The experimental results show that the improved Apriori algorithm has better efficiency than classic Apriori algorithm.

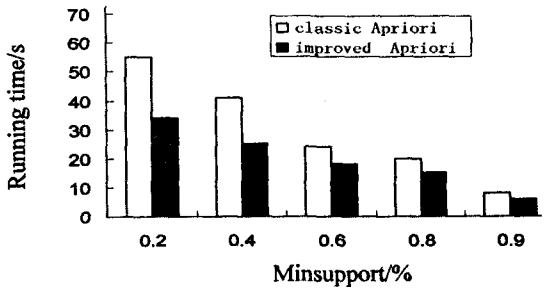

Fig 2 Comparison of the algorithm execution time before and after improvement

4 Conclusion

In this paper, mainly from the following suggestions to improve on the Apriori algorithm for mining association rules: First, algorithm uses the vertical express format of databases and map transaction database to array, therefore reducing the number of scanning database and reduce the access I/O frequency, improve utilization of space. Second, algorithm uses bitwise "and" operator and a random access characteristics of array, improve the speed of statistics of data itemset support count. That is, the number of high – performance computing to replace the relatively inefficient computing. Direct generation of frequent k – itemsets, rather than have a candidate itemsets, so reduce the system cost. Third, change the way of joining and comparing by using the orderly property of itemsets. Fourth, based on property 3, reduce the size of generating frequent k – itemsets. From the theoretical and experi-

(下转第 92 页)

声、拉普拉斯噪声到柯西噪声，它们的密度函数的拖尾从无逐渐变厚，噪声取较大值的概率逐渐变大[10]，适量噪声改善多元信号和信息处理的能力被中和。

4 结束语

基于互信息研究了四种典型噪声下多阈值系统中的 SR 现象。通过理论计算和计算机仿真得到：多阈值系统中，当输入信号在阈上时，噪声总是恶化系统的性能；当信号在阈下时，适量的噪声能够改善系统的性能，即 SR 现象存在。随着阈值系统阈值的增大，随机谐振功效降低，最佳噪声强度逐渐增大。相同阈值时，四种典型噪声作用下的互信息随噪声强度的增加的所能达到的最大值不同，也即 SR 的功效不同，均匀噪声下 SR 的功效最好，柯西噪声下 SR 的功效最差。

基于相关系数讨论多阈值系统中的随机谐振也能得到类似的结论。这些结果说明了多阈值系统中 SR 现象可能存在，且对噪声具有一定的鲁棒性，拓广了 SR 在多元信号与信息处理中的应用。

参考文献：

[1] Benzi R, Parisi G, Stuera A. A theory of stochastic resonance in climatic change[J]. SIAM Journal on applied mathematics, 1983,43(3):565－578.

[2] Benzi R, Stuera A, Vulpiani A. The mechanism of stochastic resonance[J]. J Phys A, 1981,14(5):453－457.

[3] Gammaitoni L, Hänggi P, Jung P, et al. Stochastic resonance [J]. Rev Mod Phys, 1998,70(1):223－287.

[4] 王友国，吴乐南. 极大阈值网络中的噪声提高 Fisher 信息[J]. 南京邮电大学学报：自然科学版，2007,27(2):68－71.

[5] 王友国，吴乐南. 离散时间系统中的噪声辅助信号传输[J]. 电子学报，2009,37(10):231－234.

[6] 张 雷，宋爱国. 随机共振在信号处理中应用研究的回顾与展望[J]. 电子学报，2009,37(4):811－818.

[7] Greenwood P E, Ward L M, Wefelmeyer W. Statistical analysis of stochastic resonance in a simple setting[J]. Physical Review E, 1999,60(4):4687－4695.

[8] Chapeau－Blondeau F, Rousseau D. Noise improvements in stochastic resonance from signal amplification to optimal detection[J]. Fluctuation and Noise Letters, 2002,2(3):221－233.

[9] Chapeau－Blondeau F. Stohcastic resonance and the benefit of noise in nonlinear systems[M]. Berlin, Heidelberg: Springer－Verlag, 2000:137－155.

[10] Wang Youguo, Wu Lenan. Stochastic Resonance in Nonlinear Signal Detection[J]. International Journal of Signal Processing,2006,2(2):108－113.

[11] 王友国，吴乐南. 极大并联阈值网络中噪声改善信号的相关性[J]. 数据采集与处理，2006,21(4):409－412

[12] 王友国. 广义高斯噪声下阈值系统中的随机谐振[J]. 南京邮电大学学报：自然科学版，2006,26(6):40－42.

[13] Wang Youguo, Wu Lenan. Stochastic resonance based on correlation coefficient in parallel array of threshold devices[J]. Journal of Southeast University(English Edition), 2006,22(4):479－483.

[14] Gammaitoni L. Stochastic resonance in multi－threshold systems[J]. Physics Letters A,1995,208:315－322.

[15] Wang Youguo, Wu Lenan. Stochastic resonance and noise－enhanced Fisher information[J]. Fluctuation and Noise Letters,2005,5(3):435－442.

(上接第 88 页)

mental results analysis, the improved Apriori algorithm significantly improve the efficiency of the algorithm.

References:

[1] 郭有强.一种高效的关联规则维护算法研究与实现[J].计算机技术与发展，2007,17(10):123－126.

[2] 袁万莲，郑 诚，翟明清.一种改进的 Apriori 算法[J].计算机技术与发展，2008,18(5):51－53.

[3] 钱雪忠，孔 芳.关联规则挖掘中对 Apriori 算法的研究[J].计算机工程与应用，2008,44(17):138－140.

[4] 段仰广，韦玉科.基于循环十字链表的频繁模式挖掘算法[J].计算机技术与发展，2009,19(10):73－77.

[5] 司开君，毛宇光.一种新的基于数据流的数据模型[J].计算机技术与发展，2006,16(1):1－3.

[6] Han J, Pei J. Mining frequent patterns by pattern growth: Methodology and implications[J]. J. SIGKDD Explorations (Special Issue on Scalable Data Mining Algorithms), 2000(2):142－152.

[7] Han J W, Kamber M. Data Mining: Concepts and Techniques [M]. 2nd ed. Beijing: China Machine Press, 2007:147－155.

[8] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//In Proc. VLDB'94. [s.l.]:[s.n.], 1994.

[9] Brin S, Motwani R, Ullman J D, et al. Dynamic itemset counting and implication rules for market basket data[C]//In Proc. ACM SIGMOD'97. [s.l.]:[s.n.], 1997:235－241.

[10] Cai W J, Zhang X H. Survey of Association Rule Generation [J]. Computer Engineering, 2001(5):31－33.

[11] 惠明滨，张凤鸣.一种基于栈变换的高效关联规则挖掘算法[J].计算机研究与发展，2003,40(2):330－335.

[12] Han J W, Pei J, Yin Y W, et al. Mining frequent patterns without candidate generation: a frequent－pattern tree approach[J]. Data Mining and Knowledge Discovery, 2004, 8(1):53－87.