Improved Apriori Algorithm

Guowei Jiang[1],Jianjun Wang
Hebei University of Economics and Business
Shijiazhuang, 050061, Hebei Province, China
E-mail: 1532198525@qq.com

Classic Apriori algorithm has the problem of multiply scanning database, and pattern matching is a time-consuming issue. In this paper, we propose an improved Apriori algorithm based on the classical Apriori algorithm. Improved algorithms numbers a binary number for each set. If number 1 in logical AND result is not less than k-1, connect the two item sets and get candidate k item sets. The new algorithm avoids a large number of pattern matching when connecting. Application examples show that time complexity of it is reduced, and the algorithm is effective and feasible.

ISCC 2015
18-19, December, 2015
Guangzhou, China

[1]Speaker

1.Introduction

Association rule mining has been one of important researches about data mining. It studies interesting patterns hidden in transaction databases、relational databases and other information storage. In 1993, the famous American scholar, R.Agrawl, proposed single dimensional Boolean association rule mining algorithm, then he made famous Apriori algorithm based on frequent itemsets[1-2]. The main idea of this algorithm is an iterative method obtaining (k+1) frequent items by k frequent items.Typical example of association rules is supermarket goods basket data analysis. It analyzes the relationship between different commodities that customer put into the basket, and then get the customers' buying habits. When mining association rules, how to improve the efficiency of the algorithm is very important. There are already a lot of improved algorithms based on Apriori algorithm[3-4]. Although these algorithms have advantages and mining performance are significantly better than traditional Apriori algorithm, in general, they are still relatively complicated algorithm. On the basis of the analysis of Apriori algorithm, an improved algorithm is proposed, which makes the algorithm in the compressing data source, improving the connection judgment and reducing the scanning of unnecessary transaction obvious efficiency.

2.Apriori Algorithm and Deficiencies

2.1Apriori Algorithm

Apriori algorithm is the most classic association rules mining algorithm, which uses an iterative search method step by step. First, scan the database and add up count for each item, and find a collection of frequent one set according to minimum support. This is denoted by L_1. L_1 is used to find out frequent two sets L_2, L_2 is used to find out frequent three sets L_3, keep looping until we can not find frequent k itemsets. We need a full scan of the database to find L_k every time.

Apriori algorithm uses a transcendental property (Apriori property) to compress the search space, so that the efficiency of frequent itemsets produced layer by layer increases. Apriori algorithm assumes that item of transactions and item sets are ordered by dictionary order.

2.2Apriori Algorithm Analysis and Inadequate

The traditional Apriori algorithm deficiencies following aspects:

(1)In the process of generating frequent k itemsets, you need to repeatedly scan the transaction database. The size of the candidate set determines the number of scanning the transaction database. If the size of the candidate k itemsets C_k is $|C_k|$, you need to scan $|C_k|$ second database, and some scans are unnecessary, which potentially greatly increase I/O loading.

(2)When generating candidate k-itemsets, comparison of determine the connection conditions are too many times, and it will produce a large number of candidate k itemsets, and many of them are duplicate and non-frequent. If you can prejudge some connections of some items and other items are non-frequent itemsets, avoiding these useless join operations, the efficiency of the algorithm can be improved.

(3)When calculating the candidate set frequency, Apriori scans all transactions many times, spending a lot of time on pattern matching of the candidate set and the transaction. If in the process of scanning the transaction database, items or transactions which are unnecessary to scan can be prejudged, it also can improve the efficiency of the algorithm.

2.3Apriori Algorithm Related Definitions and Theorems

Definition 1 {I1,I2,...,Im} is a collection of items, among them Ii∈I.

Definition 2 Suppose task-related data D is a transaction database,D={T_1,T_2,…,T_n}. Each of these transactions Ti is a collection of items, meeting $T_i \subseteq I$ .Each transaction has an identifier Tid.For $\forall T_i$ there is only one binary number corresponding to it, T_i={t1,t2,…,tn}, where ti=0 or 1.

Definition 3 AND operation. $T_i \wedge T_j$={t1,t2,…,tn}∧{t1,t2,…,tn}={s1,s2,…,sn}, where si=0 or 1. According to the definition of AND operation, the number of 1 in results of AND operation is the repeat number of T_i and T_j. When generating k candidate itemsets, if the number of 1 is not less than k-1, T_i and T_j can be connected.

3.Apriori Improvement

3.1Algorithm Improvement

(1)The storage structure: vertical structure

Improve data storage structure. The main idea is to use the new data structure[5-6], the current level data structure is converted into the vertical data structure correspondingly. The new structure consists of itemsets and transaction list that contains the itemset, for example, the table 1 transaction database D is converted into table 2.

Tid	Items
T_1	ABE
T_2	ACD
T_3	BD
T_4	CDE

Table 1: the original transaction database

A	B	C	D	E
T_1	T_1	T_2	T_2	T_1
T_2	T_3	T_4	T_3	T_4
			T_4	

Table 2: the item transaction database

The number of transactions that support k-itemsets through such transformation is easy to calculate. The transaction set supporting item A is T_A, the transaction set supporting item B is T_B, the itemsets not only supporting A but also supporting B are the same transaction between the transaction set A and the transaction set B, namely: $T_{AB}=T_A \cap T_B$.

After application of this data structure, only in the calculation frequency of frequent 1 item sets it scans the transaction database. Afterwards,calculate frequency of the candidate k itemsetsC_k (k>1) by frequent k-1 itemsets C_{k-1}. In addition, in the calculation of the support of the candidate item sets, it avoids the pattern matching, and greatly improves the speed of the algorithm.

(2)Join Optimization

1) According to the definition that items can be connected, if l_x, $l_y \in L_{k-1}$, located $l_x[i]$ for the l_x in the i-th item, if $l_x[1]=l_y[1] \wedge l_x[2]=l_y[2] \wedge ... \wedge l_x[k-2]= l_y[k-2] \wedge l_x[k-1]<l_y[k-1]$, the join condition is satisfied. When generating k candidate itemsets, firstly compare the k-2 items of itemsets are the same or not, if they are different, do not connect them. This will eliminate part of the pattern matching problem, and is easy to improve the efficiency of the algorithm.

2) Number the candidates. In generating candidate item sets, according to definition 3, you will do lots of logical AND operations, ALU components of the computer can quickly achieve them. When the frequent itemsets' length are long, it greatly speeds up the production rate of the candidate item sets.

Initially, scan the database to calculate the number of candidates, and it is assumed to n. The number of items in the frequent 1 itemsets (sorted by dictionary) is set to n-bit binary number, only their location (in dictionary order) is 1, and the remaining bits are all 0. At this point you can connect any two items, modify binary number once connected, the letter position is 1.Construct tree with Ii as root, if the number of occurrences of {IiIj}is greater than or equal to the minimum support number, Ij is Ii's branch. if the number of occurrences of {IiIjIk} is greater than or equal to the minimum support number, Ik is Ij's branch, and so on. You finally get n trees, if the number of 1 in the binary number corresponding to leaves is max, the branch is frequent itemset.

E.g., L_1 = {{A},{C},{D},{E}}, A is numbered 1000, C is numbered 0100, D is numbered 0010, E is numbered 0001. You can connect any two items, it generates three trees, child nodes of one tree with A as the root are CDEF, branches are {AC} 1100,{AD} 1010 and {AE} 1001. Child nodes of one tree with C as the root are DE, branches are {CD} 0110,{CE} 0101. Another tree is {DE} 0011. Suppose they are frequent, when generating the candidate three itemsets, for example, tree A, {AC}$\wedge${AD} =1100$\wedge$1010 = 1000, the number 1 in result is 1, which is greater than or equal to k-1 (that is 1). So {AC} and {AD} can be connected to form {ACD} 1110, namely D is a child node of C.

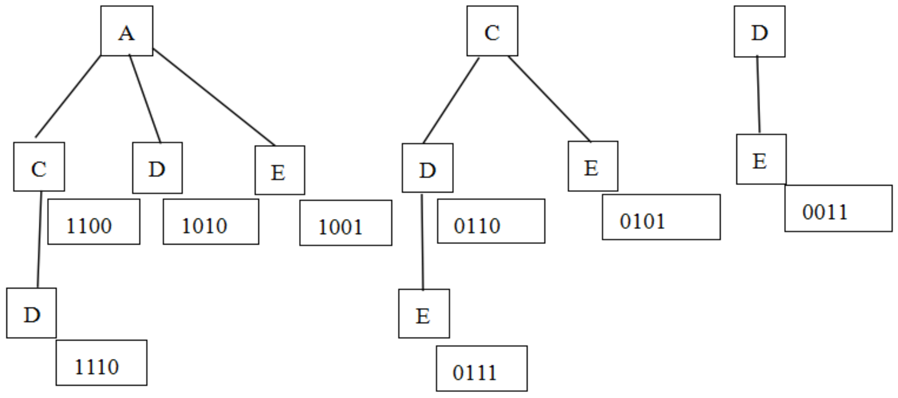

Figure1: spanning tree

Suppose spanning tree is as shown above, scan the binary number of leaves, and get a maximum of 1 is 3, so frequent itemsets is three sets. Deeply traversal spanning tree and obtain three frequent sets are {ACD} and {CDE}.

(3)Pruning Optimization

If the k dimensional data itemset X={i1,i2,...,ik}, there is a $j \in X$ that makes $|L_{k-1}(j)|<k-1$, then X is not frequent item. $L_{k-1}(j)$ denotes the number of j containing in the set L_{k-1}.

Proof: Suppose X is frequent k itemsets, its' k k-1 subset of the items are in L_{k-1}. However, in the k k-1 subset generated from X, each project j∈X appears totally k-1 times. For any j∈X there is $|L_{k-1}(j)| \geq k-1$, which is a contradiction, so X is not frequent items.

According to this property, remove all frequent items including j included in the frequent itemsets L_{k-1}, thereby obtaining a new smaller set of frequent k-1 project set. Examples are as follows, provided minimum support min sup = 3.

Tid	Items	Tid	Items
T_1	I1,I3,I5	T_6	I1,I4,I6
T_2	I2,I4,I6	T_7	I3,I4,I5,I6
T_3	I1,I3,I4,I5	T_8	I1,I3,I5
T_4	I2,I3,I4	T_9	I3,I5,I6
T_5	I1,I3,I5,I6	T_{10}	I1,I3,I5

Table 3: the original transaction database

Step 1 Scan original transaction database D (as shown in Table 3) once, and obtain the vertical data structure, candidate set C_1 and frequent set L_1.

Step 2 frequent 1 itemsets self-join produces candidate two itemsets C_2, C_2={{I1,I3}, {I1,I4},{I1,I5},{I1,I6},{I3,I4},{I3,I5},{I3,I6},{I4,I5},{I4,I6},{I5,I6}}. According to (1), calculate respectively frequencies of the candidate 2 itemsets, and remove sets whose frequency is less than 3, then get frequent 2 itemsets L_2, L_2={{I1,I3},{I1,I5},{I3,I4},{I3,I5},{I3,I6}, {I4,I6},{I5,I6}}.

Step 3 Apply of the above property when two itemsets self-join produces candidate three itemsets C_3. $|L_2(I1)|=2$, $|L_2(I3)|=4$, $|L_2(I4)|=2$, $|L_2(I5)|=3$, $|L_2(I6)|=3$. Since $|L_2(I1)|=2<3$, so get rid of all the items contain I1, similarly remove all items that contain I4. Thereby L'_2={{I3,I5}, {I3,I6}, {I5,I6}}. Then L'_2 self-join generates three sets of candidate C_3={{I3,I5,I6}}, the frequency count is three. So frequent item set is {I3,I5,I6}.

3.2Description of Improved Algorithm Steps

(1)First, scan the source transaction database. In the scanning process, record transaction codes support for each item, and all items are sorted according to the dictionary. Then number a number of length n for each and assign them, count frequency for each item. You should delete frequency count that is less than the minimum support, resulting in frequent 1 set L_1.

(2)Frequent 1 set self-join produces candidate two sets C_2. Connect any two items and change the corresponding number, then construct tree. By calculating the intersection of two items obtain transaction code set supporting the itemsets, remove itemsets whose support count is less than the minimum support, then get frequent 2 itemset L_2.

(3)And so on, to the frequent k-1 itemsets, use firstly of connecting optimization methods and pruning optimization methods and remove k itemsets which are frequent impossibly. If number 1 in logical AND result is greater than or equal to k-1, connect the two itemsets and get candidate k itemsets. By calculating intersection of transactions which support frequent k-1 sets in candidate k itemsets(located $x \in L_{k-1}$, $y \in L_{k-1}$, then $x \infty y \in C_k$, namely $T_{x \infty y} = T_x \cap T_y$), get transaction sets supporting k itemset, then delete itemsets whose frequency is less than the minimum number, finally get frequent k item sets L_k, and construct frequent k item sets tree.

(4)Repeat (3) until no more frequent itemsets generate. Deeply traversal spanning tree and obtain frequent k sets.

3.3Experimental Analysis

Advantages of improved Apriori algorithm：

(1)The algorithm just scan once the database, the time spent is Apriori's 1/ $\sum_{k=1}^{n}|C_k|$

(2)AND operations of n dimensional Boolean vectors replaces pattern matching between transaction items.It will greatly reduce the resource consumption of ALU, and improve pointing efficiency.

(3)The algorithm eventually generates m trees. Using "depth-first algorithm", select and calculate a tree with the maximum levels(maximum depth), generate maximum frequent itemsets as soon as possible. This will further enhance the operational efficiency of the algorithm, reducing the use of computing resources.

The algorithm can achieve lightweight calculation.

In order to test the efficiency of the improved algorithm, this paper carried out experiments on a large number of databases. The data size is fixed at 5000 records (using IBM's DBminer2 synthesis tool). Parameters of the synthesized data are as follows: the average number of items each record contains T=10, the average length of itemsets I=4, projects number N=20. Under the same conditions with the hardware configuration, efficiency of Apriori algorithm and Apriori improved algorithm is tested, experimental data is as shown in Table 4. From Figure 2, we can see the improved algorithm has some improvements in time.

min sup (%)	Apriori running time(s)	improved Apriori running time (s)
0.1	51	42
0.2	30	21
0.5	20	10
1	10	7
2	4	2

Table 4: test result

The test results comparison chart according to test data is as follows.

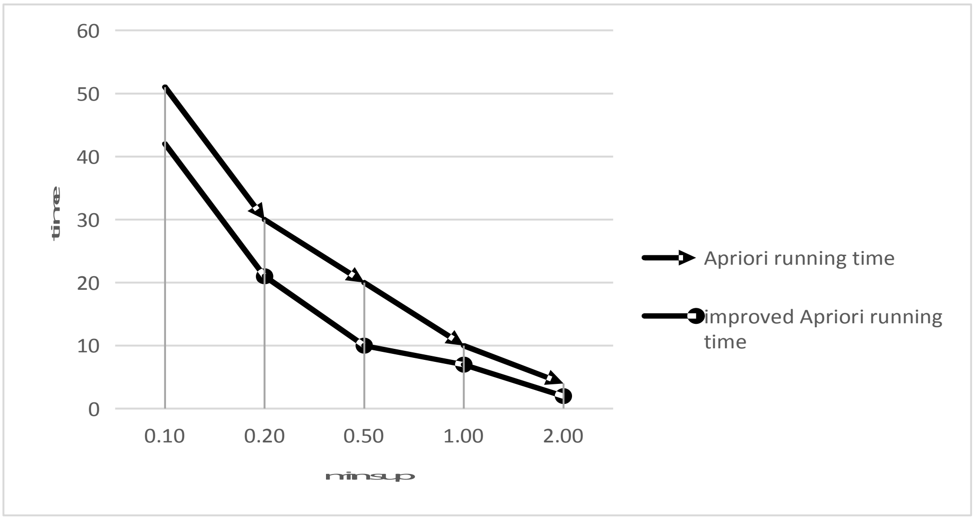

Figure 2: test results comparison chart

4.Conclusion

This article begins with the theoretical basis of Apriori algorithm, extend its basic theory and obtain an improved Apriori algorithm. The algorithm is improved in three aspects. First is data storage, scan the entire database only one time, when counting only transaction codes match instead of matching each item in transaction. The second point is, use the nature to remove invalid candidate sets. Third, number the binary digits for itemsets, if number 1 in logical AND result is not less than k-1, connect the two itemsets and get candidate k itemsets. It avoids a large number of pattern matching. Thus the improved algorithm has a higher efficiency than the classical Apriori algorithm.

References

[1]AGRWAL R,SRIKAN R. *Fast algorithms for mining association rules in large database*[C]//Proceeding soft the 20th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers,1994:487-499.

[2]Jiawei Han, Xiaofeng Meng, *Data mining concepts and technologies the third edition of the original book*[M], Mechanical Industry Press, Beijing, August 2012, 160-166, (InChinese).

[3]Yan Shen, Shunlin Song, *Efficient data mining algorithm on large-scale data collection*[D], PhD Thesis, Jiangsu University, June 2013, (InChinese).

[4]Ke Luo, Caiwang He, *an association rules extraction algorithm based on Apriori algorithm*[J], Computer and Digital Engineering, 2006, 34 (4):48-51, (InChinese).

[5]Yunfeng Li, Jianwen Chen, Daijie Cheng, *Study about association rules mining and improvement of Apriori algorithm*[J], Computer Engineering and Science, 2002, 24 (6):65-68, (InChinese).

[6]Wandan Zeng, XubobZhou, Bo Dai, *Matrix algorithm of mining association rules*, Computer Engineering[J], January 2006, 32 (2):45-47, (InChinese).