A Mining Algorithm of Maximal Frequent Itemsets Based on M-Bisearch

Meilin Zeng

(Jiangxi Industrial and Trade Vocational and Technical College 330038)

Keywords: Data mining; Association rules; Frequent itemsets; Machine learning

Abstract. The core theory of big data analysis is data mining. Association rule mining algorithm is an important branch of data mining. It contains two steps: generation of frequent itemsets and generation of association rules. The algorithm overhead in the generation of frequent itemsets is very high. Starting from the nature of the maximal frequent itemsets, the idea of M-Bisearch is used on the basis of changing the data storage structure. The storage space is compressed to reduce the number of scans and reduce the computational overhead of support, so as to achieve the purpose of improving algorithm execution efficiency. Experiments show that the improved algorithm has obvious advantages when dealing with frequent itemsets mining in long-term mode.

Introduction

Data mining algorithm is the theoretical core of big data analysis, and association rule mining algorithm is an important branch of many data mining methods. Now it has become the most active and mature research direction in data mining field. Its purpose is to explore the database. Useful interdependencies between different attributes. The Apriori algorithm proposed by Agrawal and Srikant in 1994 is the most representative and authoritative algorithm in the association rule mining algorithm. Many later algorithms are based on the Apriori algorithm to optimize and extend [1~2]. However, there are two bottlenecks in the Apriori algorithm: 1) Scanning the transaction database multiple times requires a large I/O load; 2) Generating a large number of candidate sets, which is a challenge to the runtime and main memory space.

Domestic and foreign scholars have done a lot of research and put forward a series of improved algorithms. Documents [3~4] reduce the number of tasks to be scanned and reduce the time for scanning database D; documents [5~7] use Hash technology to generate candidate item sets efficiently, while using only transaction lengths not less than the item set length is possible including the nature of the itemsets reduces the time of scanning the database; literature [8] changes the storage structure of the itemsets, reduces the number of times of scanning the database by using dynamic item set counting; literature [9~10] changes the storage form of data from the transaction contains Items become items containing transactions, while improving the connection method that generates the candidate set reduces duplicate connections.

The above improvements to the Apriori algorithm all begin with the connection of the data storage structure and the candidate candidate set, avoiding multiple scans of the database and reducing the time complexity of the algorithm. However, some of the above algorithms require a large amount of storage space, some data structures are overly complex, and some still have large iteration overheads, and there is room for improvement in reducing the storage complexity and iteration of the algorithm. Through the study of the above-mentioned literature, a new improvement method is proposed: using the vertical database structure, the idea of using M-BISEARCH to find the maximum frequent itemsets, and then directly according to the nature of the largest frequent itemsets.

It generates all frequent items. This method improves the efficiency of the algorithm by simplifying the data storage method, supporting counting method and frequent item set generation process. After experimental demonstration, the mining efficiency of data based on M-Bisearch algorithm is greatly improved.
Improved Algorithm Implementation Principle and Process

Introduction to VDS

The vertical database structure (VDS) is relative to the horizontal database structure. Traditional database is a transaction. For units, data items are contained by transactions, and VDS is an item, and the set of transactions that contain the item is composed of a TidList, which is called a support transaction list. The TidList contains the sequence number of the transaction that contains the item and counts support items in descending order. VDS can reduce the storage of data in the transaction database and reduce the data redundancy on the basis of ensuring the transaction data integrity. At the same time, changing the item set count reduces the number of times the database is scanned.

Improvement Ideas

The original transactional database is first converted to a VDS database, which is sorted in descending order by the count of transactions in the item’s supported transaction list. On this basis, a transaction length count table is generated at the same time. The table includes the transaction length and the number of transactions that reach the length. The conversion of the database structure is shown in Table 1 and Table 2, and follows the above example. The transaction length count table is shown in Table 3.

Table 1  Horizontal data structure         Table2  Transformed vertical database structure

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>No</th>
<th>I</th>
<th>s</th>
<th>Sc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A/B</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>COE</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>ACEF</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>ABCE</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>BDF</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>CE</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>ADE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3  Transaction length statistics

<table>
<thead>
<tr>
<th>Length</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Then, people calculate the number of transactions in each transaction length interval from the largest to the smallest, find a length value M, make the transaction count of M and M greater than the transaction length value satisfy the minimum supportability threshold. At the same time, the transaction length value is greater than M. The count sum is less than the minimum support...
threshold. The M value is determined as follows:

\[
\sum_{T: N\text{-}\text{Items} \subseteq T, T \subseteq D} \leq \text{min}_\text{sup} \cdot N \geq M
\]

Finally, according to the idea of binary search, all maximal frequent item sets are found and all frequent item sets are generated. According to the M value and the shortest frequent set length 1 up to take the integer average, set \((M + 1) / 2 = i\), according to the frequent 1_itemsets generate i_itemsets, the corresponding items of the support transaction list for the intersection operation. The support count table is compared with \text{min}_\text{sup}. Let \(x\) be an i_item set. If \(\text{count}(x) \geq \text{min}_\text{sup}\), calculate \((i + M)/2\) to round up. If \(\text{count}(x) < \text{min}_\text{sup}\), calculate \((1+i)/2\) Rounds up and continues the support comparison according to this binary search method. Until the next length is different from the current item set length, use another length to generate the candidate set, and finally make the length \(k\). The item set is a frequent item set, and the item set of length \(k+1\) is an infrequent item set. Then, the maximum frequent item set set \(MF\) is added. Finally, all the maximal frequent item sets are found, and then all frequent item sets are generated according to the nature of the maximal frequent itemsets.

Similar to the Apriori algorithm, the improved algorithm is also connected and pruned. In this paper, the algorithm obtains the \(k\_\text{item set through the frequent 1\_item set concatenation operation, and then the support degree obtained by the intersection operation of TidList, if it is the infrequent item set, it will be classified into the pruning set and the downward direction. Perform a binary search when it is a frequent item set, perform a binary search upwards, compare the new candidate set with it, and perform a pruning operation.}

**Example Description**

Using the data in Table 1 to set \(\text{min}_\text{sup}=0.4\), then the VDS database is shown in Table 2. The transaction length statistics are still shown in Table 3.

Since \(\text{min}_\text{sup}=0.4\), the transaction length support is \(\text{support}(4) = 2 /10\), \(\text{support}(3) = 7 /10\), \(\text{support}(2) = 10/10\), so the maximum value of the maximum frequent item set length M is 3.

According to the binary search idea, the maximal frequent itemsets are searched on the length interval [1,3]. (1+3) = 2, so mining starts from the 2_itemsets. Scanning the VDS database for frequent 2_item sets is shown in Table 4.

<table>
<thead>
<tr>
<th>Item</th>
<th>Support List</th>
<th>Support Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA</td>
<td>3, 5, 7, 10</td>
<td>4</td>
</tr>
<tr>
<td>EC</td>
<td>2, 3, 4, 5, 8</td>
<td>5</td>
</tr>
</tbody>
</table>

Because it is a 2_item set, itemsets that do not meet the supportability threshold directly use the 1_item set as the maximum frequent itemset. The support of EA and EC satisfies the condition of \(\text{min}_\text{sup} = 0.4\). Because the length of the option set is 2 and the length of the upper limit 3 is adjacent, the candidate set of another length value is directly calculated without taking the integer average upward. EAC, EAD, ECD, according to the nature of the non-frequent itemsets, the superset of the infrequent itemsets must be pruned, then the candidate set becomes EAC. At this time, the EAC support count is obtained by the following intersection operations: E. TidList\(\cap\)A. TidList\(\cap\)C. TidList\(=\)2, \(\text{Support(EAC)}<\text{min}_\text{sup}\), so EAC is an infrequent itemset. Because EA, EC itself is a frequent itemsets, and their direct supersets are all non-frequent itemsets. Available EA and EC are the maximum frequent itemsets. Therefore, the four D, B, EA, and EC maximal frequent itemsets are obtained. According to the properties of the subset of maximal frequent
itemsets, which are the collection of all frequent itemsets, there are E, A, C, D, B, EA, and EC 7
Frequent itemsets. According to the Apriori algorithm, seven frequent item sets A, B, C, D, E, AE,
and CE will be obtained. It is exactly the same as the frequent itemsets obtained by the improved
algorithm. Through example analysis, we can see that the improved algorithm is effective and
accurate. The performance of the improved algorithm will be further verified experimentally later.

**Algorithm Description**

Input: D- raw transaction database, min_sup—minimum support threshold
Output: Frequent Itemset Fk

1. Generate_VDSandTable(D);  // Scan original database to generate vertical database VD and
transaction length statistics table.
2. Max_length (Table);
3. Right_length = Max_length;
4. Left_length = 1;
5. Generate_VDS(VD, min_sup);
6. For (i = 0; i < Item. size − 1; i++) {Add frequent itemsets to L1}
7. For (k = 2; k != null; k) {Pk = join (L1, k);
8. Ck = Cut (Pk); Lk = Count (Ck);
9. if (k = Right_length And k _ itemsets are non-frequent items or k = Left_length and the K
itemset is a frequent item) {Then writes the K itemset at this point to the maximum frequent itemset
set(Max_frequent_itemsets);}
10. If (Length. Ck ≥ min_sup){If (Right_length − k > 1) k = (k+ Right_length) /2 Upward
rounding;
11. Else k = Right_length; Else if (k-Left_length > 1)k=( k-Left_length) /2 Upward rounding;
12. Else k = Left_length;
13. P_k = \emptyset; C_k = \emptyset;}
14. Function description:

Generate_VDandTable(D): scan original database D, generate vertical database VD and
transaction length statistics table Table; Max_length (Table): calculate the maximum length value
of frequent itemsets from the transaction length statistics table;
Generate_VDS(VD, min_sup): delete infrequent 1_item sets based on vertical database and
min_sup;
Join(L1,k): The k_itemsets are obtained by the intersection of frequent 1_item sets;

**Experimental Results and Analysis**

To verify the performance of the algorithm, the improved algorithm is compared with the Apriori
algorithm and compared with the Apriori_BL algorithm in [7].

Experimental environment: Intel(R) Core(TM) i-5 3470 CPU @3.20 GHz 3.20 GHz; memory 4.00
GB; 650 G hard disk; Windows 7*64 bit operating system.

**Experiment 1**

The data uses the Extended BAKERY data set in [11], which has a total of 7500 transactions.
Each transaction consists of 8 attributes, and each attribute has 0 to 5 attribute values. Randomly
take 5,000 data for experiments. The results are shown in Figure 1 and Table 5.
From Figure 1, we can see that for the short-term frequent item set mining problem of Extended BAKERY data set, the improved algorithm is a single Apriori_ BL algorithm with improved storage structure in [8]. When the support degree threshold is low, the algorithm efficiency has a certain increase, but the gap is not great. With the increase of the support degree threshold, the gap between the two is gradually reduced. When Min_ sup=0.6, the complexity of the algorithm in the paper leads to its running time greater than that of the Apriori_ BL algorithm.

From Table 5, we can see that the mining results of the improved algorithm are consistent with the mining results of the Prior algorithm, and further illustrate the correctness and completeness of the improved algorithm.

**Experiment 2**

Still using the Extended BAKERY data set, the three algorithms were tested for changes in the size of the data set (transactions from 5000 to 75000), and their running time was changed to set the supported threshold Min_ sup=0.2. The experimental results are shown in Figure 2.

From Figure 2, we can see that when the transaction set is large, the running time of the improved algorithm is significantly lower than the running time of the other two algorithms. However, as the transaction set is gradually reduced, the running speed of the algorithm in the text is not reduced by the other two algorithms. The reduction is large. When the transaction set size is reduced to 20000, the runtime of the Prior_ BL algorithm is lower than the improved algorithm.
Conclusion

Based on the deep research of Apriori algorithm, the paper points out the limitation of Apriori algorithm and proposes a new improved algorithm. Starting from the nature of the largest frequent itemsets, the idea of M-Bisearch is used on the basis of changing the data storage structure, and the number of scans and the computational overhead of support are reduced by compressing the storage space, so that the purpose of improving the execution efficiency of the algorithm can be achieved. It greatly improves the efficiency of the algorithm. The experimental results also show that the algorithm does improve the efficiency of the Apriori algorithm. It is believed that this has certain significance for solving the problem of lack of information and other issues related to data explosion in the current era.

References


About the Author:
Zeng Meilin, Bachelor, Librarian, The main research direction is library information construction. E-mail:1027205415@qq.com