**Improvement of Apriori Algorithm Based on Vector and Vertical Array**

Zhen-yu GUO¹,a and Tian-huang CHEN²,b,*

¹,²WuHan University of Technology, Wuhan 430063, China
a guozy0818@126.com, b thchen57@126.com
*Corresponding author

**Keywords:** Data mining, Association analysis, Apriori algorithm, Vector array, Vertical array.

**Abstract.** In the data mining method of association analysis, the classic Apriori algorithm of discovering frequent item sets may multiple scanning the source database, produce a large number of candidate and repeatedly pattern matching, which leads to low time efficiency of the algorithm. Based on the analysis of the array based algorithm, an improved algorithm is proposed in this paper. The main idea is to scan the source database once and use vector arrays and vertical arrays to represent the transactions, improve the strategy of the join step and the prune step when candidate frequent(k+1)-item sets were generated from frequent(K)-item sets as well as the pattern matching strategy. The experimental results show that the time complexity of the improved algorithm is reduced greatly.

**Introduction**

Association rules [1] is the core of database knowledge discovery, is an important research direction of data mining. It mainly researches on transaction database, relational database and other information storage forms, to find hidden and interesting rules between large number of data items. In 1993, the famous American scholar R. Agrawal and others put forward the Boolean association rules mining for the first time, then puts forward the famous frequent item sets based Apriori algorithm [2], its most typically applications are in the shopping sales analysis. Apriori algorithm for mining association rules is one of the most classical algorithm, it is named according to the characteristics of frequent item sets of prior knowledge [3]. The algorithm uses a hierarchical order search loop method to complete the work of frequent item sets mining. Using the known (K - 1) d frequent item sets to generate K dimension frequent item sets candidate frequent item sets, and then scanning the database again to judge whether the candidate frequent item sets is a frequent item sets.

The Apriori algorithm is effectively to produce all association rules, but there are some problems on efficiency: scanning the database too many times, the candidate set is too large and the test of candidate item sets need to spend a lot of time. This leads to the low efficiency of Apriori algorithm. So how to improve the efficiency of the algorithm is very important when mining association rules. Literature [4] proposed an Apriori algorithm based on array, the algorithm mapped the transaction database to a Boolean array, then all operation is based on the array. The improved algorithm uses the array to convert the scanning of the database to the memory array scanning, the algorithm only scans the database once, saving storage I/O time, improves the efficiency.

In this paper, based on the improved method mentioned before a new improve algorithm is put forward based on vector and vertical array. Through the compression storage of transactional database and optimization of pattern matching strategy between candidate item sets and transaction, the efficiency of the algorithm is improved greatly.
Introduction of Apriori Algorithm Based on Array

Some Terms of Association Rules

Set $I = \{i_1, i_2, \ldots, i_m\}$ as a m dimension data collection; $D = \{T_1, T_2, \ldots, T_n\}$ as a n dimension Transactional database; $T \subseteq I$, every transaction $T$ is represented by a globally unique identifier TID; any subset of $I$ namely $X$ is called data item sets, if $\forall i \in X$, all have $i \in T$, then $X \subseteq T$, we call this transaction $T$ contains data set $X$; the support of data item set $X$ is $sup(X) = |X(T)| / |D|$, among them $|X(T)|$ and $|D|$ respectively the number of elements of $X(T)$ and $D$; if $sup(X) \geq \minsup$ (minimum support), then we say that $X$ is frequent item sets, an association rule is represented like $X \Rightarrow Y$, among them $X \subseteq I, Y \subseteq I$ and $X \cap Y = \emptyset$, the confidence level of association rule $X \Rightarrow Y$ can be defined as follows:

$$\text{conf}(X \Rightarrow Y) = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)}$$

The problem of mining association rules is to find the association rules which satisfies

$$\text{sup}(X \cup Y) \geq \minsup \quad (1)$$
$$\text{conf}(X \Rightarrow Y) \geq \minconf \quad (2)$$

We call the rules which satisfies the condition (1) and (2) at the same time the strong association rules.

The Description of Apriori Algorithm Based on Array

Association rule mining algorithm based on array\textsuperscript{[5]}, scans the database only once, this algorithm suppose each transaction in the database as a row in a two dimensional array, with columns represent the items and rows represent the transactions, by scanning the corresponding column we can get the frequency of the corresponding item.

The execution step of the algorithm describes as follows:

- Scan the database once, converts the database items to corresponding Boolean constants, that is, the items contains in the transaction is represented by “1”, correspondingly the items not contains in the transaction with “0”, the converted items are stored in a two dimension array. Then by calculating the sum of each column in the array, we can get the frequency of each item set. Compare the frequency with minsup(minimum support), if equals to or greater than the minsup, we call the item set is a frequent 1-itemsets.

- Generate the candidate frequent (K+1)-item sets according to the frequent K-item sets. This step is consistent with the connection and pruning steps of Apriori algorithm.

- Generate frequent K-item sets according to the candidate frequent K-item sets. In order to calculate the frequency of each candidate item sets, scan the corresponding column in the array, if the value is “1”, add one to the frequency.

- Repeat the steps above, until no frequent item sets appears.

Algorithm Analysis

Algorithm Advantages

The execution step of the algorithm describes as follows:

- Scans the database only once, reduces the number of I/O.

- When calculating the frequency of the item sets, using the structure characteristics of the two-dimensional array, scanning the corresponding column directly, rather than scanning the entire array, to improve search speed.
Algorithm Disadvantages

While generating candidate frequent item sets, may produce a large number of candidate item sets of no value, this increases both the time cost of array scanning and the connection time to generate the item sets.

The connection routine compares the same sets too many times, and also can produce many less frequent item sets which can be determined in advance. If we can determine them in advance, we can avoid the useless join operation, thus we can improve the efficiency of the algorithm.

After the generation of every candidate item set, the algorithm need back scanning the array to determine whether the candidate frequent item set is a frequent item set. During the scanning procedure, some item sets are scanned many times which can be determined that don’t need to. If we can avoid these unnecessary scans, we can improve the efficiency of the algorithm.

Improvement of Apriori Algorithm Based on Array

The Related Properties and Theorem of Frequent Item Sets

Definition 1 Let \( x \in L_k, y \in L_k \), if \( x_1 = y_1 \land x_2 = y_2 \land \ldots \land x_{k-1} = y_{k-1} \land x_k < y_k \), then we call \( x_1x_2\ldots x_{k-1}x_ky_k \) is a connection of \( x \) and \( y \), denoted by \( x \bowtie y \).

Definition 2 Let the collection of transaction IDs which supports \( x \) denoted by \( T_x \), if \( x \in L_k \), then we denote the collection of transactions which supports \( x \) by \( T^f_x \), that is \( T^f_x = \{ t_{id} \mid t \in D \land x \subseteq t \land x \subseteq L_k \} \).

Property 1 The necessary condition for a \( K \) dimension data item set \( X \) be a frequent item set is all its \( K-1 \) dimension subset are frequent item sets.

Property 2 If there exists one \( K-1 \) dimension subset of a \( K \) dimension item set \( X \) which is not a frequent set, then \( X \) is not a frequent item set.

Theorem 1 Let \( K \) dimension item set \( X = \{i_1, i_2, \ldots, i_k\} \), if there exists \( j \in X \), satisfies \( |L_{k-1}(j)| < k-1 \), then \( X \) is not a frequent item set.

Corollary If there exists an item set \( e \) contains item \( p \) in \( L_{k-1} \), which satisfies \( |L_{k-1}(p)| < k-1 \), then all the item sets which generated by items in \( L_{k-1} \) other than \( e \) connects with \( e \) will not be a frequent item set.

Theorem 2 When calculating the frequency of a \( K \) dimension item set \( C_k \), if the length of the transaction \( T \) is shorter than \( K \), then transaction \( T \) needs not to be scanned.

Theorem 3 Let \( x \in L_k, y \in L_k \), if \( x_1 = y_1 \land x_2 = y_2 \land \ldots \land x_{k-1} = y_{k-1} \land x_k < y_k \), then \( T^l_{x \bowtie y} = T^l_x \cap T^l_y \).

Core Idea of Improvement

Related Data Structure. Considering the disadvantages of the original Apriori algorithm, this paper put forward two kinds of data structure to improve the execution efficiency of the algorithm.

Vector Array

This structure consists of a series of two dimension array \( A_i \) (i refers to the length of the transaction) which has different number of columns. According to the length of each transaction, the transaction set is divided into \( i \) groups, the same transaction only recorded once, the number of the occurrences is recorded respectively. In the two dimension arrays, the column represents the item, the row represents the transactions. Also the transactions are denoted by a new mark \( W_i \).

Vertical Array

This vertical structure can be seen as a data structure corresponding to the project transaction. With column on behalf of the item, all of the transaction number is stored in the corresponding column. This structure is made on the basis of the vector array mentioned before.

Example of the Data Structures

Assume that the transaction data set is as shown in Table 1, then the corresponding vector array and vertical array is as shown in Table 2 and 3.
Table 1. The original transaction set D.

<table>
<thead>
<tr>
<th>TID</th>
<th>Item list</th>
<th>TID</th>
<th>Item list</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>I1, I3, I5</td>
<td>T7</td>
<td>I3, I4</td>
</tr>
<tr>
<td>T2</td>
<td>I2, I4</td>
<td>T8</td>
<td>I1, I3, I5</td>
</tr>
<tr>
<td>T3</td>
<td>I1, I4</td>
<td>T9</td>
<td>I3, I6</td>
</tr>
<tr>
<td>T4</td>
<td>I3, I4, I5</td>
<td>T10</td>
<td>I3, I4</td>
</tr>
<tr>
<td>T5</td>
<td>I1, I3, I6</td>
<td>T11</td>
<td>I1, I3, I5</td>
</tr>
<tr>
<td>T6</td>
<td>I1, I4</td>
<td>T12</td>
<td>I3, I6</td>
</tr>
</tbody>
</table>

Table 2. The corresponding vector array.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>I2</td>
<td>I4</td>
<td>1</td>
<td>W5</td>
<td>I1</td>
</tr>
<tr>
<td>W2</td>
<td>I1</td>
<td>I4</td>
<td>2</td>
<td>W6</td>
<td>I3</td>
</tr>
<tr>
<td>W3</td>
<td>I3</td>
<td>I4</td>
<td>2</td>
<td>W7</td>
<td>I1</td>
</tr>
<tr>
<td>W4</td>
<td>I3</td>
<td>I6</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Vertical array generated by the vector array.

<table>
<thead>
<tr>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>I6</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2</td>
<td>W1</td>
<td>W3</td>
<td>W1</td>
<td>W5</td>
<td>W4</td>
</tr>
<tr>
<td>W5</td>
<td>W4</td>
<td>W2</td>
<td>W5</td>
<td>W6</td>
<td>W7</td>
</tr>
<tr>
<td>W7</td>
<td>W5</td>
<td>W3</td>
<td>W6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Design of the Improvement Algorithm**

Using the vector array to simplify the operations generating candidate frequent item sets. The vector array generated by scanning the data set records each item of the transactions, the length of each transaction and the times the transaction appears in the data set. While generating (K+1) candidate item sets from K candidate item sets, we can delete the transactions whose length is shorter than (K+1) according to Theorem 2. When calculating the frequency, the same transaction only need to be calculated once, the number of it is recorded in the Cnt array. The transactions are recoded in this structure, as is shown in Fig. 3-2.

Using the vertical array to simplify pattern matching operation. According to the improving idea of literature[6], we can easily calculate the transactions which support the K-item set: as is known, TAB represents the transactions that supports item A and B, TC represents the transactions that supports item C, then according to Theorem 3, we know that the transactions which supports A, B and C is the intersection of the two sets, that is:

\[ T_{ABC} = T_{AB} \cap T_C \]

In the vertical array, that is calculating the intersection of the corresponding items. What need to do is matching the new transaction codes, rather than matching each item of the transactions. Due to the order codes of the transactions, which calculating the intersections, you don’t need to recycle your scanning the original set, we can get the intersection by scan the two sets only once, this greatly reduces the time cost of the algorithm.

Reduce the time of comparisons when judging connection. According to Theorem 1 and its Corollary, before generating candidate K-item set Ck by self-join of Lk-1, we can calculate frequent item set \(|Lk-1(j)|\), that is the frequency of every item in Lk-1. If we have \(|Lk-1(j)| < k-1\), denoted by \(I^{'=} \{j||Lk-1(j)|< k-1\}\), then we can get a much smaller frequent (K-1) item set by deleting the items of Lk-1 which included in I’, we denote this smaller set by L’k-1.
Step Description of the Improved Algorithm. According to the saying all above, the steps of the improvement Apriori algorithm based on vector array and vertical array put forward in this paper can be described as follows:

Scan the original transaction data set once to generate the corresponding vector array, then generate the corresponding vertical array according to the vector array. Statistics the transactions that support each item by the vertical array (that is transactions below each item), collaborated with the vector array, we can get the support count of each transaction, then delete the items with support count less than the minimum support, frequent 1-item sets are obtained then.

Delete the transaction codes with transaction length shorter than 2 in the vertical array, then connect each two transactions in the frequent 1-item sets to generate the candidate frequent 2-item sets. By calculating the intersection of each item which support candidate frequent 2-item set from the vector array, we can get the transaction set that supports each item. By deleting the transactions with count less than the minimum support to generate the frequent 2-item set.

The rest can be done in the same manner, we can generate the candidate k-item set by the following steps: firstly, delete the transaction codes which a length(the corresponding Cnt value in the vector array) less than K in the vertical array according to Theorem 2; for the frequent (K-1)-item set, delete the items which can not be a frequent K-item set according to Theorem 1 and its Corollary; secondly, generate the candidate K-item set according to Definition 1 by connection of each two item; finally, generate the item sets supports each K-item set by calculating the intersection of each (K-1)-item set according to Theorem 3, delete the transactions with supports less than the minimum supports to get the frequent K-item sets.

Repeat the operation mentioned in 3), until no frequent item appears.

Experimental Results and Analysis

Data Set Used in the Experiment

The experiment uses the mushroom database provided by literature[7]. The database contains 8124 records which keep track of the 23 properties of a mushroom namely: hat shape, color, the shape of the neck, neck color, smell, living environment, toxic or not. Each property has 2 to 12 enumeration values. The attribute stalk-root has 2480 sample with a default value. Part of the database is shown in Fig. 1 below.

Here, I take the attribute 1-15 and 19-23 into consideration, divide the multiple-valued attributes to get 111 properties.
Experimental Results and Analysis

In order to compare the execution efficiency of the original algorithm and the improved algorithm, I conduct experiments based on the database mentioned before, and get a set of experimental results. The experiment environment: the operating system is Windows 7 Professional 64bit, CPU is the Intel (R) Core (TM) 2.4 GHz, internal storage is 4 GB. The Original algorithm and the improved algorithm are implemented using the Java programming language, the development environment is Eclipse Luna, the experimental data sets are stored in the local database MySQL 5.6. Some preprocessing procedure is taken on the original database firstly, the binary data (0 and 1) is used to indicate the presence or absence of a certain attribute, this operation improves the calculation speed.

The running time cost of the two algorithms are shown in Fig. 2 below. From the Fig. we know that the improved algorithm put forward in this paper needs much less time to analysis the association rules in the same data set under the same minimum support. This indicates the improved algorithm greatly increase the efficiency of the original algorithm, which proved the effectiveness of the core idea of this paper.

![Figure 2. running time cost under different support rate.](image)

Conclusions

By analyzing the advantages and disadvantages of the original Apriori algorithm based on array, considering several characters of the frequent item set, this paper put forward an improvement algorithm based on vector and vertical array. The improved algorithm modified the strategy of the join step and the prune step when candidate frequent(k+1)-item sets were generated from frequent(k)-item sets as well as the pattern matching strategy, this simplified complexity of the algorithm, this paper also put forward the steps of the improved algorithm. The Simulation experiment results show that the improved algorithm run faster than the original algorithm, the efficiency improved greatly.

References


