Advanced Apriori Algorithms

Ms.Aarti Patil, Ms. Seema Kolkur, Ms.Deepali Patil

Abstract— Association rule mining is an important field of knowledge discovery in database. The apriori algorithm is the classic algorithm in association rule mining. This paper compares the three apriori algorithms based on the parameters as size of the database, efficiency, speed and memory requirement.

Index Terms— Knowledge Discovery, Apriori Algorithm, ODAM,FARMA

1 INTRODUCTION

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. Association rule mining is defined as: Let \( I = \{i_1, i_2, \ldots, i_n\} \) be a set of \( n \) binary attributes called items. Let \( D = \{t_1, t_2, \ldots, t_m\} \) be a set of transactions called the database. Each transaction in \( D \) has a unique transaction ID and contains a subset of the items in \( I \). A rule is defined as an implication of the form \( X \implies Y \) where \( X, Y \subseteq I \) and \( X \cap Y = \emptyset \). An example rule for the supermarket could be \{butter, bread\} \implies \{milk\} meaning that if butter and bread are bought, customers also buys milk. To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence.

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent item sets in a database.
2. Second, these frequent item sets and the minimum confidence constraint are used to form rules [1].

There are number of algorithms used to generate association rules such as Apriori algorithm, Eclat algorithm, FP-growth algorithm.

Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions or having no timestamps.

As is common in association rule mining, given a set of itemsets (for instance, sets of retail transactions, each listing individual items purchased), the algorithm attempts to find frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

The purpose of the Apriori Algorithm is to find associations between different sets of data. It is sometimes referred to as "Market Basket Analysis". Each set of data has a number of items and is called a transaction. The output of Apriori is sets of rules that tell us how often items are contained in sets of data [6].

2 IMPROVED APRIORI ALGORITHM

In classical Apriori algorithm, when candidate itemsets are generated, the algorithm needs to test their occurrence frequencies. The manipulation with redundancy will result in high frequency in querying, so tremendous amount of resources will be expended in time or in space. Therefore the improved algorithm was proposed for mining the association rules in generating frequent k-item sets. Instead of judging whether these candidates are frequent item sets after generating new candidates, this new algorithm finds frequent item sets directly and removes the subset that is not frequent, based on the classical Apriori algorithm.

The improvement is for reducing query frequencies and storage resources. The improved Apriori algorithm mines frequent item sets without new candidate generation.

Improved Algorithm

The improved algorithm is described in following steps:

Input:
- \( D \), a database of transactions
- \( \text{Min\_sup} \), the minimum support count threshold

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemset \( C1 \). The algorithm simply scans all of the transaction to count the number of occurrences of each item.
2. The set of frequent item sets, \( L1 \), is determined by comparing the candidate count with minimum support count which contains candidate 1-itemsets satisfying minimum support.
3. To generate the set of frequent 2-itemsets, \( L2 \), the algorithm generate a candidate set of 2-itemsets and then the transactions in \( D \) are scanned and the support count of each candidate item set in \( C2 \) is accumulated and then...
3. To calculate the weight of each candidate itemset \( C_k \), this approach scans the array data structure and the items contained in \( C_k \) are accessed and their weight is obtained by summing the decimal equivalent of each item in the transaction.

4. Then calculate the support value for each item. To calculate the support value for each candidate itemset \( C_k \), this approach scans the array data structure and the items contained in \( C_k \) are accessed and the value of support is obtained by counting the number of decimal equivalent appeared in the transaction.

5. If a certain number of generations have not passed then repeat the process from the beginning otherwise generate the large itemsets by taking the union of all \( L_k \).

6. Once the large itemsets and their supports are determined, the rules can be discovered in a straightforward manner as follows:

   - If \( I \) is a large itemset, then for every subset \( a \) of \( I \), the ratio \( \text{support}(I) / \text{support}(a) \) is computed.

   - If the ratio is at least equal to the user specified minimum confidence, then the rule \( a \Rightarrow (1a) \) is output. Multiple iterations of the discovery algorithm are executed until at least \( N \) itemsets are discovered with the user specified minimum confidence, or until the user specified minimum support level is reached.

   - After finding all the itemsets using minimum support this algorithm uses Leverage measure introduced by Piatesky to filter the found item sets and to determine the interestingness of the rule. Leverage measures the difference of \( X \) and \( Y \) appearing together in the data set and what would be expected if \( X \) and \( Y \) were statistically dependent. The formula of leverage is as follows:

   \[
   \text{Leverage}(X \text{and} Y) = P(X \text{and} Y) - (P(X)P(Y))
   \]

4. **Optimized Distributed Association Rule Mining Algorithm**

The performance of Apriori association rule mining algorithm degrades for various reasons. It requires \( n \) number of database scans to generate frequent \( \{n\} \)-itemset.

It does not recognize transactions in the dataset with identical itemsets if that data set is not loaded into the main memory. Therefore, unnecessarily occupies resources for repeatedly generating itemsets from such identical transactions. For example, if a dataset has 10 identical transactions, the Apriori algorithm not only enumerates the same candidate itemsets 10 times but also updates the support counts for those candidate item sets 10 times for each iteration.

Directly loading a raw data set into the main memory won’t find a significant number of identical transactions because each transaction of a raw data set contains both frequent and infrequent items. To overcome these problems, we don’t generate candidate support counts from the raw data set after the first pass. This technique not only reduces the average transaction length but also reduces the data set size significantly.

ODAM eliminates all globally infrequent 1-itemsets from every transaction and inserts them into the main memory; it reduces the transaction size (the number of items) and finds
more identical transactions. This is because the data set initially contains both frequent and infrequent items. However, total transactions could exceed the main memory limit.

ODAM removes infrequent items and inserts each transaction into the main memory. While inserting the transactions, it checks whether they are already in memory. If yes, it increases that transaction’s counter by one. Otherwise, it inserts that transaction into the main memory with a count equal to one. Finally, it writes all main-memory entries for this partition into a temp file. This process continues for all other partitions [5].

5 COMPARATIVE STUDY

We have discussed different algorithms for association rule mining on different size of database. First we have seen the improved Apriori algorithm which takes less time for generating frequent item set. Second we have seen the Feature Based Association Rule Mining Algorithm which is efficient than other algorithms and it speeds up the data mining process. Third we have seen the Optimized Distributed Association Rule Mining Algorithm which works on distributed database. The comparative study of all these algorithms is given in tabular form as below:

Table 1: Comparative Study

<table>
<thead>
<tr>
<th>No.</th>
<th>PARAMETERS</th>
<th>IMPROVED APRIORI ALGORITHM</th>
<th>FARMA</th>
<th>ODAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Database Size</td>
<td>Small</td>
<td>Large</td>
<td>Very Large (Distributed)</td>
</tr>
<tr>
<td>2.</td>
<td>Database Scan</td>
<td>N times</td>
<td>At most Once</td>
<td>N times on different database server.</td>
</tr>
<tr>
<td>4.</td>
<td>Memory requirement</td>
<td>Large</td>
<td>Less</td>
<td>Less than FARMA</td>
</tr>
<tr>
<td>5.</td>
<td>Speed</td>
<td>Slow</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

6 CONCLUSION

Association Rule mining is one of the core data mining task. The Apriori algorithm is most representative algorithm for association mining. The classical Apriori algorithm has some disadvantages therefore in this paper we have studied different algorithms from which the Feature Based Association Rule Mining Algorithm works best for the large database and distributed database. Optimized Distributed Association Rule Mining Algorithm (ODAM) gives work properly.

REFERENCES