Optimization of Association Rule Mining Techniques Using Ant Colony Optimization

Babita Rani* and Shruti Aggarwal

SGGSWU, Fatehgarh Sahib, India

Accepted 20 November 2013, Available online 01 December 2013, Vol.3, No.5 (December 2013)

Abstract

Data Mining is used to discover the knowledge from large amount of databases and transform it into a flexible structure. Association rule mining (ARM) is the essential part of data mining process. Finding good quality of association rules between items in large databases has been an important and challenging association rule mining problem. The rules mined through ARM algorithms are used for decision making. The good quality of rules helps in better decision making. Optimization of apriori algorithm to generate strong association rules so that good qualities of rules are mined. Apriori algorithm is used to generate all significant association rules between items in the database. On the basis of Association Rule Mining and Apriori Algorithm, a new algorithm is proposed based on the Ant Colony Optimization algorithm to improve the result of association rule mining. Ant Colony Optimization (ACO) is a meta-heuristic approach and inspired by the real behaviour of ant colonies. First association rules generated by Apriori algorithm then find the rules from weakest set based on the threshold value and used the Ant Colony algorithm to reduce the association rules and discover the better quality of rules than apriori. The research work proposed focuses on reducing the scans of databases by optimization and improving the quality of rules generated for ACO.

Keywords: Data Mining, Association Rule Mining (ARM), Apriori Algorithm, Ant Colony Optimization(ACO), FP-Growth.

1. Introduction

In recent years, Data Mining (DM) has become one of the most valuable tools for extracting and manipulating data and for establishing patterns in order to produce useful information for decision-making (Sharma, A et al 2012). DM starts with the collection and storage of data in the data warehouse. DM is a process of discovering the useful knowledge from the large amount of data where the data can be stored in databases, data warehouses (A data warehouse is a “subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making). The data warehouse supports on-line analytical processing (OLAP), the functional and performance requirements of which are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported by the operational databases (Reddy, G et al 2010). DM also called the Knowledge Discovery in Database (KDD). KDD is used to extract the useful information from the large database or data warehouse. With DM techniques, it is possible to find relationship between diseases, effectiveness of treatments, identify new drugs etc.

*Corresponding author Babita Rani is a Research Scholar and Shruti Aggarwal is working as Asst. Prof.
frequent itemsets. It is the one of the most popular ARM algorithm. However, nowadays, the transaction datasets have become far larger than they were 10-15 years ago. The Apriori Algorithm now’s faces two problems in dealing with large datasets, first of all it requires multiple scan of transaction database, incurring a major time cost, in addition, it generates too many candidate sets which take up quite a lot of memory space. So there must be need to improve the Apriori Algorithm. Apriori Algorithm is improved by many of the researcher’s using with the different techniques. Improved Apriori Algorithm (Santhi, R et al 2012) (Dhanda, M et al 2011) (Singh, J et al 2013) reduced the scanning time by removing the unnecessary transactional records from the database and also reduced the redundant generation of sub-items during pruning the candidate itemset. Improved algorithm introduced an attribute Size of Transaction (SOT), containing number of items in individual transaction in database. Improved algorithm has optimized and efficient.

2. Association Rules

Relationships between the data called the associations. In general, association rule is an expression of X⇒Y form, where X is antecedent and Y is consequent. Association rule shows that how many times Y has occurred if X has already occurred depending on the support and confidence value. Many algorithms to generating association rules were presented over time. Some well known algorithms are Apriori and FP-Growth.

The problem of ARM stated as: Given a dataset of transactions, a threshold support (min_sup), and a threshold confidence (min_conf); Generate all association rules from the dataset (having the transactions records) that have support greater than or equal to min_sup and confidence greater than or equal to min_conf.

Association Rules will allow to find out rules of the type: If A then B where A and B can be particular items, values, words, etc. An association rule is composed of two item sets:

1. Antecedent or Left-Hand Side (LHS)
2. Consequent or Right-Hand Side (RHS)

There are two important basic interestingness measures for association rules, support(s) and confidence(c).

The rule A ⇒B holds in the transaction set D with:
Support: S, where s is the percentage of transactions in D that contain A ∪ B (i.e., both A and B). This is taken to be the probability, P (A ∪ B).
Confidence: C in the transaction set D if C is the percentage of transactions in D containing A that also contains B. This is taken to be conditional probability, P (B|A).

There are some commonly used terms that must be defined:
1) Itemset: An itemset is a set of items. A k-itemset is an itemset which contains k number of items.
2) Frequent itemset: This is an itemset that has minimum support.
3) Candidate set: This is the name given to a set of items that require testing to see if they fit a certain requirement. (Sharma, A et al 2012)
4) Strong rule and Weak rule: If support of the rule is greater or equal than Min_Sup and Confidence of the rule is greater or equal than Min_Conf, then association rule mark as strong rule, otherwise mark it as a weak rule.

Generally, an association rules mining algorithm contains the following steps:

1) The set of candidate k-itemsets is generated by 1-extensions of the large (k -1)-itemsets generated in the previous iteration.
2) Supports for the candidate k-itemsets are generated by a pass over the database.
3) Itemsets whose support is less than minimum support are pruned and the remaining itemsets are called large k-itemsets.

Discovering of all association rules can be decomposed into two sub-problems (Agrawal, R et al 1993):
1) Finds the frequent itemsets.
2) Frequent itemsets are used to generate the desired rules.

3. Apriori Algorithm

Apriori algorithm is the algorithm of Boolean association rules of mining frequent item sets; it was developed by R. Agrawal and R. Srikant in 1994. Apriori Algorithm employs the bottom up, level-wise search method, it include all the frequent itemsets (Yabing, J 2013):

1) Suppose a minimum support threshold (Min_sup) and a minimum confidence threshold (Min_conf).
2) Scan the dataset, generate the candidate 1-itemset C1 and the number of occurrences of each item is determined. Then generate the frequent 1-itemset L1 from the C1 by comparing candidate support count with minimum support count.
3) Generate the candidate 2-itemset C2 from L1 by multiply the L1 * L1. Scan the dataset again, and generate the frequent 2-itemset L2 from the C2 by comparing candidate support count with minimum support count.
4) Generate the candidate 3-itemset C3 from L2 by multiply the L2 * L2. Scan the dataset again, and generate the frequent 3-itemset L3 from the C3 by comparing candidate support count with minimum support count.
5) Repeatedly scan the dataset until no more frequent k-itemsets can be found. The finds for each L2 requires one full scan of the database. To improve the efficiency of level-wise generation of frequent itemsets, Apriori Algorithm has an important property called the Apriori property, presented is used to reduce the search space.

Apriori Property: All subset of a frequent itemset must also be frequent.
Apriori Algorithm is used the two-step process to find the frequent itemsets: join and prune actions:

a. Join Step: In this, join the itemsets with itself for generate the new itemsets. Itemsets are joinable if and only if there is one or more than one item is common.
b. Prune Step: If support of any itemset is less than minimum support then prune the items from the itemset and Apriori property is used for prune the items from the itemset.

4. Ant Colony Optimization

In the early 1990s, ant colony optimization (ACO) was introduced by M. Dorigo and colleagues (Barker, T et al 2005). The ACO is a meta-heuristic inspired by the behaviour of real ants in their search for the shortest paths to food sources. It looks for an optimal solution by considering both local heuristics and previous knowledge. How ants can find shortest paths from their nest to food sources?

When searching for food, initially each ant moves at random manner. While moving, each ant deposited a chemical pheromone trail on the ground. All Ants can smell pheromone. When ants choosing their way, they choose the paths marked by strong pheromone concentrations because more the pheromone trails better the path. As soon as ants find a food source, it evaluates the quality and the quantity of the food and carries some of it when ants back to their nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quality and quantity of the food. The pheromone trails must guide other ants to the food source (Agrawal, R et al 1993).

Example: 

![Image](image_url)

Figure 4.1: An experimental setting that demonstrates the shortest path finding capability of ant colonies. Between the ants’ nest and the only food source exist two paths of different lengths.

5. Methodology

In this paper the Ant Colony Optimization algorithm is applied over the rules fetched from Apriori Algorithm. The proposed method for generating better quality of association rules by Ant Colony Optimization is as follows:

1. Start
2. Load a sample of records from the database that fits in the memory.
3. Apply Apriori algorithm to find the frequent itemsets with the minimum support.
4. Frequent itemsets are divided into two categories, Candidate Generation and Probabilistic Generation.
5. Finding the better rules from the Probabilistic Generation based on Average Support, Average Confidence and Probabilistic Support Factor.
6. Apply the ACO Algorithm.
7. Results from various Scenarios will be compared and an analysis will be fetched in with comparison with other techniques.

6. Flow Chart of Proposed Work

![Flow Chart](flow_chart_url)

Figure 6.1: Flow Chart of Proposed Work.

7. Experimental Results

In order to evaluate the performance of the proposed (Ant Apriori) algorithm, Ant Apriori Algorithm program with Java 1.7 to realize the two algorithms. The data source of experiments is Breast-Cancer dataset, Contact-Lens dataset, whether dataset and Vote dataset obtained from UCI machine learning repository and know the comparative result of Apriori algorithm and the Ant Apriori algorithm. The Ant Apriori algorithm discovers
frequent itemsets and shows much greater efficiency than the Apriori algorithm. Rule set generation and optimization table with Apriori Algorithm and Using ACO.

7.1 Comparison using Number of rules generated

Experimental results for number of results generated against support and probabilistic factor has been shown graphically in figure 7.1, figure 7.3 and 7.5 for datasets Breast-Cancer; Contact-lens and weather respectively; vertical axis indicating number of rules generated, horizontal axis indicating Probabilistic support with different coloured bars indicating different values for support factor.

Comparison of apriori approach and proposed methodology has been indicated through graphical representation of number of association rules generated against support factor keeping the value of probabilistic factor at fixed value of 0.50 in figures 7.2, 7.4, 7.6 for different datasets respectively.

**Figure 7.1**: Generated rules against support and probabilistic factor for Breast-Cancer dataset.

**Figure 7.2**: Generated rules against support and probabilistic factor (=0.50) for Apriori and Ant Apriori Algorithm Approach for Breast-Cancer dataset.

**Figure 7.3**: Generated rules against support and threshold factor for Contact-lens dataset.

**Figure 7.4**: Generated rules against support and probabilistic factor (=0.50) for Apriori and Ant Apriori Algorithm for Contact-lens dataset.

**Figure 7.5**: Generated rules against support and probabilistic factor for weather dataset.

**Figure 7.6**: Generated rules against support and probabilistic factor (=0.50) for Apriori and Ant Apriori Algorithm for weather dataset.

It is clear from figure 7.1, 7.3 and 7.5 for all three datasets, rules generated using Apriori algorithm decrease as the support is increased. Similarly, for increase in the value of probabilistic factor, number of rules generated are decreased. From the figures 7.2, 7.4, 7.6, it can be concluded that rules generated using proposed methodology are less than the simple apriori algorithm which indicates that proposed algorithm is better than apriori approach.
7.2 Time and Number of Rules Comparison

Time comparison is done using run time (or execution time), which is the time to mine the frequent itemsets, against probabilistic factor and also done the number of comparison against probabilistic factor. Experimental results of number of rules comparison, time comparison have been shown graphically in figure 7.7 and figure 7.8. with red line indicating performance of the FP-Growth with ACO based algorithm and green line indicating performance of the Apriori with ACO based algorithm for dataset vote.

![Figure 7.7](image)

**Figure 7.7:** Generated number of rules against probabilistic factor for vote dataset.

![Figure 7.8](image)

**Figure 7.8** Time against probabilistic factor for vote dataset.

It can be concluded from figure 7.7 and figure 7.8 that time taken by Apriori and ACO based algorithm is less than the FP-Growth and ACO based algorithm so technique based on Apriori and ACO approach is better than FP-Growth and ACO approach.

Conclusion and Future Work

Methodology in this research study is based on the ACO algorithm for optimizing the association rules, generated through apriori algorithm. ACO is a meta-heuristic approach for solving hard combinatorial optimization problems. The good quality of rules helps in better decision making. On the basis of the association rule mining and Apriori algorithm, a new algorithm is proposed based on the Ant Colony Optimization algorithm to improve the result of association rule mining. Ant Colony Optimization optimized the result generated by Apriori Algorithm by introducing probabilistic scheme. In probabilistic section, set of good rules are found from the weakest set rules based on the support and confidence value. For this, rules are reduced and number of rules is compared with the probabilistic value. If the probabilistic value is increased, then the number of rules is decreased or vice versa. From this research work which compares the rules with the time factor it was found that if number of rules is decreased then time of work process is also decreased. Performance comparison results indicated that proposed methodology was better than the Apriori approach in rule generation approach. Comparative analysis also proves that apriori and ACO based approach is better than FP-Growth and ACO based approach as the time taken in processing of the algorithm on dataset using former approach is less than the later approach. As it can be seen that the proposed technique was found very useful from the existing techniques, but still there is a scope for improvement in the proposed approach to extend this approach to handle variety of situations and information. Proposed methodology which is implemented for single level association rules can be adopted for multi-level association rule mining. Time factor and number of rules can be reduced for FP-Growth and ACO approach. Further, comparisons based on factors other than Rule and Time-basis can be done such as memory, number of scans, size of dataset etc.

References


