Optimization of Association Rule Mining Using Genetic Algorithm

Umesh Kumar Patel
School of ICT, Gautam Buddha University,
Greater Noida, Gautam Budha Nagar(U.P.), India

ABSTRACT
Association rule mining is the most important technique in the field of data mining. Association rule mining finding the frequent patterns, associations among sets of items or objects in transaction databases, relational databases, and other information repositories. We formulate a general Association rule mining model for extracting useful information from very large dataset. The patterns that occur frequently in data are frequent patterns. Some frequent patterns are like frequent itemsets, sequential patterns and frequent substructures. A frequent itemset are those items that often appear together in a dataset. In this paper we design a method for generation of strong rule. In which Apriori algorithm is combine with FP growth algorithm to generate the rules. After that we use the optimization techniques to optimize the rule. Genetic algorithm is used to optimize the rules.

Keywords Apriori algorithm, Genetic algorithms, Association rule mining, FP growth.

1. INTRODUCTION
1.1 Data mining
Data mining is the technique of discovering interesting patterns and knowledge from large amounts of data. The sources of data can be databases, data warehouses, the Web, other information repositories. Data mining is an essential step in the process of knowledge discovery. The rapid development of computer technology, especially increased capacities and decreased cost of storage medium, has led businesses to store huge amounts of external and internal information in large databases with lower cost. Mining useful information and helpful knowledge from these large databases has thus evolved into an important research area.

1.2 Association rule
Association rule is a technique for discovering interesting relations between items in large databases. It is used to judge the strong rules that are discovered in databases using different measures of interestingness. Association rules are used for discovering relationship between products in large-scale transaction data in supermarkets. For example, the rule \{butter, bread\} \(\Rightarrow\) \{milk\} found in the sales data of a supermarket would indicate that if a customer buys butter and bread together, they are likely to also buy milk. Such information can be used as the basis for decisions about strategy in marketing such as, e.g., promotional pricing or product placements.

Association rule is defined as:
Let \(I=\{i_1,i_2,i_3, \ldots \ldots, i_n\}\) be a set of \(n\) items called attributes.
Let \(D=\{t_1,t_2,t_3, \ldots \ldots, t_m\}\) be a set of transactions called database.

Each transaction in database \(D\) has a specific identifier \(ID\) and contains a subset of the items in \(I\).
A rule is defined as an implication of the form:
\[X \Rightarrow Y\] Where \(X,Y \subseteq I\) and \(X \cap Y = \emptyset\).

Every rule is composed by two different set of items, also known as item sets, \(X\) and \(Y\). Where \(X\) is called antecedent or left-hand-side (LHS) and \(Y\) consequent or right-hand-side (RHS).

1.3 Apriori algorithm
Apriori algorithm is used for mining the frequent itemset and association rule. The algorithm use a level-wise search to find frequent itemsets from transactional database for Boolean association rules. In this \(k\)-itemsets are used to explore \((k+1)\) itemsets. An itemset which contains \(k\) items is known as \(k\)-itemset. In this algorithm,
frequent subsets are extended one item at a time from previous set and this step is known as candidate generation process. Then groups of candidates are evaluated against the data. To count candidate itemsets accurately, Apriori uses breadth-first search method and a hash tree data structure.

It identifies the frequent individual items in the database and extends them to larger and larger itemsets as long as those itemsets appear in the database. Apriori algorithm find frequent item sets that are used to determine association rules which highlight general trends in the database.

1.4 Genetic algorithms
A genetic algorithm is a searching algorithm. It search a solution space for an optimal solution to a problem. The algorithm produce a “population” of feasible solutions to the problem and lets them “evolve” over multiple generations to find better solution. Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are capture and employ to form a new population. Cycle of the Algorithm: The algorithm operates through a cycle formation of a population of strings.

- Evolution of each string.
- Selection of the perfect string.
- Genetic manipulation to form a new population of strings.

1.5 FP growth algorithm
Frequent pattern growth, or FP-growth, which adopts a divide-and-conquer strategy. First, it compresses the database representing frequent items into a frequent pattern tree, known as FP-tree, which retains the itemset association information. It then split up the compressed database into a set of conditional databases, each associated with one frequent item or “pattern” and extracting each D separately. Over each “pattern fragment,” only its associated data sets need to be examined. Therefore, this approach may significantly reduce the size of the data sets to be searched, along with the “growth” of patterns being examined.

2. METHODOLOGY
Hybrid algorithm
The proposed algorithm is a hybrid of FP Tree creation algorithm i.e. FP Tree and Apriori algorithm. This proposed algorithm can be explained into two phases.

The first phase constructs the FP Tree and second phase involves mining the FP Tree created using the hybrid(FP+Apriori) algorithm.

Phase 1:- Tree construction using FP Tree Algorithm.
Phase 2:- Tree Mining using hybrid (FP+Apriori) Algorithm.

2.1 FP tree algorithm:-
The FP-tree is constructed in the following steps:-

(a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the list of frequent items.

(b) Create the root of an FP-tree, and label it as “null.” For each transaction ‘Trans’ in D do the following. Select and sort the frequent items in ‘Trans’ according to the order of L. Let the sorted frequent item list in ‘Trans’ be [p[P]], where p is the first element and P is the remaining list. Call insert_tree([p[P],T), which is performed as follows. If T has a child N such that N.item-name=p.item-name, then increment N’s count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert_tree(P,N) recursively. 2.2 Candidate set algorithm :- getCandidate(k)
In order to perform the mining by hybrid (FP+Apriori) algorithm firstly the candidates are generated using a candidate set algorithm.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>List1=k-1 frequent item dataset from Data</td>
</tr>
<tr>
<td>2</td>
<td>n=size(List1) Initialize mylist as a blank list to contain generated frequent item dataset</td>
</tr>
</tbody>
</table>
Step 3  Repeat for i=1 to n-1
Step 4  Repeat for j=i+1 to n
Step 5  l1=list1[i]
Step 6  l2=list1[j]
Step 7  Remove the last elements from l1 and l2
Step 8  if l2 is a subset of l1 then
       Flist=append last element of l2 at end of l1
Step 9  If count(Flist)=supp then
        Add Flist to mylist
        Else
        return null
Step 10 end

2.3 Hybrid (FP+Apriori) algorithm
Input: Nodes of FP Tree as a “list2” and Sup: minimum support
Output: frequent itemsets “frequent_data”
The algorithm is implemented with the following steps:-
Step 1  Repeat following step while scanning list2 till end
       • Create list containing single item from list2
       • Add this list to mylist1
      [End of repeat]
Step 2  Add mylist1 to data
Step 3  Repeat for k=2,3,4,….
Step 4  C_k=getCandidate(k)
Step 5  If [C_k] =0 then
        Goto step 6
        Else
        Add C_k to frequent_data
Step 6  End

After obtaining the frequent itemsets, the association rules can be generated as follows
•   For each frequent itemset l, generate all nonempty subsets of l.
•   For every nonempty subset s of l, output the rule s⇒(l-s)  if
(support _count(l)/support_count(s)) ≥ min_conf, where min_conf is the minimum confidence threshold.

After it we apply the hybrid(FP+Apriori) algorithm with GA to optimize the Association rule.

3. PROPOSED FLOW CHART

![Flow Chart for Hybrid Algorithm](image-url)
4. EXPERIMENT AND RESULTS

4.1 Experiment

The experiment is done on Abalone dataset obtained from UCI machine learning repository. The data set has 4177 samples. It is composed of one nominal attribute and 7 continuous attributes and one integer attributes. The setting of parameter: the size of population N=500, maximum generation=100, crossover rate=0.006, mutation rate=0.001. The experiment was executed using software MATLAB 8.1.0.604(R2013a), Microsoft Windows XP Professional 2002 operating system.

4.2 Results

![Figure 2: Comparison between Apriori with GA and Apriori + FP with GA for the generation and fitness](image)

Figure 2 show the value of fitness based on the number of generations. The number of maximum generation in above graph is 100. Hybrid(Apriori+ FP) with GA algorithm give the approximately 4% better result in comparison to the Apriori algorithm with GA. There is a definite improvement by using hybrid algorithm comprising of Apriori along with FP and Genetic algorithm.

![Figure 3: Comparison between Apriori with GA and Apriori + FP with GA for the Rule generation](image)

Figure 3: Comparison between Apriori with GA and Apriori + FP with GA for the Rule generation
Figure 3 show the number of rules generated. The number of rules generated in hybrid (Apriori+ FP) with GA algorithm is approximately 60% less than the number of rules generated in Apriori algorithm with GA. Thus the proposed hybrid (Apriori+ FP) with GA algorithm gives the better results.

The below table represent the rule generation of Apriori with GA and Apriori + FP with GA for particular support. It is seen that number of rule generation discusses with hybrid algorithm.

Table 1 result of rule generation

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Min support</th>
<th>Apriori with GA</th>
<th>Apriori + FP with GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>76</td>
<td>42</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>77</td>
<td>43</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>79</td>
<td>44</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>83</td>
<td>48</td>
</tr>
<tr>
<td>E</td>
<td>9</td>
<td>84</td>
<td>49</td>
</tr>
<tr>
<td>F</td>
<td>10</td>
<td>84</td>
<td>49</td>
</tr>
<tr>
<td>G</td>
<td>10</td>
<td>84</td>
<td>50</td>
</tr>
<tr>
<td>H</td>
<td>10</td>
<td>84</td>
<td>50</td>
</tr>
</tbody>
</table>

Fig 4: Comparison of rules in Apriori with GA and Apriori + FP with GA with respect to various attributes

Fig 4 show the number of rules generated in Apriori with GA algorithm and in hybrid (Apriori + FP) with GA algorithm for various attributes. The number of rules generated in hybrid (Apriori + FP) with GA algorithm is less than the number of rules generated in Apriori with GA algorithm for each attributes. Thus the proposed hybrid (Apriori + FP) with GA algorithm gives the better results.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented the implementation of hybrid of Apriori and FP growth with GA for association rule mining on Abalone dataset. A data mining system has the ability to generate the thousands or more rules. So all the rules are not interesting, only small fraction of the rules would be interest to user. Generate only interesting rules is the optimization problem in data mining. In this study, to generate the association rule we have implemented Apriori algorithm with FP growth algorithm. We have used Genetic algorithm to optimize
the generated rules, we take fitness function for the optimization and find the optimum solutions that are interesting rules. We compared these two implementations on the generation and fitness they required. To improve the efficiency and accuracy of the optimization there is need of incorporating more measures in simple genetic algorithm for such data mining problems.

REFERENCES