CHAPTER 6

EXACT RARE ASSOCIATION RULE MINING

This chapter deals with the mining of rare association rules with 100% confidence called exact association rules. An algorithm is proposed to extract exact association rules.

6.1 BASIC CONCEPTS

Nowadays, the mining of rare association rules with full confidence is an interesting research topic. In order to mine the exact rare association rules, the support threshold is to be too low. A low support threshold leads to the generation of too many itemsets sometimes uninteresting itemsets whereas a high support threshold misses some rare itemsets. The real world datasets consists of items that are of non uniform in nature. Some items appear frequently in transactions and some of them appear rarely. The rare itemsets also may be interesting. A rare itemset is one consisting of rare items. It may be found by setting a low support threshold but leads to combinatorial explosion problem. But it is difficult to mine rare association rules using single support threshold based approaches like Apriori and Frequent Pattern-Growth (FP-Growth). The problem of specifying an appropriate support threshold causes rare item problem. The rare association rules are classified into two types: Exact Rules and Approximate Rules.

Definition 6.1: Exact Rules – An association rule is said to be an Exact Rule if it has low support but the items contained in it are highly correlated or the confidence of the rule is 100%. For example in Table 3.1, the itemset
{bread,jam} occurs only in 30% of the transactions but for the rule jam ⇒ bread, its confidence is 100%.

**Definition 6.2:** Approximate Rules – An association rule is said to be Approximate Rule if it has low support and low confidence. In the given transactions in Table 3.1, the itemset {milk, sugar} occurs with 30% support but the rule milk ⇒ sugar occurs with 17% confidence.

To deal with the rare association problem, a multiple minimum support framework has been adopted. This framework assigns each item in the transaction dataset a minsup called Minimum Item Support (MIS). An association is defined as interesting association if its support greater than or equal to the minimum MIS values among all its items. To extract rare itemsets, the MIS values of the itemsets are specified low. The MIS values are set relative to their actual support values, since the support of rare item is usually low. In order to find interesting association which contains rare items, the MIS value of the item should be relatively low. But, it is difficult to find an appropriate MIS value. Hence the model based on MIS is also not suitable to mine interesting rare associations specifically in datasets with widely varying item supports. It finds the rare associations according to the specified MIS value. It still misses some rare associations with 100% confidence. This chapter proposes a framework which is specially designed for the extraction of all exact rules that is the rules with 100% confidence.

### 6.2 EXACT ASSOCIATION RULE MINING FRAMEWORK

In this method, to deal with the rare association problem, a multiple minimum support framework has been designed. A framework has been designed for the extraction of all exact rules that is the rules with 100% confidence. Usually the association rule mining algorithm consists of two steps: Finding the frequent itemsets and extracting the interesting association
rules. This chapter makes a combined effort to extract the interesting rare association rules and interesting frequent association rules in a single step.

The proposed model is known an Exact_AR (Exact Association Rule) algorithm. It adopts MSApriori algorithm (Liu et al 1999) for implementation. It attempts to discover interesting associations involving rare items of 100% confidence. Each level the MIS values are derived from the support of its subsets. This algorithm assigns MIS value for each itemset and an itemset is considered to be interesting if the support and the MIS value of the itemset are equal. If an itemset is found to be interesting, the rule is immediately generated. The itemset which constitutes LHS is determined from the support of the subsets of the interesting itemset.

All items in the first level is considered to be interesting and moved to the next level. In subsequent levels, the MIS value for each itemset is computed as the minimum support among the subset of items contained in the itemset. 2-itemset candidates are generated and support for each candidate is calculated. Itemsets with zero support are eliminated.

Given an itemset(X, Y), the MIS value at the second level is calculated as follows:

\[ \text{MIS}(X, Y) = \min (\text{sup}(X), \text{sup}(Y)) \]

The item which has the lowest support in the itemset is recorded as LHS of the rule. Since the confidence value is not symmetric, the LHS part of the rule has to be determined. The rules \( X \Rightarrow Y \) and \( Y \Rightarrow X \) are not equal and the confidences for both rules are not equal. Third level candidates are generated from the FI which are found in the second level.

The MIS for 3-itemset is calculated from the support of its subsets in the previous level as shown below:
MIS(X, Y, Z) = min (sup(X, Y), sup(X, Z), sup(Y, Z))

The LHS part of the rule is the subset of the itemset which has minimum support among the subsets.

Usually, the itemwise support based approaches applies the sorted downward closure property for frequent itemset mining. According to this property, the items will be sorted in ascending order of their support. This property is not applicable here. If the items are sorted, then minimum of any item two items X, Y will be always X. That is the first item in comparison. Hence, the sorted downward closure property should not be used.

In order to determine the MIS value of an itemset, the algorithm retrieves all of its subsets and compares their supports. Hence, the algorithm incurs poor response time and memory access problems. To overcome these drawbacks, the algorithm is implemented with a double hashing data structure. One hash tree is used as a permanent structure (i.e. till the end of the program execution) and stores the candidate itemsets. Another hash is used as a temporary structure and it is used like a cache. That is, each time the second hash tree stores the previous level supports of items and it is overwritten during the next level. This makes the execution of the algorithm in an effective way.

6.3 EXACT_AR – AN EXAMPLE

The execution of the proposed algorithm on dataset D is depicted in Table 6.1. Initially the dataset is scanned to calculate the support of the first level items. During the first level, the MIS value for each item is its support. 1- itemset is generated.

According to the example given in Table 3.1, the MIS values for the itemset {bread, milk, egg, sugar, jam} are {7, 6, 5, 3, 3}. All items in the first
level is considered to be interesting and moved to the next level. In subsequent levels, the MIS value for each itemset is computed as the minimum support among the subsetset of items contained in the itemset. 2-itemset candidates are generated and support for each candidate is calculated. Itemsets with zero support are eliminated. Then the MIS value at the second level is calculated and the whole process is illustrated in Table 6.1.

**Table 6.1 Execution of Exact_AR algorithm on Dataset D**

<table>
<thead>
<tr>
<th>Item</th>
<th>sup</th>
<th>Itemset</th>
<th>sup</th>
<th>MIS</th>
<th>Itemset</th>
<th>sup</th>
<th>MIS</th>
<th>LHS</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>7</td>
<td>Bread, milk</td>
<td>4</td>
<td>6</td>
<td>Bread, jam</td>
<td>3</td>
<td>3</td>
<td>Jam</td>
<td>jam ⇒ bread</td>
</tr>
<tr>
<td>Milk</td>
<td>6</td>
<td>Bread, egg</td>
<td>3</td>
<td>5</td>
<td>Milk, sugar</td>
<td>3</td>
<td>3</td>
<td>Sugar</td>
<td>sugar ⇒ milk</td>
</tr>
<tr>
<td>Egg</td>
<td>5</td>
<td>Bread, jam</td>
<td>3</td>
<td>3</td>
<td>Milk, egg</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jam</td>
<td>3</td>
<td>Bread, sugar</td>
<td>1</td>
<td>3</td>
<td>Milk, jam</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sugar</td>
<td>3</td>
<td>Milk, egg</td>
<td>4</td>
<td>5</td>
<td>Milk, sugar</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Egg, sugar</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Itemset</td>
<td>sup</td>
<td>MIS</td>
<td>LHS</td>
<td>Rules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bread, milk, jam</td>
<td>1</td>
<td>1</td>
<td>Milk, jam</td>
<td>milk, jam ⇒ bread</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milk, egg, sugar</td>
<td>2</td>
<td>2</td>
<td>Egg, sugar</td>
<td>egg, sugar ⇒ milk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milk, bread, sugar</td>
<td>1</td>
<td>1</td>
<td>Bread, sugar</td>
<td>bread, sugar ⇒ milk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milk, bread, sugar, egg</td>
<td>1</td>
<td>1</td>
<td>Milk, bread, sugar</td>
<td>milk, bread, sugar ⇒ egg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, the MIS value for \{bread, jam\} can be calculated as $\text{MIS}\{\text{bread, jam}\} = \min(\text{sup}(\text{bread}), \text{sup}(\text{jam})) = \min(7,3) = 3$. A pattern is known to be interesting if its support is equal to MIS value. Here, the pattern \{bread, jam\} is interesting because $\text{sup}(\text{bread, jam}) = \text{MIS}(\text{bread, jam}) = 3$. 
Likewise \{\text{milk, sugar}\} is also an interesting pattern. Patterns which are not found to be interesting are pruned.

Now, the association rule can be formed by finding the left hand side (LHS) of the pattern. The item which has the lowest support in the itemset is recorded as LHS of the rule. In the example LHS of the pattern \{\text{bread, jam}\} is jam because it has the lowest support than bread. Since the confidence value is not symmetric, the LHS part of the rule has to be determined. The rules \text{X} \Rightarrow \text{Y} and \text{Y} \Rightarrow \text{X} are not equal and the confidences for both rules are not equal. Third level candidates are generated from the FI (L) found in the second level. Suppose in the set of items I, if the itemset \{\text{X,Y}\} is found to be interesting then the candidates are generated by considering items \text{X}, \text{Y} \cup \{I-Z\} where \text{Z} \notin \text{L} and \text{X,Y,Z} \in \text{I}. Itemsets with zero support are pruned. The MIS for 3-itemset is calculated from the support of its subsets in the previous level as shown below:

Consider \{\text{bread, milk, jam}\} is a 3-itemset candidate. 
\text{MIS(bread,milk,jam)}=\text{min(sup(bread,milk),sup(bread,jam),sup(milk,jam))} = \text{min(4,3,1)}=1. The support of the itemset is also equal to 1. So, the itemset is considered to be interesting. The LHS part of the rule is the subset of the itemset which has minimum support among the subsets. In this case, \{\text{milk, jam}\} and it forms the LHS of the rule.

6.4 EXACT_AR ALGORITHM

EXACT_AR Algorithm

Variables

D – Dataset

L – Frequent itemsets

C – Candidate itemset

R – Exact Rules
MIS – Minimum Item Support

LHS – Left Hand Side

**Input**

Transaction dataset D

**Output**

L  Frequent itemsets

R  Exact rules

**Exact_AR**

L_1= find frequent 1 itemsets(D)

Calculate support of I for all I ∈ L_1

Assign support(I) to MIS(I)

for (k = 2; L_{k-1} ≠ ∅, k++) do

C_k=candidate-gen (L_{k-1})

end

for each transaction t ∈ D do

C_t= subset (C_k, t);

for each candidate c ∈ C_t do c.count++;

if c.count =0 then delete c;

MIS(c)= c.count | c.count is minimum for all C_{k-1}

L_k = {c ∈ C_k | c.count = MIS(c)}

If c.count=MIS(c) then

LHS = c | c.count is minimum for all C_{k-1}

R_k =Form_Rule(LHS,c)

Return \bigcup R_k;

return \bigcup L_k;

end
Procedure Candidate-gen(L_{k-1})
for each itemset l_1 \in L_{k-1}
    for each itemset l_2 \in L_{k-1}
        perform join operation l_1 and l_2
        if has_infrequent_subset(c, L_{k-1})
            prune c;
        else
            add c to C_k;
        end if
    end
end
return C_k;

Procedure has_infrequent_subset(c, L_{k-1})
for each (k-1) subset s of c
    if s is in L_{k-1}
        return false;
    else
        return true;
    end if
end

Procedure Form_Rule(LHS,c)
RHS=c minus LHS
Write rule in the format LHS \Rightarrow RHS
end

The FI and rules are generated until there are no more frequent patterns.
6.5 EXPERIMENTAL RESULTS

The performance of the proposed algorithm is compared with Apriori_Rare in terms of the number of frequent itemsets and the number of rules. Since the Apriori_Rare algorithm generates frequent itemsets and rare itemsets like the proposed algorithm, this algorithm is taken for comparison.

The Apriori_Rare algorithm runs each time with different user specified minsup thresholds. For each dataset some optimum support thresholds have been picked up and run. The proposed algorithm is an automated algorithm which does not require any minsup values rather it calculates itself and run. The results are illustrated in Table 6.2.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Apriori-Rare</th>
<th>Exact_AR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minsup</td>
<td>#FIs</td>
</tr>
<tr>
<td>T20I6D100K</td>
<td>10%</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>0.75%</td>
<td>4,710</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>26,836</td>
</tr>
<tr>
<td></td>
<td>0.25%</td>
<td>155,163</td>
</tr>
<tr>
<td>C20D10K</td>
<td>30%</td>
<td>5,319</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>20,239</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>89,883</td>
</tr>
<tr>
<td></td>
<td>5%</td>
<td>352,611</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>1,741,883</td>
</tr>
<tr>
<td>MUSHROOMS</td>
<td>40%</td>
<td>505</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>2,587</td>
</tr>
<tr>
<td></td>
<td>15%</td>
<td>99,079</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>600,817</td>
</tr>
</tbody>
</table>

Table 6.2 shows the number of frequent itemsets (#FIs) extracted from each dataset. The FIs generated by the proposed algorithm is equal to the itemsets obtained by Apriori-Rare when the support threshold is slightly less than 0.25% in the case of T20I6D100K. For C20D10K dataset the number of FIs generated by the proposed algorithm is achieved by Apriori-Rare only
when the minsup value is less than 10%. In case of Mushrooms dataset the proposed algorithm matches with Apriori_Rare when the minsup value is set greater than 10% level.

Table 6.3 Exact Rules for Apriori_Rare and Exact_AR

<table>
<thead>
<tr>
<th>Dataset</th>
<th>APRIORI-RARE</th>
<th>EXACT_AR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minsup</td>
<td>#Rules (minconf=100%)</td>
</tr>
<tr>
<td>T20I6D100K</td>
<td>0.25%</td>
<td>8746</td>
</tr>
<tr>
<td>C20D10K</td>
<td>10%</td>
<td>377564</td>
</tr>
<tr>
<td>MUSHROOMS</td>
<td>5%</td>
<td>23564</td>
</tr>
</tbody>
</table>

Table 6.3 shows the number of rules generated by Apriori_Rare and the rules generated by the proposed algorithm. Rules are generated using Apriori-Rare by setting an optimum minsup and minconf thresholds for each dataset. The proposed algorithm assigns the item support automatically and generates rules with 100% confidence. The proposed approach generates the rules which are missed by Apriori-Rare. This shows that the proposed algorithm generates the 100% confidence rules involving rare items in an effective way.

This study presented an approach for finding both frequent and rare itemset mining together. It uses automated itemwise support thresholds for mining. This leaves the user free from finding an appropriate threshold for each dataset. These thresholds are automatically calculated and used by the algorithm. Experimental results show that the algorithm produces the most frequent items and the rare interesting items of 100% confidence rules with the use of automated support thresholds.