Open access Journal International Journal of Emerging Trends in Science and Technology

Research of Improved Association Rule Algorithms (Apriori and FP-Growth)

Authors

Nitika, Anju Gandhi
Kurukshetra University, Kurukshetra
Email: *Nitikkaa_singh@yahoo.com*

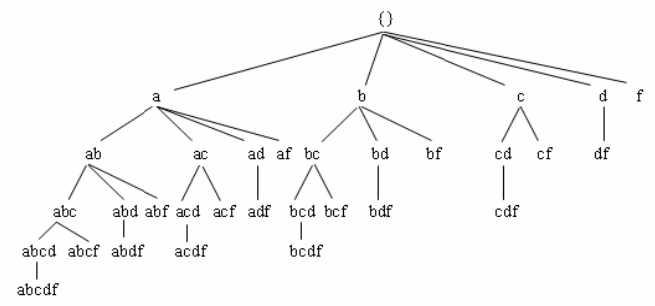
Abstract: *- Association rule mining explores interesting relationships among items in a given data set. An objective of association rule mining is to develop a systematic method using the given database and finds relationships between the different items. Goal of association rules finding associations among items from a set of transactions, which contain a set of items. In this paper we focused on explaining the fundamentals of association mining and analyze implementations of the well-known association rule algorithms. Study focuses on algorithms Apriori, FP-Growth, and Dynamic Itemset Counting. Moreover, the algorithm generates frequent item sets in order so that the result can be used expediently.*
Key Terms *– Association Rule Mining, FP-Growth Algorithm, Apriori algorithm, Frequent Itemsets.*

INTRODUCTION

Association rule mining is the most commonly used data mining operation. It uses generation of association rules based on the given set of data. These rules are defined using a stochastic approach, based on probabilities drawn up on the various attributes of the database. An association rule is a rule, which implies certain association relationships among a set of objects (such as ``occur together'' or ``one implies the other'') in a database. Given a set of transactions, where each transaction is a set of literals (called items), an association rule is an expression of the form " X => Y has support s and confidence c", where "X" and "Y" are two attributes or sets of attributes (called itemsets) in the given database.

I. Organization of ItemsetS In Association Rule Mining

Transactions are needed to be count to find the frequent itemsets. Generally, tree structure is used to organize the itemsets. Aim of the itemset tree is storing itemsets efficiently and supporting the process of transaction. Some of the association rule algorithms use different data structures which may be used for the nodes of this tree, and the other use different pruning techniques for generating the frequent-itemsets.



II. Frequent Itemsets Algorithms

The problem of mining association rules was first proposed by. Agrawal has introduced one of the most popular association rule algorithms called Apriori to find frequent itemsets. Apriori and all of the other algorithms based on the principles of knowledge of frequent itemsets "all non-empty subsets of a frequent itemset must also be frequent".

IMPROVED APRIORI ALGORITHM

Apriori, [1] the Apriori algorithm is the most popular association rule algorithm. Apriori have used bottom-up search. Using multiple database scan would have been significantly slower. Apriori algorithm works as follows:

1. The **first,** Apriori algorithm counts occurrences for the each item in the entire database and generates the set *C1* of candidate 1 – itemsets. Then, itemsets count and minimum support value are compared to find the set *L1*.
2. The **second,** algorithm use *L1* to construct the set *C2* of candidate 2 – itemsets. The process is finished when there are no more candidates. In each phase, all the transaction in the database is scanned. Finally, all frequent itemsets are returned.

New algorithm to mine association rules is a combination of Apriori, which is the most popular mining algorithm, and set operations. Principles of set operations, which are intersection and union, are used. These principles are related to lattice tree. In lattice tree, there are nodes holding frequent itemsets and transactions containing related itemsets. In order to construct (n+1)-itemsets, frequently n-itemsets are used. Hence, intersection operation is employed between the transaction sets.

If the result is greater than minimum support, it will be joined to lattice tree. If the result is lower than minimum support, it will be pruned off.

The main idea of Algorithm is outlined as follows;

Lk : Frequent itemset of size k

Ck : Candidate itemset of size k

L1 = frequent 1- items;

Generate frequent 2- itemsets using lattice Tree

k=2;

For each frequent k- itemset nodes in lattice

Ck+1 = new node; // Generated from *Lk*

For each Xi in Nodes

For each Xj in Nodes //where j>i

Ck+1 =T(Xi) ∩ T(Xj)

If *Ck+1* > min_support **then** {// Add *Ck+1* in lattice

(Node.item = Xi U □□Xj)

(Node.transaction =T(Xi) ∩ T(Xj)) }

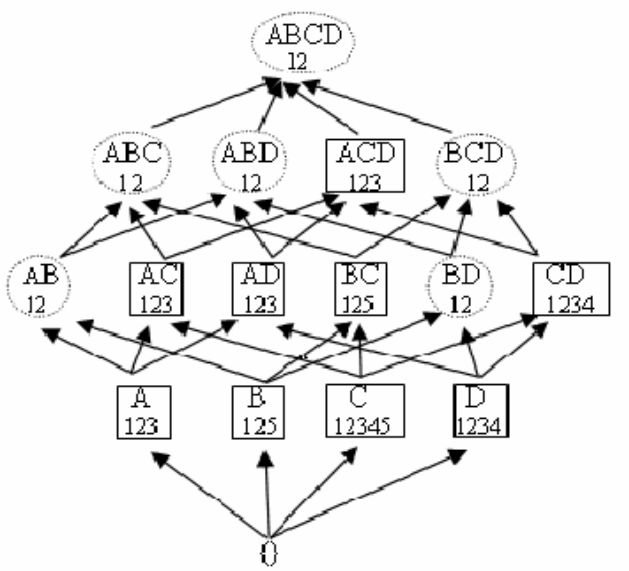
k++; End

This algorithms work as follows;

1. Scan database and find frequent 1-itemsets, at the same time obtain transaction sets, which includes the itemsets.
2. Construct Lattice tree, in order to generate all frequent 2-itemsets.
3. During the second step, prune off the nodes whose node count is lower than minimum support.
4. Find frequent itemsets by using lattice tree. Consequently, for each frequent 3, 4 …, n–itemset, scan the database to approve the consistence of the itemsets.
5. Finally, itemsets are used to generate strong rules having minimum confidence in the lattice tree.

T_id	Items in Transaction	Frequent items
1	A, B, C, D,G	D, A, B, C
2	A, B, D, F, C	D, A, B, C
3	A, C, D	D, A, C
4	C, D, F, K, L	D, C
5	B, C, L	B, C

Transaction DB3



Lattice Tree for transaction-DB3

Figure shows how this algorithm works in sample database in table (transaction-DB3). If the intersection result of the frequent nodes exceeds the support thresholds, mark the node with a square. Otherwise, mark node with a dotted circle. In this example, firstly, find frequent 1-itemsets. Next, use intersection operation to find frequently 2-itemset.

$$\left.\begin{array}{l}\{A\} \text{ in } t(123)\\ \{C\} \text{ in } t(12345)\end{array}\right\} \Rightarrow \{A\} \cup \{C\} = t(A) \cap t(C) \Rightarrow \{AC\} = 123 \cap 12345 = 123$$

Intersection principles are used to find frequent itemsets, then, for each frequent 3, 4 …, n – itemset, The scanning of the database to approve the consistence of the itemsets.

FP – Growth Algorithm

Given a transaction database and a minimum support threshold, the problem of finding the complete set of frequent patterns is called the ***frequent pattern-mining*** problem. Fp – tree is a compact data structure. *FP-Growth* is an algorithm for generating frequent itemsets for association rules. This algorithm compresses a large database into a compact, frequent–pattern–tree (FP tree) structure. Fp – tree structure stores all necessary information about frequent itemsets in a database.

Definition (FP-tree): A frequent pattern tree (or FP-tree in short) is defined below,

1. The root labeled with "null" and set of items as the children o the root. Frequent item header table, which contains items in their frequency descending order.

2. Each node contains of three fields: item name (holds the frequent item), count (number of transactions that share that node), and node- link (next node in the FP tree).

3. Frequent-item header table contains two fields, item name and head of node link (points to the first node in the FP-tree holding the item-name).

FP- Growth method only needs two database scans when mining all frequent itemsets. In the first scan, all frequent items are found. Next scan, constructs the first FP – tree that contains all frequency information of the given database. Fp – tree use compact data structure based on the following properties:

- Frequent pattern generation mining performs one scan of Database to determine the set of frequent items.
- Method needs to store each item in a compact structure, thus, more than two databases scan unnecessary.

- Each frequent item located in the FP – tree and each node hold items and count of the frequent item.
- Each item has to be sorted in their frequency descending order. So, tree construction operation performs easily.
-

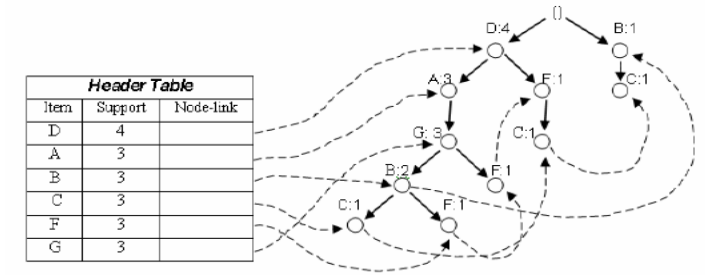
```
Algorithm

Procedure FP-growth (Tree, σ)
{
if Tree contains a single path P
then for each combination (denoted as β)
     of the nodes in the path P do
      generate pattern β ∪ σ with
               support =minimum support of nodes in β
else for each a_i in the header of Tree do
{
      generate pattern β = a_i ∪ σ  with
          support = a_i.support;

     construct β 's conditional pattern base and
     then β 's conditional FP-tree Tree_β;
     if Tree_β =∅ ;
          then call FP-growth (Tree_β, β)
}
}
```

In following example the database transaction-DB2 in table is used and constructs FP-tree in using the construction steps.

the transaction database transaction-DB2

T_id	Items in transaction	Ordered Frequent items	
1	A, B, C, D, H, G	D, A, G, B, C	Min_support 50 %
2	A, B, D, F, G	D, A, G, B, F	
3	A, D, G, F	D, A, G, F	
4	C, D, F, K, L	D, F, C	
5	B, C, L	B, C	

FP-Tree



Details of this example as follows:

First, frequent pattern generation mining perform one scan of Database to determine the set of frequent items and itemset count and all infrequent items are removed from the header table. Items are reordered according to count descending order. Then, create the root of a tree labeled with "null". Second, scan the all transactions and construct the FP – Tree. The first transaction form the first branch the tree {(D: 1), (A: 1), (G: 1), (B: 1), (C: 1)}. For the second transaction, (D, A, G, B, F) ordered frequent item list shares a common prefix (D, A, G, B) with the existing path (D, A, G, B, C), the count of each node which share the same item is incremented by 1, and one new node (C:1) is created. Linked with node of (B:2). For the third transaction (D, A, G, F) ordered frequent item list shares a common prefix (D, A, G) with the existing path. The count of each node which share the same item is incremented by 1, and one new node (F:1) is created. Linked with node of (G:3). The fourth transaction (D, F, C) ordered frequent item list shares a common prefix (D) with the existing path. The count of the node (D) which share the same item is incremented by 1, and (F: 1) is created and linked as a child of (D: 4) and (C: 1) is created and linked as a child of (F:1). The last transaction (B, C) form to the construction of the second branch of the tree. (B: 1) is created and linked as a child of the root and (C: 1) is created and linked as a child of (B:1).

REFRENCES

1. R. Agrawal, T. Imielinski, and A. Swami. “Mining association rules between sets of items in large databases”, SIGMOD'93, 207-216, Washington, D.C.
2. R. Agrawal and R. Srikant. “Fast algorithms for mining association rules.”, VLDB'94 487-499, Santiago, Chile.
3. S. Brin, R. Motwani, and C. Silverstein. “Beyond market basket: Generalizing association rules to correlations”, SIGMOD'97, 265-276, Tucson, Arizona.
4. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. “Dynamic itemset counting and implication rules for market basket analysis”, SIGMOD'97, 255-264, Tucson, Arizona, May 1997.
5. J. Han, J. Pei, and Y. Yin. “Mining frequent patterns without candidate generation”, SIGMOD'00, 1-12, Dallas, TX, May 2000.