Volume 2, No. 6, June 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

EVALUATING THE PERFORMANCE OF ASSOCIATION RULE MINING ALGORITHMS

K.Vanitha[*1], R.Santhi[2]
[1]Department of Computer Studies, Saranathan College of Engineering, Trichy,India
rkvanithamca@gmail.com
[2]Department of Computer Studies. Saranathan College of Engineering, Trichy,India
anandsanthi@rediff.com

Abstract: Association rule mining is one of the most popular data mining methods. However, mining association rules often results in a very large number of found rules, leaving the analyst with the task to go through all the rules and discover interesting ones. In this paper, we present the performance comparison of Apriori and FP-growth algorithms. The performance is analyzed based on the execution time for different number of instances and confidence in Super market data set. These algorithms are presented together with some experimental data. Our performance study shows that the FP-growth method is efficient and scalable and is about an order of magnitude faster than the Apriori algorithm.

Keywords – Apriori, FP-growth, Confidence

INTRODUCTION

Data Mining is often defined as finding hidden information in a database. Alternatively, it has been called exploratory data analysis, data driven discovery and deductive learning [2]. Among the areas of data mining, the problem of deriving associations from data has received a great deal of attention. Association rules are used to identify relationships among a set of items in database. These relationships are not based on inherent properties of the data themselves (as with functional dependencies), but rather based on co-occurrence of the data items[1].

Association Rule Mining finds application in market basket analysis. The market analysts would be interested in identifying frequently purchased items, so that the organization can adapt effective shelf space management and efficient sales strategies. Two strategically measures of significant that control the process of association rule mining are support and confidence. Support is the statistical significance of a rule while confidence is the degree of certainty of the detective associations the entire process of association mining is controlled by user specified parameters, namely, minimum support and confidence.

The main aim of this paper is to evaluate the performance of Apriori and FP-growth algorithms among the various algorithms in Association Rule Mining. The main difference between the two approaches is that the Apriori-like techniques are based on *bottom-up generation of frequent itemset combinations* and the FP-growth based ones are *partition-based, divide-and-conquer methods*. Also the Apriori algorithms generate the candidate itemsets but FP-growth does not generate the candidate itemset.

ASSOCIATION RULE MINING ALGORITHMS

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

Apriori Algorithm

Apriori is the best-known algorithm to mine association rules. It uses a breadth-first search strategy to counting the support of itemsets and uses a candidate generation function which exploits the downward closure property of support.
Apriori uses a "bottom up" approach, where 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.

A. Let's define:
   C_k as a candidate itemset of size k
   L_k as a frequent itemset of size k
B. Main steps of iteration are:
a. Find frequent set L_{k-1}
b. Join step: C_k is generated by joining L_{k-1} with itself (cartesian product L_{k-1} x L_{k-1})
c. Prune step (apriori property): Any (k – 1) size itemset that is not frequent cannot be a subset of a frequent k size itemset, hence should be removed
d. Frequent set L_k has been achieved

The **Apriori** algorithm is
L_1= {frequent items};
for (k= 2; L_{k-1} !=∅; k++) **do begin**
C_k= candidates generated from L_{k-1} (that is: Cartesian product L_{k-1} x L_{k-1} and eliminating any k-1 size itemset that is not frequent);
for each transaction t in database **do**
increment the count of all candidates in C_k that are contained in t
L_k = candidates in C_k with *min_sup*
end
return $\cup_k L_k$;

FP-growth (frequent pattern growth) uses an extended prefix-tree (FP-tree) structure to store the database in a compressed form. FP-growth adopts a divide-and-conquer approach to decompose both the mining tasks and the databases. It uses a pattern fragment growth method to avoid the costly process of candidate generation and testing used by Apriori.

FP-Growth adopts a divide-and-conquer strategy as follows. First, it compresses the database representing frequent items into a frequent-pattern tree or FP-tree, which retains the itemsets association information. It then divides the compressed database into a set of conditional databases. Each associated with one frequent item and mines each such database separately.

Definition 1 (FP-tree) A frequent pattern tree is a tree structure defined below.

A. It consists of one root labeled as "root", a set of *item prefix sub-trees* as the children of the root, and a *frequent-item header table*.
B. Each node in the *item prefix sub-tree* consists of three fields: *item-name*, *count*, and *node-link*, where *item-name* registers which item this node represents, *count* registers the number of transactions represented by the portion of the path reaching this node, and *node-link* links to the next node in the **FP-tree** carrying the same item-name, or null if there is none.
C. Each entry in the *frequent-item header table* consists of two fields, (1) *item-name* and (2) *head of node-link*, which points to the first node in the **FP-tree** carrying the *item-name*.[3]

The actual algorithm is:

Algorithm (FP-tree construction)
Input: A transactional database *DB* and a minimum support threshold ξ.
Output: Its frequent pattern tree, **FP-tree**
Method: The **FP-tree** is constructed in the following steps:

a. Scan the transaction database *DB* once. Collect the set of frequent items *F* and their supports. Sort *F* in support descending order as *L*, the *list* of frequent items.
b. Create the root of an **FP-tree**, *T*, and label it as "root". For each transaction *Trans* in *DB* do the following.
c. 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*).
d. The function *insert_tree*([*p* | *P*], *T*) 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 be linked to the nodes with the same *item-name* via the node-link structure. If *P* is nonempty, call *insert_tree*(*P*, *N*) recursively.

The FP-growth algorithm for mining frequent patterns with FP-tree by pattern fragment growth is:
Input: a *FP-tree* constructed with the above mentioned algorithm;
D – transaction database;
s – minimum support threshold.
Output: The complete set of frequent patterns.
Method:
call FP-growth(*FP-tree*, *null*).
Procedure FP-growth (*Tree*, *A*)
{
if *Tree* contains a single path *P*
then for each combination (denoted as *B*) of the nodes in the path *P* **do**
generate pattern *B* □ *A* with *support=minimum support of nodes in B*
else for each *ai* in the header of the *Tree* **do**
{
generate pattern *B* = *ai* □ *A* with *support* = *ai.support*;
construct *B*'s conditional pattern base and *B*'s conditional *FP-tree*
TreeB;
if *TreeB* ≠ Ø
then call FP-growth (*TreeB*, *B*)
}
}

METHODOLOGY

The two association rule mining algorithms were tested in WEKA software of version 3.6.1. WEKA software is a collection of open source of many data mining and machine learning algorithms, including pre-processing on data, Classification, clustering and association rule extraction.

The performance of Apriori and FP-growth were evaluated based on execution time. The execution time is measured for different number of instances and Confidence level on Super market data set.

We have analyzed both algorithms for super market data set. This data set contains 4627 instances and 217 attributes. For our experiment we have imported the data set in ARFF format.

For evaluating the efficiency, we have used the GUI based WEKA Application. The database is loaded using OpenFile in the preprocess tab. In the Associate tab we have selected the APriori and FP-growth algorithms to measure the execution time

RESULT AND DISCUSSION

In this section, we present a performance comparison of ARM algorithms. The following tables present the test results of Apriori and FP-growth for different number of instances and Confidence..

Table I Execution Time for Different Number Of Instances

No. of Instances	Execution Time (in Secs)	
	Apriori	FP-growth
3627	47	3
1689	25	2
941	8	1

As a result, when the number of instances decreased, the execution time for both algorithms is decreased. For the 3627 instances of supermarket data set, APriori requires 47

seconds but FP-growth requires only 3seconds for generating the association rules.

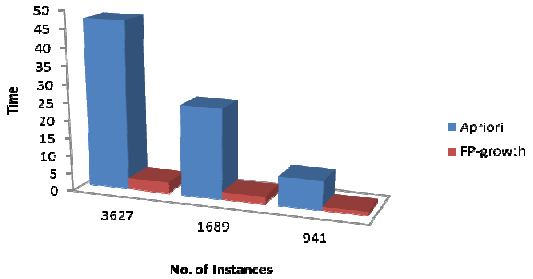

Figure1. Scalability of Apriori and FP-growth

In the above Figure 1, the performance of Apriori is compared with FP-growth, based on time. For each algorithm, three different size of data set were considered with sizes of 3627, 1689 and 941. Here the x-axis shows size of database in number of instances and y-axis shows the execution time in seconds.

When comparing Apriori with FP-Growth the FP-growth algorithm requires less time for any number of instances. So, the performance of FP-growth outperforms Apriori based on time for various numbers of instances.

Table II. Execution Time for Different Confidence Level

Confidence	Execution Time(in Secs)	
	Apriori	FP-growth
0.5	15	1
0.7	18	2
0.9	56	3

Table II summarizes that the execution time of Apriori and FP-growth.for various confidence level. When Confidence level is high, the time taken for both algorithms is also high. While the Confidence level is 0.5, the time taken to generate the association rule is 15seconds in Apriori and 1 second in FP-growth.

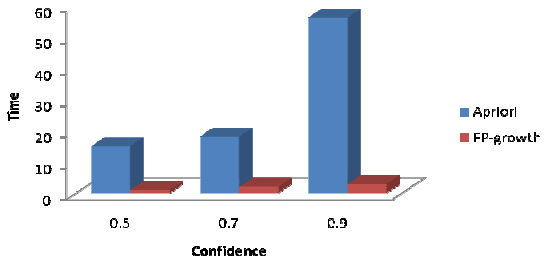

Figure2. Confidence vs. Time

Figure 2 shows the relationship between the time and confidence. In this graph, x axis represents the time and y axis represents the Confidence. The running time for FP-growth with confidence of 0.9 is much higher than running time of Apriori.

It says that, the time taken to execute the FP-growth is less compared with Apriori for any Confidence level. Thus the performance of FP-growth Algorithm is an efficient and scalable method for mining the complete set of frequent patterns

CONCLUSION

The association rules play a major role in many data mining applications, trying to find interesting patterns in data bases. In order to obtain these association rules the frequent sets must be previously generated. The most common algorithms which are used for this type of actions are the Apriori and FP-Growth. The performance analysis is done by varying number of instances and confidence level. The efficiency of both algorithms is evaluated based on time to generate the association rules. From the experimental data presented it can be concluded that the FP-growth algorithm behaves better than the Apriori algorithm.

REFERENCES

[1] M., Suraj Kumar Sudhanshu, Ayush Kumar and Ghose M.K., "Optimized association rule mining using genetic algorithm Anandhavalli Advances in Information Mining" , ISSN: 0975–3265, Volume 1, Issue 2, 2009, pp-01-04

[2] Margret H. Dunham, S.Sridhar, "Data mining Introductory and advanced topics", Pearson Education,Second Edition,2007.

[3] Daniel Hunyadi, "Performance comparison of Apriori and FP-Growth algorithms in generating association rules", Proceedings of the European Computing Conference

[4] Time.Dr (Mrs).Sujni Paul, "An Optimized Distributed Association Rule Mining Algorithm In Parallel And Distributed Data Data Mining With XML Data For Improved Response", International Journal of Computer Science and Information Technology, Volume 2, Number 2, April 2010

[5] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining Frequent Patterns without Candidate Generation" SIGMOD'2000 Paper ID: 196