Mining Association Rules Using Modified FP-Growth Algorithm

V. Ramya[1] and M.Ramakrishnan[2]

[1]Research Scholar, Department of Computer Science, Bharathiyar University, Coimbatore, Tamil Nadu,India
[1]vramya11@gmail.com

[2]Chairperson, School of Information Technology, Madurai Kamaraj University, Madurai, Tamil Nadu, India,
[2]ramkrishod@gmail.com

ABSTRACT

Association rule mining is performed first by finding the frequent itemsets. The FP-growth algorithms are the most famous algorithms which have their own shortcomings such as space complexity of the former and time complexity of the latter. Many existing algorithms are almost improved based on the two algorithms and one such is APFT, which combines the Apriori algorithm and FP-tree structure of FP-growth algorithm. The advantage of APFT is that it doesn't generate conditional & sub conditional patterns of the tree recursively and the results of the experiment show that it works fasts than Apriori and almost as fast as FP-growth. We have proposed to go one step further & modify the APFT to include correlated items & trim the non correlated itemsets. This additional feature optimizes the FP-tree & removes loosely associated items from the frequent itemsets. We choose to call this method as APFTC method which is APFT with correlation.

Keywords/ Index Term— Association Rule Mining, ARM, FP growth, frequent itemset.

1. INTRODUCTION

Association rules exhibit correlations among different data items in a transaction set. Association rules are if then statements that help uncover relationships between seemingly unrelated data in a relational database. An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent[1].

Association rules are created by analyzing data for frequent if/then patterns and using the criteria of support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database[2]. Confidence indicates the number of times the if/then statements have been found to be true. Mining association rules purely on the basis of minimum support may not always give interesting relationships between the item sets[3].

Consider a case where in a sample set of 100 transactions, item A with support SA=50 & item B with support SB=50 have a combined support of SAB =5.If the minimum support threshold is 5, it would appear as if A and B are frequent item sets because they satisfy the minimum support criteria. The drawback of this method is that only 10% of all A and 10% of all B. Another drawback of Apriori algorithm is that it can generate frequent itemsets by performing multiple scans over the database. Since scanning the database is very costly, it is important to have a method that can generate frequent itemsets with candidate set generation. The solution is to use FP-growth algorithm. FP growth algorithm improves the efficiency of the Apriori algorithm to a great extent and frequent itemsets are generated with just two scans over the database. Hence we present in this paper the mining of association rules with FP growth algorithm.

The paper is organized as follows. Introduction about basic concepts of association rule mining is briefed in section 1. The method used in the current situation termed as existing system is explained in section 2. Section 3 elaborates details about the proposed system which include the introduction about correlation concept, FP tree concept and proposed ideas. Experimental results are lucidly presented in section 4 and paper ends by noting the findings as conclusion in section 5.

2. EXISTING SYSTEM

The concept of frequent itemset was first introduced by Agarwal et al in 1993[4]. Two basic frequent itemset mining methodologies: Apriori & FP-growth, and their extensions, are introduced. Agarwal and Srikanth observed an interesting downward closure property which states that: A k-itemset is frequent only if all of its sub-itemsets are frequent. It generates candidate itemset of length k from itemset of length k-1[4]. Since the Apriori algorithm was proposed, there have been extensive studies on the improvements of Apriori, eg. partitioning technique, sampling approach, dynamic itemset counting, incremental mining and so on. Apriori, while historically significant, suffers from(1) generating a huge number of candidate sets, and (2) repeatedly scanning the database[5].

Han et al [6] derived an FP-growth method, based on FP-tree. The first scan of the database derives a list of frequent items in which items in the frequency descending order are compressed into a frequent-pattern tree or FP-tree. The FP-tree is mined to generate itemsets. There are many alternatives and extensions to the FP-growth approach, including depth first generation of frequent itemset [6];

Han et al [7] proposed hyper structure mining of frequent patterns; and an array-based implementation of prefix tree structure for efficient pattern growth mining. To overcome the limitation of the two approaches a new method named APFT [8] was proposed. The APFT algorithm has two steps: first it constructs an FP-tree & then second mines the frequent items using Apriori algorithm. [The results of the experiment show that it works faster than Apriori and almost as fast as FP-growth]. Extending this approach, we have introduced APFTC, which includes the concept of correlation to filter (reduce) the association rules that not only satisfy the minimum support but also have liner relationships among them. The computational results verify the good performance of APFTC algorithm.

3. PROPOSED SYSTEM

3.1. Correlation Concept

The concept of correlation can be used on transaction databases with the following modifications. An item ‘a’ is said to be correlated with item ‘b’ if it satisfies the following conditions: P (ab) > P(a)P(b). Here P(ab) = probability of items ‘a’ and ‘b’ occurring together in the transaction database i.e. the number of transactions in which both ‘a’ and ‘b’ occur together/total number of transactions. P(a) =The number of transactions in which ‘a’ occurs/total transactions. P(b) = The number of transactions in which ‘b’ occurs/total transactions[9]. Therefore the formula essentially represents Observed probability > Expected Probability. This condition is said to be positive correlation between items ‘a’ and ‘b’.

3.2. APFTC

We are basically deriving the frequent itemsets of size 2, so it is only appropriate to introduce the idea of correlation here. There is another change to the algorithm where we can calculate the support of each branch at the time of construction of calculation N Table itself instead of traversing the tree again later. This is a more economical way of calculating support than the one suggested in the original paper where repeated traversal of the tree is necessary for support calculation.

Algorithm APFT [ ]

Input: FP-tree, minimum support threshold ∋

Output : all frequent itemset

Apriori(T,ϵ)
$L_1 \leftarrow \{\text{large } 1-\text{itemsets}\}$
$k \leftarrow 2$
while $L_{k-1} \neq \emptyset$
$C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a\} - \{c \mid \{s \mid s \subseteq c \wedge |s| = k-1\} \nsubseteq L_{k-1}\}$
for transactions $t \in T$
$C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$
for candidates $c \in C_t$
$count[c] \leftarrow count[c] + 1$
$L_k \leftarrow \{c \mid c \in C_k \wedge count[c] \geq \epsilon\}$
$k \leftarrow k+1$
return $\bigcup_k L_k$

By introducing correlation coefficient, we continue for each node qi != root on the prefix path of q All Paths of Tree[i].add (qi.item-name);//AllPathsOfTree is an array of all paths from q to root. if NTable has a entry N such that N.Item-name= q i.item-name.

All Paths of Tree [i].support=q.count;//here we have the individual path //and its support stored to be used //later q = q.tablelink;

3.3. FP Tree

The construction of fp-tree is a straight forward procedure which follows from the above transaction database to the tree structure shown below Once the tree has been constructed we proceed with the APFT algorithm with construction of an N Table for each of the nodes. We start with the node 4 which is at the bottom of the header table for the given fp tree. Let us take the minimum support value as 2 for this example.

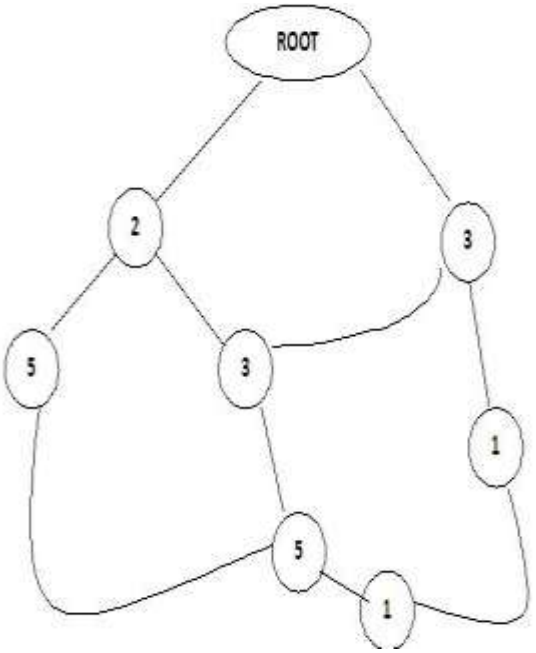

Figure 1 : FP Tree

ii. Ntable For Node 5

Calculations for Correlation with item 5:

P (5, 2) =4/5=0.8

P (5) P (2)=4/5*4/5=0.64

Hence P (5, 2)>P (5) P (2)

P (5, 3) =3/5=0.6

P (5) P (3)=4/5*4/5=0.64

Hence P (5, 3) <P (5) P (3)

The support count of node 2 is 4. It is clear that the Ntable alongwith Apriori's candidate generation step to successively generate supersets of the smaller itemsets and then perform the pruning step by calculating support by scanning the tree paths instead of scanning the entire database as is the case with apriori[10]. The candidate support calculation procedure is as shown in the diagram below each path from the node to the root is stored with that path's support.

4. EXPERIMENTAL RESULTS

We are using T1014D100K and Mushroom datasets for the experiment as inputs. Each data set contains 870 and 119 items respectively[13]. The algorithm PFTC is reportedly working efficiently and in many cases, it's much faster than FP-Growth. The results are found to be more interesting than association rules mined by FP-Growth although they are the subsets of itemsets mined by FP-Growth. The above graph shows the nature of the three algorithms with varied minimum support. It can be observed that APFTC has completed the task in very less time when compared to the other two algorithms. The above graph shows the number of itemsets generated with respect to varying minimum support.

Supporting our idea, our proposed algorithm has generated equal to the number of itemsets which are highly correlated when compared to the other algorithms. The above graphs gives the itemsets generated with respect to varying size[11]. The itemsets generated by our method are equal to or less than those generated by the other two algorithms in some cases. It can be concluded from the above results that the proposed method outperforms as expected proving to be efficient in time consumed and also in retrieving the most correlated itemsets[12].

5. CONCLUSION

Determining frequent objects (item sets, episodes, sequential Patterns) is one of the most important fields of data mining. It is well known that the way candidates are defined has great effect on running time and memory need, and this is the reason for the large number of algorithms. We presented frequent pattern mining is expecting transpose representation to relieve current methods from the traditional bottleneck, providing scalability to massive Data sets and improving response time. In order to mine patterns in databases with more columns than rows, we proposed a complete framework for the transposition: we gave the item set in the transposed database

of the transposition of many classical transactions ID. Then we gave a strategy to use this framework to mine all the itemset satisfying we used dynamic approach which is better than tradition approach for finding longest common subsequence.

REFERENCES

[1] Jeetesh Kumar Jain, Nirupama Tiwari, Manoj Ramaiya "A Survey: On Association Rule Mining", International Journal of Engineering Research and Applications, February-2013

[2] Yunlong Song, Ran Wei , "Research on Application of Data Mining Based on FP-Growth Algorithm for Digital Library", IEEE,July, 2011

[3] Kuldeep Singh Malik, Neeraj Raheja "Improving performance of frequent item set algorithm" International Journal of Research in Engineering & Applied Sciences, March, 2013

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In VLDBY94, pp. 487-499.

[5] Chen zhuo, Lu nannan, Li shiqi , Han tao "Research of Association Rule Mining Algorithm Based on Improved FP-Tree", I.J. Engineering and Manufacturing, October, 2012

[6] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation (PDF), (Slides), Proc. 2000ACM-SIGMOD Int. May 2000.

[7] J. Pei, J. Han, and H. Lu. Hmine: Hyper-structure mining of frequent patterns in large databases. In

[8] ICDM, 2001, pp441–448.

[9] Banu Priya.M*, Umarani. V "Enriching the Efficiency of Association Rule Mining using Enhanced IFP-Growth Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, January, 2013

[10] Ding Zhenguo, Wei Qinqin, Ding Xianhua"An Improved FP-growth Algorithm Based on Compound Single Linked List"International Conference on Information and Computing Science(IEEE)March,2009

[11] LIU Jian-ping, WANG Ying, YANG Fan-ding "Incremental mining algorithm Pre-FP in association rules based on FP-tree" First International Conference on Networking and Distributed Computing (IEEE), October,2010

[12] Jyoti Jadhav, Lata Ragha, Vijay Katkar "Incremental Frequent Pattern Mining", International Journal of Engineering and Advanced Technology (IJEAT),August, 2012

[13] Chia-Han Yang and Don-Lin Yang "IMBT - A Binary Tree for Efficient Support Counting of Incremental Data", International Conference on Computational Science and Engineering(IEEE) ,August,2009

