Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 2 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree

By Sujatha Dandu, B.L.Deekshatulu & Priti Chandra

Aurora's Technological and Research Institute, Hyderabad, India

Abstract - Frequent itemset mining plays an important role in association rule mining. The Apriori & 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 [11], which combines the Apriori algorithm [1] and FP-tree structure of FP-growth algorithm [7]. 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 : data mining, correlation, correlation coefficient.

GJCST-C Classification : H.3.3

Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree

Sujatha Dandu [α], B.L.Deekshatulu [σ] & Priti Chandra [ρ]

Abstract - Frequent itemset mining plays an important role in association rule mining. The Apriori & 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 [11], which combines the Apriori algorithm [1] and FP-tree structure of FP-growth algorithm [7]. 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 : data mining, correlation, correlation coefficient.

I. Introduction

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.

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. 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.

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 are involved in association rules together. So the relationship between A and B cannot be of much use even though they occur together more than the support value. A new method is required wherein we measure not only the support but also the confidence of B occurring when A occurs & vice versa. This way we can make sure that the interestingness of the rules is preserved. The concept of correlation is introduced in order to filter the result from the association rules that not only satisfy the minimum support criteria but also have a linear relationship amongst them. This approach combines the concepts of FP-growths tree generation technique with Apriori's candidate generation step along with a correlation condition so as to improve the interestingness of rules as well as to optimize the space and time consumption.

II. Existing System

The concept of frequent itemset was first introduced by Agarwal et al in 1993.Two basic frequent itemset mining methodologies: Apriori & FP-growth, and their extensions, are introduced. Agarwal and Srikanth [2] 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. Since the Apriori algorithm was proposed, there have been extensive studies on the improvements of Apriori, eg. partitioning technique[3], sampling approach[4], dynamic itemset counting [6], incremental mining [5] & so on.

Apriori, while historically significant, suffers from(1) generating a huge number of candidate sets, and (2) repeatedly scanning the database. Han et al [7] 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 [8]; H-mine, by [9] which explores a hyper structure mining of frequent patterns; and an array-based implementation of prefix tree structure for efficient pattern growth mining [10]. To overcome the limitation of the two approaches a new method named APFT [11] was proposed. The APFT algorithm has two steps: first it constructs an FP-tree & then second mines the frequent items using Apriori

Author α : Aurora's Technological and Research Institute, Hyderabad, India. E-mail : sujatha.dandu@gmail.com
Author σ : Distinguished Fellow, IDRBT, Hyderabad, India.
E-mail : deekshatulu@hotmail.com
Author ρ : Scientist, Advanced System Laboratory, Hyderabad, India.
E-mail : priti_murali@yahoo.com

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.

III. Proposed System

a) Correlation Concept

The concept of correlation can be extended to 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
Therefore the formula essentially represents

Observed probability > Expected Probability

This condition is said to be positive correlation between items 'a' and 'b'.

b) APFTC

The idea of correlation is introduced at step 5 of the APFT algorithm. We are basically deriving the frequent itemsets of size 2 at this step, 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 L
L = L1;
for each item Ii in header table, in top down order
LIi = Apriori-mining (Ii);
Return L = {LULi1ULi2U…Lin};
```

```
Pseudo code Apriori-mining (I i )
1.  Find item p in the header table which has the same
    name with Ii (
2.  q = p.tablelink;
3.  while q is not null
4.  for each node qi != root on the prefix path of q
5.  if N Table has a entry N such that N. Item-name
    = q i. item-name
6.  N. Item-support = N. Item-support + q.count;
7.  else
8.  add an entry N to the N Table;
9.  N.Item-name = q i. item-name;
10. N.Item-support = q.count;
11. q = q.tablelink;
12. k = 1;
13. Fk = {j | j"NTable)j.Item-support*minsup}
14. repeat
15. k = k + 1;
16. Ck = apriori-gen(Fk-1) ;
17. q = p.tablelink;
18. while q is not null
19. find prefix path t of q
20. Ct = subset(Ck, t);
21. for each c"+,t
22. c.support = c.support + q.count;
23. q = q.tablelink;
24. Fk = {c | c"+k,),c.support *,-./012}
25. until Fk = &
26. return LI i = Ii U F1 U F2 U…U Fk // Generate
```

Introducing correlation coefficient at line4, 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.

N.Item-support = N.Item-support + q.count; else

Check For Correlation Between qi and q
PA=Map Support (q.itemName);
PB=Map Support (qi.item-name);
P(AB)= q.count;
If(P(AB)>P(A)P(B)/transaction Count)
. add an entry N to the NTable;
. N.Item-name = q i. item-name;
. N.Item-support = q.count;

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;

c) Example

We follow an example of a simple database with 6 transactions as shown below.

Transaction Database
1 3 4
2 3 5
1 2 3 5
2 5
1 2 3 5

i. *FPTREE*

The fp-tree construction is a fairly straight forward procedure which follows from the above transaction database to the tree structure shown below

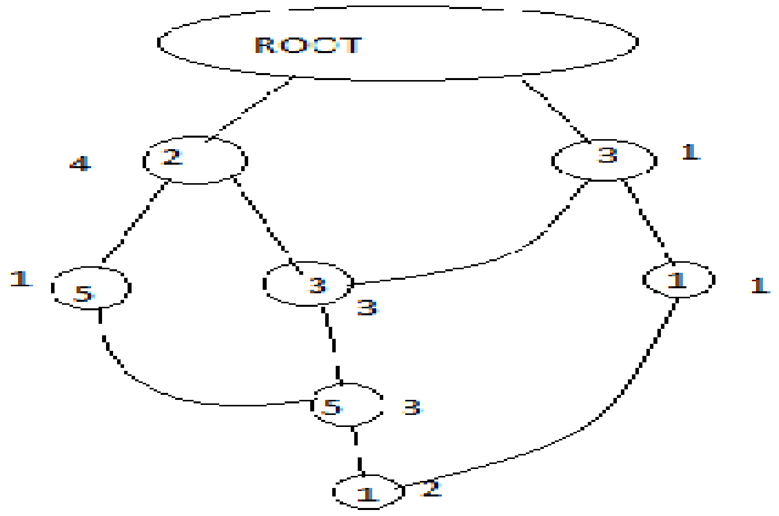


Figure 1 : FP-TREE

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 of 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.

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)

NODE	SUPPORT
2	4

OUTPUT
5 2:4

As shown in the figure we use the Ntable along with Apriori's candidate generation step to successively generate supersets of the 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.

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.

IV. Results

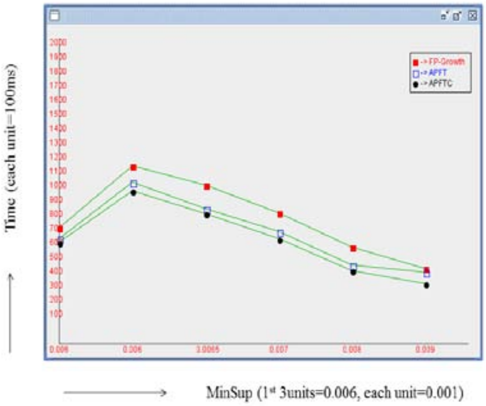
a) Datasets

The following datasets have been used in the experiment as inputs

DATASETS	Items	Max|t|	Avg|t|	Size
T10I4D100K	870	29	10	100000
Mushroom	119	23	23	8124

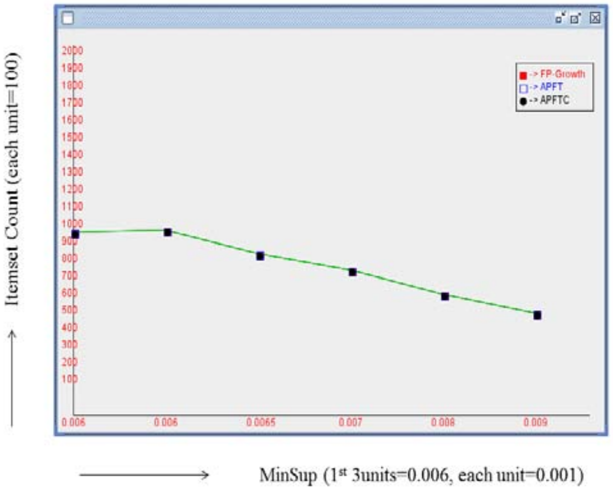
The algorithm APFTC is reportedly working efficiently and in many cases, it's much faster than FP-Growth. The results are found to be more interesting than association rules mined by FP-Growth although they are the subsets of itemsets mined by FP-Growth.

Figure 1 : Minimum Support vs. Time



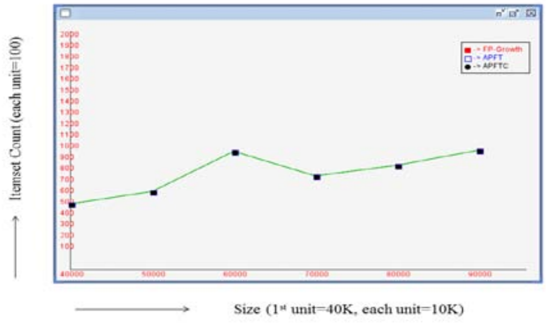
The above graph shows the nature of the three algorithms with varied minimum support. It can be observed that APFTC has completed the task in very less time when compared to the other two algorithms.

Figure 2 : Minimum Support vs. Count



The above graph shows the number of itemsets generated with respect to varying minimum support. Supporting our idea, APFTC has generated equal to the number of itemsets which are highly correlated when compared to the other algorithms.

Figure 3 : Size vs. Count



The above graphs gives the itemsets generated with respect to varying size. The itemsets generated by APFTC 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 APFTC performs as expected proving to be efficient in time consumed and also in retrieving the most correlated itemsets.

References Références Referencias

1. Data Mining – Concepts and Techniques – Jia Wei Han and Micheline Kamber
2. SPMF – A Sequential Pattern Mining Framework – Philippe Fournier-Viger
3. Contrasting Correlations by an Efficient Double-Clique Condition - Aixiang Li, Makoto Haraguchi, and Yoshiaki Okubo
4. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. InProc.1993 ACM-SIGMOD Int. Conf. Management of Data, Washington, D.C., May 1993, pp 207–216.
5. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In VLDBY94, pp. 487-499.
6. J.S. Park, M.S. Chen, and P.S. Yu. An effective hash- based algorithm for mining association rules. InSIGMOD1995, pp 175-186.
7. J. S. Park, M. S. Chen, and P. S. Yu. An effective hash- based algorithm for mining association rules. Proceedings of ACM SIGMOD International Conference on Management of Data, San Jose, CA, 1995, pp175-186.
8. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. Proceedings of the 21st International Conference on Very large Database, 1995.
9. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation (PDF), (Slides), Proc. 2000ACM-SIGMOD Int. May 2000.
10. AgarwalR, AggarwalC, PrasadVVV.A tree projection algorithm for generation of frequent itemsets. In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining), 2000.
11. J. Pei, J. Han, and H. Lu. Hmine: Hyper-structure mining of frequent patterns in large databases. In ICDM, 2001, pp441–448.