International Journal of Computing Science and Information Technology, 2014, Vol.02 (01), 01-05
ISSN: 2278-9669, January 2014 (http://ijcsit.org)

Effectiveness of various FPM Algorithms in Data Mining

Dr. Telkapalli Murali Krishna
Asst. Prof, Department of Computing, Jimma Institute of Technology, Jimma University, Jimma, Ethiopia
murali2007tel@yahoo.com

Abstract

Data Mining is retrieval of Knowledge from large amounts of data. A Frequent pattern is a pattern that appears in a data set frequently. It may be an itemset, subsequence or substructures. A set of items that appear frequently together in a transactional database is called Frequent Itemset. Frequent Itemset Mining is the essential step in association rule mining and in finding correlations. FPM also plays an important role in identifying interesting relationships among data. The detection of interesting correlation relationships among large business transaction tuples can help in decision-making process and customer shopping behavior analysis (i.e., market-basket analysis). FPM helps the business people to develop marketing strategies for gaining profits. There are many algorithms that have been proposed for finding frequent itemset mining in a transactional dataset. They are Apriori, FP-Growth, Vertical Partitioning, RELIM etc. In this paper, I compare the effectiveness of these algorithms and proposed a new algorithm in an advanced approach. The new thought of this algorithm is derived from existing algorithms. The effectiveness of this new algorithm can be achieved with less number of scans and better intermediate steps.

Keywords: Data Mining, Association Rule, FP-Tree, Frequent Itemset, Transactional Database

1. Introduction

Innovation of knowledge from vast amounts of data is called data mining or knowledge extraction. One of the data mining functionalities is association rule mining. The data is in the form of transactional or relational model. Association rule mining is helpful in finding the frequent item set in the transactional database.

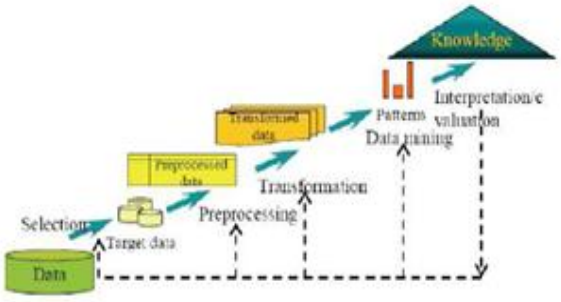

Fig 1: The transition from raw data to valuable knowledge

Frequent item set mining is used for analyzing customer buying habits by finding associations between the dif-ferent items that customers place in their shopping baskets.

The two algorithms for mining frequent itemsets are Aprori (Candidate Generation and Test) and FP-Growth (Pattern Growth). For improving the efficiency, research has been conducted for an effective and efficient mining of frequent itemset.

2. Problem Statement

Let K = $\{k_1, k_2, k_3 \ldots\ldots, k_n\}$ be the set of items and "D" is the task relevant database; which contains a set of transactions where each transaction 'T' is a set of items such that T ⊂ K. An itemset that contains 'n' items is called n-itemset. The Association rule is an implication of X→Y, where X ⊂ K, Y ⊂ K and X∩Y=ϕ. The rule X→Y holds support 'S' in D and confidence 'C'. Support 'S' is the percentage of transactions in D that contain XUY and confidence 'C' is the percentage of transactions in 'D' that containing X that also contain 'Y'.
The number of transactions that contain the itemset is called frequency or support count.

2.1 Mining Frequent Itemset

When a transactional database 'D' and Sup_{min} are given, the problem of finding the complete set of frequent itemsets with less computational time.

Table 1: Sample Transactional Database

Trans ID	List of Item IDs
T1	11,12,14,15,16,20,22
T2	11, 16, 17, 25
T3	13, 15, 18, 19
T4	11, 13, 14, 15, 17, 19
T5	11, 12, 13, 15, 17,21
T6	15, 18, 23
T7	11, 12, 13, 15, 16 20
T8	11, 13, 14, 21
T9	11, 13, 15, 17, 23
T10	11, 13, 15, 17,24

and $Sup_{min} = 4$.

2.2. Solution using APRIORI (Join & Prune steps)

C1

Itemset	Support count
11	8
12	3
13	7
14	3
15	8
16	3
17	5
18	2
19	2
20	2
21	2
22	1
23	2
24	1
25	1

C2 = L1 x L1

L2

Itemset	Support count
11,13	6
11, 15	6
11, 17	5
13,15	6
13, 17	4
15, 17	4

C3 = L2 X L2

Itemset	Support count
11, 13, 15	5
11,13,17	4
11,15,17	4
13,15,17	4

C4

Item set	Support Count
11,13,15,17	4

In a large transactional database, to do market basket analysis, Apriori algorithm is good. But it takes many scans of the database and to check with the candidate's minimum support. Candidate generation and test needs a large I/O and also more search space.

2.3 FP – Growth

The first scan of the database is same as Apriori. An FP-Tree is constructed with root as null. The set of frequent items sorted in the descending support count and denote it as L.

L = {<11:8>, <15:8>,<13:7>,<17:5>}

Itemset	Support count
11	8
13	7
15	8
17	5

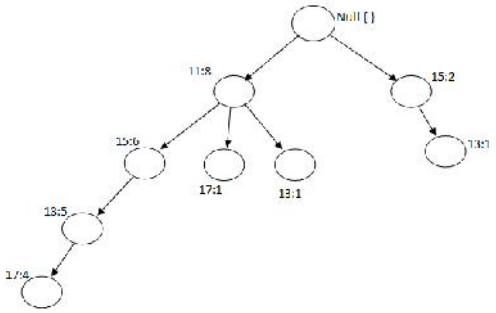

Fig 2: FP-Tree Construction

Table 2: Mining FP-Tree by creating Conditional sub-pattern bases

Item	Conditional Pattern base	Conditional FP-Tree	Frequent Items generated
17	{11,15,13: 1} {11:1}	{11:5, 15:4,13:4}	{11,17:4} {15,17:4}{13,17:4}
13	{11,15: 5} {11:1} {15:1}	{11:6, 15:6}	{11,13:4} {15,13:4}
15	{11:6}	{11:6}	{11,15:4}

FP-Growth method transforms the problem of finding long frequent patterns to searching for shorter patterns recursively and then concatenating the suffix. There is no candidate generation, but the pointers need to store in memory require large space. This method considerably reduces the search cost.

The other methods such as Vertical data format and RE-LIM have their own deficiencies.

Vertical Layout is faster than Apriori. Transform the horizontally formatted data into vertical format by scanning the data set once. The support count of an itemset is the length of TID_set of the itemset. TID_set takes substantial memory & time for the set intersection and also require more memory space compared with horizontal layout.

RELIM (Recursive Elimination) algorithm works better than all the earlier algorithms and is basically inspired by FP-Growth algorithm. There is no candidate generation but it requires pre-processing on database in the first stage which makes more computational cost. The main strength of this algorithm is not speed, but the simplicity of its structure.

3. NEW SUGGESTED APPROACH

The main objective is to minimize candidate set generation and reduced execution time, so that the algorithm generates frequent itemsets in a fast manner.

In the new proposed approach there is no widespread candidate set generation and no repeated scanning of the database to determine support count.

Algorithm Steps

1. Scan the Database 'D'
2. Calculate Frequent 1-Itemset
3. Sort the 1-itemset by ascending order of support (if the items are of same support arrange them either in ascending order or in alphabetical manner)
4. Construct Frequent-2 table by right neighbor method and scan the database
5. Construct the Tree with binary values (strings) in each leaf.
6. P= 3

While {height of the tree increased
Generate P-layer Tree ( )
P = P +1
}

7. List the frequent items from the tree.

3.1 Implementation

Database 'D' is scanned and min. support is 4. The resultant frequent 1-itemset is {<11:8>, <13:7>, <15:8>, <17:5>}. Arrange this list in the ascending order of support count. Then revised list will be {<17:5>, <13:7>, <11:8>, <15:8>}. Both 11 and 15 are of same support, as the list IDs are represented in number, you have to arrange in ascending order. If the item IDs are represented in alphabets like A, B, C, D and any two IDs are of same support, arrange them in alphabetical order.
In step 6, a third layer of tree is generated. The itemset {17,13,15} is checked. The subsets if this itemset {17,13: 4} {17,15:4} {13,15:6} all are frequent and there is no equal support pruning in its subsets. The Bitwise 'AND' operation of the strings '0001100011' and '0001100011' is performed and the output is '0001100011'. That is the support of itemset {17,13,15} is 4 which is equivalent to {17,13}. Item Id '15' is put into the equal-support set of {17,13} and there is no new child to be added.

Next itemset {13,11,15} is checked and all its subsets {13,11:6} {13,15:6}{11,15:6} are frequent. Perform the Bitwise 'AND' operation of the leaf nodes binary strings '0001101111' and '0011101011'. The resultant string is '0001101011' and the number of '1' bits is 5. So the new support itemset {13,11,15} is 5, which is not equal to {13,11}. Hence a new child is added to {13,11}.

3.2 Frequent-2 Table

TID	Ordered ID	{17, 13}	{17,11}	{17, 15}	{13, 11}	{13,15}	{11, 15}
T1	11,15	0	0	0	0	0	1
T2	17,11	0	1	0	0	0	0
T3	13,15	0	0	0	0	1	0
T4	17,13,11,15	1	1	1	1	1	1
T5	17,13,11,15	1	1	1	1	1	1
T6	15	0	0	0	0	0	0
T7	13,11,15	0	0	0	1	1	1
T8	13,11	0	0	0	1	0	0
T9	17,13,11,15	1	1	1	1	1	1
T10	17,13,11,15	1	1	1	1	1	1
Sup Count		4	5	4	6	6	6

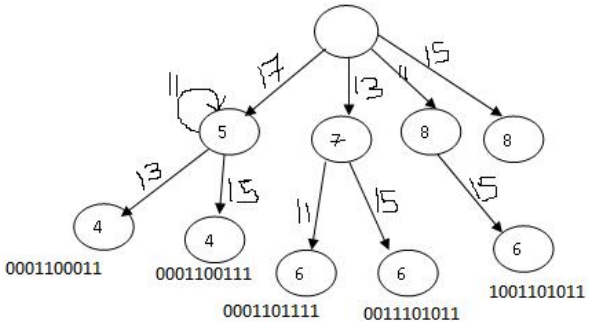

Fig 3: Generation P-Layer Tree

The value of 'P' is incremented and P=4. There is only one node in layer 4 and the height of the tree does not increase & the algorithm terminates.

All the frequent itemsets are written according to the tree in Figure 4 and they are {17,13,15} {13,11,15}.

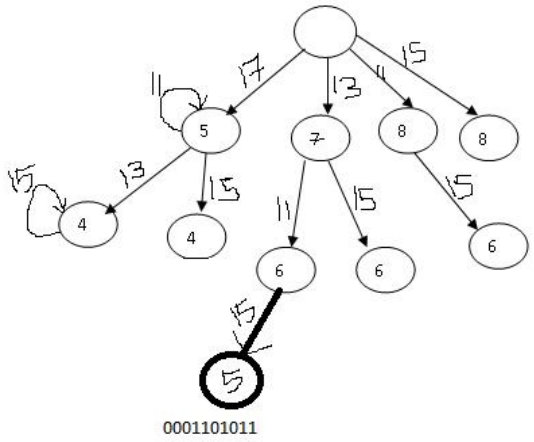

Fig 4: Generation of New P-Layer Tree

4. Conclusion

In the new approach, there is no candidate set generation and also in Frequent-2 table , the summation of each column data gives the support count. There are no complex tree traversal methods. Bitwise AND operation finds the support count of new node in a fast manner, when compared with Apriori, FP-Growth, Vertical Partitioning and RELIM.

The two basic FPM techniques namely Apriori and FP-Growth are effectively integrated and hence the execution time efficiency is improved.

REFERENCES

[1] Aggrawal.R; Imielinski.t; Swami.A. 1993. *Mining Association Rules between Sets of Items in Large Databases*. ACM SIGMOD Conference. Washington DC, USA.

[2] J. Han, H. Cheng, D. Xin, X. Yan. August 2007. *Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery*, vol., no., pp.55–86, 15(1).

[3] *Data Mining, Concepts and Techniques*, Second Edition, J. Han and M. Kamber.

[4] C.Borgelt. 2003. *Efficient Implementations of Apriori and Eclat*. In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany.

[5] Agarwal, R. C., Agarwal, C. C. and Prasad, V. V. V. (2001) *A tree projection algorithm for generation of frequent item sets*. Journal of Parallel and Distributed Computing, 61(3), Pp. 350–371.

[6] Deepak Garg et. al. "*Comparative Analysis of Various Approaches Used in Frequent Pattern Mining*" (IJACSA) International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence.

[7] Han.J., Pei.J., Yin. Y.2000. *Mining frequent patterns without candidate generation*. In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD).

[8] J. Pei, J. Han; R. Mao. 2000. Closet: *An efficient algorithm for mining frequent closed itemsets*. In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

[9] Goswami D. N et. al. "*An Algorithm for Frequent Pattern Mining Based On Apriori*" (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, Pp. 942-947.

[10] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, T. Yiu. Nov 2005. Mafia: *a maximal frequent itemset algorithm*. IEEE Transactions on Knowledge and Data Engineering, vol., no., pp. 1490–1504, 17(11).

[11] Agrawal.R., Srikant.R. September 1994. *Fast algorithms for mining association rules*. In Proc. Int'l Conf. Very Large Data Bases (VLDB), pp.487–499.