A Novel Approach to Mine Frequent Itemsets Using Maximal Apriori and FP-tree Method

Bharat Gupta[1], Dr. Deepak Garg[2], Karun Verma[3]

[1, 2, 3] *Department of Computer Science, Thapar University, Patiala, Punjab, India*
E-mail : [1]bharatgupta35@gmail.com, [2]dgarg@thapar.edu, [3]karun.verma@thapar.edu

Abstract

With a large transactional database, it is time consuming and difficult task to mine the frequent itemsets. Apriori and FP-tree algorithms are the two classical algorithms for frequent itemsets mining. But it suffers from many problems like repetitive database scan, huge candidate generation or memory efficiency problem during mining with large transactional datasets. This paper presents the idea to reduce the database scan by first finding the maximal frequent itemsets in the database and then all its subset consider as frequent according to Apriori property. Then prune the database by just considering only those transactions which are 1-itemset frequent but not include in maximal frequent itemsets and then mine the remaining left frequent itemsets by constructing the FP-tree. Thus it reduces the memory constraints and helps to efficiently mine frequent itemsets in less time.

Key words : *Data mining, Frequent itemsets, Apriori algorithm, FP-tree algorithm.*

1. Introduction

Association rule mining is one of the most important data mining problems. The purpose of association rule mining is the discovery of association relationship among a set of frequent and infrequent items [7]. The mining of association rules include two sub problems [5].

1. Finding all frequent itemsets that appear more often than a minimum support threshold, and
2. Generate association rules using these frequent itemsets.

The first sub problem i.e. mining frequent itemsets from the large transactional databases is a fundamental and the most difficult task. It means association rule mining is mainly depends upon the efficiently mined frequent itemsets, although many algorithms have been proposed, which can be categorised into two main classes: First is the candidate generate-and-test approach and second is the pattern growth approach. The first class algorithms, such as Apriori [1] as well as many subsequent studies [4, 9, 10]. In each iteration of the candidate generate-and-test approach, pairs of frequent k-itemset are joined to form candidate (k+1)-itemset, and then scanned the database to verify their supports. The Apriori algorithm achieves good reduction on the size of candidate sets, however. It takes many scans of the database to check the candidate's supports as much as the most long length of patterns, so, when there exist a large number of frequent patterns and long patterns, candidate generate-and-test approach may suffer a large overhead of I\O. Its subsequent improvements [4, 9] may produce frequent itemset early but not completely. The second class comprises pattern-growth methods [2], over the past few years, several pattern-growth algorithms have been proposed. A FP-growth algorithm uses the tree structure to store the contents, instead of generating candidates as in Apriori, it mine the using FP-tree recursively by building Pattern base and conditional tree. The second approach is more efficient but needs more memory to store pattern tree as for large database it is difficult to accommodate all items in main memory as tree structure.

As it is very common in retail selling database that two or more items sell or purchase together many times [6] therefore database contains same set of items many times, by using this concept aim to overcome the limitation of above existing approaches [1, 2] and propose a novel approach to mine frequent itemsets from a large transactional database without the candidate generation with the help of existing techniques. First at the beginning, In the first scan same set of occurring transactions are found from the database and save in the two dimension array with the count of repetition. Then find the maximal frequent itemsets if contained in array i.e. find the maximal transactions whose occurrence is greater than the user specified limit (support). Take all its non empty subsets of maximal frequent itemsets as frequent [4]. In this way most of the frequent itemsets are found. For the remaining itemsets which are frequent but not include in maximal frequent itemsets will be mined by using tree structure. Therefore database is pruned by considering only those transactions which contain the 1-itemset frequent items but not include in maximal frequent itemsets. In this way now construct the FP-Tree only for pruned (reduced) database. Thus memory consumption for FP-tree is now less due to reduced database.

The further organisation of this paper is as follows. In Section 2, we briefly review the Apriori method and FP-tree

ISSN : 0975 - 7686

algorithm. Sections 3 propose an efficient algorithm for mining frequent tem set based upon maximal Apriori and FP-tree structure.

2. Problem Statement

Let be a set of items and is considered the dimensionality of the problem. Let D be the task relevant database which consists of transactions where each transaction T is set of items such that. A transaction T is said to contain itemset X, which is called a pattern, i.e. . A transaction is a pair which contains unique identifier Tid and set of items [8].

Lemma1: A transaction T is said to be maximal frequent if its pattern length is greater than or equal to all other existing transactional patterns and also count of occurrence (support) in database is greater than or equal to specified minimum support threshold [4].

Lemma2 : An itemset X is said to be frequent if its support is greater than or equal to given minimum support threshold i.e. count(X)>minsup [1].

Transactional database D and minimum support threshold is given, the problem is to finding the complete set of frequent itemsets is known as frequent itemsets mining problem [1].

3. Literature Review

3.1 Apriori algorithm

Apriori algorithm is a bottom-up, breadth-first approach [1]. The frequent itemsets are extended one item at a time. It employs an iterative approach known as a level wise search, where k-itemsets are used to explore (k+1)-itemsets [1]. The main steps of Apriori algorithm are show as follows [5]:

Step 1 : Find 1-itemset frequent: Scan the database D; count the number of occurrences of each item and make the member of candidate itemset.The set of items whose count is bigger or equal with the min-support. Consider the itemset is as.

Step 2 : Join *: To find next level, a set of candidate k-itemsets is generated by joining with itself. (I.e. Cartesian product (*).This set of candidates is denoted.

Step 3 : Prune infrequent items: is a superset of , so that members of may or may not be frequent, but all of the frequent k-itemsets are included in . Scan of the database to determine the count of each candidate in would result in the determination of (i.e. all candidates having a count no less than the minimum support count is frequent by definition). , however, can be huge, and so this could involve heavy computation. To reduce the size of k, the *Apriori property* is used as follows [3]. *If an itemset is frequent then all its non empty subsets are frequent and vice versa.* Hence, if any (k 1)-subset of a candidate k-itemset is not frequent in k-1, then the candidate cannot be frequent either and so can be removed from.

Step 4 : Repeat the steps mentioned above until no new candidate itemsets generate.

The transactional database comprises of large database therefore Apriori algorithm has certain limitation on the speed and efficiency. Large number of generated candidate sets results in the low efficiency of the mining algorithm. Secondly, so many records of the database result in too many I/O spending. Therefore it doesn't consider as a good algorithm for mining frequent itemset from today's large databases.

3.2 FP-tree Algorithm

FP-tree algorithm [2] is based upon the recursively divide and conquers strategy; first the set of frequent 1-itemset and their counts is discovered. With start from each frequent pattern, construct the conditional pattern base, then its conditional FP-tree is constructed (which is a prefix tree.). Until the resulting FP-tree is empty, or contains only one single path. (Single path will generate all the combinations of its sub-paths, each of which is a frequent pattern). The items in each transaction are processed in L order. (i.e. items in the set were sorted based on their frequencies in the descending order to form a list).

The detail step is as follows: [5]

FP-Growth Method: Construction of FP-tree

1. Create root of the tree as a “null”.
2. After scanning the database D for finding the 1-itemset then process the each transaction in decreasing order of their frequency.
3. A new branch is created for each transaction with the corresponding support.
4. If same node is encountered in another transaction, just increment the support count by 1 of the common node.
5. Each item points to the occurrence in the tree using the chain of node-link by maintaining the header table.

After above process mining of the FP-tree will be done by Creating Conditional (sub) pattern bases:

1. Start from node constructs its conditional pattern base.
2. Then, Construct its conditional FP-tree & perform mining on such a tree.
3. Join the suffix patterns with a frequent pattern generated from a conditional GP-tree for achieving FP-growth.
4. The union of all frequent patterns found by above step gives the required frequent itemset.

In this way frequent patterns are mined from the database using FP-tree.

Definition : Conditional pattern base [5]

A “subdatabase” which consists of the set of prefix paths in the FP-tree co-occuring with suffix pattern.eg for an itemset X, the set of prefix paths of X forms the conditional pattern base of X which co-occurs with X.

Definition : Conditional FP-tree [5]

The FP-tree built for the conditional pattern base X is called conditional FP-tree.

A FP-tree is a compressed tree structure which contains complete information of frequent itemsets. But it suffers from a problem when database is too large then it cannot fit into the main memory.

4. Improved Mining Algorithm

In this Section, a new algorithm based upon the improved Apriori and the FP-tree structure is present.

In a large transactional database like retailer database it is common that multiple items are selling or purchasing simultaneously therefore the database surly contains various transactions which contain same set of items. Thus by taking advantage of these transactions trying to find out the frequent itemsets and prune the database as early as possible without generating the candidate itemset and multiple database scan, results in efficiently usage of memory and improved computation.

This proposed algorithm is based upon the Apriori property [1] i.e. all non empty subsets of the frequent itemsets are frequent.

This proposed algorithm has two procedures. In first procedure, find all those maximal transactions which are repeating in the database equal to or greater than min user defined support also known as maximal frequent itemset [4]. Then get all nonempty subsets of those maximal frequent itemset as frequent according to Apriori property. Scan the database to find 1-itemset frequent elements. There may be many items found which are 1-itemset frequent but not include in maximal frequent transactions. Therefore prune the database by just considering only those transactions from the database which contain 1-itemset frequent elements, but not include in the maximal frequent itemsets. Now this pruned database is smaller than the actual database in the average cases and no item left in best case.

For the second procedure, pruned database is taken as input and scan the pruned database once find 1-itemset frequent and delete those items from transaction which are not 1-itemset frequent. Then construct the FP-tree [2] only for pruned transactions. In this way it reduces the memory problem for FP-tree because the database is reduced in most of cases. In best case no need to build FP-tree because all elements are found in first procedure. In the worst case if there is no maximal frequent transaction exist, then only second procedure run and also computational performance is same as FP-tree. The key of this idea to prune the database after finding the maximal frequent itemsets and formation of FP-tree for a pruned database thus reduce memory problem in FP-tree and make the mining process fast. The more detail step as follows:

Procedure1:

Input: Database D, minimum support

Step 1: Take a 2- dimensional array; Put the transaction into 2-dimmension array with their count of repetition.

Step 2: Arrange them in increasing order on the basis of the pattern length of each transaction.

Step 3: Find maximal transactions (k-itemset) from the array whose count is greater than or equal to the minimum support known as maximal frequent itemsets or transactions. If k-itemsets count is less than minimum support then look for k-itemsets and (k-1)-itemsets jointly for next (k-1) maximal itemsets and so on until no itemsets count found greater than minimum support. If no such transaction found then go to Procedure2.

Step 4: Once the maximal frequent transactions found, than according to Apriori property consider all its non empty subsets are frequent.

Step 5: There are itemsets remaining which are not included in maximal frequent itemset but they are frequent. Therefore find all frequent 1-itemset and prune the database just considering only those transactions which contain frequent 1-itemset element but not include in maximal frequent transaction.

Output: some or all frequent itemsets, Pruned database D1.

Procedure2:

Input: Pruned database D1, minimum support

Step 1: Find frequent 1-itemset from pruned database; delete all those items which are not 1-itemset frequent.

Step 2: Construct FP-tree for mine remaining frequent itemset by following the procedure of FP-tree algorithm [2] as discussed above in section 2B.

Output: remaining frequent itemsets

For the clarity of presentation, let's take an example:

Example:

Suppose table 1 is a database of retailer, D. There are 10 transactions. Suppose the minimum support is 2.

Table 1. Sample Retailer Transactional Database

Tid	List of items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3
T10	I1,I2,I3,I5

Procedure 1:

Input: Database D, Minimum support = 2

Step1: After scanning a database put items in 2-dimensional array with the count of repetition.

Table 2. Itemsets in array

2-itemset	count	3-itemset	count	4-itemset	count
{I2,I3}	2	{I1,I2, I4}	1	{ I1,I2,	2
{I1,I3}	2	{I1,I2, I3}	1	I3, I5}	
{I2,I3}	2	{I1,I2, I4}	1		

Step 2 : Find maximal itemset (4-itemset). Check weather its count is greater or equal to specified support, its count is 2 in our case which is equal to given support therefore this transaction is considered as maximal frequent. (If its count is less than support value then we scan k-1 and k-itemset in array for k-1 maximal itemset jointly and so on until finding all maximal frequent itemset from a array. i.e. 3-itemset and 4-itemset for checking 3-itemset maximal and so on).

Step 3 : According to Apriori property subset of maximal frequent itemset is also considered as frequent .i.e. Maximal frequent itemset: {I1, I2, I3, I5}

All subsets are frequent (Apriori Property) i.e. {I1, I2, and I3}, {I1, I2, I5}, {I2, I3, I5}, {I2, I3}, {I2, I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I3, I5}, {I1}, {I2}, {I3}.

Step 4 : Scan the database for finding the above mined support.

Step 5 : Find 1-itemset frequent from database, it is found that I4 which is frequent but not include in maximal frequent itemset. (There may be many items remain which are not include in maximal frequent itemsets, in our case only 1 item is there).

Prune the database by considering only transaction which contains I4 itemset.

Output : Some frequent itemsets ({I1, I2, I5}, {I2, I3, I5}, {I2, I3}, {I2, I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I3, I5}, {I1}, {I2}, {I3}), Pruned database.

Table 3. Pruned Database

Tid	List of items
T2	I2,I4
T4	I1,I2,I4

Procedure 2:

Input : Pruned database, minimum support = 2

Step 1 : Find frequent 1-itemset from pruned database with support = 2, It is found I1 is not frequent therefore delete it. (In this case delete I1).

Table 4. Frequency of Itemsets

Itemset	Frequency
I2	2
I4	2
I1	1

Transactions become: T2: I2, I4 and T4: I2, I4.

Step 2: Construct FP-tree for remaining transaction in pruned database.

1. In this case I2 and I4 have same frequency, therefore no need to arrange in L order (descending order of their frequencies).
2. A new branch is created for each transaction. In this case single branch is created because of same set of transactions

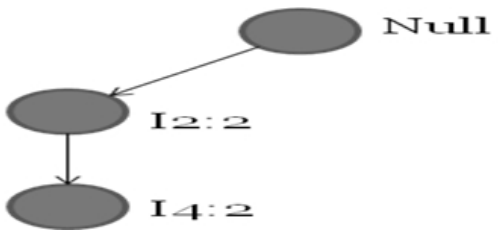

Fig. 1. FP-tree for above transaction

3. Construct Conditional pattern base and FP-tree only for item I4..

Table 5. Mining the FP-tree by creating conditional (Sub-) pattern base

Item	Pattern base	Conditional FP-tree	Frequent Item
I4	{I2: 2}	{I2:2}	{I4,I2:2}

Thus by this procedure we can easily find unmined frequent itemset i.e. {I4, I2} which some of previous algorithm [4] are not able to find. Now we get all the frequent itemset in a particular database.

Output: remaining itemsets ({I4, I2})

5. Advantages and disadvantages of algorithm

This approach is well suited for transactional database where database may large or small but having same transactions occur many times. If the repetition count crosses the minimum support then it is very easy for this algorithm to find frequent itemsets as early as possible because all those transactions considered as frequent and as per apriori property its subsets i.e. all items contain in those transactions also consider as frequent. There may be the case where no transaction repeats, in that case only the second procedure runs i.e. FP-tree will build for all the transactions and frequent itemset will be found and computation cost become equal to FP-tree. Therefore it can say that, this algorithm is based on the approach to find maximal frequent itemset firstly so that most of the frequent itemset can be mined early and also overcome the limitation of previous approach [4] by constructing the FP-tree for the remaining itemsets left in first procedure. Thus memory utilization for

tree structure should be minimised and fit into main memory due to pruning of database in first step. Thus in this way it saves much time and space.

This type of approach is much better than Apriori and FP-tree in terms of scanning and memory utilization because it doesn't produce larger number of candidates and also doesn't need to scan whole database again and again. In average cases the input for the construction of FP-tree is partial database so that whole FP-tree can accumulate in main memory easily.

But this approach is only better for transactional databases where many items sell or purchase (occur) in one set so that maximal itemset contains many items. This algorithm loose mean when no maximal frequent itemset occur in database then all the frequent items can be mined by only constructing the FP-tree.

6. Conclusion

In this research work attempt has been made to develop an algorithm which is an improvement over Apriori and FP-tree, using an approach of improved Apriori algorithm for a transactional database. With the above discussion it shows that this algorithm produces frequent itemsets completely. This approach doesn't produce candidate itemsets and building FP-tree only for pruned database so that fit into main memory easily. Thus it saves much time and space and considered as an efficient method. But there are many disadvantages also as discussed which can be improved in future research.

References

[1] R.Agrawal, R.Srikant, "Fast algorithms for mining association rules", Proceedings of the 20th Very Large DataBases Conference (VLDB'94), Santiago de Chile, Chile, 1994, pp. 487-499.

[2] J.Han, J.Pei and Y.Yin., "Mining frequent patterns without candidate Generation", in: Proceeding of ACM SIGMOD International Conference Management of Data, 2000, pp.1-12.

[3] Jie Gao, Shao-jun Li, "A Method of Improvement and Optimization on Association Rules Apriori Algorithm", Proceedings of the 6th World Congress On Intelligent Control and Automation, 2006, pp.5901-5905.

[4] D.Sun, S.Teng, W.Zhang, H. Zhu, "An Algorithm to Improve the Effectiveness of Apriori", Proceeding of the 6th IEEE International Conference on Cognitive Information, 2007, pp.385-390.

[5] Jiawei Han, M.Kamber, "Data Mining-Concepts and Techniques", Morgan Kanufmann Publishers, Sam Francisco, 2009

[6] C.Zhang, J.Ruan, "A Modified Approach with Its Application in Industrial Cross-Selling Strategies of the Retail Industry", Proceeding of Internatonal Conference on Electronic Commerce and Business Intelligence, 2009, pp.515-518.

[7] Ling Zhou, Stephen Yau, "Efficient association rule mining among both frequent and infrequent items", Computers and Mathematics with Applications, 2007, 54, pp. 737-749.

[8] Q.Lan, D.Zhang, B.Wu, "A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree", Proceeding on IEEE International Conference on Global Congress on Intelligent System, 2009, pp.360-364.

[9] W.LIU, J.CHEn, S.Qu, W.Wan, "An Improved Apriori Algorithm", Proceeding on IET International Communication Conference on Wireless Mobile & Computing (CCWMC 2009), pp.221-224.

[10] S.P Latha, DR. N.Ramaraj, "Agorithm for Efficient Data Mining", Proceeding on IEEE International Computational Intelligence and Multimedia Aplications 2007, pp. 66-70.

Bharat Gupta, he is Member of IEEE Computer Society. He is student of Masters in Engineering in Computer Science at Thapar University, Patiala (Punjab). His research interests are Algorithm design and complexity issues for Data Mining.

Mr. Karun Verma, He is Member IEEE .He is the assistant Professor in computer science department in Thapar University, Patiala (Punjab). He has more than 10 publications to his credit. His research interests are Natural Language Processing.

Dr Deepak Garg, Senior Member of ACM and is Chair, ACM North India SIGACT. He is Senior Member IEEE and Secretary IEEE Delhi Section Computer Society. He has more than 50 publications to his credit. His research interests are Algorithm design and complexity issues. He is having 15 years of research experience.