

## COMPARATIVE STUDY OF FREQUENT ITEMSET MINING ALGORITHMS APRIORI AND FP GROWTH

<sup>1</sup>Dr. Banumathi, A. and <sup>2,\*</sup>Dhurga Devi, M.

<sup>1</sup>Assistant professor, Govt. Arts., College, Karur, Tamil Nadu 639007

<sup>2</sup>Student of Cs Department, Govt Arts College, Karur, Tamil Nadu 639007

### ARTICLE INFO

#### Article History:

Received 19<sup>th</sup> February, 2018  
Received in revised form  
01<sup>st</sup> March, 2018  
Accepted 28<sup>th</sup> April, 2018  
Published online 31<sup>st</sup> May, 2018

#### Key Words:

Frequent Itemset Mining,  
Apriori, FP-Growth.

### ABSTRACT

In this paper generating frequent item sets are discussed: Apriori and FP-growth algorithm. In apriori algorithm candidates are generated and testing is done which is easy to implement but candidate generation and support counting is very expensive in this because database is checked many times. In the fp-growth, there is no candidate generation and requires only 2 passes over the database but in this the generation of fp-tree become very expensive to built and support is counted only when entire dataset is added to fp tree. The comparison of these algorithms are present as in this paper which shows better performance.

Copyright © 2018, Banumathi and Dhurga Devi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Citation: Dr. Banumathi, A. and Dhurga Devi, M. 2018. "Comparative Study Of Frequent Itemset Mining Algorithms Apriori And FP Growth", International Journal of Development Research, 8, (05), 20596-20600.

### INTRODUCTION

In recent years amount of data in the database has increased rapidly. The increasing size of the database has led to growing interest in extraction of useful information from the bulk of data. Data mining is a technique useful for attaining useful information from vast databases. Implicit information within a database can be very useful in tasks such as marketing, financial forecast etc. This information has to be derived efficiently. Frequent itemset mining discovers significant relationships among variables or items in a dataset. Association rule mining (Ritu gang) searches for relationships between items in a dataset. It finds association among set of items in transactional database. Each transaction is a list of items. Association rules (Aimammoyaid, 2009) is in form  $A \Rightarrow B$  which means customer buys A also tends to buy B. To mine association rule, basic concepts of support and confidence are needed. Support  $s$  is the probability that a transaction contain  $(X, Y)$ . Confidence  $C$  is the measure of the strength of the association rule, suppose the confidence of the association rule  $x \Rightarrow y$  is 90%, it means that 90% of the transactions that contain  $X$  also contain  $Y$  together. Also minimum support and minimum confidence is needed to eliminate the unimportant association rules.

Such that the association rules is hold when it is greater than the minimum support and minimum confidence.

Equation for support and confidence:

Support  $(A \Rightarrow B) = \text{Probability}(A \cap B)$   
Confidence  $(A \Rightarrow B) = \text{Probability}(B/A)$ .

### Apriori Algorithm

The apriorialgorithm (Agarwal 2001) is firstly proposed by R. Aggarwal for mining frequent itemset. In data mining, Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

Apriori algorithm follows two phases:

#### Generate Phase

In this phase candidate  $(k+1)$ -itemset is generated using  $k$ -itemset; this phase creates  $C_k$  candidate set.

#### Prune Phase

In this phase candidate set is pruned to generate large frequent itemset using "minimum support" as the pruning parameter.

\*Corresponding author: Dhurga Devi, M.,  
Student of Cs Department, Govt Arts College, Karur, Tamil Nadu 639007.

This phase creates Lk large itemset Fig 1 shows the pseudo code for apriorialgorithm:

**Apriori\_Algo (L,C,k)**

**Pass 1**

- Generate the candidate itemsets in C1 2.
- Save the frequent itemsets in L1

**Pass k**

- Generate the candidate itemsets in Ck from
- the frequent itemsets in Lk-1
  - Join Lk-1 p with Lk-1q, as follows:

**Insertinto Ck**

**select** p.item1 , p.item2 , . . . , p.itemk-

l,q.itemk-1 **from** Lk-1 p, Lk-1q

**where** p.item1 = q.item1 , . . . p.itemk-

2 = q.itemk-2 , p.itemk-1 < q.itemk-1

- Generate all (k-1)-subsets from the candidate itemsets in Ck
- Prune all candidate itemsets from Ck where some (k-1)-subset of the candidate itemset is not in the

**Frequentitemset Lk-1**

Scan the transaction database to determine the support for each candidate itemset in Ck

Save the frequent itemsets in Lk.

- Suppose min. support count required is 2 (i.e. min\_sup = 2/9 = 22 % )

**Table 1. Database containing 9 transactions**

TID	List of items
T100	I1 ,I2 ,I5
T200	I2 ,I4
T300	I2 ,I3
T400	I1 ,I2 ,I4
T500	I1 ,I3
T600	I2 ,I3
T700	I1 ,I3
T800	I1 ,I2 ,I3 ,I5
T900	I1 ,I2 ,I3

**Step 1:** Count the number of transactions in which each item occurs (Table 2.a)

**Step 2:** In this step we remove all the items that are bought less than 2 times from the table (Table 2.b)

**C1**

**Step 3:** Make all the pairs of items by using property JOIN L1 with L1and count how many times each pair is bought together (Table 3.a)

**Step 4:** Remove all the item pairs with number of Transactions less than two (Table3.b)

**Table 2. First scan of Apriori (Scan for count of each candidate)**

**Table (2.a)**

Item set	Sup-Count
I1	6
I2	7
I3	6
I4	2
I5	2

**Table (2.b)**

Item set	Sup-Count
I1	6
I2	7
I3	6
I4	2
I5	2

**C2**

**Table (3.a)**

Item set	Sup-Count
I1,I2	4
I1,I3	1
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

**L2**

**Table (3.b)**

Itemset	Sup-Count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

Table 3: The second scan of A-priori (Generate C2 and Scan D for count of each Candidate).

**Step 5:** To make the set of three items we need one more rule (it's termed as self-join).

It simply means, from the Item pairs in the above table, we find two pairs with the same first Item.

**C3**

**Table (4.a)**

Itemset	Sup-Count
I1,I2,I5	2
I1,I2,I4	1
I1,I2,I3	2
I2,I3,I4	0
I3,I4,I5	0

**L3**

**Table (4.b)**

Itemset	Sup-Count
I1,I2,I3	2
I1,I2,I5	2

Table 5: The third scan of A-priori (Generate C3 and Scan D for count of each Candidate)

- While we are on this, suppose you have sets of 3 items say ABC, ABD, ACD, ACE, BCD and you want to generate item sets of 4 items you look for two sets having the same first two alphabets.

ABC and ABD -> ABCD  
ACD and ACE -> ACDE

**Step 6:** According to above statement I1, I2, I3, I5 is generated whose minimum support is less than 2. so this is not frequent.

Thus the set of three items that are bought together most frequently are I1, I2, I3 and I1, I2, I5.

#### ADVANTAGES

- Use large itemset.
- Easy to implement.
- Easily parallelized.

#### Disadvantage

- It may need to generate a huge no of candidate sets.
- Assumes transactional database is memory resident.
- Support count is expensive because require many database scan.

#### Fp-growth algorithm

The FP-Growth Algorithm [Han, 2001], proposed by Han in, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns [Han, 2001; Ritu gang; Jagrati Malviya, 2015] e.g. the Apriori Algorithm

Major steps in FP-growth is

**Step1-** It firstly compresses the database showing frequent item set in to FP-tree. FP-tree is built using 2 passes over the dataset.

**Step2:** It divides the FP-tree in to a set of conditional database and mines each database separately, thus extract frequent item sets from FP-tree directly. It consist of one root labeled as null, a set of item prefix sub trees as the children of the root, and a frequent .item header table. Each node in the item prefix sub tree consists of three fields: item-name, count and node link where--- item-name registers which item the node represents; count registers the number of transactions represented by the portion of path reaching this node, node link links to the next node in the FP- tree. Each item in the header table consists of two fields---item name and head of node link, which points to the first node in the FP-tree carrying the item name.

#### FP-Tree structure

The frequent-pattern tree (FP-tree)[6] is a compact structure that stores quantitative information about frequent patterns in a

database. Han defines the FP-tree as the tree structure defined below:

#### One root labeled as “null” with a set of item-prefix subtrees as children, and a frequent-item-header table

- Each node in the item-prefix subtree consists of three fields: Item-name: registers which item is represented by the node;
- **Count:** the number of transactions represented by the portion of the path reaching the node;
- **Node-link:** links to the next node in the FP-tree carrying the same item-name, or null if there is none.

#### Each entry in the frequent-item-header table consists of two fields

- **Item-name:** as the same to the node;
- **Head of node-link:** a pointer to the first node in the FP-tree carrying the itemname.

The FP- Growth algorithm for mining frequent patterns using FP-Tree is follows:

**Input:** A transaction database (D) and minimum support threshold ( $\xi$ ).

**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 BUA with support = minimum support of nodes in B;

else (for each  $a_i$  in the header of Tree) do

{  
generate pattern B =  $a_i \cup A$  with support =  $a_i$ .support;  
construct B's conditional pattern base and then B's  
conditional FP-Tree Tree B;

if (Tree B  $\neq \emptyset$ )

{  
call FP-growth (Tree B, B) } }

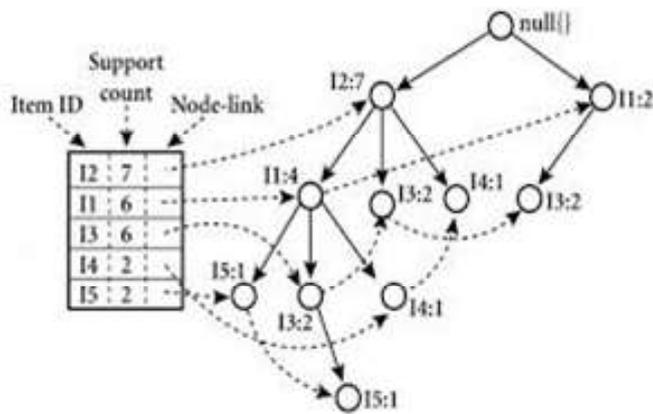
Let us create the FP-tree for the example from Table 1:

- First we scan the database and determine the set of frequent items (1-itemsets) and their support counts(frequencies):  
 $L = \{\{I2:7\}, \{I1:6\}, \{I3:6\}, \{I4:2\}, \{I5:2\}\}$
- Then we create the root of the FP-tree and label it with “null”
- We take each transaction, sort the items according to descending support count, and create a branch for it. For example the scan of the first transaction “T100:I1, I2, I5”, which contain tree items: I2, I1 and I5 in sorted

descending, leads to the construction of the first branch of the tree: (I2:1), (I1:1), (I5:1).

- The second transaction T200 contains the items I2 and I4. This would result a branch where I2 is linked to the root and I4 is linked to I2. However this branch would share a common prefix, i2, with the existing path for T100. Therefore we instead increment the count of the I2 node by 1 and create a new node (I4:1), which is linked as a child of (I2:2).

In general when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1 and nodes for the items following the prefix are created and linked accordingly. To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. In this way the problem of mining frequent pattern in database is transformed to that of mining the FP-tree.



FP-tree

The FP-tree is mined as follows: Start from each frequent length-1 pattern, as an initial suffix pattern, construct its conditional pattern base, a sub-database, which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern, then construct its conditional FP-tree and perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

The following table shows the frequent pattern generated for each node:

Conditional Item Pattern BaseFp-Tree Frequent pattern Generated

- {{I2, I1:1}, {I2, I5:2}, {I1, I5}} {I2, I1(I2:2, I1:2)I5:2}, {I2, I3:1}} {I1, I5:2}
- {{I2, I1:2}, I4}} {I2:1}} (I2:2) {I2, I4:2}
- {{I2, I1:2} (I2:4, I1:2), {I2, I3:4}, {I1, I3}} {I2:2}, (I1:2), I3:4}, {I2, I1, I3:2}, {I2, I1:2}, (I2:4)I1:4}
- I1 {{I2:4}} (I2:4) {I2, I1:4}

ADVANTAGES

- It compresses the database.
- Require only 2 pass over database.
- There is no candidate generation.
- Faster than apriori.
- Reduces search cost

Disadvantage

- It may not fit in main memory.
- FP tree is expensive to build.
  - takes time to build but once built frequent itemset can be obtained easily
  - Support can only be calculated once the entire dataset is added to fp-tree.

Comparison of apriori and fp-growth algorithms

Parameters Apriori FP-growth

Algorithm

Technique Use AprioriIt constructs property and join conditional frequent prune property pattern tree and conditional pattern base from database which satisfy minimum support. Memory Due to large no Due to compact Utilization of candidate structure and no generation candidate require large less memory memory space. Space Number of Multiple scans for Scan the database Scans generating candidate only twice set. Time Execution time is Execution time more as time is lesser than the Wasted in producing Apriori algorithm candidate every time.

Conclusion

Frequent itemset mining is an important task in association rule mining. It has been found useful in many applications like market basket analysis, financial forecasting etc. We have discussed about classical algorithm Apriori and Fp growth using thses approach ,going to all candidate itemset for each level has to be discovered , the length of the frequent itemset, more the number of candidate generation. Projected tree method is efficient in terms of speed but utilizes more space. These disadvantages can be overcome by using techniques like hashing, partitioning etc. In this paper study of item set mining algorithms is done and on the basis of that study comparison is given between them.

REFERENCES

Agarwal, R., Aggarwal, C. and Prasad, V.V.V. 2001. A tree projection algorithm for generation of frequent itemsets. Journal of Parallel and Distributed Computing, 61:350–371.

Aimammoyaid said, Dr. Dominic, P.D.D., Dr. Azween B Abdullah. 2009. Comparative study if FP. Growth Variations, IJCSNS, Vol9, May.

Algorithms. IJCSNS.

character set that apriori and Fp-Growth Frquent.

Cornelia Gyorödi and Robert Gyorödi. A Comparative Study of Association Rules Mining Algorithms.

Han, J., Kamber, M. 2001. "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, USA, ISBN 1558604898.

Jagrati Malviya, Anju Singh, Bu Bhopa, 2015. An FP tree based approach for extracting frequent parrern from large database by applying parallel and partition projection,IJCA(0975-8887) Vol 144-no.18, Mar.

Jeff Heaton, Ft-lauderdale ,Comparing dataset

Kavitha, M., Tamilselvi, S.T. Comparative Study on Aprio Algorithms And Fp Growth Algorithm With Pros and Cons.

Rajesh, K. Ahir, 2013. A Comparative Study of Algorithms for Mining Frequent Patterns, International Journal of Computer and Communication Engineering Vol 2, issue 12, Dec 2013.

Ritu gang, Preethigulia, Comparative study of Frequent Itemset mining algorithms Apriori and FP-Growth, International journal of computer application(0975-8887).

Santhosh Kumar, B.and Rukmani. K.V. Implementation of Web Usage Mining Using APRIORI and FP Growth

\*\*\*\*\*