# FREQUENT PATTERNS FOR MINING ASSOCIATION RULE IN IMPROVED APRIORI ALGORITHM

**Ms. Jyoti B. Deone, Asst. Prof. Vimla Jethani**

**ABSTRACT:**
An important aspect of data mining is to discover association rules among large number of item sets. Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. The main problem is the generation of candidate set. In this thesis we have presented a different algorithm for mining frequent patterns in large datasets using transposition and Boolean matrix of the database with modification of the Apriori-like algorithm. The main advantage of this method is that the database stores in different formats in each iteration. Database is filtered and reduced by generating the transaction ID for each pattern. In proposed method we describe the weighted Apriori algorithm, in that we collect the large amount of items divided and categorized into groups that is use of bitpartition technique to reduce the huge computing time and also to decrease the database size. Due to this the efficiency of algorithm increases.

**INDEX TERMS**
Data mining, association rule, candidate set, Apriori algorithm.

## I. INTRODUCTION

Data mining is used to deal with size of data stored in the database, to extract the desired information and knowledge [1]. Data mining has various techniques to perform data extraction. Association technique is the most effective data mining technique among them. It discovers hidden or desired pattern among the large amount of data. It is responsible to find correlation relationships among different data attributes in a large set of items in a database. Since its introduction, this method has gained lot of attention. Author of [1] has analyzed that an association analysis [5] is the discovery of hidden pattern or clause that occur repeatedly mutually in a supplied dataset. Association rule finds relations and connection among data and datasets given. An association rule is a law which necessitates certain relationship with the objects or items. Such association's rules are calculated from the data with help of the concept of probability.

## II APRIORI ALGORITHM

The algorithm [5] is designed to find associations in sets of data in a database. You find this algorithm dealing with transactions (batches of operations that must all happen together) and it attempts to find subsets that reach a certain value. For example, out of 10000 transactions, how many times did person 'A' buy an item under 10 dollars but not a toothbrush. Before going in to details of Apriori algorithm we will first see the definitions of some common terminologies which are used in algorithm [3].

### II.I ITEMSET

Itemset is collection of items in a database which is denoted by I = {i1, i2, …, in}, here 'n' is the number of items.

### II.II TRANSACTION

Transaction is a database entry which contains collection of items. Transaction is denoted by and $T \subseteq I$.

A transaction contains set of items

T= {i1, i2, .., in}.

### II.III MINIMUM SUPPORT

Minimum support is the condition which should be satisfied by the given items so that further processing of that item can be done. Minimum support can be considered as a condition which helps in removal of the in-frequent items in any database. Usually the Minimum support is given in terms of percentage.

### II.IV FREQUENT ITEMSET (LARGE ITEMSET)

The itemsets which satisfies the minimum support criteria are known as frequent itemsets. It is usually denoted by 'Li' where 'i' indicates the i-itemset.

**II.V CANDIDATE ITEMSET**

Candidate itemsets are items which are only to be considered for the processing. Candidate itemset are all the possible combination of itemset. It is usually denoted by 'Ci' where 'i' indicates the i-itemset.

**II.VI SUPPORT**

Usefulness of a rule can be measured with the help of support threshold. Support helps us to measure how many transactions have such itemsets that match both sides of the implication in the association rule. Consider two items A and B. To calculate support of A->B the following formula is used,

Supp(A->B) = (number of transactions containing both(A&B) (Total number of transactions)

**II.VII CONFIDENCE**

Confidence indicates the certainty of the rule. This parameter lets us to count how often a transaction's itemset matches with the left side of the implication with the right side. The itemset which does not satisfies the above condition can be discarded. Consider two items A and B. To calculate confidence of A->B the following formula is used,

Conf(A->B)=(number of transactions containing both A& B) (Transactions containing only.

Note: Conf(A->B) might not be equal to conf (B->A).

**II.VIII APRIORI ALGORITHM**

Apriori algorithm works on two concepts-

a. Self Joined

b. runing.

Apriori uses a level wise search where K-itemsets are used to find (K+1)-itemset.

1) First, the set of frequent1-itemsets is found which is denoted by C1.

2) The next step is support calculation which means the occurrence of the item in the database. This requires scanning the complete database.

3) Then the pruning step is carried out on 'C1' in which the items are compared with the minimum support parameter. The items which satisfy the minimum support criteria are only taken into consideration for the next process which is denoted by 'L1'.

4) Then the candidate set generation step is carried out in which 2-itemsets are generated this is denoted byC2.

5) Again the database is scanned to calculate the support of the 2-itemset. According to the minimum support, the generated candidate sets are tested and only the itemset which satisfies the minimum support criteria are further used for 3-itemset candidate set generation.

6) This above step continues till there is no frequent itemset or candidate set that can be generated. We can easily understand the concepts used by the Apriori with the help of following example.

**II.IX DRAWBACKS OF APRIORI ALGORITHM**

- Large number of in-frequent itemsets are generated which increases the space complexity.
- Too many database scans are required because of large number of itemsets generated.
- As the number of database scans are more, the time complexity increases as the database increases.
- Due to these drawbacks, there is a necessity in making some modification in the Apriori Algorithm which we will see further in this paper.

In traditional Apriori meaningless frequent itemset exist that increases the database scans and requires lots of storage space. Dividing itemsets into broad categories and setting the weighted values of categories, we can calculate the weighted support and confidence. Further steps like pruning and selection can be done according to the minimum weighted support and confidence.

**III. LITERATUREREVIEW**

**III.I APRIORIALGORITHM BASED ON TRANSPOSE DATABASE:**

This algorithm mainly concentrated on:

(1) For reducing frequent itemset and

(2) For reducing storage space as well as the computing time.

In the case of large datasets like Wal-Mart datasets, this algorithm is very much useful for reducing the frequent patterns and also reducing the database size for every subsequent passes.

For example: In this Algorithm, the number of occurrence of frequent k-itemsets when k- item sets are generated from (k-1)-itemsets is computed. If 'k' is greater than the size of the database 'D', there is no need to scan database 'D' which is generated by (k-1)-itemsets according to the Apriori property and it can be remove automatically. Below table shows the notations used during the 3 Algorithms specified here:

**A. TRANSPOSE DATABASE [6]:**

A simple approach is if we implement in transposed database, then result is very fast. Recently, different works proposed a new way to mine patterns in transposed database where a database with thousands of attributes but only tens of objects. In Apriori Algorithm each phase is counting the support of prune pattern candidate from database. Transposed database used for frequent pattern mining in which database represented in transposed form. And for counting the support we find out by longest common subsequence is stored or update in database so that next time instead of whole transaction we search from these filter transaction string.

**ALGORITHMIC STEPS ARE DESCRIBED AS BELOW [6]:**

1. First the function Apriori-gen (LTk-1) is called and to generate candidate k-transaction set by frequent k- transactions.
2. Checking whether candidate transactions *CT* are joined into candidate k-transactions or not. It proceeds by calling function recursively has infrequent transactions (ct, LTk-1). If it is true, it means the set of transactions are not frequent and should be removed. Otherwise, scan database DT.
3. The occurrence of frequent k-transaction is computed by generating (k-1)-transactions from k-transactions. If k-transaction is greater than the size of database DT, it is not needed to scan database DT which is generated by (k-1)-transactions based on the lemma 1, and it can be deleted.
4. If the size of database DT is greater than or equal to 'k', then call function *subset* (CTk, dt), which computes frequent pattern using a subsequent iterative level-wise approach based on candidate generation.

**ALGORITHM 1: ALGORITHM**

Input: A transposed database DT and the user defined minimum support thresholds.
Output: The complete set of frequent patterns
Step 1: Convert Database D into transpose form DT
Step 2: Compute *CT1* candidate transaction sets of Size-1 and finds the support count.
Step 3: Compute the large transaction sets (*LT*) of size-1 (i.e. for all *CT1* is greater than or equal to minimum support.)
*LT1* = {Large 1-transactionset (*LT*)};
For (k=2;*LTk-1*= 0;k++) do
Begin
*CTk*= *Apriori-gen*(*LTk-1,ct*);
//new candidate transaction sets
End
Return*LT*= ∪*kLTk*;

**ALGORITHM 2: APRIORI-GEN(LTK-1), GENERATE CANDIDATE SETS**

For all transactions p∈*LTk-1*do begin
For alltransactions q∈*LTk-1*do begin
If p.transaction1= q.transaction,…
p.transactionk-2=q.transactionk-2
p.transactionk-1< q.transactionk-1 then
begin
ct=p∞q;
If *has_infrequent_subset*(*ct, LTk-1*) then
deletect;
Else
For all transaction set t∈DT do begin
If count (t)<k then deletet;
Else begin
Ct=subset(CTk, t);
End; End
For all candidate transactions ct∈CTi do begin
*CT.count*= *CT.count*+1;
End; End;
*LTk*={*ct*∈*CTk*|*CT.count*≥*s*};
End; End;
End; End;
ReturnCTk;

**ALGORITHMS 3: HAS_INFREQUENT_SUBSET(CT, LTK-1)**

//checking the elements of candidate generation
For all(k-1)-sub transaction set of ct do
Begin

If t∈LTk-1 then return true;

else return false;

End.

The main advantage of the proposed algorithm for frequent patterns discovery are, it reduces the size of the database after second pass and, the storage space and saves the computing time.

## IV PROPOSED ALGORITHM

Association rule mining is an important task in data mining. Association rules are frequently used by retail stores to assist in marketing, advertising, floor placement and inventory control. Most of the association rule mining algorithms will not consider the weight of an item. Weighted association is very important in KDD. In analyzing market basket analysis, people often use Apriori Algorithm, but Apriori generates large number of frequent itemsets. One alternate approach to Apriori is partitioning technique. The proposed method to find weighted frequent itemsets using partitioning and bitvector. By example, it is proved that partitioning technique can improve the efficiency by reducing the number of candidates.

### IV.I IMPROVED ALGORITHMFOR WEIGHTED APRIORI:-

There are various methods to improve Apriori Algorithm efficiency. 1) Hashbased itemset counting 2) Transaction reduction, 3) Partitioning, 4) Sampling, 5) Dynamic itemset counting.

Weight of each item represents the importance of the item in the transaction database. Weighted association rules are derived from this weighted items based on some methods. This method to find the weighted frequent itemsets using Partitioning method.

Weight of an item: The weight of each item is assigned to reflect the importance of each item in the transaction database.

Weighted support Degree): weighted support of rule

$X \Rightarrow Y$

$$\left( \sum_{Ti \in (x \cup y)} \frac{wi}{K} \right) \left( (s(x \rightarrow y) \right)$$

$$\square i \; )(s(x \rightarrow y))$$

### DATA PARTIONING APPROACH FOR FREQUENT ITEMSET-

A Partitioning Technique can be used that requires just two scans to mine frequent itemsets. The main goal of this division of process into local and global computation is to process one fragment in the main memory at a time, to avoid multiple scans over D from secondary storage. Partitioning may improve the performance of finding large itemsets in several ways.

### ALGORITHM FOR WEIGHTED FREQUENT ITEMSETS

Supermarkets/marts use multi-dimensional data and the items are categorized into several classes based on the type of transactions. Here we are taking 9 types of transaction. Set weights to each category of transactions based on importance of items.

Consider the transaction database in above table and assign the weights to each transaction.

Algorithm:

Input: transactional data base *D*, weight of item *Wi*, support S, Weighted Support *Ws*.

Output: weighted frequent item sets

Method:

| Category of transaction | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 |
|---|---|---|---|---|---|---|---|---|---|
| Weight assigned | 1 | 0.9 | 0.7 | 0.8 | 0.7 | 0.9 | 0.6 | 1 | 0.9 |

Table 4.1 –Weight assigned to each transaction

**Step 1:** Read the transactional data base *D*, and map into Boolean

**Step 2:** Partition the transactional database using skipping fragments. Apply the method from step3to9 for each cluster.

**Step 3:** Find bitvector of each item in a cluster.

**Step 4:** If support (Si) <minimum support(S) then Prune item. Items which satisfy minimum support are frequent itemsets.

**Step 5:** Perform AND operation among the bitvectors of each 1 frequent itemsets to generate candidate itemsets i.e. to generate 2 itemsets.

Step 6: Find weighted support of each itemsets(X).

Weighted support(X) = (WI+WJ+------Wn)/

Pattern length * Support

**Step 7:** if weighted support(X)<=minimum weighted support prune corresponding itemset.
Step 8: obtain the 3- weighted frequent item sets in similar fashion.
**Step 9:** continue the same operation till no frequent items exist.
**Step 10:** combine local weighted frequent item sets in each cluster and weighted frequent item sets which
1. And let minimum support is 2. Map the above transactional data base into Boolean Matrix
2. Using below formulas, we have to calculate minimum support.
Supp(x,y)=$\sum Wi$(Ti∈(X⊆Y)/K)(S(X→Y))w minsup
Conf(X,Y)=$\sum Wi$(Ti∈(X⊆Y)/K)(S(X→Y))w minsup
3. Map the above transactional database into Boolean Matrix:

| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Table 4.2-Transactional database into Boolean matrix

4. Partition the transactional database into 2 Clusters:
Cluster1:

| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Table 4.3 – Cluster1data

Cluster2:

| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

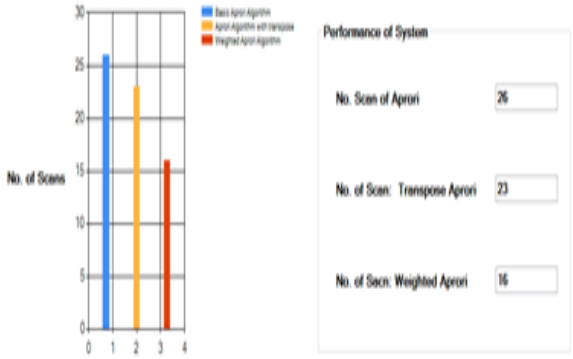
Table4.4-Cluster2 data

5. Apply Algorithm: Cluster1:
1. Prune items as T1, T3 not satisfying minimum support {2};
2. Find bitvectors of each item:
BV {T2}:00010011
BV {T4}:01000010
BV {T5}:11100100
BV {T6}:10110010
BV {T7}:01110110
BV {T8}:10001000
BV {T9}:00010001
3. T2, T4, T5, T6, T7, T8, T9 are frequent 1 itemsets. Perform bitwise AND operation among 1 itemsets to get 2 itemsets.
I. BV {T2}:00010011 ^ BV {T4}:01000010:
00000010 weighted support
Here we calculate weighted support by formulas:
W.S=(0.9+0.8)/2 ×1=8.5%so weighted support of T2, T4=8.5%
II. BV {T2}:00010011 ^ BV
{T5}:11100100:00000000
W.S= (0.9+0.7)/2 X0=0% T2, T5=0%
III. BV {T2}:00010011 ^ BV
{T6}:10110010:00010010
W.S= (0.9+0.9)/2 X2=18% T2,
W.S= (0.9+0.9)/2 X2=18% T2,
T6=18%T2, T7=15%
T2, T8=0%T2, T9=18%
T4, T5= 7.5%T4, T6=8.5%
T4, T7=14% T4, T8=0%
T4, T9=0%T5, T6=16%
T5, T7=19.5% T5, T8=8.5%
T5, T9=0% T6, T7=22.5%
T6, T8=9.5%
T6, T9=0%
T7, T8=0%
T7, T9=7.5% T8, T9=0%
4. Item set (T2, T5) ,(T2, T8) ,(T4,T8), (T4,T9), (T5,T9), (T6,T9), (T7,T8), (T8,T9) having less Minimum support percentage hence it will be pruned.
5. Now find the weighted 3itemsets. sets.

## V. EXPERIMENTAL ANALYSIS

To evaluate the efficiency and effectiveness of the improved algorithm, we performed an extensive study of two algorithms: Apriori-like with transpose and weighted Apriori algorithm, on both real time synthetic datasets with different ranges. All the experiments were examined on Pentium IV machine 1 GBRAM, running Microsoft Windows 7. Two algorithms, Apriori and weighted Apriori algorithm were implemented in .net framework. Also we got the real time Wal-Mart database with 2280 itemsets and 4200 elements. The running scan comparison between improved algorithm and Apriori algorithm are shown in the Figure with minimum support ranges from 1 percentage (%) to 5 percentages (%).




The importance of weighted Apriori algorithm is to reduce the number of items in each and every scan and also reduce the size of the original dataset. There are three aspects to make this algorithm better than the original one. Firstly, when the candidates are being produced, instead of dealing with all the items of the previous large set, only the elements which having the same transaction IDs are crossed. At the same time, generating frequent patterns, it may reduce the computing time dramatically and the size of the database is reduced. Secondly, by pruning, the number of elements in the candidate sets is decreased once more based on modified database. Finally, the computing time and storage space are saved.

## VI. CONCLUSION AND FUTURE WORK

In this paper we have discussed on different Apriori Algorithms which provides a new technique in generating rules. These Algorithms are more efficient than the traditional Algorithm and provide faster results in terms of time complexity. Comparison has been made which discusses on various attributes. Association rule mining will be an important aspect as data in the world is increasing day by day. The main advantages of transpose data base DT to reduce the number of scanning and redundancy by the time of generating sub-transaction set tests and verifying them in the database. In order to discover frequent patterns in massive datasets with more columns than rows, it has been presented a complete framework for the transposition; the itemset in the transposed database of the transposition of many classical transactions is given. One more technique we describe i. e. Matrix Apriori in this we use binary matrix which to reduce the database scans and it store data in 2-D array format. Also it has been compared the classical Apriori Algorithm with a proposed Weighted improved Algorithm. In this Algorithm we use bitpartition technique and apply this algorithm on group of data due to this improve the efficiency and less scan required as compared to all other algorithms. Hence, the proposed Algorithm is very much suitable for a massive datasets.

## VII. REFERENCE

[1] Pranay Bhandari, K. Rajeswari, Swati Tonge, Mahadev Shindalkar improved Apriori Algorithms-Survey.

[2] D. Gunaseelan, P. Uma College of Computer & Information Systems JAZAN University, Kingdom of Saudi Arabia – An Improved Frequent Pattern Algorithm for Mining Association Rules – International Journal of Information and Communication Technology Research in 2012.

[3] Agrawal R, Imielinski T, Swami A, ―Mining Association Rules between Sets of Items in Large Databases, ‖ In: Proc of the ACM SIGMOD International conference on Management of Data, Washington DC, 1993, pp, 207-216.

[4] Pranay Bhandari, Rajeswari, Swati Tonge, ‖ Improved Apriori Algorithm‖ in 2012 computer science department.

[5] International Institute for Infrastructural, Hydraulic, and Environmental Engineering, P. O. Box 3015, 2601 DA Delft, The Netherlands, ”Predictive Data Mining : Practical Examples ‖ in march 2000.

[6]A Novel Approach to Mine Frequent Itemsets of Process Models for Dyeing Process using Association Rule Mining – International Journal of Computational Engineering Research ||Vol, 03||Issue, 5||.

[7] V. Vaithiyanathan, Swati Tonge, Rashmi Phalnikar, ‖ Mining Association Rules using Hash Table‖ International Journal of Computer

Applications (0975–8887) Volume 57–No.8, November2012.
[8] A. B. M. Rezbaul Islam and Tae-Sun Chung– An Improved Frequent Pattern Tree Based Association Rule Mining Technique ‖ ,IEEE, International Conference on Information Science and Applications (ICISA), 2011
[9] R Agrawal and R Srikant ― Fast Algorithm for Mining Association Rules ‖ Proceedings of VLDB conference pp 487– 449, Santiago, Chile, 1994.
[10] Bayardo R. J., 1998 efficiently mining long patterns from databases. SIGMOD'98: Proceedings of the 1998 ACMSIGMOD international conference on Management of data, Seattle, USA, pp. 85-93.
[10] Han J., Kamber M and Pei, Data Mining Concepts and Techniques, the Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, July2011.

**Ms. Jyoti Deone**
M.E Student (Computer Engineering),
RAIT College of Engineering,
Navi Mumbai.

**Ms. Vimla Jethani**
Asst. Professor (Computer Engineering),
RAIT College of Engineering,
Navi Mumbai.