'FUTURIZM-2014, (14TH & 15TH March 2014), 'P.D.E.As College of Engineering, Manjari[Bk]', 'IIP Cell,'Manjari Road, Hadapsar, Pune-412307. India.', 'Chief Editor:Dr.K.R.Harne', Editors: Prof R V Patil, Prof Niraja Jain

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-11 , August, 2014

REVIEW ON MATRIX BASED EFFICIENT APRIORI ALGORITHM

Sachin Nagpure, Nitesh Fender, Preeti Deshmukh
Priyadarshini College of Engineering, RTM Nagpur University, Nagpur, India.
Department of Computer Technology, Priyadarshini College of Engineering, Nagpur, India
(sachinvnagpure@gmail.com, niteshfender@gmail.com, preeti.deshmukh8@gmail.com)

***Abstract* - These Apriori Algorithm is one of the well-known and most widely used algorithm in the field of data mining. Apriori algorithm is association rule mining algorithm which is used to find frequent itemsets from the transactions in the database. The association rules are then generated from these frequent itemsets. The frequent itemset mining algorithms discover the frequent itemsets from a database. However, running the frequent itemset mining algorithms with every update in the database is inefficient and is the foremost issue for research. This problem is often referred as the dynamic update problem of frequent itemsets. One of the approach is to dynamically mine the frequent itemsets. In this paper, dynamic frequent itemset mining algorithm, which is called Matrix Apriori, is reviewed.**

***Keywords* - Apriori, association rule, frequent itemsets, data mining, candidates, support count, and confidence.**

I. INTRODUCTION

Data mining is used to discover patterns and relationships in data with emphasis on large databases. The purpose of data mining is to extract desired knowledge from the large database and using it to make useful decisions. Association rule mining is finding the frequent patterns, associations or correlations among the set of items or elements in the databases [2].

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). The algorithm is used to find frequent itemsets from transactions in databases and then for finding association rules from frequent itemsets [6].

Databases are updated continuously; when new transactions arrive to the database or some transactions are needed to be deleted from the database, the frequent itemset mining algorithms should be re-run in order to find up-to-date frequent itemsets. Since re-running the algorithms is time consuming, a dynamic frequent itemset mining algorithm, which is based on Matrix Apriori Algorithm, is proposed.

Apriori algorithm represents the candidate generation approach. It generates candidate (k+1) itemsets based on frequent k-itemsets. Apriori is a Breadth First Search Algorithm (BFS) [1].

Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence are used to form the association rules.

APRIORI ALGORITHM

The **Apriori Algorithm** is one of the best-known association rule mining algorithms. It uses prior knowledge of frequent itemset properties and runs an iterative approach called level-wise search. That is, k itemsets are used to explore (k +1) itemsets (they are called candidate itemsets before testing them against the database) by eliminating the candidates that do not satisfy the minimum support. This process terminates when no frequent or candidate set can be generated. The efficiency of the level-wise generation of frequent itemsets is improved by the Apriori Property: "All nonempty subsets of a frequent itemset must be frequent" [4]. By means of this property, many unnecessary candidate generation and support counting are eliminated. This algorithm needs to scan the database with each iteration [2].

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-6 , March, 2014

II. BASIC CONCEPTS

A. Association Rule Mining

Given a set of transactions T, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction. The goal of association rule mining is to find all rules having

TID	ITEMS
1	Mouse, keyboard
2	Mouse, mobile, headphone, charger
3	Keyboard, mobile, headphone, speakers
4	Mouse, keyboard, mobile, headphone
5	Mouse, keyboard, mobile, speakers

- ✓ Support ≥ minimum support threshold
- ✓ Confidence ≥ minimum confidence threshold [4].

B. Itemset

- ✓ A collection of one or more items

‹ Example: {mouse, keyboard, mobile}

- ✓ k-itemset - An itemset that contains k items [4]

Fig. 1. Table with five transactions and respective items.

C. Support Count (σ)

– Frequency of occurrence of an itemset in transactions
– E.g. σ({mouse, keyboard, mobile}) = 2

D. Support

- ✓ Fraction of transactions that contain an itemset

– E.g. s({mouse, keyboard, mobile}) = 2/5

E. Frequent Itemset

An itemset whose support is greater than or equal to a minimum support threshold [4]

F. Association Rule

The association rules X → Y with a minimum support and confidence, is the ratio of support of X **U** Y to support of X

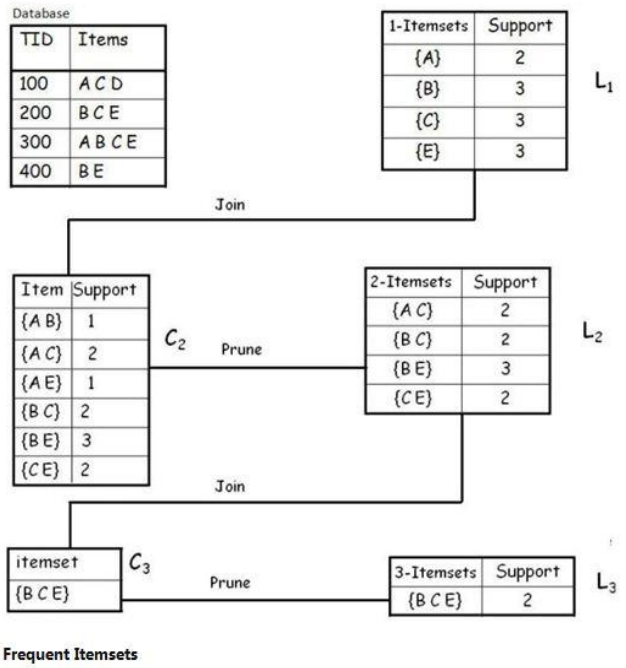
III. APRIORI ALGORITHM

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

It is a two-step approach:

1. Frequent Itemset Generation
   – Generate all itemsets whose support ≥ minimum support
2. Rule Generation
   – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset.



HOW APRIORI WORKS:

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-6 , March, 2014

- ✓ Find all frequent itemsets:
  - Get frequent items:
    - Items whose occurrence in database is greater than or equal to the minimum support threshold.
  - Get frequent itemsets:
    - Generate candidates from frequent items.
    - Eliminate the candidates that don't satisfy minimum support to find the frequent itemsets.
- Generate strong association rules from frequent itemsets:
  - Rules which satisfy the minimum support and minimum confidence threshold.

The algorithm scans the database for each transaction and for each item in the each transaction; all the items are considered as candidates for the first iteration and the frequent-1 candidates are generated with their support count. Then the items or candidates which satisfy the minimum support count are then added to frequent-1 itemset list and rests of the items are eliminated.

For second iteration, frequent-2 candidates are generated by making the join of frequent-1 itemsets with itself. Again these frequent-2 candidates are checked against their support count and the items which satisfy the minimum support count are added to frequent-2 itemset list and rests of the items are eliminated. In this way each iteration wise the frequent-n itemsets are generated. When there are no more frequent itemsets available in the database then the algorithm terminated.

Example:
A database has five transactions. Let the minimum support = 50% and minimum confidence = 80%.

Step 1: Find all Frequent Itemsets

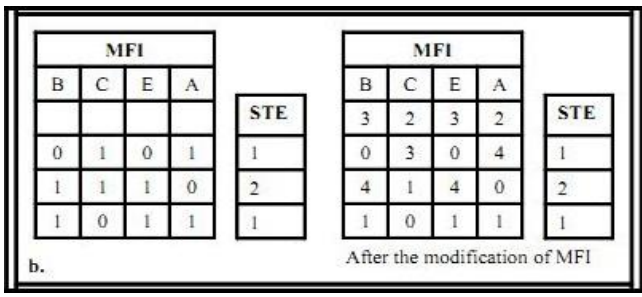
Solution:
Step 2: Generate strong association rules from the frequent itemsets

Rules	Support (X Y)	Support(X)	Confidence
{A} -> {C}	2	2	100
{B} -> {C}	2	3	66.66666667
{B} -> {E}	3	3	100
{C} -> {E}	2	3	66.66666667
{B} -> {C E}	2	3	66.66666667
{C} -> {B E}	2	3	66.66666667
{E} -> {B C}	2	3	66.66666667
{C} -> {A}	2	3	66.66666667
{C} -> {B}	2	3	66.66666667
{E} -> {B}	3	3	100
{E} -> {C}	2	3	66.66666667
{C E} -> {B}	2	2	100
{B E} -> {C}	2	3	66.66666667
{B C} -> {E}	2	2	100

The association rules whose confidence is 100 % are then added to final association rules.

IV. MATRIX APRIORI ALGORITHM

The **Matrix Apriori Algorithm** is a frequent itemset mining algorithm. In this algorithm the frequent items are stored in a matrix called MFI (Matrix of Frequent Items) and the supports of the patterns are stored in a vector called STE. Initially the database is scanned in order to determine frequent items. These items are sorted in a descending support count and trimmed to those that are above the minimum support value to create the frequent items list as in Figure (IV.a). The sorted frequent items list is the basis for the order of columns of the MFI. Subsequently, in the second scan, MFI and STE are built as follows [2].



International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-6 , March, 2014

The first row of the MFI is left empty. This row will be updated later in the modification. Therefore, inserting rows to MFI begins after this empty row. For each new transaction in the database, a row of zeros and ones is inserted according to the following rule.

The row is constructed by using the order in the frequent items list. For each item in the list, either "1" or "0" is added to the column of the row if the transaction contains the item or not. If the transaction is already included in the MFI, then it is not stored again in a new row, but its STE is incremented by "1".

After the construction of the matrix, it is modified in order to speed up the frequent itemset search. The first line of the MFI was left empty for this modification process. For each column, beginning from the first row, each cell value is set to the number of the row where the cell value is equal to "1" in the unmodified matrix. If there are not any values of "1" in the remaining rows of the unmodified matrix, the cell value is set to "1". In Figure (IV.b), the construction and modification of the matrix are shown [2].

After the matrix construction, frequent itemsets are found as in (IV.c). Beginning from the item that has the least support count; the item is compared with the items found on its left in order to find frequent itemsets. Following that, their support counts are calculated. The support count of an itemset is found by sequentially adding the related rows of STE from top to bottom.

Matrix Apriori works without candidate generation scans database only twice and uses Apriori Property. Also, the management of matrix is easy [2].

TID	Items
001	A C D
002	B C E
003	B C E
004	A B E

1-Itemsets	Support Count
{B}	3
{C}	3
{E}	3
{A}	2
~~{D}~~	~~1~~

Minimum support = 50%
Frequent items = { B, C, E, A}

a.

Frequent Itemsets	Support Counts
CE	2
BE	3
BC	2
BCE	2

c.

V. DYNAMIC MATRIX APRIORI ALGORITHM

TID	Items
005	C D
006	C F

Additions (*A*)

1-Itemsets	Support Count
{C}	5
{B}	3
{E}	3
{A}	2
{D}	2
{F}	1

Minimum support = 50%
IS_{new} = { C, B, E, A, D, F }

MFI				
B	C	E	A	D
3	2	3	2	2
0	3	0	4	1
4	1	4	0	0
1	0	1	1	0

Before additions

STE
1
2
1

a.

In order to provide dynamic mining of frequent itemsets, the matrix is constructed by the minimum support value i.e. 1. That means all items are kept in the MFI without considering their frequencies. So that, flexibility for support change is enabled as well. So finding frequent itemsets with any support threshold is easy. Since all items are kept in the MFI, frequent itemsets can be calculated from the item that satisfies the minimum support when the support threshold is changed.

The database is scanned and the support counts of items are calculated. All items are arranged in a descending order of support counts without considering if the item's support count is more than the minimum support or not. After that the MFI is constructed and then modified as in Matrix Apriori as in Fig(V.1.b).

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-6 , March, 2014

When new transactions arrive or some transactions are to be deleted, the modified MFI is updated and named as ISnew as shown in fig(V.1.a). First, the new transaction is checked whether it is existed in the MFI or not. If it exists, its STE is incremented by "1"; if it does not exist, it is added to the MFI. Adding to the MFI is done by setting the cell value to the row number of transaction where the "1" value is stored in the MFI. When the item does not exist in the remaining transactions of the incremental database, the cell value is set to "1". Finally, according to the change of the 1-itemset ordered list of items, the order of items in the MFI is changed as in Figure (V.2.c).

Figure V.1. Dynamic Matrix Apriori - Before Update.

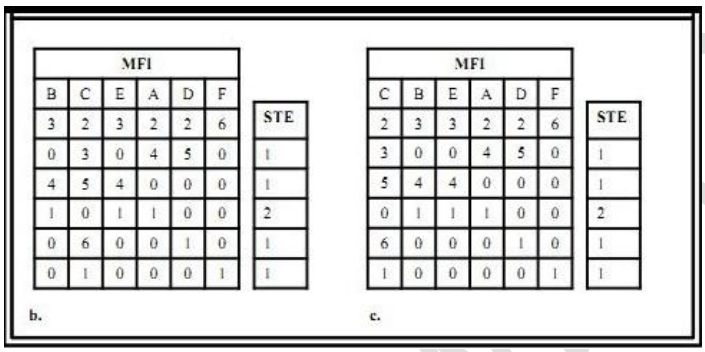


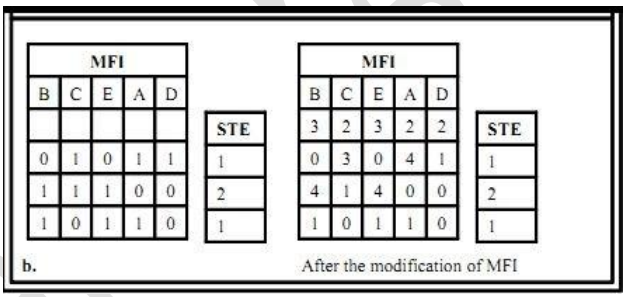
Figure V.2. Dynamic Matrix Apriori - After Update.

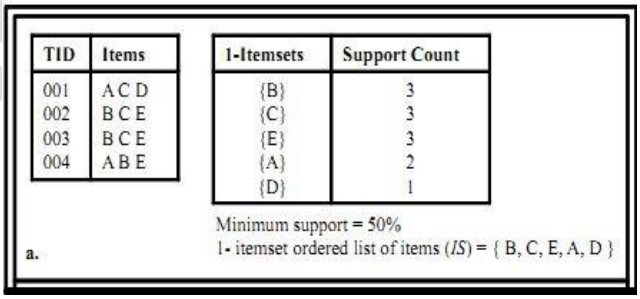
When new transactions arrive to the database, additions are scanned and 1-itemset ordered list is updated (in Figure V.2.a).

A new column is included to the MFI for the new item, F.

The MFI is searched whether the first transaction (CD) exists in the MFI or not. It does not exist in the MFI, so a new row is added to the MFI as the row number 5 and the MFI is updated as follows. Since the transaction contains C and D, the last MFI rows where these items have value "1" are found. For C, this is row number 3 and the cell value "1" is updated as "5". For D, this is row number 2 and the cell value "1" is also updated as "5". On row number 5 each item has value "1" for this transaction. For the following transaction, (C F), the procedure is repeated. This time value "1" of C on the row number 5 is updated as "6" and on row 6, the cell value "1" is set. F exists in this transaction for the first time. So the cell value of first row of F is set to "6" and the cell value of row number 6 is set to "1" (see Figure V.2.a and Figure V.2.b).

The columns of the MFI are rearranged according to the 1-itemset ordered list of items (see Figure V.2.c).





VI. CONCLUSION

Dynamic Matrix Apriori algorithm transformed the management of matrix into much easier way. It works without candidate generation, scans database only twice and uses Apriori Property. The need to scan the database each time is completely removed. The dynamic update problem of apriori algorithm is removed. Dynamic Matrix Apriori algorithm definitely has evolved in a great way and definitely has a long way to grow.

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-6 , March, 2014

REFERENCES

[1] Jaishree Singh , Hari Ram , Dr. J.S. Sodhi, Improving Efficiency of Apriori Algorithm Using Transaction Reduction International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013

[2] Damla Oguz "Dynamic Frequent Itemset Mining Based On Matrix Apriori Algorithm" June-2012 Graduate School of Engineering and Sciences of Izmir Institute of Technology

[3] Professor Anita Wasilewska APRIORI ALGORITHM Lecture Notes

[4] Data Mining Association Analysis: Basic Concepts and Algorithms Introduction to Data Mining by Tan, Steinbach, Kumar

[5] K. Geetha, Sk. Mohiddin, "An Efficient Data Mining Technique for Generating Frequent Item Sets", In: Proceeding of IJARCSSE, ISSN 2277-128X, Vol. 3, Issue 4, April 2013.

[6] Agrawal, R., T. Imieli ´ nski, and A. Swami (1993). Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data

[7] Sunita B.Aher, Lobo L.M.R.J.,"A Comparative Study of Association Rule Algorithms for course Recommender System in E-learning", In: Proceeding of IJCA, ISSN 0975-8887, Vol. 39-No.1, February 2012.

[8] Amornchewin, R. and W. Kreesuradej (2007, dec.). Incremental association rule mining

using promising frequent itemset algorithm. In Information, Communications Signal

Processing, 2007 6th International Conference on, pp. 1 –5.

[9] Oguz, D. and B. Ergenc (2012). Incremental itemset mining based on matrix apriori algorithm. In 14th International Conference, DaWaK 2012. Springer.

[10] Yildiz, B. and B. Ergenc (2010). Comparison of two association rule mining algorithms without candidate generation. In Proceedings of the 10th IASTED International Conference on Artificial Intelligence and Applications, SIGMOD