International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 32151-32159
© Research India Publications
http://www.ripublication.com

Matrix Approach of Apriori Algorithm using Subset Count Technique

T.Sasikala[1] and K.Vimal Reddy[2]

Assistant professor & UG Scholar
Department of Computer Science and Engineering,
Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Bangalore, India.
jivishnaa@gmail.com&kallemvimalreddy@gmail.com

Abstract

In present days, since the size of the datasets is being increased day by day, extracting knowledge from this huge data set has become a very big issue. Particularly speaking about transactional data set, there were many approaches to find the frequent item sets. Many classical algorithms and techniques of Apriori have been proposed and used to find the frequent item sets from the huge transactional dataset. The classical algorithms scans the database repeatedly for finding frequent item sets which generally takes more execution time. In this paper we propose a new improved matrix approach with subset count to find the frequent item sets. First the given data is converted to matrix form where the items with less than minimum support count and duplicate transactions are removed, secondly each transactions are scanned for item sets and if this item set is not present in frequent item sets list the we find the subsets of this item set and add them to subset count list by incrementing the count of a particular subset. If the count of any subset is greater than minimum support then the subset is added to frequent item sets list. In this paper we have compared the existing apriori algorithm with improved matrix approach based on execution time.

Key words-Apriori algorithm, frequent itemset, support count.

I. INTRODUCTION

Data mining is a process of extracting knowledge from huge datasets. This knowledge is intern used for decision making purpose. Data mining techniques are used to extract knowledge or information from huge set of transactional database. Association rule mining, is a technique under data mining, which is a two-step process. First step is finding the frequent item sets and second step is to discover the association rules from the frequent items obtained in first step. Association rule is used for discovering the interesting relationships among the frequent items. The Association rule is considered to be strong association rule if its support and confidence are greater than the user specified support and confidence. Support count is the total number of transactions containing the item sets and support is a value obtained by the support count value with the number of transactions. If A and B are two items in a transactional data set then support for this two items is given by, support (A, B)=P(AUB). Here we took only two-itemset, it can be either 1-itemset, 2-itemset,, …n-itemset, where n is the number of items in a dataset. If the itemsets support is greater than the minimum support then it is added to frequent item sets list. The association rule A=> B is said to have the confidence value of c, if c% of transaction in dataset that contains A also contains B. confidence is given by, confidence (A, B)=P(AUB)/P(A). Here in this paper we are focusing on first step of association rule mining, i.e., finding the frequent item sets.

II. LITERATURE SURVEY

For finding the frequent item sets, we have referred to many techniques such as candidate set generation approach which was proposed by Sreekanth and Agarwal [1], improved apriori algorithm based on matrix data structure [2]. Transaction reduction of apriori algorithm [3].Candidate set generation approach [1][5], is an iterative approach known as level wise search, where k frequent item sets are used to find (k+1) frequent item sets. It has two steps. First step is a join step, to find Lk a set of candidate k-itemsets, it is generated by joining L_{k-1} with itself. This set of candidates is denoted by C_k. Second step is the prune step, where the members of C_k whose count is less than the min support count will be removed.We have implemented the existing apriori algorithm using candidate set and compared with our improved matrix approach using subset count.

III. IMPROVED MATRIX APPROACH WITH SUBSET COUNT

Improved matrix approach with subset count has two phases: First phase is to find the compressed matrix (Final_Matrix) for the given data set and second phase is to generate the frequent item set using subset count approach.

A. Phase 1:

First the given data set is scanned transaction by transaction and is converted to first_matrix form containing transactions ID as rows and items as columns. The value corresponding to this transaction ID and item will be either 0 or 1, where 0 represents absence of the item in that particular transaction ID and 1 represents item is present in that particular transaction ID. Once the first_matrix is ready, check for the count of items. If the item count is less than min_suppcount then remove the particular item column from the first_matrix. Add one more column IC (Item Count) to the first matrix. Where IC represents the value with the number of items present in the particular transaction. Rearrange this first_matrix in the descending order of their Item Count (IC). Now the resultant matrix is named as second_matrix.

Add one more column named Transaction Count (TC) to the second_matrix. Delete all the duplicate transactions and keep only distinct transactions in second_matrix by incrementing the Transaction Count value of the distinct transactions by number of duplicate transactions. The resultant matrix is the compressed matrix of given dataset. It is called as final_matrix.

B. Phase 2:

Scanning the transaction/row in final_matrix, and check whether the item set of this transaction is present in Frequent Item Sets (FIS) list or not.If item set is not present in FIS list then, check if the Transaction Count (TC) value of a particular transaction is greater than or equal to the minimum support count (min_supp), if it is greater than or equal to the min_suppthen add all the subsets of the particular transaction to the Frequent Item Sets (FIS) list. If Transaction count Value is not greater than or equal to min_supp then find all the subsets and add them to the Subset Count (SC) list as a key value pair, where key is the subset and value represents its count. If a subset is already present in Subset Count (SC) list, then its count value will be incremented by Transaction Count Value of the particular transaction. After scanning each row in final_matrix and updating the Subset Count list, we check if the count of any subset is greater than or equal to min_supp, if it is greater than or equal to min_supp count then it will be added to FIS list.If item set is already present in FIS list, then leave the transaction/row and shift to next transaction, leaving the Subset Count List and Frequent Itemset list undisturbed.

C. Pseudo code of the proposed algorithm:

*Input:*Database of transactions(D);min_sup.

*Output:*Frequent Item Sets in D (FIS)

Method:

```
1.    Final_matrix =final_Matrix(D, min_supp);
2.    Subset Count List = null, FIS = null;
3.    For each transaction in final_matrix
4.    If IS not in FIS
5.    Subset of IS = Subsets(IS);
6.    Subset Count List = {Subset CountList}
      +{subset of IS}
7.    For each set in Subset Count List
8.    If set.count>= min_supp Then
9.    FIS = FIS U set;
10.   End If
11.   End For
12.   End If
13.   End For
```

IV. ILLUSTRATIVE EXAMPLE

Consider the Transactional dataset (D) containing the Trans ID and list of items as shown in Table 1withmin_supp is 4.

TABLE 1: Dataset (D)

Trans_ID	Items
T1	A1, A2, A4, A5
T2	A2, A4
T3	A2, A3
T4	A1, A2, A4
T5	A1, A3
T6	A2, A3
T7	A1, A3
T8	A1, A2, A3, A4, A5
T9	A1, A2, A3

Phase 1: To generate the final_matrix.

Step-1: *T*he given dataset (D) in Table 1 is converted to first_matrix form as show inTable 2, where each row represents the Trans_ID and each column represents the items. If a cell value is 0, then the particular item is not present in the corresponding transaction. If a cell value is 1, then the particular item is present in the corresponding transaction.

TABLE 2: First_matrix

TID/Items	A1	A2	A3	A4	A5
T1	1	1	0	1	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	1	0	1	0	0
T6	0	1	1	0	0
T7	1	0	1	0	0
T8	1	1	1	1	1
T9	1	1	1	0	0

Step-2: From Table2 itemA5 has the total count 2, which is less than min_supp count (4). SoA5 column is removed from the first_matrix. One more column IC is added to the first_matrix, where IC value represents the number of items present in a transaction. Arrange the transactions in descending order of IC as shown in Table2. This newly formed matrix is called as second_matrix.

TABLE 3: Second_matrix

TID/Items	A1	A2	A3	A4	IC
T8	1	1	1	1	4
T1	1	1	0	1	3
T9	1	1	1	0	3
T4	1	1	0	1	3
T2	0	1	0	1	2
T3	0	1	1	0	2
T5	1	0	1	0	2
T6	0	1	1	0	2
T7	1	0	1	0	2

Step-3: Add another column TC to the second_matrix and initially assign the value as one for all the transaction. Scan the first transactionsand check for duplicates and if duplicates exists then delete the duplicate transactions corresponding to the current transaction and increment the TC value of current transaction with the number of its duplicate transactions. Now newly formed matrix is called final_matrix. From Table3 T1 and T4 are same, so delete T4 and increment the TC value of T1 by 1. Now the new TC value of T1 is 2.Similarly T6 is duplicate of T3, so delete T6, and increment the TC value of T3 by 1. New TC value of T3 is 2. T7 is duplicate of T5, so delete T7, and increment the TC value of T5 by 1. New TC value of T5 is 2.

TABLE 4: Final_matrix

TID/Items	A1	A2	A3	A4	IC	TC
T8	1	1	1	1	4	1
T1	1	1	0	1	3	2
T9	1	1	1	0	3	1
T2	0	1	0	1	2	1
T3	0	1	1	0	2	2
T5	1	0	1	0	2	2

Phase 2 : To generate Frequent item list from the subset count.Initially Subset Count (SC) list= { } and Frequent Item Sets (FIS) list = { }.

Step-1: Fromthe final_matrix as shown in Table4, the algorithm scans the transaction (T8) and find the subsets of {A1, A2, A3, A4} and add them to SC list. Assign the count value for this subsets as 1, since TC value for this T8 is 1. After scanning transaction (T8) the Subset Count list is like
SC = {
{A1}-1, {A2}-1, {A3}-1, {A4}-1,

{A1, A2}-1, {A1, A3}-1, {A1, A4}-1, {A2, A3}-1, {A2, A4}-1, {A3, A4}-1,
{A1, A2, A3}-1, {A1, A2, A4}-1, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1}

Now if any of the above subsets count is greater than or equal to min_supp (4) then add those item sets to the Frequent Item Sets. Since there is no subset with count equal to greater than 4, FIS = { }.

Step-2: After scanning the transaction (T4) with items {A1, A2, A4} and with transaction count as 2 the subsets and their count in SC will look like
SC = {
{A1}-3, {A2}-3, {A3}-1, {A4}-3,
{A1, A2}-3, {A1, A3}-1, {A1, A4}-3, {A2, A3}-1, {A2, A4}-3, {A3, A4}-1,
{A1, A2, A3}-1, {A1, A2, A4}-3, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1
}

Now if the above subsets count is greater than or equal to Minsupp count (4) then add those item sets to the frequent item sets. Since there is no subset with count equal to greater than 4, FIS = { }.

Step-3: After scanning the transaction (T9) with items {A1, A2, A3} and with transaction count as 1 the subsets and their count in SC list will look like
SC = {
{A1}-4, {A2}-4, {A3}-2, {A4}-3,
{A1, A2}-4, {A1, A3}-2, {A1, A4}-3, {A2, A3}-2, {A2, A4}-3, {A3, A4}-1,
{A1, A2, A3}-2, {A1, A2, A4}-3, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1 }

Now if the above subsets count is greater than or equal to Minsupp count (4) then add those item sets to the frequent item sets. Since the subsets {A1}, {A2}, {A1, A2} has the count 4. These subsets are added to frequent item sets. Now FIS is like, FIS = { {A1}, {A2}, {A1, A2} }

Step-4: Scan the transaction (T5) with items {A1, A3} and with transaction count as 2. Since the items{A1, A3} is not present in frequent item set, find its subsets and increse the subset count. Now the subsets and their count in SC list will look like
SC = {
{A1}-4, {A2}-4, {A3}-4, {A4}-3,
{A1, A2}-4, {A1, A3}-4, {A1, A4}-3, {A2, A3}-2, {A2, A4}-3, {A3, A4}-1,
{A1, A2, A3}-2, {A1, A2, A4}-3, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1
}
FIS = { {A1}, {A2}, {A3}, {A1, A2}, {A1, A3} }

Step-5:After scanning the transaction (T6) with items {A2, A3} and with transaction count as 2 the subsets and their count in SC list will look like
SC = {
{A1}-4, {A2}-6, {A3}-6, {A4}-3,
{A1, A2}-4, {A1, A3}-4, {A1, A4}-3, {A2, A3}-4, {A2, A4}-3, {A3, A4}-1,
{A1, A2, A3}-2, {A1, A2, A4}-3, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1
}
FIS = { {A1}, {A2}, {A3}, {A1, A2}, {A1, A3}, {A2, A3} }

Step-6: After scanning the transaction (T2) with items {A2, A4} and with transaction count as 1 the subsets and their count in SC list will look like
SC = {
{A1}-4, {A2}-7, {A3}-6, {A4}-4,
{A1, A2}-4, {A1, A3}-4, {A1, A4}-3, {A2, A3}-4, {A2, A4}-4, {A3, A4}-1,
{A1, A2, A3}-2, {A1, A2, A4}-3, {A1, A3, A4}-1, {A2, A3, A4}-1,
{A1, A2, A3, A4}-1
}

Finally FIS List = { {A1}, {A2}, {A3}, {A1, A2}, {A1, A3}, {A2, A3}, {A2, A4} }

V. EXPERIMENTAL RESULTS.

Experiments related to this paper is carried on intel core A5 processor with windows 8 operating system. The execution time of classical apriori algorithmis compared with the improved matrix approach with subset count technique, with different records and various supportcount.The comparison result shows that execution time of the proposed technique is very less when compared with the existing Apriori algorithm of Candidate set generation approach [1]. The graphs below are ploted by taking the number of transactions on x-axis and execution time on y-axis.

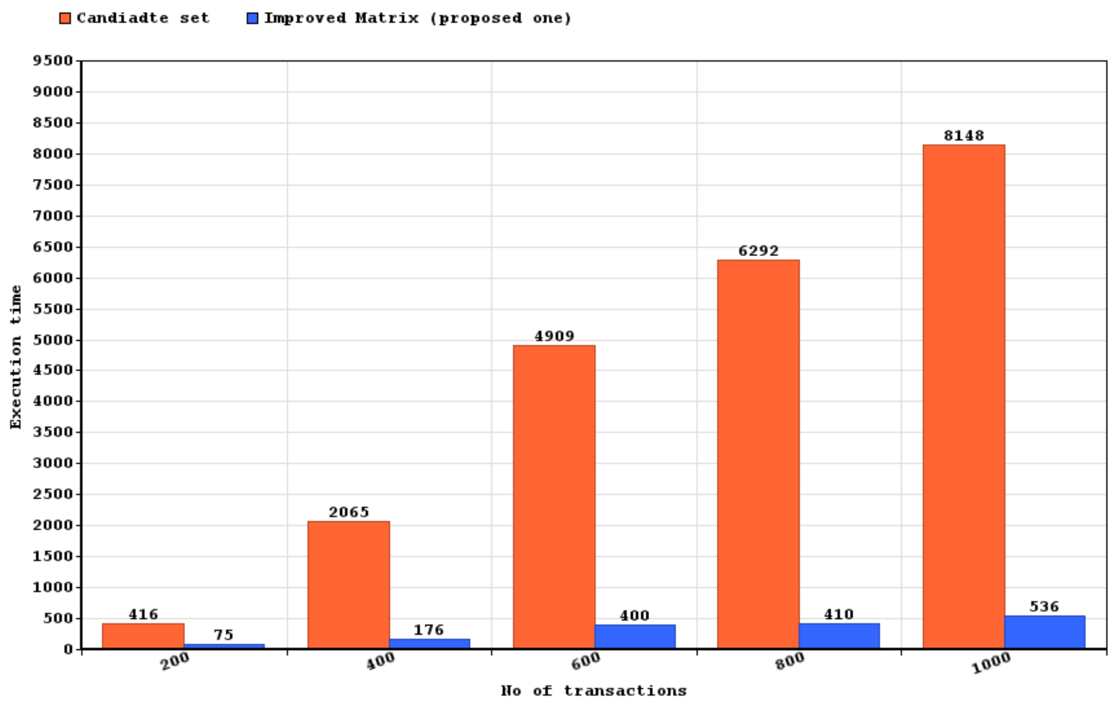

Fig.1:Execution time with different records(minsupp= 20).

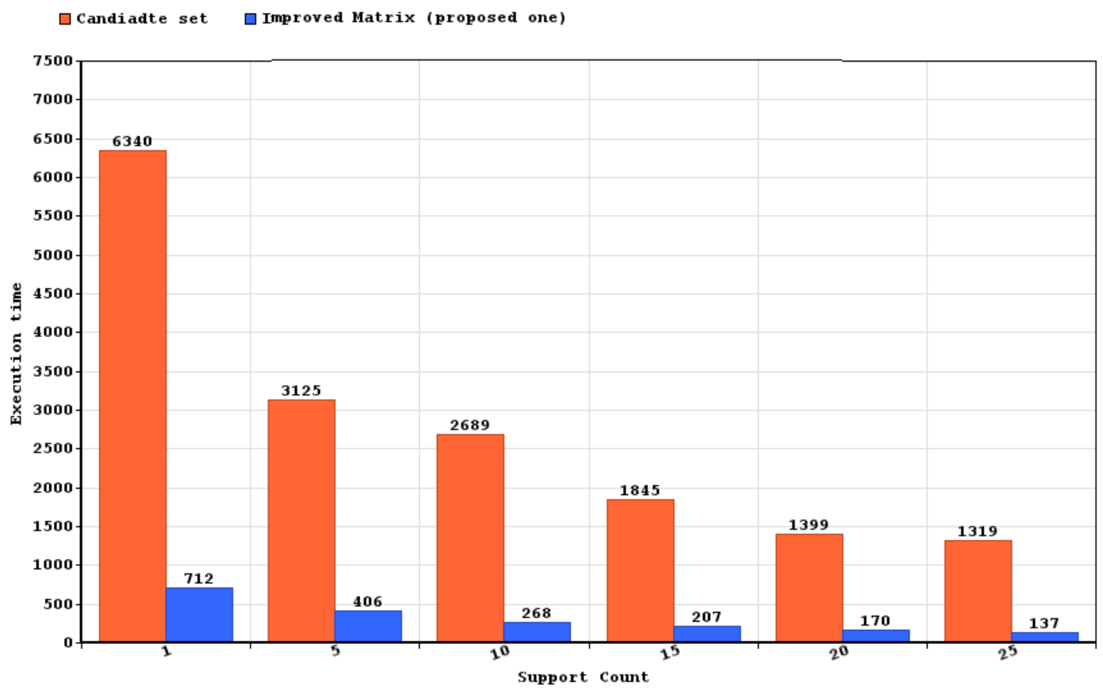

Fig. 2: Execution time for different minsupport count.

VI. CONCLUSION

In this paper we analysed various algorithms for finding frequent item set. The newly proposed approach can be used in real time scenarios for finding the frequent item sets for large data sets, since its execution time is less and also occupies very less memory. Here two dimensional array is used to convert the given dataset to matrix form. Instead some other data structure can be used to transfer the given data set to matrix form.

REFERENECES

[1] Agrawal, R.andSrikant, R. 1994, Fast algorithms for mining association rules in largedatabases. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499.

[2] ShaliniDutt, NaveenChoudhary, DharmSingh, 2014, “A Improved AprioriAlgorithm based on Matrix Data Structure”, Double Blind Peer Reviewed International Research Journal, Volume 14, Issue 5, Version 1.0, PP..

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

[4] Zhuang Chen, ShibangCai, Qiulin Song and Chonglai Zhu, 2011, “A Improved Aproiri Algorithm based on Pruning Optimization and Transaction Reduction, ” AMISEC, 2nd IEEE International Conference, pp1908-1911.

[5] Han, “DataMining concepts and Techniques” Second Edition. Morgan Kaufmann Publisher, 2006, pp.123-134.