Improved Apriori Algorithms- A Survey

Rupali Manoj Patil

ME Student, Computer Engineering

Shah And Anchor Kutchhi Engineering College, Chembur, India

Abstract:- Rapid expansion in the Network, Information technology and Internet facilities causes more and more data generation from the users. More importance is being given on handling of this data using the data mining rules for developing associations. Apriori algorithm is a standard association rule mining algorithm and also is one of the most significant algorithms. It is used to extract frequent item sets from huge database and get the association rule for discovering the knowledge but it has some limitations of frequently scanning database and generating a huge number of candidate sets. This paper discusses about three improved Apriori algorithms, what new techniques used and how they are more efficient as compared to traditional Apriori algorithm that is also discussed.

Keywords:- *Data mining*, *Association rule mining*, *Apriori algorithm, Improved Apriori.*

*****

1. INTRODUCTION

With the development in technology, all present applications are generating and collecting huge amount of data. As a result, data mining requires more effective and efficient algorithms to convert this vast data into knowledge. Data mining can be defined as a process of extracting hidden patterns and predictive information from huge amount of data. This hidden knowledge can be extracted using methods like, Association Rule Mining, Clustering, Sequence analysis, Classification or Forecasting. Association Rule mining is an important research direction in data mining which is used to describe association between item sets of transaction database. Many algorithms come under association rule mining but Apriori Algorithm is one of the standard algorithm. It was introduced by Agarwal et al. in 1993. Apriori Algorithm is based on the property of frequent item sets "All subsets of frequent item sets is frequent and all supersets of a nonfrequent item set is nonfrequent." This property is used in Apriori Algorithm to determine all the frequent item sets.

1.1 Apriori algorithm

Apriori Algorithm works on two concepts:

a) Self-Joining and b) Pruning.

Apriori uses a level wise search where k-item sets are used to find (k+1) item set.

Simple steps of Apriori algorithm

1) First scan the whole transaction database and find 1-candidate item sets which is denoted by C1.
2) Then support is calculated. Support means the occurrence of the item in the database.
3) Next step is pruning step, it is carried out on C1. In this the items which satisfies the minimum support criteria are only considered for frequent-1 item set, L1.
4) Then self-joining step is carried out in which L1 joins itself to get C2 i.e. candidate-2 item sets.
5) Then Pruning of C2 to get L2.
6) Self-joining of L2 to get C3 i.e. candidate-3 item set.
7) Repeat above steps till there is no frequent or candidate item sets.

We can understand the concepts use by Apriori Algorithm with the help of following example. Table 1 shows simple database having 4 transactions. TID is transaction ID, an unique identification for every transaction.

Table 1 Database D

TID	Items
T1	1 2 4
T2	3 4 5
T3	1 3 4 5
T4	3 5

By performing the first step, which is scanning the whole database to get C1 as shown in Table 2.

Table 2 C1

Itemset	Support
{1}	2
{2}	1
{3}	3
{4}	3
{5}	3

Here assuming minimum support as 2. Next step is the pruning step in which each itemset support is compared with the minimum support. Only those itemset whose support is greater than or equal to minimum support are taken in L1.

Table 3 L1

Itemset	Support
{1}	2
{3}	3
{4}	3
{5}	3

Now Candidate-2 itemset are created from L1. Table 4 shows all possible combinations that can made from Table 3 itemset.

Table 4 C2

Itemset	Support
{1 3}	1
{1 4}	2
{1 5}	1
{3 4}	2
{3 5}	3
{4 5}	2

Now again pruning step is carried out using minimum support condition. We will get following Table 5.

Table 5 L2

Itemset	Support
{1 4}	2
{3 4}	2
{3 5}	3
{4 5}	2

Now from L2 create candidate-3 itemset which is denoted as C3. Table 6 describes C3.

Table 6 C3

Itemset	Support
{1 3 4}	1
{1 3 5}	1
{1 4 5}	1
{3 4 5}	2

After pruning on the basis of minimum support condition frequent-3 itemset, L3 is created. Table 7 describes it.

Table 7 L3

Itemset	Support
{3 4 5}	2

Above steps continued till there are no frequent or candidate set that can be generated.

Pseudo Code –

C_k: Candidate itemset of size k
L_k: frequent itemset of size k
L_1: {frequent itemset};
For (k=1; L_k !=Ø; k++) **do begin**
C_{k+1} = candidates generated from L_k;
for each transaction t in database do
Increment the count of all candidates in C_{k+1} that are contained in t
L_{k+1} = candidates in C_{k+1} with minimum support
end
return$\cup_k L_k$;

Drawbacks of Apriori Algorithm:

♦ Scan database many times repeatedly for finding candidate itemset which increases the CPU overhead.
♦ More time is wasted to hold vast number of candidate sets.
♦ In case of large dataset, this algorithm is not efficient.
♦ For large dataset large number of infrequent item set are generated which increases the space complexity.
♦ for large dataset support computation will be more.

Due to these drawbacks Apriori should get improved which we will study further in this paper.

2. Improved Apriori Algorithms

2.1 Improved Apriori based on matrix

First step in this improved Apriori is to make a Matrix library. Prof. Dr. K. Rajeswari [3] had transformed the list of items into a matrix library. The matrix library contains a binary representation where ‘1’ indicates item present in transaction and ‘0’ indicates it is absent. The third column in the example dataset is binary representation of the items in the transactions.

Improved Algorithm:

Pseudo code:

‘Ti’ Transaction in the database D, **I_k:** Candidate item set, **A_{mn}:** Matrix representation of transactions of size mxn, **m:** No. of transaction, **n:** No. of items, MAT (Ij): single item set support.

1) I/P: Database D, Minimum_support
2) O/P: L_k frequent item set
3) For (i=1; i<=m; i++) do begin
4) For each transaction Ti,
5) For (j=1; j<=n; j++) do begin
6) If (Ij present in Ti)
A[i] [j] =1
Else
A[i] [j] =0
7) End for

8) End for
9) For (j=1; j<=; j++) do begin
10) For each item generate support matrix
11) For (i=1; i<=m; i++) do begin
12) MAT[j] [i] = A[i] [j]
//Arrange column value to get support of each item set
13) End for
14) End for
//For frequent itemset-2
15) MAT [Ij, I_k] = MAT [Ij] & MAT [I_k]
16) Support for frequent itemset-2 is calculated

Example:

Table 8 Database D

TID	List of Items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I4
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

First step is to make matrix library.

Table 9 Matrix library

TID	List of Items	I1	I2	I3	I4	I5
T1	I1,I2,I5	1	1	0	0	1
T2	I2,I4	0	1	0	1	0
T3	I2,I4	0	1	0	1	0
T4	I1,I2,I4	1	1	0	1	0
T5	I1,I3	1	0	1	0	0
T6	I2,I3	0	1	1	0	0
T7	I1,I3	1	0	1	0	0
T8	I1,I2,I3,I5	1	1	1	0	1
T9	I1,I2,I3	1	1	1	0	0

For 1-itemset matrix represented is used (i.e.)
MAT (I1) =100110111
MAT (I2) =111101011
MAT (I3) =000011111
MAT (I4) =011100000
MAT (I5) =100000010
Minimum support is 3 so I5 is not considered.
For 2-itemset take logical "AND" of MAT (I1) and MAT (I2)
MAT (I1) = 100110111
MAT (I2) = 111101011
MAT (I1, I2) =100100011

Support can be calculated as:
Support (I1, I2) = Nos. of times appearing together/Total Transaction) = 4/9.
Same procedure can be followed for all possible item set. This algorithm needs to scan the database only once so frequently scanning problem of basic Apriori is solved and also this algorithm does not requires to find the candidate set when searching for frequent item set ,so amount of memory utilization is also less as compared to traditional Apriori.
Table 10 provides computational time of Apriori and improved Apriori.

Table 10 Computation time for Apriori and Improved Apriori

Record no.	Apriori computing time(ms)	Improved Apriori Computing time(ms)
500	1787	35
1000	8187	108
1500	44444	178
2000	46288	214
2500	97467	292
3000	199253	407
3500	226558	467
4000	310379	569
4500	155243	470
5000	208685	572

2.2 Improved Apriori based on Transaction Reduction

This improved Apriori algorithm [2] is to reduce the time consuming for candidate itemset generation. Original Apriori scan all the transactions for candidate-2 itemset, this proposed improved algorithm split the two items and get the minimum support between them using L1 and the item who is having less support count only those transactions are scanned.

Steps for C_k generation:
1) Scan all transactions to generate L1 table L1 (items, their support, their transaction IDs).
2) Construct C_k by self-join.
3) Use L1 to identify the target transactions for C_k.
4) Scan the target transactions to generate C_k.

Improved Algorithm:
//Generate items, items support, their transaction ID
1) L1=find _frequent_1_itemsets (T);
2) For (k=2; L_{k-1} !=Ø; k++) {
//Generating the C_k from the L_{k-1}
3) **C_k** =candidates that are generated from L_{k-1};
//getting the item with minimum support in C_k using L1, (k>=1).

4) x = Get_item_minimum_support (C_k, L1);
//getting the target transaction IDs that contain item x.
5) Target = get_Transaction_ID (x)
6) For each transaction t in Target Do
7) Increment the count of all the items in C_k that are found in Target;
8) L_k = items in C_k >= minimum support;
9) End;
10)}

Example:

Table 11 Database D

TID	List of Items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I4
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Table 12 Candidate-1 item-set

Items	Support count
I1	6
I2	7
I3	5
I4	3
I5	2

Step 1: From the above database generate candidate-1 item set, which contains items with their support count.
Step 2: Generate Frequent-1 item set (L1) which consists of items with their support count and the transaction Ids that contain these items. Eliminate the candidates which are not frequent or whose support count < Minimum_support. Here minimum support is 3 so item I5 is deleted in frequent-1 item set. Which is shown in Table 13.

Table 13 L1: Frequent-1 item-set

Items	Support	T_IDs
I1	6	T1,T4,T5,T7,T8,T9
I2	7	T1,T2,T3,T4,T6,T8,T9
I3	5	T5,T6,T7,T8,T9
I4	3	T2,T3,T4

Step 3: Generate candidate-2 item set from L1. Min is the minimum support count. I1 is having less support count than I2, so only I1 transaction Ids are scanned by the algorithm. Candidates I1, I4 and I3, I4 will be deleted from frequent-3 item set because it is not satisfying minimum support condition.

Table 14 L2: Frequent-2 item-set

Items	Support count	Min	Found in
I1,I2	4	I1	T1,T4,T5,T7,T8,T9
I1,I3	4	I3	T5,T6,T7,T8,T9
I1,I4	1	I4	T2,T3,T4
I2,I3	3	I3	T5,T6,T7,T8,T9
I2,I4	3	I4	T2,T3,T4
I3,I4	0	I4	T2,T3,T4

Step 4: Same process is carried out to generate 3-itemset depending on L1 table. For a given frequent item set Lk ,all nonempty subsets that satisfy the minimum confidence are found then all candidate association rules are generated.

Table 15 L3: Frequent -3 item set

Items	Support	Min	Found in
I1,I2,I3	2	I3	T5,T6,T7,T8,T9
I1,I2,I4	1	I4	T2,T3,T4
I1,I3,I4	0	I4	T2,T3,T4
I2,I3,I4	0	I4	T2,T3,T4

In above example L3 is not satisfying minimum support criteria so all candidates are deleted and association rules are generated from L2.

1.3 Improved DC Apriori Algorithm

Prof. Jioling Du et.al [1] had proposed improved DC_Apriori algorithm. This proposed algorithm uses the logical "And" operation for reference and uses a map structure to store the item sets table then takes the intersection to obtain the corresponding support, avoiding frequently database scanning.

Improved DC Algorithm:
Pseudocode description:
Input: Transaction database, Minimum support min_sup
Output: Frequent item sets L
1. For each Transaction in D {
2. While (getline (strSreamItem, strItem, ' ')){
//get transaction t's Item
3. If (t.item==c1.item)
//Item already present
4. then c1.Item.Tid.insert(t);
//Add transaction Id

5. Else //Item not present
6. Then c1.insert (Item & Tid);
//Insert new Item and Tid
7.}
8.}
9. For each c1 in C1 {
//Judge the support of each 1-candidate c1
10. L1= {c1є C1 lc1.sup>= min_sup};
//Generate frequent 1-item sets L1
11.}
12. for (k=2; L_{k-1} !=NULL; k++){
//If L_{k-1} is empty, stop the circulation
13. L_k= apriori_gen ($L_{k-1} \infty$ L1);
//Generate frequent k-item sets L_k
14.}
15. Return L= L_{k-1};
//The last frequent item sets

The process to generate frequent k-item sets L_k:

1. for all l_{k-1} є L_{k-1}
// l_{k-1} containing (k-1) elements is subset of L_{k-1}
2. for all l_1 є L1
3. If (l_{k-1}.Tid== l_1.Tid) & (l_{k-1} l_1.sup>=min_sup)
4. Then l_k = l_{k-1} l_1;
//Take the intersection of both l_{k-1} and l_1's transactions list, then add the item which is generated by $l_{k-1} \infty l_1$ and which support is bigger than the min_sup into the frequent k-item sets L_k.
5. Else
6. Then l_1→next;
//otherwise l_1 moves to the next frequent 1-item set
7. End if
8. End for
9. End for
10. Return L_k;

Example:

Table 16 Database D

TID	List of Items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I4
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

This algorithm scan the whole database and reorganized it with Item, Tid structure. It is using vector to store each Item.

Table 17 Frequent-1 item set table

Item	Tid						
I1	1	4	5	7	8	9	
I2	1	2	3	4	6	8	9
I3	5	6	7	8	9		
I4	2	3	4				
I5	1	8					

Table 18 Frequent 2-Item sets table

Item	Tid			
I1I2	1	4	8	9
I1I3	5	7	8	9
I1I5	1	8		
I2I3	6	8	9	
I2I4	2	3	4	
I2I5	1	8		

3. COMPARISON OF IMPROVED APRIORI

Attributes	Improved Algorithm based on matrix	Improvement based on transaction reduction	Improved DC algorithm
New Technique used	Binary matrix which to reduce the database scan	Vertical data format and Min column to reduce database scan	Vertical data format to reduce database scan for finding support
Number of scans	1	1	1
Storage structure	2-D array	Normal Database	Dynamic array

used			vectors

4. CONCLUSION

In this paper we have studied drawbacks of Apriori Algorithm and about 3 Improved Apriori algorithms which uses new technique in generating rules. These algorithms are far better than the traditional Apriori Algorithm and improves time complexity. Comparison has been made which discusses different attributes. These improved algorithms may be implemented and tested against different datasets to find their efficiencies.

REFERENCES

[1] Jiaoling Du, Xiangli Zhang, Hongmei Zhang and Lei Chen, Research and Improvement of Apriori Algorithm, Sixth International Conference on Information Science and Technology, Dalian , China ;May16 IEEE, pp.117-121.

[2] Rucha Kale, Sharayu Fukey, An Improved Apriori algorithm ,International Journal of Scientific Engineering and Applied Science(IJSEAS)-Volume-2, Issue-5, May 16.

[3] K. Rajeswari, Improved Apriori Algorithm-A Comparative study using different objective measures. International Journal of Computer Science and Information Technologies (IJCSIT), Vol.6 (3), 2015, 3185-3191.

[4] Jiawei Han, Micheline Kamber , Jian Pei, Data Mining Concepts and Techniques, Third Edition, Morgan Kaufmann Publisher, Elsevier.

[5] Akshita Bhandari, Ashutosh Gupta, Debasis Das, Improvised Apriori algorithm using frequent pattern tree for real time applications in data mining, Procedia Computer Science 46(2015)644-651,International Conference on Information and Communication Technologies (ICICT 2014).

[6] X. Luo, W. Wang, "Improved Algorithms Research for Association Rule Based on Matrix", 2010 International Conference on Intelligent Computing and Cognitive Informatics,pp.415-419,Jun2010 IEEE.

[7] Lingjuan Li, Min Zhang, "The Strategy of Mining Association Rule Based on Cloud Computing", 2011 International Conference on Business Computing and Global Informatization, Sept2011 IEEE.

[8] Rao S, Gupta P, "Implementing Improved Algorithm over Apriori Data Mining Association Rule Algorithm, International Journal of Computer Science and Technology, pp.489-493, Mar.2012.

[9] K. Vyas, S. Sherasiya, "Modified Apriori Algorithm using Hash based Technique", IJARIIE, Vol-2, Issue-3, pp. 1229-1234, 2016.

[10] Zhuang Chen, S. CAI, Q. Song and C. Zhu, "An Improved Apriori based on Pruning Optimization and Transaction Reduction", IEEE conference, Jun 2011, pp.1908-1911.

[11] A. Mohammad, B. Arkok, "An Improved Apriori algorithm'', International Journal on Natural Language Computing (IJNLC) vol.03, Feb 2014