International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-5,May 2016
ISSN: 2395-3470
www.ijseas.com

AN IMPROVED APRIORI ALGORITHM

RUCHA KALE[1], SHARAYU FUKEY[2]
[1] V.J.T.I.MUMBAI-19
[2] V.J.T.I.MUMBAI-19

Abstract

Since the development in the networks, information technology, Internet facilities is attracting more and more data generation from the users, data is being generated massively and increasing like a fire. Due to spreading of this information with high speed, more emphasis is being given on handling of such data and using the data mining rules for deriving associations. The provided study describes the Apriori algorithm in association rule mining algorithm in detail and also the algorithm implementation is discussed. Also it proposes a new optimization algorithm called APRIORI-IMPROVED which is based on the insufficient of traditional Apriori. It illustrates the optimization in terms of generation, transactions, compression, and performance and so on. APRIORI-IMPROVED provides a hash structure an efficient horizontal data representation and optimized strategy of storage for saving time and space. The performance study shows that APRIORI-IMPROVED is much faster than simple Apriori.

Keywords: *Author Guide, Article, Camera-Ready Format, Paper Specifications, Paper Submission.*

1. Introduction

In spite of rapid development in modern computer technology and Database technology could support the store and quickly retrieve the huge scale databases or data warehouses, but these tactics were only to gather these massive data and not to organize them effectively and use the knowledge hidden with them, which eventually lead to today's world's phenomenon of "rich data but poor knowledge"[1]. The emergence of data mining technology met people needs by providing them only the data specific to them and facilitating them suggestions to make faster choices in selecting some data item sets. Association rule mining is to find relationships and correlations amongst these items in large databases consisting transactions[1].

An influential algorithm Apriori, was first discussed by R. Agrawal and R. Shrikant. Apriori provides an iterative approach known as level-wise and breadth first approach, in which k- data item sets are used to generate (k+1)-data item sets. In terms of the feature of Apriori property, called anti monotone, one can efficiently generate candidate item sets, by discarding unnecessary remaining ones. Apriori algorithm uses a two-step process Join and Prune[2]. However there are two major drawbacks of the algorithms based on generated and tested candidate item sets (1) Database has to be scanned multiple number of times in order to generate candidate item sets. Multiple scans indeed increases the I/O load and it's respective time consumption. (2) A lot of CPU time will be consumed by the generation of huge candidate sets and calculation of their support [4].

1.1 2. Apriori Algorithm Summary

Apriori algorithm is one of the most efficient algorithms used for mining the frequent data sets of Boolean association rules. The algorithm is an approach based on the two-stage frequency, where the design of Association rule mining algorithm can be distinguished into two sub-issues:

(1) Find all the item sets those having support greater than the minimum support, which is known as the frequent items.
(2) Based on these obtained frequent sets, all the association rules will be derived, and for each frequent item set A, all non-

empty subset of A will be further derived if it satisfies the condition as stated as below
Support(A) / Support(a) >= minimum_confidence, to generate the association rule A-a.
That is from the frequent sets found in the first step, to exploit the rules with confidence not less than minimum confidence minimum_confidence has to be specified by the user.[1]

Apriori algorithm consists two sub processes named as Apriori-generate () and subset ().
Apriori-generate() actually generates a candidate, further uses an Apriori property that is all non-empty frequent sets must also be frequent item sets; in order to delete all those candidates of the non-frequent subsets obtained. After the completion of generation of all the candidates, the database D will be scanned, as well as for each of the transactions, use the subset () sub procedure for identifying all the candidate subsets and make cumulative count for each of the subsets. Finally all candidates meet the minimum support of frequent item set L[2].

3. Apriori Illustration

The following example represents general representation of Apriori Algorithm.

(A) Generating Frequent item sets.
Following table (1) shows the transaction table consisting 4 datasets |D|=4
Let us consider minimum support count be 2.

TID	ITEMS
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

The specific process is described as below:

(1) Scan the complete database in order to initialize the source data and to form the candidate-1 with all the itemsets C1 in an aggregation.
(2) To find the frequent 1-itemsets L1, each of the candidate itemsets are checked for the occurrences and the condition for minimum_support count is tested. This condition enables to eliminate all other non-required itemsets that cannot participate into generation of association rules.

(B) Generate association rules from frequent item sets
After obtaining frequent item sets, the next step is to find the association rules from them respectively.
Here, the conditional property can be calculated using the support for item sets

*Confidence (X -> Y) = ∂ (X -> Y) / ∂x*100% = support(X U Y) / support(X) * 100%*

International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-5,May 2016
ISSN: 2395-3470
www.ijseas.com

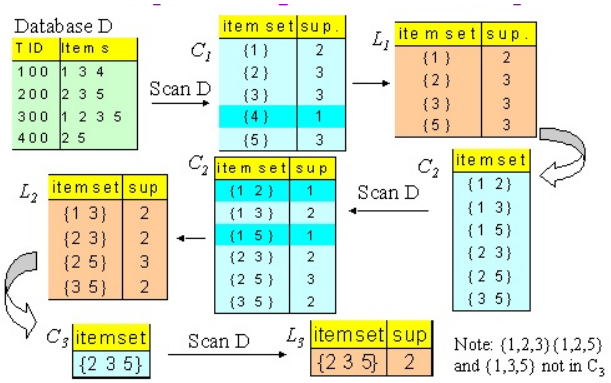


Fig (1) Selection of frequent item sets in Apriori transitions.

Depending on the above selected frequent items and their respective eliminations depending on the minimum support criterion, association rules are derived based on the conditions of condition.

4. Basic Apriori algorithm evaluation

The basic Apriori algorithm works efficiently and suitably but it consists of certain shortcomings that are difficult to overcome. Those are enlisted below:

(1) Repeated scanning of the database which indeed increases the CPU overhead.

(2) This can generate comparatively large candidate set, from L(k-1) to generate k item sets, and Ck proves to be the exponential growth.

(3) Reducing the affairs need to be scanned in later circle, the one affair does not contain all (k+1)-item sets

(4) The fitness landscape of the algorithm is very narrow. It only considers a single Boolean association rule mining. However in practical applications, there may be more complex scenarios such as multi-dimensional database, multivolume and multi-layer association rules.

5. APRIORI-IMPROVED ALGORITHM

The following section will address the improved Apriori ideas.

In the process of Apriori, the following definitions are needed:

Definition 1: Suppose T={T1, T2, … , Tm},(m≥1) is a set of transactions, Ti= {I1, I2, … , In},(n≥1) is the set of items, and k-item set = {i1, i2, … , ik},(k≥1) is also the set of k items, and k-item set ⊆ I.

Definition 2: Suppose σ (item set), is the support count of item set or the frequency of occurrence of an item set in transactions. Definition 3: Suppose Ck is the candidate item set of size k, and Lk is the frequent item set of size k [4].

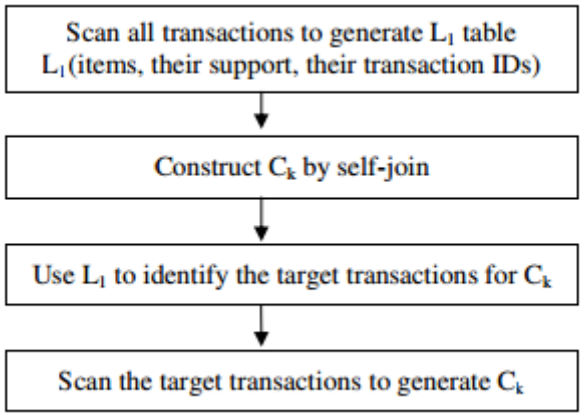


Fig (2) Steps representing Ck generation

The improvement of algorithm can be elaborated as follows[4]:

//Generate items, items support, their transaction ID

(1) L1 = find_frequent_1_itemsets (T);

(2) For (k = 2; Lk-1 ≠Φ; k++) {
//Generating the Ck from the LK-1

(3) Ck = candidates that are generated from Lk-1;
//getting the item with minimum support in Ck using L1, (1≤w≤k).

(4)) x = Get _item_minimum_support(Ck, L1);
// getting the target transaction IDs that contain item x.

(5) Tgt = get_Transaction_ID(x);
(6) For each transaction t in Tgt Do
(7) Increment the count of all the items in Ck that are found in Tgt;
(8) Lk= items in Ck ≥ minimum_support;
(9) End;
(10) }

6. IMPROVED-APRIORI IMPLEMENTATION

Assume we have transaction set D has 9 transactions, and the minimum support = 3. The transaction set is shown in Table.2

T_ID	Items
T_1	I_1, I_2, I_5
T_2	I_2, I_4
T_3	I_2, I_4
T_4	I_1, I_2, I_4
T_5	I_1, I_3
T_6	I_2, I_3
T_7	I_1, I_3
T_8	I_1, I_2, I_3, I_5
T_9	I_1, I_2, I_3

Table(2) Transaction table considered for Improved-Apriori implementation

Items	support	
I_1	6	
I_2	7	
I_3	5	
I_4	3	
I_5	2	deleted

Table(3) The candidate-1 item set

Initially, all the transaction are scanned to get frequent 1-itemset l1 which consists of the items as well as their support count and the transactions ids that contain these items, and then eliminate the candidates that are not frequent or their respective support are less than the minimum_support. The frequent 1-itemset is shown in table 4.

Items	support	T_IDs	
I_1	6	$T_1, T_4, T_5, T_7, T_8, T_9$	
I_2	7	$T_1, T_2, T_3, T_4, T_6, T_8, T_9$	
I_3	5	T_5, T_6, T_7, T_8, T_9	
I_4	3	T_2, T_3, T_4	
I_5	2	T_1, T_8	deleted

Table(4) Frequent_1 item-set

The next step is generating candidate 2-itemset from L1. To get support count for every item set, each item set is spitted in 2-itemset into two elements then l1 table is used to determine the transactions where you can find the item set in, rather than searching for them in all transactions. for example, let's take the first item in table.4 (I1, I2), in the original Apriori we scan all 9 transactions to find the item (I1, I2); but in our proposed improved algorithm we will split the item (I1, I2) into I1 and I2 and get the minimum support between them using L1, here i1 has the smallest minimum support. After that we search for item set (I1, I2) only in the transactions T1, T4, T5, T7, T8 and T9.

Items	support	Min	Found in	
I_1, I_2	4	I_1	$T_1, T_4, T_5, T_7, T_8, T_9$	
I_1, I_3	4	I_3	T_5, T_6, T_7, T_8, T_9	
I_1, I_4	1	I_4	T_2, T_3, T_4	deleted
I_2, I_3	3	I_3	T_5, T_6, T_7, T_8, T_9	
I_2, I_4	3	I_4	T_2, T_3, T_4	
I_3, I_4	0	I_4	T_2, T_3, T_4	deleted

Table (5) Frequent_2 item sets

The same process is carried out to generate 3-itemset depending on L1 table, as it is shown in table 6.

Items	support	Min	Found in	
I1, I2 , I3	2	I_3	T_5, T_6, T_7, T_8, T_9	deleted
I1, I2 , I4	1	I_4	T_2, T_3, T_4	deleted
I1, I3 , I4	0	I_4	T_2, T_3, T_4	deleted
I2, I3 , I4	0	I_4	T_2, T_3, T_4	deleted

Table (6) Frequent_3 item sets

For a given frequent item set LK, all non-empty subsets that satisfy the minimum confidence are

found, and then all candidate association rules are generated. Using the original Apriori and our improved Apriori, the obvious difference between number of scanned transactions with our improved Apriori and the original Apriori will be considerably observed. From the table 7, number of transactions in 1-itemset is the same in both of sides, and whenever the k of k-item set increases, the gap between our improved Apriori and the original Apriori increase from view of time consumed, and hence the time consumed to generate candidate support count will be considerably reduced.

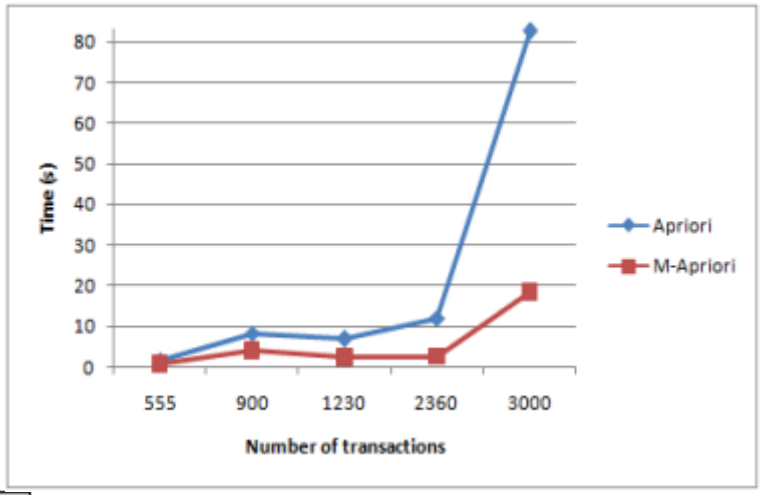
	Original Apriori	Our improved Apriori
1-itemset	45	45
2-itemset	54	25
3-itemset	36	14
sum	135	84

Table (7) Number of transaction scanned elements

We developed an implementation for original Apriori and our improved Apriori, and we collect 5 different groups of transactions as the follow:

- T1: 556 transactions.
- T2:800 transactions.
- T3: 1250 transactions.
- T4: 2500 transactions.
- T5: 3100 transactions.

The first experiment compares the time consumed of original Apriori, and the proposed improved algorithm by applying the above five groups of transactions in the implementation. The result is shown in following Figure 3.



Figure(3) : Time consuming comparison for different groups of transactions

The second experiment is used to compare the time consumed of original Apriori, and our proposed algorithm by applying the one group of transactions through various values for minimum support in the implementation. The corresponding result is shown in following Figure 4.

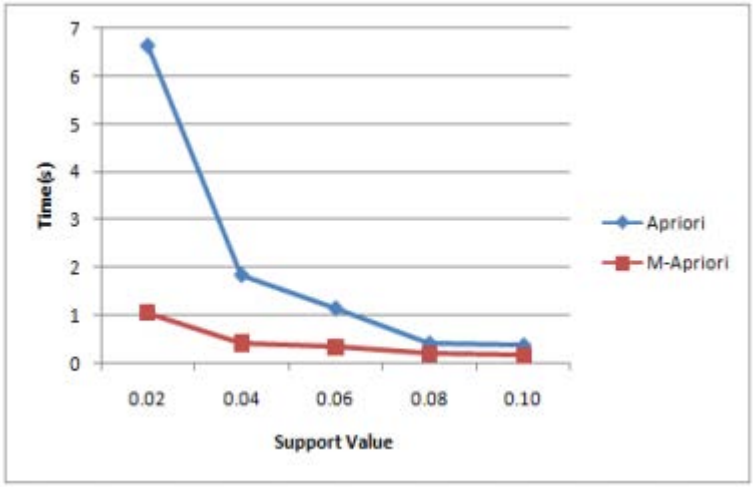


Figure (4): Time consuming comparison for different values based on minimum support

Evaluation of IMPROVED-APRIORI:

The time consuming in improved Apriori in each group of transactions is less than it in the original Apriori, and the difference gradually increases corresponding to the number of transactions[5]

T	Original Apriori (S)	Improved Apriori (S)	Time reducing rate (%)
T_1	1.776	0.677	61.88%
T_2	8.221	4.002	51.32%
T_3	6.871	2.304	66.47%
T_4	11.940	2.458	79.41%
T_5	82.558	18.331	77.80%

Table (8) The time reducing rate of improved Apriori on the original Apriori as per the number of transactions

As the value of minimum support increase the rate is decreased also. The average of reducing time rate in the improved Apriori is 68.39%.

Min_Sup	Original Apriori (S)	Improved Apriori (S)	Time reducing rate (%)
0.02	6.638	1.056	84.09%
0.04	1.855	0.422	77.25%
0.06	1.158	0.330	71.50%
0.08	0.424	0.199	53.07%
0.10	0.382	0.168	56.02%

Table (9) The time reducing rate of improved Apriori on the original Apriori according to the values of minimum support obtained.

4. Conclusions

In this paper, an improved Apriori is proposed towards decreasing the time consumed in transactions scanning for candidate item sets by decreasing the total number of transactions to be scanned. The gap between our improved Apriori and the original Apriori increases from view of time consumed whenever the k of k-item set increases; and the gap between our improved Apriori and the original Apriori decreases whenever the value of minimum support increases from view of time consumed. The time consumed to generate candidate support count in our improved Apriori is less than the time consumed in the original Apriori; our improved Apriori reduces the time consuming by 67.38%. As this is proved and validated by the experiments and obvious in figure 3, figure 4, table 8 and table 9.

References

(1) Yanxi Liu, "Study on Application of Apriori Algorithm in Data Mining", Second International Conference on Computer Modeling and Simulation, 2010

(2) Yuntao Liu, Qing Liu, Danweng Zheng, Qingping Deng, Zhanpeng Tang, Jian Ying, "Application and Improvement Discussion about Apriori Algorithm of Association Rules Mining in Cases Mining of Influenza treated by contemporary famous old Chinese medicine", IEEE International Conference on Bioinformatics and Biomedicine Workshops, 2012

(3) Yong Li, "The Java Implementation of Apriori Algorithm based on Agile Design Principles",.

(4) Riu Chang, Zhiyi Liu, "An Approved Apriori Algorithm", International Conference on Electronics and Optoelectronics (ICOEO), 2011

(5) Gang Yang, Hong Zhao, Lei Wang, Ying Liu, "An Implementation of Improved Apriori Algorithm", Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, July 2009

(6) Mohammad Al-Maolegi , Bassam Arkok, "An Improved Apriori Algorithm" , International Journal on Natural Language Computing (IJNLC) Vol.03, February 2014