International Journal of Research, Science, Technology & Management
ISSN Online: 2455-2240, Volume 10 Issue III, March 2018

IMPROVING THE EFFICIENCY OF APRIORI ALGORITHM IN DATA MINING

Jhunjhun Awasthi

ABSTRACT

Apriori algorithm is the classic algorithm of association rules, which finds the entire frequent item, sets. When this algorithm is applied to dense data the algorithm performance declines due to the large number of long patterns emerge. In order to find more valuable rules, this paper proposes an improved algorithm of association rules, the classical Apriori algorithm. Finally, the improved algorithm is verified, the results show that the improved algorithm is reasonable and effective, can extract more valuable information.

KEYWORDS- Association rule mining, Apriori, Candidate item sets, Data Mining

1. INTRODUCTION

Data mining is one of the most dynamic emerging researches in today's database technology. To extract the valuable data from large databases, it is necessary to explore the databases efficiently. It is the analysis step of the KDD (Knowledge Discovery and Data Mining) process. It is defined as the process of extracting interesting (non-trivial, implicit, previously unknown and useful) information or patterns from large information repositories such as: relational database, data warehouses etc. The goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. The problem of mining association rules from transactional database was introduced in [9]. The concept aims to find frequent patterns, interesting correlations, and associations among sets of items in the transaction databases or other data repositories. Association rules are being used widely in various areas such as telecommunication networks, drug

analysis, risk and market management, inventory control etc.[2]

Association rule are the statements that find the relationship between data in any database. Association rule has two parts "Antecedent" and "Consequent". For example {bread} => {butter}. Here bread is the antecedent and butter is the consequent. Antecedent is that item which is found in the database, and consequent is the item that is found in combination with the first i.e. the antecedent [12].

Formal definition [3]: Let I = {i1,i2, ..., in} be a set of items. Let D be a set of task relevant data transactions where each transaction T is a set of items such that T ⊆ I. A unique TID is associated with each transaction. Let A be a set of items. A transaction T is said to contain A if and only if A ⊆ T. An association rule is implication of the form A ⇒ B, where A ⊂ I, B ⊂ I, and A ∩ B = null.

The ARM technique is based on several threshold values which are explained below:

- Support: is defined as the percentage/fraction of records that contain XUY to the total number of records in the database. It is given as: Support(XUY)= Support(XY)/Total number of transactions.
- Confidence: is defined as percentage of transaction in database containing X that also contain Y. It is given as: confidence = Support(XUY)/Support(X).
- Lift: is the ratio of the probability that X and Y occur together to the multiple of the two individual probabilities for X and Y. It is given as: lift = Pr(X,Y) / Pr(X).Pr(Y).
- Conviction: Unlike lift, it measures the effect of the right-hand-side not being true. It inverts the ratio and is given as : conviction = Pr(X).Pr(not Y) / Pr(X,Y).[11]

2. APRIORI ALGORITHM

Apriori is the most classical and famous algorithm for mining frequent patterns. Apriori algorithm was introduced by [9]. The algorithm works on categorical attributes and employs bottom up strategy. It is based on the Apriori property which is useful for trimming irrelevant data. It states that any subset of frequent item-sets must be frequent.

2.1 DESCRIPTION OF THE TYPICAL APRIORI ALGORITHM

Apriori employs an iterative approach, where k-item sets are used to explore (k+1)-item sets. First, the set of frequent 1-itemsets is found by scanning the database to count the occurrences for each item, and then collecting those items that satisfy minimum support defined. The result set obtained is denoted as L1. Now, L1 is used to find the set of frequent 2-itemsets, L2 and so on, until no more frequent k-item sets can be found. Thus the finding of each frequent k-item set requires one full scan of the database. To improve the efficiency of frequent item sets level-wise generation, an important property called the Apriori property, is used. It reduces the search space. Apriori property states that "All nonempty subsets of a frequent item set must also be frequent". Thus it is divided into a two-step process which is used to find the frequent item sets: join and prune actions.

A) THE JOIN STEP

A set of candidate k-item set is generated by joining Lk-1 with itself. This set of candidates is denoted as Ck.

B) THE PRUNE STEP

The members of Ck may or may not be frequent, but all of the frequent k-item sets are members of Ck. A scan of the database is required to determine the count of each candidate in Ck. The scan of the database result in the formation of Lk i.e. it contains all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk. The Apriori property is used to reduce the size of Ck, as follows. Any (K-1)-item set that is not frequent cannot be a subset of a frequent k-item set. Hence, if any (K-1)-subset of the candidate k-item set is not in Lk-1, then the candidate item cannot be frequent either and so can be deleted from Ck [5].

3. RELATED WORK

Rehab H. Alwa and Anasuya V. Patil (2013) [1] described a novel approach to improve the Apriori algorithm through the creation of Matrix- File. For this purpose MATLAB is used, where the database transactions are saved. Thus repeated scanning is avoided and particular rows & columns are extracted and perform a function on that rather than scanning entire database. The results are visualize using graphical form display and then interpreted. The novel approach showed a very good result in comparison to the traditional

Apriori algorithm because there is a pruning process to those columns whose item count is less than the value of minimum support. Hence the size of the Matrix reduces which saves a lot of time, and an improvement in the speed is noticed by reducing the redundant scanning of the database.

Jaio Yabing (2013) [4] proposed an improved algorithm of association rules, the classical Apriori algorithm. The improved algorithm is implemented and the results show that the improved algorithm is reasonable and effective, can extract more value information. Although this process can decline the number of candidate item sets in Ck and reduce time cost of data mining, the price of it is pruning frequent item set, which could cost certain time. For dense databases, the efficiency of this algorithm is higher than Apriori algorithm.

Jaishree Singh et al (2013) [5] proposed an Improved Apriori algorithm which reduces the scanning time by cutting down not required transaction records as well as reduce the redundant generation of sub-items during pruning the candidate item-sets, which can form directly the set of frequent item-sets and eliminate candidate having a subset that is not frequent. The improved algorithm not only optimizes the algorithm by reducing the size of the candidate set of k-item-sets, Ck, but it also reduces the amount of I / O spending by cutting down transaction records in the database. The performance of Apriori algorithm is optimized so that we can mine association information from massive data faster and better.

Jose L. Balcazar (2013) [6] proposed a measure to the confidence boost of a rule. Acting as a balance to confidence and support, it helps to obtain small and crisp sets of mined association rules and solves the well-known problem such as rules of negative correlation may pass the confidence bound. He analyzed the properties of two versions of the notion of confidence boost, one being a natural generalization of the other. They developed algorithms to filter rules according to their confidence boost, and compared the given concept to some similar notions in the literature, and described the results of experimentation employing the new notions on standard benchmark datasets. He also described an open source association mining tool that embodies one of our variants of

confidence boost in such a way that the data mining process does not require the user to select any value for any parameter.

Mohammed Al- Maolegi and Bassam Arkok (2013) [8] indicated the limitation of the original Apriori algorithm of wasting time for scanning the whole database searching on the frequent item-sets, and presented an improvement on Apriori by reducing that wasted time depending on scanning only some transactions. They performed experiments with several groups of transactions, and with several values of minimum support applied on the original Apriori and the improved Apriori, and the results showed that the improved Apriori reduces the time consumed by 67.38% in comparison with the original Apriori, and makes the improved Apriori algorithm more efficient and less time consuming.

Rina Raval et al (2013) [10] performed a survey on few good improved approaches of Apriori algorithm such as Record Filter and Intersection approach, Improvement based on set size frequency and trade list, Improvement by reducing candidate set and memory utilization, Improvement based on frequency of items etc. They found that mostly improved Apriori algorithms aims to generate less candidate sets and yet get all frequent items. They concluded that many improvements are needed basically on pruning in Apriori to improve efficiency of algorithm.

Apriori Algorithm is the classical algorithm for association rule mining. From the above reviews it has been found that Apriori Algorithm is simple and easy to implement but still it has several drawbacks such as it requires multiple scan of the database. Moreover for candidate generation process it takes more memory space and time. The rules generated consist of items which are irrelevant. In case of large databases redundant rules are generated. Moreover the Apriori Algorithm works only on a single value of confidence and support throughout the algorithm. Thus it is not suitable for dense databases. [13]

4. PROPOSED ALGORITHM

A. THEORY OF ALGORITHM

Classical Apriori algorithm generates large number of candidate sets if database is large. And due to large number of records in database results in much more I/O cost. In this project, we proposed an optimized method for

Apriori algorithm which reduces the size of database along with reducing the number of candidate item sets generated. In our proposed method, we introduced an attribute named SizeOfTransaction (SOT), containing number of items in individual transaction in database. The deletion process of transaction in database will made according to the value of K. Depending on the value of K, algorithm searches the same value for SOT in database. If value of SOT matches with value of K then those transactions are deleted from the database. For reducing the size of the candidate item sets formed before the candidate item sets Ck are generated, further prune Lk-1, count the times of all items occurred in Lk-1, delete item sets with this number less than k-1 in Lk-1. In this way, the number of connecting items sets and the size of the database to be scanned will be decreased, so that the number of candidate items will decline.

B. PSEUDO CODE OF THE ALGORITHM

Input: D: Database of transactions;
min_sup: minimum support threshold

Output: L: frequent itemsets in D

Method:

1) L1=find_frequent_1-itemsets(D);

2) For(k=2;Lk-1≠∅; k++){

3) Prune1(Lk-1)

4) Ck=apriori_gen(Lk-1, min_sup);

5) for each transaction t∈D{

6) Ct=subset(Ck,t);

7) for each candidate c∈Ct

8) c.count++;

9) }

10) Lk={ c∈Ck |c.count≥min_sup };

11) if(k>=2){

12) delete_datavalue(D, Lk, Lk-1);

13) delete_datarow (D, Lk); }

14) }

15) return L=UkLk ;

Procedure apriori_gen(Lk-1:frequent(k-1)-itemsets)

1) for each itemset l1∈ Lk-1 {

2) for each itemset l2∈ Lk-1 {

3) if(l1 [1]= l2 [1])∧ (l1 [2]= l2 [2]) ∧...∧(l1 [k-2]= l2 [k-2]) ∧(l1 [k-1]< l2 [k-1]) then {

4) c=l1 ∞l2;

5) for each itemset l1∈Lk-1 {

6) for each candidate c ∈Ck {

7) if l1 is the subset of c then

8) c.num++; }}}}}

9) C'k={ c∈Ck |c.num=k};

10) return C'k;

Procedure delete_datavalue

(D:Database; Lk: frequent (k)-itemsets; Lk-1: frequent(k-1) - itemsets)

1) for each itemset i ∈Lk-1 and i ∉ Lk{

2) for each transaction t∈D{

3) for each datavalue∈t{

4) if (datavalue=i)

5) update datavalue=null;

6) }}

Procedure delete_datarow (D: Database Lk:frequent(k) - itemsets)

1) for each transaction t∈D{

2) for each datavalue∈t{

3) if(datavalue!=null and datavalue!=0){

4) datarow.count++; }}

5) if(datarow.count<k){

6) delete datarow;}

7)}

Procedure Prune 1(Lk-1)

1) for all itemsets L1∈Lk-1

2) if count(L1)≤k-1

3) then delete all Lj from Lk-1

4) return L'k-1 // go back and delete item sets with this

number less than k-1.

C. EXAMPLE OF THE ALGORITHM

Let us consider a transaction database as shown in table 1.

TABLE I: Transaction database

TID	ITEMS	SOT
T1	A,B,D	3
T2	A,B,C,D	4
T3	A,B,E	3
T4	B,E,F	3
T5	A,B,D,F	4
T6	A,E	2
T7	C	1
T8	E,F	2

Suppose the minimum support count min_sup=2. The algorithm proceeds as follows (fig 1):

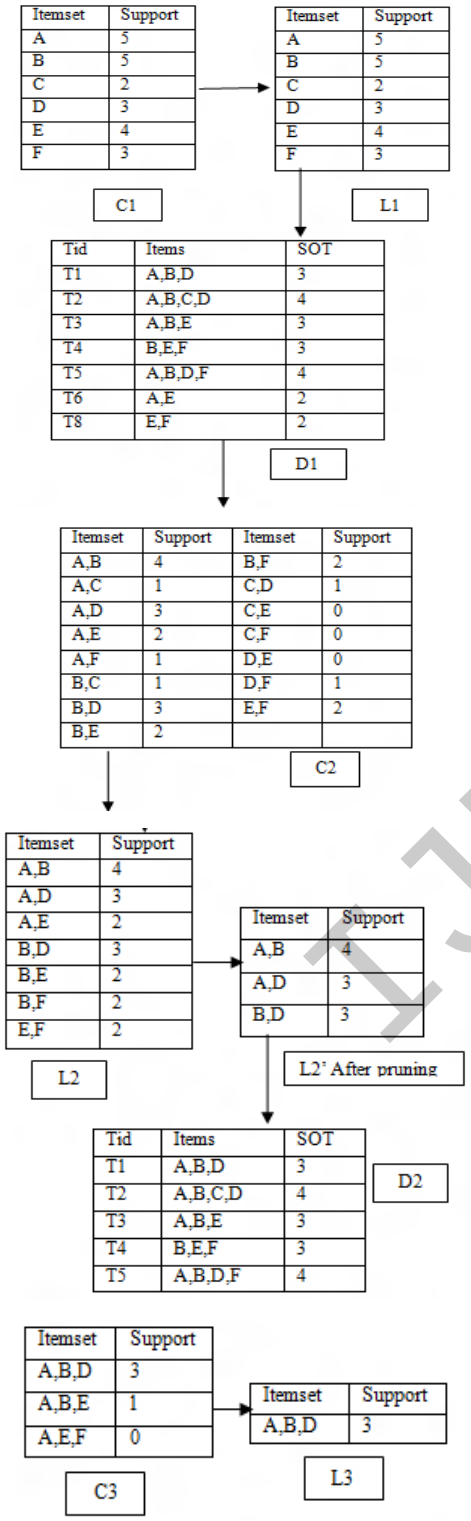

Figure 1: Example of proposed algorithm

Step 1: Firstly, the SOT column is added to the database.

Step 2: In the first iteration, each item is a member of candidate 1-itemsets,C1. The algorithm simply scans the database to count the occurrences of each item.

Step 3: This algorithm will then generate number of items in each transaction. We called this Size_Of _Transaction (SOT).

Step 4: Because of min_sup=2, the set of frequent 1-itemset, L1 can be determined. It consists of the candidate 1-itemset, satisfying the minimum support threshold.

Step 5: As there are no items with support less than 2 nothing is deleted. In addition, when L1 is generated, now, the value of k is 2, delete those records of transaction having SOT=1 in D. And there won't exist any elements of C2 in the records we find there is only one data in the T7. Thus after deleting the transaction we obtain transaction database D1.

Step 6: To discover the set of frequent 2-itemsets, L2, the algorithm uses the join L1 ∞L1 to generate a candidate set of 2-itemsets, C2.

Step 7: The transactions in D1 are scanned and the support count and SOT of each candidate item-set in C2 is accumulated.

Step 8: L2, the set of frequent 2-itemsets is now then determined, consisting of those candidate 2-itemsets in C2 having minimum support greater or equal to the specified value.

Step 9: Now L2 is pruned by deleting the item sets with number of occurrences in Lk-1 less than k-1 to before candidate item sets occur i.e. itemsets {B,F}, {A,E}, {B,E}, {E,F} are deleted.

Step 10: After L2 is pruned, we find that the transactions T6 and T8 consists of only two items. Now, the value of k is 2, delete the transactions having SOT=2. And there won't exist any elements of C3 in the records. Therefore, these records can be deleted and we obtain transaction database D2.

Step 11: To discover the set of frequent 3-itemsets, L3, the algorithm uses the join L2'∞ L2' to generate a candidate set of 3-itemsets C3. There are a number of elements in C3. According to the property of Apriori algorithm, C3 needs is pruned.

Step 12: The transactions in D2 are scanned and the support count of each candidate itemset in C3 is accumulated. Use C3 to generate L3.

Step 13: L3 has only one 3-itemsets so that C4 =null. The algorithm will stop and give out all the frequent itemsets.

Step 14: Algorithm will be generated for Ck until Ck+1 becomes empty.

5. ANALYTICAL ANALYSIS OF THE PROPOSED ALGORITHM

The basic thought of this proposed algorithm is similar with the existing apirori algorithm, that is it gets the frequent item set L1 which has support level larger than or equal to the given level of the users support via scan the database D. Then iterate the process and get L2，L3……Lk.. But still the proposed algorithm differs from the existing algorithm.

The size of the database is reduced by cutting the database using the SizeofTransaction attribute. The optimized algorithm prunes Lk-1 before Ck is generated. In other words, frequent item set in Lk-1 which is going to connect are counted. According to the result obtained item sets with number less than k-1 in Lk-1 are deleted to

decrease the number of the connecting item set and removal of some elements that does not satisfy the conditions. Next the transactions whose value of SOT is less than k-1 are removed from the database. This decreases the possibility of combination, thus declining the number of candidate item sets in Ck, and reduces the number of times to repeat the process and the number of transactions that need to be scanned from iterations. For large database, this algorithm can thus save time and increase the efficiency of data mining. This is what Apriori algorithm does not have.

Although this process can decline the number of candidate item sets in Ck and reduce time cost of data mining, the price of it is pruning frequent item set, which could cost certain time. For dense database such as, telecom, population, multinational companies, census, etc. as they consist of large amounts of long forms, the efficiency of this algorithm is higher than Apriori.

6. CONCLUSION

In this paper, Apriori algorithm is improved based on the properties of cutting database and pruning of the frequent item-set. The typical Apriori algorithm has performance bottleneck in the large data processing so that we need to optimize the existing algorithm with variety of methods. Thus the optimized algorithm prunes Lk-1 before we generate Ck. Thus decreasing the number of connecting item sets and removing the item-sets which does not satisfies the conditions. The improved algorithm proposed optimizes the algorithm by reducing the size of the candidate set Ck and also reducing the I/O spend by cutting down transaction records in the database. The proposed algorithm also reduces the time to repeat the process. For large database, this algorithm can save time cost and increase the efficiency of data mining. Thus the performance of Apriori algorithm is optimized so that we can mine association information from massive data faster and better.

Although this improved algorithm has optimized the existing algorithm and is more efficient but it has overhead to manage the new database after every generation of Lk. Tus, there should exist some approach which requires less number of scans of database. Another solution might be division of large database among processors.

REFERENCES

[1]. A. Rehab H. Alwa and B. Anasuya V Patil, "New Matrix Approach to Improve Apriori Algorithm" In International Journal of Computer Science and Network Solutions, Vol. 1 No 4 December 2013.

[2]. http://en.wikipedia.org/wiki/Data mining

[3]. Ila Chandrakar and A. Mari Kirthima, "A Survey On Association Rule Mining Algorithms", In International Journal Of Mathematics and Computer Research, ISSN: 2320-7167, Vol 1, Issue 10, Page No. 270-272, November 2013.

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

[5]. Jiao Yabing, "Research of an Improved Apriori Algorithm in Data Mining Association Rules", In Internatinal Journal of Computer and Communication Engineering, Vol. 2,No. 1, January 2013.

[6]. Jose L. Balcazar, " Formal and Computational Properties of the Confidence Boost of Association Rules " In ACM Transactions on Kmowledge Discovery from Data, Vol 7, No. 4, November 2013.

[7]. K.Vanitha and R. Santhi, "Using Hash Based Apriori Algorithm to Reduce the Candidate 2- Itemsets for Mining Association Rules ", In Journal of Global Research for Computer Science, Volume 2, No. 5, April 2011.

[8]. Mohammed Al- Maolegi and Bassam Arkok, " An Improved Apriori Algorithm for Association Rules" In International Journal on Natural Language Computing Vol. 3, No. 1, February 2014.

[9]. Rakesh Agrawal, R., T. Imielinski, and A.Swami, "Mining association rules between sets of items in large databases" In Proceedings of the 1993 ACM SIGMOD International Conference

on Management of Data- SIGMOD '93, pp. 207-216, 1993.

[10]. Rina Rawal, Prof Indrjeet Rajput, Prof. Vinit kumar Gupta "Survey on several improved apriori algorithms" In IOSR Journal of Computer Engineering, Volume 9, April 2013.

[11]. Shweta and Kanwal Garg, "Mining Efficient Association Rules Through Apriori Algorithm Using Attributes And Comparative Analysis Of Various Association Rule Algorithms" In International Journal Of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6, June 2013.

[12]. Sotiris Kotsiantis, Dimitris Kanellopoulas, "Association Rules Mining: A Recent Overview", GESTS International Transactions on Computer Science and Engineering, Vol. 329(1), pp. 71-82, 2006.

[13]. Qiankun Zhao and Sourav S. Bhowmick, "Association Rule Mining: A Survey", Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003116, 2003.