AN IMPROVED APRIORI-BASED ALGORITHM FOR ASSOCIATION RULE MINING

1NEESHA SHARMA, 2DR. CHANDER KANT VERMA
1M.Tech Student, DCSA, Kurukshetra University, Kurukshetra, India
2Assistant Professor, DCSA, Kurukshetra University, Kurukshetra, India

ABSTRACT
Association rule mining is an important part of the data mining process. It aims at finding some hidden information or relationships among the attributes of the database. Various algorithms have been proposed for this task. But there are certain limitations of each. In this paper, one of the most widely applicable algorithms i.e. Apriori algorithms has been considered. Firstly, the basic working of the algorithm has been given along with its limitations. Then the improvement in the algorithm has been suggested.

KEYWORDS: Apriori, Association rule mining.

1. INTRODUCTION
Association rule mining tries to find such relationships among the attributes of the database which may be helpful in the task of decision making. It finds its major application in the field of market basket analysis. But it is not restricted to this field only.

Association rules are a set of statements that depict certain relationships between data in any database. An association rule is of the form \( \{X\} \Rightarrow \{Y\} \), where the left hand side i.e. \( \{X\} \) is called the antecedent and the right hand side i.e. \( \{Y\} \) is called the consequent [9]. A more formal definition of association rules is: Let \( I=I_1,I_2,...,I_n \) be a set of \( n \) distinct attributes, \( T \) be any transaction that contains a set of items such that \( T \subseteq I \), \( D \) be a database with different transaction records \( T_s \). An association rule is a statement of the form \( X \Rightarrow Y \) where \( X, Y \subseteq I \) are the sets of items called itemsets and \( X \cap Y = \emptyset \) [10].

The association rules selected after each step depend on some interestingness measure. These interestingness measures prune the non-relevant or uninteresting association rules found at a particular step. Major interestingness measures in association rule mining are
support and confidence. They have been defined as follows:

- **Support:** An association rule \( \{X\} \Rightarrow \{Y\} \) has a support \( s \) in the database \( D \) if \( s\% \) of the transactions in \( D \) contains both \( X \) and \( Y \)[7].

- **Confidence or Predictability:** An association rule \( \{X\} \Rightarrow \{Y\} \) has a confidence \( c \) if \( c\% \) of the transactions in the database \( D \) that contain \( X \) also contains \( Y \)[7].

The paper is organized into four sections. Section 2 gives the related work. Section 3 describes the Apriori algorithm. Section 4 gives the suggested improvement in the Apriori algorithm. Section 5 concludes the paper.

### 2. RELATED WORK

In 2009, Huan Wu et al. [2] proposed an improved algorithm IAA based on the Apriori algorithm. IAA adopts a new count based method to prune candidate itemsets and uses generation record to reduce total data scan amount. Experiments demonstrate that their algorithm outperforms the original Apriori and some other existing ARM methods. Through pruning candidate itemsets by a new count-based method and decreasing the amount of scan data by candidate generation record, this algorithm can reduce the redundant operation while generating frequent itemsets and association rules in the database.

In 2010, Goswami D.N. et al [1] considered three different frequent pattern mining approaches (Record filter, Intersection and Proposed Algorithm) based on classical Apriori algorithm. The conventional algorithm of association rule discovery proceeds in two and more steps but in their approach discovery of all frequent items take the same steps but it takes lesser time as compared to the conventional algorithm. They concluded that that in this new approach, they have the key ideas of reducing time. They performed a comparative study of all approaches on dataset of 2000 transaction. In these approaches Record filter approach proved better than classical Apriori algorithm, Intersection approach proved better than Record filter and finally proposed algorithm proved better than other frequent pattern mining algorithm. This approach is really going to be fruitful in saving the time in case of large database.

In 2011, K. Vanitha and R. Santhi [6] described an implementation of Hash based Apriori. They analyzed, theoretically and experimentally, the principal data structure of their
solution. This data structure is the main factor in the efficiency of their implementation. They proposed an effective hash based algorithm for the candidate set generation. Explicitly, the number of candidate 2-itemsets generated by the proposed algorithm is, in orders of magnitude, smaller than that by previous methods, thus resolving the performance bottleneck. Their approach scans the database once utilizing an enhanced version of Apriori algorithm. The generation of smaller candidate sets enables us to effectively trim the transaction database size at a much earlier stage of iterations, thereby reducing the computational cost for later iterations significantly. Determining frequent objects is one of the most important fields in data mining. This algorithm can achieve a smaller memory usage than the Apriori algorithm. It is well known that the way candidates are defined has great effect on running time and memory need. They have presented experimental results, showing that the proposed algorithm always outperforms Apriori. Hash based Apriori is most efficient for generating frequent itemset than Apriori.

In 2012, Sunil Kumar et al [11] proposed a new algorithm which takes less number of scans to mining the frequent itemsets from the large database which leads to mine the association rule between the data. They addressed the importance of knowledge mining from a large data set and overview of existing algorithm and its flaws and innovative solution with a new algorithm for data mining from the large dataset. It was seen in the observation and analysis that the new proposed algorithm performs better than the existing Apriori. The relative performance of the proposed algorithm using probability and matrix under different minsupport specify its excellence and features. The improved algorithm can be used in many areas such as medical, image processing and database and ERP etc. with a reduced time and space complexity requirements.

In 2013, Jiao Yabing [4] proposed an improved version of Apriori algorithm for association rule mining. Apriori algorithm is the classic algorithm of association rules, which enumerate all of the frequent itemsets. When this algorithm encountered dense data due to large number of long patterns emerge, this algorithm's performance declined dramatically. In order to find more valuable rules, a new algorithm
has been proposed. Finally, the improved algorithm is verified, the results show that the improved algorithm is reasonable and effective, can extract more valuable information.

3. THE APRIORI ALGORITHM

3.1 BASIC ALGORITHM

Apriori algorithm is the most widely used algorithm for association rule mining. Apriori algorithm is a milestone in the history of association rule mining. In Apriori algorithm, firstly the candidate itemsets are generated. Then the database is scanned for checking the support of these itemsets to generate frequent 1-itemset. During the first scan, 1-itemsets are generated by rejecting those itemsets whose support is below the threshold. In the consequent passes, the candidate k-itemsets are generated after (k-1)th pass over the database by joining (k-1) itemsets. The pruning of the non-interesting itemsets is done according to the Apriori property which states that the subset of a frequent itemset must also be frequent [8]. Apriori works in two steps [6]:

The ‘Join step’ constructs new candidate sets. A candidate itemset is an itemset that either is frequent or infrequent with respect to the support threshold. Higher level candidate itemsets (C_k) are generated by joining previous level frequent itemsets L_{k-1} with itself. The ‘Prune step’ filters out candidate itemsets whose subsets are not frequent. This is based on Apriori property as a result of which every subset of a frequent itemset is also frequent.

The pseudo code for Apriori algorithm is as follows [3]:

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

**Output**: L, frequent itemsets in D

**Method**:

1. \(L_1=\text{find} \_\text{frequent} \_1\text{-itemsets} (D)\);
2. for (k=2; L_{k-1}≠\emptyset; k++){
3. \(C_k=\text{Apriori}\_\text{gen}(L_{k-1}, \text{min} \_\text{sup})\);
4. for each transaction \(t\in D\{
5. \(C_t=\text{subset}(C_k,t)\);
6. for each candidate \(c\in C_t\)
7. \(c.\text{count}++ \);
8. }
9. \(L_k=\{c\in C_k | c.\text{count}\geq \text{min} \_\text{sup}\}\)
10. }
11. return \(L=\cup_k L_k\);

**Procedure** Apriori_gen \((L_{k-1}: \text{frequent}(k-1)\text{-itemsets})\):

1. for each itemset \(l_1\in L_{k-1}\{
2. for each itemset \(l_2\in L_{k-1}\{
3. if([l_1[1]=l_2[1])\land ([l_1[2]=l_2[2])\land\ldots\land([l_1[k-2]=l_2[k-2])\land([l_1[k-1]<l_2[k-1])\text{then}\{
4. \(c=l_1\infty l_2;\)
(5) if has_infrequent_subset(c, Lk-1) then
(6) delete c;
(7) else add c to Ck;
(8) });
(9) return Ck;

Procedure has_infrequent_subset(c: candidate k-itemset; Lk-1: frequent(k-1)-itemsets)
(1) for each (k-1)-subset s of c {
(2) if s ∉ Lk-1 then
(3) return true; }
(4) return false;

The working of the algorithm is explained below through an example (Figure 1). A database has ten transactions. Let the support specified by the user be 3.

The result of applying the Apriori algorithm on the given database is obtained after L3. The frequent itemset obtained is \{2, 3, 4\}.

3.2 LIMITATIONS OF APRIORI
Although the working of Apriori algorithm is very simple, but there are certain limitations of this algorithm which are listed below [5]:

a) It only explains the presence and absence of an item in transactional databases.
b) In case of large dataset, this algorithm is not efficient.
c) In Apriori, all items are treated equally by using the presence and absence of items.
3.3 WAYS TO IMPROVE APRIORI

Various methods have been suggested in [5] which can help in improving the efficiency of Apriori algorithm.

a) Transaction Reduction: transactions that do not consist of frequent item-sets are of no importance in the next scans for searching frequent item-sets.
b) Hash based item-set counting: hashing table is used for counting the occurrences of itemsets.
c) Partitioning: for any item-set i.e. frequent in database, then that item-set must be frequent in at least one of the partition of database.
d) By adding attribute Weight and Quantity: means how much quantity of item has been purchased.
e) By adding attribute Profit: that can give the valuable information for business and customers.
f) By reducing the number of scans.
g) By removing the large candidates that cause high Input/output cost.

4. PROPOSED ALGORITHM

The proposed method represents the database in the form of a matrix .the rows of the matrix represent the transactions while each column of the matrix represents the items or the attributes of the database.

\[ A_{ij} = \begin{cases} 1 & \text{if transaction } i \text{ has item } j \\ 0 & \text{elsewhere} \end{cases} \]

An additional row is inserted in the matrix which gives the count of the \( j \)th column. An additional column is considered in the database which acts as a tag of each transaction and gives the size of each transaction. Three factors are considered for pruning of items in the database. Those candidate itemsets are pruned where any of the its subset is not frequent (Apriori property). Those candidate itemsets where count is less than the minimum support threshold are pruned. Those rows where size-of-transaction tag is
less than the minimum support are deleted from the database. This method cuts the size of the database. So for next scan by the algorithm, a smaller database is presented. The proposed algorithm is as follows:

1) Represent the database as a matrix where each row represents the transaction of the database and each column represents the item in the database.

2) Insert an additional column acting as a tag for each transaction called as size-of-transaction(SOT).

3) Insert an additional row that counts the occurrences of each item in the database.

4) $L_1=$ find_frequent_1-itemsets($D$);

5) For($k=2; Lk\neq \Phi; k++$){

6) $C_k=\text{Apriori\_gen}(L_{k-1}, \text{min\_sup})$;

7) For each transaction $t \in D$

8) $C_t=\text{subset}(C_k,t)$;

9) For each candidate $c \in C_t$

10) $c\cdot \text{count}++$;

11) }

12) $L_k=$ \{$c \in C_k | c\cdot \text{count} \geq \text{min\_sup}$ \};

13) if($k=2$){

14) $\text{delete\_datavalue}(D, L_k, L_{k-1})$;

15) $\text{delete\_datarow}(D, L_k)$;

16) }

17) return $L=U_k L_k$;

Procedure \text{Apriori\_gen}(L_{k-1}: \text{frequent}(k-1)-itemsets)

1) For each itemset $l_1 \in L_{k-1}$

2) For each itemset $l_2 \in L_{k-1}$


4) $c=l_1 \bigcirc \mid l_2$

5) For each candidate $c \in C_k$

6) if $l_1$ is the subset of $c$

7) Then $c\cdot \text{num}++$;

8) $C'_{k}=$ \{$c \in C_k | c\cdot \text{num}=k$\};

9) return $C'_{k}$;

Procedure \text{delete\_datavalue}(D; L_k: \text{frequent}(k)-itemsets; L_{k-1}: \text{frequent}(k-1)-itemsets)

1) For each itemset $i \in L_{k-1}$ and $i \notin L_k$

2) For each transaction $t \in D$

3) For each datavalue$\in t$

4) If (datavalue=$i$)

5) Update datavalue=null;

6) }

Procedure \text{delete\_datarow}(D; L_k: \text{frequent}(k)-itemsets)

1) For each transaction $t \in D$

2) If (SOT<$\text{min\_sup}$)

3) Delete datarow;

4) }

The proposed algorithm has been explained through the following example. The same example has been solved by using Apriori algorithm with support=3.
The proposed algorithm is based on the property of cutting down the database after each step. Various methods have been proposed to cut down the database so that the efficiency of the Apriori algorithm could be increased. Here, the database is cut by comparing the rows and columns with the minimum support specified by the user. The number of iterations of the proposed algorithm comes out to be equal to the original Apriori algorithm. But, unlike the original Apriori, the proposed method does not scan the complete database after each step. It has to scan only a smaller version of database every time.

Figure 2: Finding Frequent Itemsets Using Proposed Algorithm
The result of applying the proposed algorithm on the given database is obtained after L3. The frequent itemset obtained is \{2, 3, 4\}.

5. CONCLUSION

In this paper, Apriori algorithm is improved on the basis of the property of cutting database. The typical Apriori algorithm has one major performance bottleneck in the massive data processing. It needs to scan the same large database every time it has to calculate the candidate itemsets. This is very much time consuming tasks thus making the algorithm slow. The proposed algorithm in this paper not only optimizes the algorithm by cutting down transaction records in the database. Thus, every time when candidate generation process is carried out, the algorithm is presented with smaller number of transactions. The number of scans of the database made by the algorithm is the same but size of the database is reduced after each step. Although this proposed algorithm is optimized and efficient over the original Apriori algorithm but it has overhead to manage the new database after every generation of \(L_k\). Moreover, if the minimum support specified by the user is 1, then the algorithm will behave exactly like the original Apriori algorithm.

REFERENCES


