An Enhanced Apriori Algorithm for Frequent Pattern Matching

Bhukya Krishna and Geetanjali Amarawat

1CSE, CMR Technical Campus, Medchal, Hyd. krish685@gmail.com
2CSE, Madhav University, Aburoad, Rajasthan. gits.princess@gmail.com

Abstract

The progressions in the field of database innovation have made it conceivable to store a colossal measure of data. Data mining strategies have been broadly utilized for removing non-paltry data from such huge measures of data. It is helpful in numerous applications like key basic leadership, monetary conjecture, and medicinal conclusion and so forth. Data mining can be connected either as a elucidating or as a prescient device. Affiliation manage mining is one of the functionalities of data mining. This postulation proposes a couple of strategies for moving forward affiliation govern mining, affiliation administer covering up, and post mining. The way toward creating affiliation rules includes the errand of finding the set of all the successive thing sets and creating promising standards. This proposal proposes a procedure for determining the continuous thing sets with a solitary output of the exchanges database in the circle. Amid this single database examine data about thing sets and their events are caught in a table kept in the principle memory. While determining the regular thing sets, this table is examined rather than the plate.

Sometimes, the quantity of incessant thing sets to be produced is vast. All these thing sets are hard to deal with and oversee. This issue can be understood by mining Maximal Frequent Sets (MFS) alone. A MFS of length m suggests the nearness of 2m-2 visit thing sets. Subsequently, all the continuous thing sets can be promptly surmised from the MFS. Along these lines, the era of MFS lessens the time taken for acquiring all the continuous thing sets. Another quicker procedure is proposed in this proposition for mining the MFS, which decreases the quantity of database sweeps to the most extreme of two and furthermore keeps away from the era of competitors. Despite the fact that data mining has a ton of benefits, it has a couple of bad marks moreover. Delicate data contained in the
database might be brought out by the data mining devices. Distinctive methodologies are being utilized to shroud the delicate data. It is watched that a large portion of the concealing algorithms in the current writing, work at the exchanges level to conceal some touchy data. This is a tedious stride in the concealing procedure. In this theory, two new procedures have been proposed to lessen the time multifaceted nature of the concealing procedure. One strategy accomplishes the decrease of time many-sided quality by altering the structure of the Frequent Pattern Growth Tree. The other strategy accomplishes the decrease by proper dividing of the database. Hypothetical analysis of the algorithms demonstrates that the stowing away of data limits the time and space complexities to a incredible degree.
1. Introduction

The challenging task is to extracting useful information from the large collection of data in Dataware house and data base. Around the world lot of research is underway to discover the knowledge from the large collection of data in data warehouse. In this process many algorithms has been proposed to identify the associations between the data in the database, leads to mine the association rule among the data. Association rules are used for knowledge discovery and to take useful managerial decision in the organization based on the results of associations among data stepping toward to make a smarter system. In this regard, the first algorithm Apriori was proposed in the year 1994 by Agarwal and Srikanth to mine the frequent item set. Time constraint and efficiency of algorithms leads to lot of research in the area of algorithm to build efficient algorithm which takes less time and few number of database scans to mine frequent item set and association rule. Association rule is based mainly on discovering frequent item sets. Association rules are frequently used by retail stores to assist in marketing, advertising, inventory control, predicting faults in telecommunication network. The remaining part of this paper is organized as follows: Section 2 contains Apriori Algorithm with worked example. In section 3, elaborate the proposed algorithm for transaction reduction technique and worked example. Section 4 contains conclusion.

2. Background

A lot of association rule algorithms have been developed in the last decades [11-13], which can be classified into two categories: (1) breadth-first search (BFS) or candidate-generation-and-test approach such as Apriori [14], (2) depth-first search (DFS) or pattern-growth approach [15-18]. With BFS the support values of all (k - 1) itemsets are determined before counting the support values of the k-itemsets. In contrast, DFS recursively descends following the tree structure defined above. Each of the algorithms is characterized by its strategy to a) traverse the search space and b) determine the support values of the itemsets as shown in figure 1. In addition an algorithm may employ specific optimizations for further speeding up. The most popular algorithm of this type is Apriori [16, 19] where the downward closure property of itemset support was introduced. Apriori makes an additional use of this property by pruning those candidates that have an infrequent subset before counting their supports. This optimization issue becomes possible because BFS ensures that the support values of all subsets of a candidate are known in advance. Apriori counts all candidates of a cardinality k together in one scan over the database. The critical part is to look for the candidates in each of the transactions. For this purpose, the work in [16] introduces a so called hash-tree structure. The items in each transaction are used to descend in the hash-tree. Whenever they reach one of its leaves, they find a set of candidates having a common prefix that is contained in the transaction. Then these candidates are searched in the transaction that has been encoded as a bitmap [16]. In the case of success, the counter of the
candidate in the tree is incremented.

Figure 1: Systematization of the Algorithms

- Apriori Tid [16] is an extension of the basic Apriori approach.
- SETM algorithm [17] was motivated by the desire to use SQL to compute large itemsets, the candidate itemsets are generated on-the-fly during the pass as data is being read.
- DIC is a further variation of the Apriori-Algorithm [18, 19]. DIC softens the strict separation between counting and generating candidates. Whenever a candidate reaches min-sup; that is even when this candidate has not yet "seen" all transactions, DIC starts generating additional candidates based on it.
- The Partition-Algorithm [20, 21] uses set intersections to determine support values.

**Apriori Algorithm for Generating Frequent Itemsets**

Apriori is the best-known algorithm to mine association rules. It uses a breadth-first search strategy to count the support of itemsets and uses a candidate generation function which exploits the downward closure property of support.

Apriori algorithm employs an iterative approach known as level-wise search, where k-item sets are used to explore (k+1)-item sets. First, the set of frequent 1-itemsets L1 is found. Next, L1 is used find the set of frequent 2-itemsets L2. Then L2 is used to find the set of frequent 3-itemsets L3. The method iterates like this till no more frequent k-item sets are found.

We can easily understand the concepts used by the Apriori with the help of following example. Table 1 shows a transactional database having 9 transactions. **TID** is a unique identification given to the each transaction. Let minimum support value be 2 (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>ITEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>I1, I2, I5</td>
</tr>
<tr>
<td>T2</td>
<td>I2, I4</td>
</tr>
<tr>
<td>T3</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T4</td>
<td>I1, I2, I4</td>
</tr>
<tr>
<td>T5</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T6</td>
<td>I2, I3</td>
</tr>
<tr>
<td>T7</td>
<td>I1, I3</td>
</tr>
<tr>
<td>T8</td>
<td>I1, I2, I3, I5</td>
</tr>
<tr>
<td>T9</td>
<td>I1, I2, I3</td>
</tr>
</tbody>
</table>
Figure 2: Generation of frequent item sets by applying Apriori Algorithm

**Pseudo Code**

Initialize: \( k := 1 \), \( C_1 \) = all the 1-item sets;
read the database to count the support of \( C_1 \) to determine \( L_1 \).
\( L_1 \) := \{ frequent 1-item sets \}; \( k:=2; //k \) represents the pass number//
while (\( L_k \) - \( L_{k-1} \) ≠ ø) do
begin
\( C_k \) := gen_candidate_itemsets with
the given \( L_{k-1} \)
prune(\( C_k \))
for all transactions \( t \in T \) do
increment the count of all candidates
in \( C_k \) that are contained in \( t \);
\( L_k \) := All candidates in \( C_k \) with
minimum support ;
k := \( k + 1 \);
end
Answer := \( U_k L_k \)

**Existing Algorithms**

**The General Apriori Algorithm**
The general Apriori algorithm is:
\( T \): Transactional data base
\( C_k \): Candidate item set of size \( k \)
\( L_k \): Frequent item set of size \( k \)
\( s \): Support
Apriori(\( T, s \))
\( L_1 \) ← \{ large 1-item set that appear in more than or equal to \( s \) transactions \}
k ← 2
While $L_{k-1} \neq \phi$

$C_k \leftarrow \text{Join}(L_{k-1})$

For each transaction $t$ in $T$
For each candidate $c$ in $C_k$
If ($c \subset t$) then
  $\text{count}[c] \leftarrow \text{count}[c] + 1$
End If
End For
End For

$L_k = \phi$

For each candidate $c$ in $C_k$ //Prune
If ($\text{count}[c] \geq s$) then
  $L_k \leftarrow L_k \cup \{c\}$
End If
End For

$k \leftarrow k + 1$

End While
Return $L_k$
End Apriori

In general Apriori algorithm, for each candidate in $C_k$, frequency is calculated by scanning transactional database. After calculating frequencies for all candidates in a $C_k$ these frequencies are compared with support, $s$ and exclude candidates with frequencies less than $s$. This results in generation of $L_k$.

The general Apriori algorithm has some flaws:

- The transactional database is scanned repeatedly. This is because every candidate of candidate set ($C_k$) generated after Join operation must be checked in all transactions of transactional database for the presence of candidate.
- If there are adequate transactions then the general Apriori algorithm is not apt.

**Drawbacks of Apriori Algorithm**

- Large Number of infrequent itemset are generated which increases the space complexity.
- Too many database scans are required because of large number of itemsets generated.
- As the number of database scans are more the time complexity increases as the database increases.

**3. Proposed Enhanced Apriori Algorithm**

**Enhanced Apriori Algorithm**

In classical Apriori algorithm, when candidate itemsets are generated, the algorithm needs to test their occurrence frequencies. The manipulation with redundancy will result in high frequency in querying, so tremendous amount of
re-sources will be expended in time or in space. Therefore the improved algorithm was proposed for mining the association rules in generating frequent k-item sets. Instead of judging whether these candidates are frequent item sets after generating new candidates, this new algorithm finds frequent item sets directly and removes the subset that is not frequent, based on the classical Apriori algorithm.

The improvement is for reducing query frequencies and storage resources. The improved Apriori algorithm mines frequent item sets without new candidate generation.

**Figure 3: The Proposed Enhanced Apriori Algorithm**

**Deterministic-function Pruning based Enhanced Apriori Algorithm**

**Input:** Transaction database $D$

**Output:** Non-coincidental frequent itemsets

**Procedure:**

1. **For all transactions** $t \in D$
   
   - $C_t = \text{subset}(C_1, t)$
   
   - **For all candidates** $c \in C_t$

2. **C1 = Min_sup(C1)**
   
   **For** $k = 2; L_k-1 \neq \emptyset; k++$

3. **Ck = Apriori(Lk-1)**
   
   **For all transactions** $t \in D$

4. **Ct = subset(Ck, t)**

5. **For all candidates** $c \in C_t$
Improved Algorithm Steps

The improved algorithm is described in following steps:

Input:
- D, a database of transactions
- Min_sup, the minimum support count threshold

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemset C1. The algorithm simply scans all of the transaction to count the number of occurrences of each item.
2. The set of frequent item sets, L1, is determined by comparing the candidate count with minimum support count which contains candidate 1-itemsets satisfying minimum support.
3. To generate the set of frequent 2-itemsets, L2, the algorithm generate a candidate set of 2-itemsets and then the transactions in D are scanned and the support count of each candidate item set in C2 is accumulated and then repeating the step 2.
4. Then D2 is determined from L2.
5. Generate C3 candidates from L2 and scan D2 for count of each candidate and then repeating step 2.
6. At the end of the pass, determine which of the candidate item sets are actually large, and those become the seed for the next pass.
7. This process continues until no new large item sets are found (Fig. 1).

The improved Apriori algorithm reduces the number of database scans and the redundancy while generating subtests and verifying them in the database. Because of which this algorithm takes less time for generating frequent item set as compared to classical Apriori algorithm [1].

Figure 4: Example of Generation of Candidate Item Set and Frequent Item Set
Although it is a first algorithm proposed in this field of frequent pattern mining, but with the time a number of modified algorithm were designed to improve the efficiency of time, memory management and remove the complexity of process. Here we are presenting a different approach in Apriori algorithm to count the support of candidate item set. Basically his approach is more appropriate for vertical data layout, since Apriori basically works on horizontal data layout. In this new approach, we use the set theory concept of intersection. In Classical Apriori algorithm, to count the support of candidate set each record is scanned one by one and check the existence of each candidate, if candidate exists then we increase the support by one. This process takes a lot of time, requires iterative scan of whole database for each candidate set, which is equal to the max length of candidate item set. In modified approach, to calculate the support we count the common transaction that contains in each element’s of candidate set, by using the intersect query of SQL. This approach requires very less time as compared to classical Apriori.

4. Results

The experiments were performed on machine using an Intel core i5 processor with 450 GB HDD and 4GB RAM. To carry out the performance studies, we implemented the basic Apriori algorithm and QR algorithm. Apriori algorithm alone and Apriori algorithm using significant attribute selection were analyzed and compared. Analysis has been done by varying various parameters like support, confidence, number of records, number of dimensions Basic Apriori algorithm:

![Figure 5: Results](image)

The traditional Apriori algorithm is one of the best algorithms used for association rule mining. But the only disadvantage it faces is that it is not scalable and hence fails for high dimensional datasets. Moreover it takes exponential time as the dimension of dataset increases. Experimentation shows that the traditional Apriori algorithm takes infinite and large amount of time in
generating association rules compared to Apriori algorithm using significant attribute selection. There is approximately a reduction of about 40% time than the basic algorithm when varied with number of records and about 35% when varied with number of dimensions. This has been depicted in table 2.

TABLE 2: Time(s) vs Number of Records

<table>
<thead>
<tr>
<th>NO OF RECORDS</th>
<th>APRIORI</th>
<th>APRIORI WITH QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>45000</td>
<td>30</td>
<td>16.57</td>
</tr>
<tr>
<td>90000</td>
<td>59.11</td>
<td>27.17</td>
</tr>
<tr>
<td>135000</td>
<td>88.3</td>
<td>40.22</td>
</tr>
<tr>
<td>180000</td>
<td>101.48</td>
<td>54.46</td>
</tr>
<tr>
<td>225000</td>
<td>114.41</td>
<td>67.04</td>
</tr>
</tbody>
</table>

5. Conclusion

Throughout the last decade, a lot of people have implemented and compared several algorithms that try to solve the frequent item set mining problem as efficiently as possible. For example, a very often used implementation of the Apriori algorithm is, An Enhanced Scaling Apriori for Association Rule Mining Efficiency.

Nevertheless, when we compared his implementation with ours, the performance of both algorithms showed immense differences. We can conclude that in this new approach, we have the key ideas of reducing time. As we have proved above how the Enhance apriori take less time than that of classical algorithms. That is really going to be fruitful in saving the time in case of large database. This key idea is surely going to open a new gateway for the upcoming researcher to work in the field of the data mining.

Acknowledgement

The author would like to express their sincere gratitude to the Chairman C. Gopal Reddy, Director Dr. A. Raji Reddy and Supervisor(s) for their constant encouragement and co-operation.

References


