A Dynamic Solution for Association Rules Mining

P.AMARANATHA REDDY, D.HARSHA VARDHAN RAJU, P.GOVARDHAN

1,2Dept. Of CSE, School of Eng. & Tech, SRI Padmavati Mahila University, Tirupati, 3Dept. Of CSE, SVCET.

E-mail: amaranatha.p@gmail.com, mailinharsha@gmail.com, govardhan.ctr@gmail.com

ABSTRACT

We consider the problem of discovering association rules between items in a large database of sales transactions. We present a new algorithm for solving the problem that is fundamentally different from the known algorithms. This algorithm solves most of the major problems of well known association rules mining algorithms like Apriori and FP-tree. In this algorithm we generated mutually independent candidates as per our requirement in a single scan. It is very flexible to use in incremental mining system. During the interactive mining process, users may change the threshold of support according to the rules. In our algorithm once candidates are generated we can change the support without repeating the whole process. It is suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database. In our algorithm insertion of these datasets is possible without repeating of the whole process. We can generate the candidates for required items and required itemsets in a single scan. This makes effective memory utilization and improves the performance.

Keywords: Association Rules Mining, Frequent Itemsets, Support, Confidence, Mutually Independent Candidates.

1. INTRODUCTION

Association rule mining, one of the most important and well researched techniques of data mining. It aims to find interesting association or correlation relationships among a large set of data items. With massive amounts of data continuously collected and stored, many industries are becoming interested in mining association rules from their databases. The discovery of interesting association relationships among huge amounts of business transaction records can help in many business decision making process. Consider an example, supermarket with a large collection of items. Typical business decisions like the management of the supermarket has to decide which product has to put on sale, how to design coupons particular products, how to place merchandise on shelves in order to maximize the profit, etc. Analysis of past transaction data is a commonly used approach in order to improve the quality of such decisions. Until recently, however, only global data about the cumulative sales during some time period (a day, a week, a month, etc.) was available on the computer. Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub-problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Since the second sub problem is quite straight forward, most of the researches focus on the first sub problem. The first sub-problem can be further divided into two Sub problems candidate large itemsets generation process and frequent itemsets generation process. We call those itemsets whose support exceed the support threshold as large or frequent item-sets, those itemsets that are expected or have the hope to be large or frequent are called candidate itemsets [6]. In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. In this paper, we surveyed the most recent existing association rule mining techniques. The organization of the rest of the paper is as follows. Section 2 provides the preliminaries of basic concepts and their notations. Section 3 describes the well known algorithms and their drawbacks. Section 4 describes our proposed Single scan mutually independent candidate’s generation algorithm. Finally, Section 5 concludes the paper.

2. BASIC CONCEPTS AND NOTATIONS

Let I=I1, I2, …, Im be a set of m distinct attributes, T be transaction that contains a set of items such that T ⊆ I, D be a database with different transaction records Ts. An association rule is an implication in the form of X → Y, where X, Y are subsets of I. X is called antecedent while Y is called consequent, the rule means X implies Y. There are two important basic measures for association rules, support(s) and confidence(c) [4].

2.1 Support(s)

Support(s) of an association rule is defined as the percentage/fraction of records that contain X E Y to the total number of records in the database. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item.
2.2 Confidence(c)
Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain \( X \cap Y \) to the total number of records that contain \( X \). Confidence is a measure of strength of the association rules, suppose the confidence of the association rule \( X \cap Y \) is 80%, it means that 80% of the transactions that contain \( X \) also contain \( Y \) together.

\[
\text{Confidence}(X|Y) = \frac{\text{Support}(XY)}{\text{Support}(X)}
\]

For example consider the following itemset.

For rule \( \text{Milk} = \text{Sugar} \)
\[
\text{Support}(s) = \text{support(\{Milk, Sugar\})} = 50\%
\]
\[
\text{Confidence}(c) = \frac{\text{support(\{Milk, Sugar\})}}{\text{support(\{Milk\})}} = 60\%
\]

3. WELL KNOWN ASSOCIATION RULES MINING ALGORITHMS AND THEIR DRAWBACKS
Association rule mining is a well explored research area, we will only introduce some basic and famous algorithms for association rule mining [3] [4].

3.1 Apriori Algorithm
Apriori is a great improvement in the history of association rule mining, Apriori algorithm was proposed by Agrawal in [Agrawal and Srikant 1994] [1] [6]. It follows two step process consisting of join and prune actions. In the join step candidate itemsets are generated, then in the prune step database is scanned to check the actual support count of the corresponding itemsets and remove the itemsets whose support is below the threshold.

During the first scanning of the database all the 1-itemsets and support count of each item is calculated and the frequent 1-itemsets are generated by pruning those itemsets whose supports are below the pre-defined threshold. The candidate \( k \)-itemsets are generated by joining the frequent \( k-1 \)-itemsets. All the candidate \( k \)-
itemsets are pruned whose support is below the pre-defined threshold. Terminate when no frequent or candidate set can be generated. The algorithm is shown in Figure 2.

Bottlenecks of the Apriori algorithm are complex candidate generation process that uses most of the time, space, memory and multiple scan of the database. Changing of support may lead to repetition of the whole mining process. Another limitation is it is not suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database, those insertions may also lead to a repetition of the whole process if we employ Apriori algorithm.

3.2 FP-Tree (Frequent Pattern Tree) Algorithm

Pattern mining, is another milestone in the development of association rule mining. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-Tree was introduced by Han et al in [Han et al. 2000] [2] [6].

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support count. The set of frequent items is sorted in the order of descending support count.

An FP-Tree is then constructed as follows. First, create the root of the tree labeled with “null”. Scan the database second time. The items in each transaction are processed in order (i.e., sorted according to descending support count) and a branch is created for each transaction. To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of links.

The mining of the FP-tree proceeds as follows. Start from each frequent length 1 pattern construct its conditional pattern base then construct its FP-tree, and perform mining recursively on such tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

3.3 MUTUALLY INDEPENDENT CANDIDATES ALGORITHM

4.1 Mining process

In this, we divided the total algorithm process into two phases. In the first phase candidate item sets are generated, and then in the second phase frequent itemsets are generated.

Phase-1 Candidate itemsets generation

In the whole process of our algorithm scanning is performed only once. Whenever the scanning is completed all the candidates are ready for processing. Start scanning from the first transaction; generate all possible subsets of items in the transaction except null set. For each subset based on the size of subset make entries in appropriate candidate with support count value as 1. If the corresponding candidate is not available create new one.

Likewise, if scanning is performed for kth transaction, make all possible subsets for items in the transaction except null set. For these subsets, based on the size in appropriate candidate for the corresponding subset
increment the support count as previous support count value + 1 if the subset is already available else made new entry with support count value 1.

This process is performed until the last transaction. Once the scanning is completed all the candidates are ready. From these candidates we can get the frequent itemsets based on common or individual support count.

Now we consider some special cases where our algorithm works better than other algorithms. Suppose in one of the supermarket management is not interested to generate item Ik frequent itemsets because item Ik is banded in the market or outdated item. And the management is only interested to generate 3-itemset frequent items because only 3 racks in shelve are empty.

In this type of requirement generation of all the frequent itemsets is waste of process, time and memory. In our algorithm all the generated candidates are mutually independent i.e. change or deletion of one candidate doesn’t affect another candidate. By this future we can control the generation of candidates in this algorithm by specifying the uninteresting candidates or uninteresting items. Hence this makes the effective memory utilization and improves the performance.

Phase-2 Frequent itemsets generation
After the completion of first phase all the candidates are ready. By using these candidates we will produce frequent itemsets for the specified support for individual candidate or common support for all candidates.

It is very flexible to use in interactive mining system. During the interactive mining process, users may change the threshold of support according to the rules. In our algorithm once candidates are generated we can change the support without repeating the whole process.

4.2 Pseudo code:

\[ C_k : \text{Candidate itemset of size } k \]
\[ L_k : \text{Frequent itemset of size } k \]
\[ S_k : \text{Support for Candidate } C_k \]
\[ UC[ ] : \text{Uninteresting Candidates} \]
\[ UI[ ] : \text{Uninteresting Items} \]

for \( i = 1; i != \text{null}; i++ \) do begin

for each transaction \( t \) in database do

generate subsets for itemset in transaction

if each item in subset not present in UI[ ] and subset not belongs to UC[ ]

increment the support count for each subset in appropriate candidate if subset is available

else insert as a new entry in the with support count 1.

end

for each candidate \( C_k \)

generate \( L_k \) with threshold support \( S_k \)

return \( F_k \), \( L_k \).

4.3 Benefits of our approach
In our algorithm all the generated candidates are mutually independent i.e. change or deletion of one candidate doesn’t affect another candidate. This feature provides so many benefits compare with the remaining association rules algorithms.

4.3.1 Controlling uninterested candidates generation
In our algorithm all the generated candidates are mutually independent i.e. change or deletion of one candidate doesn’t affect another candidate. With this future we can control the generation of candidates by specifying the uninteresting candidates or uninteresting items. This feature makes the effective memory utilization and improves the performance.

4.3.2 Interactive mining
It is very flexible to use in interactive mining system. During the interactive mining process, users may change the threshold of support according to the rules. In our algorithm once candidates are generated we can change the support without repeating the whole process.

4.3.3 Incremental mining
It is suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database. In our algorithm insertion of these datasets is possible without repeating the whole process (i.e. scanning and candidates generation for previous data). Take the transaction \( T_k \) may add to our Transaction data base in future then we can perform the candidate generation without scanning the old database.

Fig. 4: describes the Mutually independent candidate generation in which generation of 2-itemset candidate is uninterested and item I4 is uninterested and generate frequent itemset with support = 7, 4 for 1-itemset candidate, support = 3, 2 for 3-itemset candidate, support = 2 for 4-itemset.

When scanning is performed for the first transaction i.e. \( T_{100} \) corresponding items are \{I1, I2, I5\}. Possible subsets for this except {} (null set) are \{{I1}, {I2}, {I5}, {I1, I2}, {I1, I5}, {I2, I5}, {I1, I2, I5}\}. Among these 2-itemsets are uninterested hence \{I1, I2\}, \{I1, I5\}, \{I2, I5\} are eliminated for candidate generation and I4 is
uninterested but I4 is not present in I1. From remaining subsets size of \{I1\}, \{I2\}, \{I3\} is 1, so kept these in C1 with sup. Count=1. And size of \{I1, I2, and I5\} is 3, so kept this in C3. With this scanning of first transaction is completed coming to T200 corresponding items are \{I2, I4\}. Possible subsets for this except {} (null set) are \{\{I2\}, \{I4\}, \{I2, I4\}\}. From these subsets \{I2, I4\} is eliminated because of set size 2, i.e uninterested items set and \{4\} is eliminated because of uninterested item. Remaining \{I2\} size is 1 and \{I2\} is already available in C1 so simply increment count of \{I2\}. (i.e sup. Count of \{I2\} is 2). This process is continues till the last transaction and the final results are shown in Fig 4.

![Fig 4 Mutually independent candidate’s algorithm mining process](image)

Transaction Tk is for future purpose. If new transaction itemsets added to our database in future then the scanning process is carried out from that transaction without repeating the whole process again.

5. EXPERIMENTAL CONDITIONS
In our experiment to have a good look and feel we have used JSP (Java Server Pages) which is also available on every system nowadays as a third party application hence, helps in an easy deployment. We did not use any kind of database in developing the code. For maintaining the transaction itemsets data, candidate itemsets and frequent itemsets we used files. The Apache Tomcat software processes the server requests from the client.

6. CONCLUSION & FUTURE WORK
We surveyed the well known existing association rule mining techniques and their drawbacks. We presented new algorithm, mutually independent candidates generated algorithm it resolves most of the major problems of existing algorithms. Our algorithm is very flexible to use in interactive mining system and suitable for incremental mining. Scanning database is performed only once hence it improves the performance. We can eliminate the generation of uninteresting candidates and uninteresting items which makes the effective memory utilization.

Although it solves most of the problems we need very good user interaction in our mining process. If the user have very good knowledge on items which are interested and which are uninterested, which itemsets are interested and which itemsets are uninterested this algorithm provides best mining process.
ACKNOWLEDGEMENTS
This work was supported by Dept. Of CSE in School of Engineering & Technology, SRI Padmavati Mahila University, Tirupati. We thanks all my colleagues for their valuable suggestions and we thanks our Head of the department for encouraging us to do this research.

REFERENCES