Recommending An Insurance Policy Using Association Rule Mining

Nilkamal More
Assistant Professor
Department of Information Technology
K.J. Somaiya college of Engineering Vidyavihar, Mumbai-400077

Suchitra Patil
Assistant Professor
Department of Information Technology
K.J. Somaiya college of Engineering Vidyavihar, Mumbai-400077

Abstract

Improving the efficiency of ascertaining the frequent itemsets is a crucial issue in association rule mining algorithms. This paper illustrates the use of Apriori algorithm for Life Insurance Corporation for recommending a policy to a customer who is interested in taking a policy. For recommending a policy to a customer, the information about existing policy holders is taken into consideration. This paper also analyzes the performance of “An Improved Apriori Algorithm based on Matrix” [1].

Keywords: Apriori Algorithm, Boolean Matrix, Frequent Itemsets, Association Rules.

I. INTRODUCTION

Association rule mining is one of the most important and well researched techniques of data mining. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Apriori algorithm is one of the most famous classical algorithms of association rule mining. Apriori algorithm for association rule mining based on matrix can be used to recommend a life insurance policy for a person based on the existing records.

It is one of the most important algorithms but it has two severe deadlocks:

A. Huge candidate sets.
   - $10^4$ frequent 1-itemset will generate $10^7$ candidate 2-itemsets.
   - To dissolve a frequent pattern of size 100, example: \{a1, a2, a3, ..., a100\}, one needs to generate $2^{100} \sim 10^{30}$ candidates.

B. Multiple scans of database. At every stage entire data set needs to be scanned. Needs (n+1) scans, n is the length of the longest pattern.

Candidate set generation in this algorithm will again be a problem.

II. RELATED WORK

A. Basic Concepts & Basic Association Rules Algorithms:

Let $I = I_1, I_2, ..., I_m$ be a set of $m$ distinct attributes, $T$ be transaction that contains a set of items such that $T \subseteq I$, $D$ be a database with different transaction records $T$. An

Association rule is an implication in the form of $X \Rightarrow Y$, where $X, Y \subseteq I$ are sets of items called item set. $X$ 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). Since the database is large and users concern about only those frequently purchased items. Usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ 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.

Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ 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 $X \Rightarrow Y$ is 80%, it means that 80% of the transactions that contain $X$ also contain $Y$ together.

In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemsets. Itemsets of some length $k$ are referred to as $k$-itemsets.

Generally, an association rules mining algorithm contains the following steps:
The set of candidate k-itemsets is generated by 1-extensions of the large (k-1)-itemsets generated in the previous iteration. Supports for the candidate k-itemsets are generated by a pass over the database. Items that do not have the minimum support are discarded and the remaining itemsets are called large k-itemsets. This process is repeated until no larger itemsets are found. The AIS algorithm was the first algorithm proposed for mining association rule [4]. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item, for example we only generate rules like $X \cap Y \Rightarrow Z$ but not those rules as $X \Rightarrow Y \cap Z$. The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database. Apriori is more efficient during the candidate generation process [5]. Apriori uses pruning techniques to avoid measuring certain item sets, while guaranteeing completeness.

### III. Proposed System

The proposed system uses an Apriori algorithm based on matrix. The block diagram for the proposed system will be as shown in figure below:

Proposed system uses the Improved Apriori Algorithm based on Matrix. Association rule mining algorithm is a perfect solution on this problem as it will find the association between the policies for getting a benefit from it.

Algorithm analyzes the database which has information about life insurance policies. The database gives information about death benefit, bonus and other information about the policy. The transaction database consist of set of policies which are taken together frequently by the customers of life insurance company. The algorithm will find the policies which can be taken together to earn maximum benefit based on the basic information of a customer like age, amount that the person want to invest and so on.

As shown in TABLE 1, the transaction database has 8 transactions (8 persons who are insured) $D=\{T01,T02,T03,T04,T05,T06,T07,T08\}$ the item sets (policies which are taken together) are $I=\{a,b,c,d,e,f,g,h\}$ and the minimum support or threshold is 2.

### Table 1

<table>
<thead>
<tr>
<th>Transaction No.</th>
<th>Item sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>T01</td>
<td>d, e, g</td>
</tr>
<tr>
<td>T02</td>
<td>b, c, h</td>
</tr>
<tr>
<td>T03</td>
<td>a, c, d, f</td>
</tr>
<tr>
<td>T04</td>
<td>b, e, f, g</td>
</tr>
<tr>
<td>T05</td>
<td>d, e, g</td>
</tr>
<tr>
<td>T06</td>
<td>a, c, d, f</td>
</tr>
<tr>
<td>T07</td>
<td>b, e, f, g</td>
</tr>
<tr>
<td>T08</td>
<td>a, b, c, d, e</td>
</tr>
</tbody>
</table>

A. Find out the Mart

As shown in TABLE 2, create the matrix according the transaction database. If an item in a transaction, the position was set 1, Or else set 0

### Table 2

<table>
<thead>
<tr>
<th></th>
<th>T01</th>
<th>T02</th>
<th>T03</th>
<th>T04</th>
<th>T05</th>
<th>T06</th>
<th>T07</th>
<th>T08</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Obviously, there is only one “1” in the row “h”, it is less than the min support 2, so we should delete row “h” and don’t consider it again.
B. Find out the largest frequent itemsets

We can find out the largest frequent itemsets by simplifying the TABLE 2 (the row “h” had been deleted).

Operations as follows:
- As shown in TABLE 2, the number of the most items in an itemset is 5, but only one transaction “T08” has 5 items, so the number of transactions had 5 items is less than the min support “2”. But there are 5 transactions have 4 items or more: T03, T04, T06, T07 and T08.
- We should simplify the matrix according to 4 items. As shown in TABLE 3.
- As shown in TABLE 3, the transactions “T03”, “T04” have an itemset contained 4 items {a, c, d, f}, we do the “AND operation” to the rows “a”, “c”, “d”, “f”, the result is 2 which is greater than or equal to the min support “2”, so the itemsets {a, c, d, f} is one of the frequent itemsets. Because the transaction “T03” has only one itemset contained 4 items, so we can delete this column. We can know itemsets {b, e, f, g} also is a frequent itemsets when we use the same way to deal with the transaction “T04”. Now we delete the column “T04” and simplify the matrix again, we get an empty matrix, so the frequent itemsets are {a, c, d, f} and {b, e, f, g}.

C. Find out others frequent itemsets

Because we have know the largest frequent itemsets {a, c, d, f} and {b, e, f, g}, so we can get some other frequent itemsets according Nature 1:
- 2-frequent itemsets: {a, c}, {a, d}, {a, f}, {a, c}, {a, d}, {a, f}, {b, c}, {b, d}, {c, e}, {c, g}, {a, d}, {a, f}, {b, e}, {b, f}, {b, g}, {c, e}, {c, g}, {d, e}, {d, g}.
- 3-frequent itemsets: {a, c, d}, {a, c, f}, {a, d, f}, {c, d, f}, {b, e, f}, {b, e, g}, {b, f, g}, {e, f, g}.

So, we don’t count theirs Support again, which reduce the times of connecting and pruning.

1) Step 1:

Find out the 2-frequent item sets: According the TABLE (row “h” had been deleted) and the 2-frequent itemsets had been know, we can know the candidates of 2-frequent itemsets are: {a, b}, {a, e}, {a, g}, {b, c}, {b, d}, {c, e}, {c, g}, {d, e}, {d, g}.

It has only 19 itemsets, but the Apriori algorithm has 21 itemsets. Do the “AND operation” to the TABLE (row “h” had been deleted) and we can know the support of the itemsets {b, c}, {d, e}, {d, g} is less than the min support. So the 2-frequent itemsets are: {b, c}, {b, d}, {b, g}, {c, e}, {c, g}, {d, e}, {d, g}.

2) Step 2:

Find out the 3-frequent itemsets: As shown in TABLE 4, it is the result of deleting the transactions which the number of items less than 3 in TABLE 4 and simplifying the matrix. The more of the number of items we compute, the simpler of the matrix is. It will be easier counting the support. We should let the itemsets {b, c}, {d, e}, {d, g} in the 2-frequent itemsets as the center of connecting and pruning when we want to find out the candidates of 3-frequent itemsets. So, they are: {b, c, f}, {d, e, f}, {d, e, g}, {d, f, g}.

Perform “AND operation” to the TABLE 4 and we can know only the support of the itemsets {d, e, g} is less than the min support. So the 3-frequent itemsets are: {d, e, g}, {a, c, d}, {a, c, f}, {a, d, f}, {c, d, f}, {b, e, f}, {b, e, g}, {b, f, g}, {e, f, g}.

Now we have to found out all the frequent itemsets.
Recommending An Insurance Policy Using Association Rule Mining

So at the end of the program we will get set of policies which can be taken together to earn the maximum benefit

IV. RESULT AND DISCUSSION

The table below compares the time taken by the two algorithms in computing the frequent itemsets for different values of minimum support.

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>Computation Time(milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>156</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

V. CONCLUSION

Hence, it can be resolved that the Improved Apriori algorithm based on Matrix has better performance statistics as compared to the basic Apriori for Association Rules Mining algorithm as it requires less time for computation of the frequent itemsets.

REFERENCES


[2] Quantitative association rules mining algorithm based on matrix Huizhen Liu, Shangping Dai, Hong Jiang Department of Computer Science Huazhong Normal University Wuhan, China

