AMOEBA

T. Sudhir[1],
Associate Professor,

Naga Sriharsha Mulugu[2],
M.Tech Student,

Abstract

Association is one of the data mining techniques which generate frequent patterns. These frequent patterns may be 2-dimensional or multi-dimensional. Every association rule is built based on support and confidence factors. An instance is concluded with a frequent pattern if and only if satisfies the mentioned support and confidence values. Support and confidence values are user-defined. It may vary in according to the framework. Association rules generated in these cases depends on dimension of rule to be generated and various factors associated. These association rules never generate a chain of frequent patterns. The proposed algorithm is designed to generate chain of frequent patterns.

Keywords:

Amoeba, Association, Data Mining, Frequent Patterns.

Introduction

Data mining is interdisciplinary of computer science that deals with discovering of patterns in large data sets. The knowledge extraction is done with the help of some common tasks like association based learning.

Association rule based mining is used to find interesting relationships between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. Association rules are found using certain algorithms like apriori, which are used to find a set of frequent item sets. In these algorithms, for any given large database, initially a data set with some transactions is taken. Based on those transactional data sets, support and confidence factors are defined. Using these factors and the transactional data set, a set of frequent pattern item set is found. The process is repeated until no new item sets are discovered.

There are certain problems with the above technique. In the previous scenario, with each new frequent item set to be generated a new set of transactional data sets are required to be found. This is a time consuming process. Secondly, the use of transactional data sets is irrelevant. The construction and maintenance is costly.

Hence we design an approach that doesn't require the creation of transactional data sets.

Algorithm – AMOEBA

AMOEBA[1] is an algorithm, which finds out a chain of probable frequent items for a given attribute values. This algorithm falls under association and data mining. This algorithm includes probability distribution of data values, determining attributes, and determined attribute values by other attributes (using functional dependencies) are evaluated.

This algorithm is only suitable for certain applications which involve large data sets. Let us assume a scenario, that a customer is required to buy an item X from a departmental store. The previous algorithms suggests the customer the related set of items to be bought with the item X. Our proposed algorithm gives the set of all related and frequent items that can be bought with item X.

For any given random attribute X in a data set, we can find related items along a row or a column. We can traverse in a data set in row to the right or to the left, similarly for a column up or down. A frequent chain of item sets can be found in any a direction (the basic principle of AMOEBA). The algorithm may search in any direction depending on the user interestingness. This interestingness is proposed by the probability distribution among the attributes. Hence the name AMOEBA.

The proposed algorithm used two basic procedures feed and defend. They are named after the nature of the species AMOEBA. The procedure feed, searches for the frequent patterns either to the left or to the right in the dataset.

Procedure Amoeba

Inputs: A dataset, an attribute
Output: A frequent set of related attributes with highest probability.
Variables used: flag, inputAttrib, result, dataPoint
procedure Amoeba()
  inputAttrib:=read value
  dataPoint := obtain a random row from dataset

```
    index:= find the index of the attribute
    Randomly call any one of procedure
    feedleft(dataPoint, inputAttrib,index)
    feedRight(dataPoint, inputAttrib,index)
end procedure Amoeba
```

Procedure feedLeft

```
Inputs: An Attribute, A random row from the dataset
Output: set of frequent attributes to the left in the dataset.
procedure feedLeft()
   if ! flag
     find all the related attributes with respect to
inputAttrib.
     tempAttrib:= save all related attribute
     j:= column index of attribute
     for each i:=0 to j do
        find max probability among temp.
        result:= tempAttrib
        break
     end for
     feedLeft()
  else
      return
  end if
end procedure feedLeft
```

Procedure feedRight

```
Inputs: An Attribute, A random row from the dataset
Output: set of frequent attributes to the right in the
dataset.
procedure feedRight()
   if ! flag
     find all the related attributes with respect to
inputAttrib.
     tempAttrib:= save all related attribute
     j:= column index of attribute
     for each i:=j to maxcolumns in dataset do
        find max probability among temp.
        result:= tempAttrib
        break
     end for
     feedRight()
  else
      return
  end if

end procedure feedRight
```

The above procedures are used to find the chain of frequent attributes. The procedure roughly stated is used. Initially, for the attribute X, the set of frequent item sets are found along with their probabilities. Among those the item with highest probability ratio is considered let us say Y. Again the frequent items for Y are calculated along with that of X. The highest among items obtained from X, Y are found. The process is repeated until no further items are found along that particular direction.

The procedure Amoeba takes a dataset and an attribute for which the chain of items are to be found. From the dataset the algorithm first finds the attribute X in the dataset. After finding the attribute column in the dataset, then the algorithm either calls the procedure feedLeft or feedRight. It is random calling the procedures but both of them are executed. The procedure feedLeft finds the attributes to the left of the attribute in the dataset and feedRight to the right. Both the procedures are almost identical except the way they search for attributes in the dataset.

Instead of generating any transactional tables, the procedure simply finds the frequent item and displays them. We require no further memory for saving the contents. The procedure feedRight or feedLeft depicted above initially finds all the list of items regarding the input attribute X. Amongst them, it finds the attribute that has highest support factor. This is calculated by the defendUp or defendDown procedures. These are used to find the maximum probability among the given set of attributes. After finding the attribute Y that has the maximum support and confidence among the others of X, the procedure adds Y to X X->Y, then it in turn calls the same procedure. The process is repeated until no further attributes related are found.

The same process is repeated in different directions calling their respective procedures feedLeft, feedRight, defendUp, defendDown. The algorithm can be stopped at any period of time by providing certain constraints like 2Dimesional, 3Dimensional, etc,

Using this algorithm we can find a chain of items say as many as possible. Unlike other algorithms in which the result is limited, the proposed algorithm produces all the possibilities for the given attribute X.

Advantages

This proposed algorithm doesn't require the dataset to be in proper order. This algorithm is not time consuming and doesn't require building transactional data sets. The algorithm can be used for finding multi-dimensional item sets.

The limitation for this algorithm is that it is to be used for only certain applications under certain requirements.

Efficiency

As the above proposed algorithm uses four procedures, the complexity of the above algorithm is O(mn) in the worst case scenario, where m is the length of the dataset and n being the number of rows. As the dataset has to be searched for each and every possible attribute for any given X.

- Though Amoeba takes has longer time in execution, it is flexible. It generates a variety of attributes, which are much more than apriori algorithm.

- Apiori Algorithm has less time complexity compared to the proposed ones. But with the time required to build the transactional datasets, our algorithm would have done the job. That makes amoeba preferable.

Conclusion

The proposed algorithm has been designed based on a single dataset used for departmental stores. In future this can be extended for many applications of data mining. Provided with the variety of attributes, in future the algorithm can be extended to work for any number of input attributes.

References

[1] Toshinori Munakata, Masashi Aono, Masahiko Hara, "Amoeba based Knowledge Discovery",IJCS, 2010, Vol. 4, No. 5, 453-462.

[2] Pranay Bhandari, Rajeswari, Swati Tonge, Mahadev Shindalkar, "Improved Apriori Algorithm", IJACEN, 2013, Volume - 1 Issue – 2.

[3] L. Lu and P. Liu, "Application In Supermarket,"pp. 441–443.

Authors Biography

1. T.sudhir pursuing phd at acharya nagarjuna university ,working as associate professor at Vasireddy Venkatadri Institute of Technology, Nambur, guided several projects, having 10 years of experience ,areas of interest is medical mining and image processing.

2. M. Naga Sriharsha currently pursing M.Tech at Vasireddy Venkatadri Institute of Technology. Having an experience of 3 and a half years as Assistant Professor in the department of CSE.