Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 31 – 36

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

A New Algorithm in Association Mining, Amoeba for Finding Frequent Patterns Using Functional Dependency and Probability

Sudhir Tirumalasetty[a,*], Sreenivasa Reddy Edara[b]

[a]*Dept. of Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology, Guntur 522508, India*
[b]*Dept. of Computer Science & Engineering, Acharya Nagarjuna University, Guntur 522510, India*

Abstract

In Data Mining, research on association mining has expanded widely in many application areas. APRIORI and FP-Growth are prominent algorithms in association mining for finding frequent patterns. These algorithms have their own shortcomings like space complexity and time complexity. Moreover constructed transaction data set in obligatory for these algorithms as input. Improved versions of these approaches have reduced the prior mentioned shortcomings. A new algorithm, Amoeba, was proposed to find chain of possible frequent patterns. This algorithm excludes construction of transaction dataset and calculating thresholds and includes probable occurrences of attribute values using functional dependencies.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: Data Mining; Association Rules; Frequent patterns; APRIORI; FP Growth; Amoeba; Functional Dependency

1. Introduction

Data mining is interdisciplinary area 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. In the past few years the chore of discovering frequent pattern in large databases is very significant and has been reviewed in large extent. If there are a large number of patterns then finding frequent patterns is computationally expensive.

Association rules are formed using “if then” statements. These statements (or rules) help in discovering frequent

* Corresponding author. Tel.: +0-984-970-0724;
E-mail address: sudhir2918@gmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.01.045

patterns between two sets of attributes. One set of attributes is called antecedent (if) and the other is called consequent (then). Antecedent determines another attribute whereas consequent gets determined by antecedent. Any item found in the data set can be treated as antecedent. A consequent is an item that is found in a combination with the antecedent.

Association rules are formed by exploring data. These rules are formed by using "if then" statements along with two measures: support and confidences. *Support* is defined as the frequency of items appearing in the database. *Confidence* is defined as the number of times the "if then" statements have been found to be true. The minimum support value may not always result in interesting relationships between the item sets for generating association rules. Sometimes certain algorithms are involved in the construction of transaction data set and in other algorithms involves in construction of frequent-pattern trees.

The two prominent algorithms used in association mining for finding frequent patterns are APRIORI and FP-Growth.

The APRIORI[4] algorithm involves prior construction of transaction data set before finding frequent patterns. An assumed support and calculated confidence factors are calculated for finding frequent patterns. At an instance, support and frequent pattern set size may vary in every iteration of this algorithm. In every iteration of APRIORI algorithm some transactions within transaction data set gets excluded. Excluded transactions are not items that can be treated as useless items of a data set because these same items may become frequent items in later transactions and also because transactions are not static. Suppose if this algorithm is re-executed with same transactional data set and altered support value, results in change of outcomes of frequent items for every iteration. This concludes that the outcome of frequent patterns of a transaction data set changes with change in support value. As data set is not static with time, association rule mining also changes with time.

The FP-Growth Algorithm[3] mines the complete set of frequent patterns by pattern fragment growth which is an efficient and scalable method. It uses an extended prefix-tree structure called frequent-pattern tree (FP-tree) for storing compressed and crucial information about frequent patterns. The performance of this algorithm is enhanced when compared with APRIORI as this algorithm discovers frequent item sets without using candidate generations. This algorithm uses a divide-and-conquer strategy. The central part of this algorithm is the usage of a special data structure named frequent-pattern tree (FP-tree), which holds the item set association information.

Construction of transactional data set, assumptions of support, calculating confidence factors and construction of frequent pattern tree are prerequisites for some algorithms in association mining. Without these calculations certain association algorithms can't move forward. Moreover data set is not static. A new algorithm named AMOEBA[1] is proposed based on the characteristics of unicellular organism amoeba. This algorithm is intended to overcome the pre calculations of some association mining algorithms.

2. Evolution of Algorithm Amoeba

Amoeba is a unicellular organism which does not have a definite shape and belongs to phylum protozoa[5]. August Johann Rosel von Rosenhof in 1757 discovered the first microscopic amoeba[6]. The name *"amibe"* was given to it by Bory de Saint-Vincent,[2] from Greek *amoibe*, meaning change. Amoeba moves by using pseudopodia or *"false feet"*. Though several hypotheses have been proposed to explain the mechanism of amoeboid movement, the exact movement of amoeba is still unknown. These characteristic features of amoeba lead to the evolution of a new association mining algorithm AMOEBA.

Amoeba moves in a direction which is not specific. This is due to the presence of false feet in amoeba. This feature enabled for the evolution of this new algorithm. This can be termed as attribute value determining. This determination can be achieved by using functional dependency i.e. determining an attribute value by another attribute value. The determination also includes, at what percentage an attribute value determines other attribute distinct values.

3. Working of Amoeba

This algorithm works mainly on two principles:

- Determining another attribute value in a data set using an attribute value.

  Or

  Determining another attribute value in a data set which determined the attribute value.
- Probability of an attribute value being determined by an attribute value.

Algorithm:

Input: A discrete dataset, an attribute
Output: A frequent set of related attributes with highestprobability.
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

Algorithm feedRight:
Input: An Attribute, A random row from the dataset.
Output: set of frequent attributes to the right in thedataset.
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

Algorithm feedLeft
Input: 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

Example:
Consider the following, an instant data set with five attributes as shown in Fig. 1. to explain how AMOEBA algorithm works:

Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
N	J	A	D	G
M	J	A	D	G
O	K	B	D	H
N	L	A	E	H
O	L	B	F	I
N	K	C	F	I
M	J	A	F	G
M	J	B	E	G

Fig. 1. An instance data set

- At random select an attribute say attribute 3 and also select at random a discrete value of attribute 3 say **A**as shown in Fig. 2.

Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
N	J	A	D	G
M	J	A	D	G
O	K	B	D	H
N	L	A	E	H
O	L	B	F	I
N	K	C	F	I
M	J	A	F	G
M	J	B	E	G

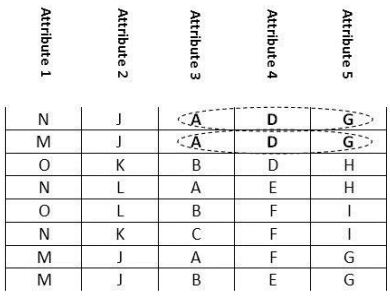
Fig. 2. Random attribute value selection

- Attribute 3's distinct value **A** determines values **D, E** and **F** of Attribute 4 but **A** determines **D** with maximum probability when compared to **A** determining **E** and **F** as shown in Fig. 3.

Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
N	J	A	D	G
M	J	A	D	G
O	K	B	D	H
N	L	A	E	H
O	L	B	F	I
N	K	C	F	I
M	J	A	F	G
M	J	B	E	G

Fig. 3. Determining other attribute value

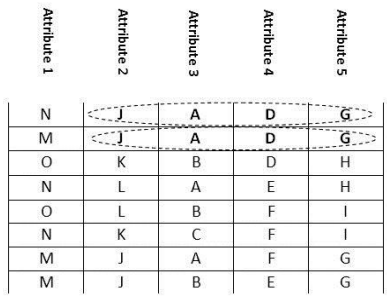
- Attribute 3's distinct value **A** and Attribute 4's distinct value **D** together determines value **G** of Attribute 5 with maximum probability as shown in Fig. 4.



Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
N	J	A	D	G
M	J	A	D	G
O	K	B	D	H
N	L	A	E	H
O	L	B	F	I
N	K	C	F	I
M	J	A	F	G
M	J	B	E	G

Fig. 4. Determining remaining attribute value

- Attribute 3's distinct value **A,** Attribute 4's distinct value **D** and Attribute 5's distinct value **G** together gets determined by the value **J** of Attribute 2 with maximum probability as shown in Fig. 5.



Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
N	J	A	D	G
M	J	A	D	G
O	K	B	D	H
N	L	A	E	H
O	L	B	F	I
N	K	C	F	I
M	J	A	F	G
M	J	B	E	G

Fig. 5. Determining remaining attribute value

- Attribute 3's distinct value **A,** Attribute 4's distinct value **D,** Attribute 5's distinct value **G** and Attribute 2's distinct value **J** together gets determined by a distinct of value of Attribute 1 is of equal probability.
- Hence the chain of frequent items is obtained. This chain contains attribute values in the order; determining values followed by get determined by values. In the above example frequent items chain is **{A, D, G, J}.**
- Every time an attribute value may not determine or get determined by its adjacent attribute values.

4. Results

The algorithm AMOEBA is executed on standard data set supermarket.arff which is taken from www.cs.waikato.ac.nz/~ml/weka/datasets.html. This data set is considered with three instances of tuples. The cardinality of these three instances is 3627, 1689 and 941 tuples respectively. These instances are considered for comparing the execution speed of algorithms Apriori, FP-Growth and Amoeba in seconds. The execution speed of new algorithm Amoeba is faster than Apriori and slow when compared to FP Growth algorithms. This comparison is shown in Table. 1.

Table 1. Execution times of algorithms Apriori, FP-Growth and Amoeba in seconds.

No. of Instances	Apriori	FP-Growth	Amoeba
3627	47	3	22.3
1689	25	2	13.7
941	8	1	5.2

The advantage of algorithm Amoeba is free from construction of FP Growth tree and doesn't include the values of factors like support and confidence. The algorithm Amoeba doesn't require a transaction data set. Moreover the algorithm is not limited in dimension of frequent pattern set.

5. Conclusion

The algorithm AMOEBA results in a chain of frequent items set with a minimum of size 2. The chain of frequent items is not identical for all times, it varies with time as data set is not constant. The probability of frequent items in the frequent item chain decreases if the frequent item set size increases. In the above mentioned example frequent items chain is { A , D , G, J }. The frequent items sets for this chain are { A , D }, { A , D , G } and { A , D , G, J }. The probability of frequent items of these frequent items sets decreases with increase in size of frequent item set i.e. frequent item set of size 2 has highest probability and frequent item set greater size has less probability.

Algorithm AMOEBA is free from construction of transaction data set, calculation factors like support and confidence and construction of frequent pattern trees. The limitation of this algorithm AMOEBA is, input data set must be discretized because determination through probability can be defined on discrete values. Selection of initial attribute value, influence the evolution of frequent items chain for the algorithm AMOEBA. This is due to, if the determination values of other attributes by initial attribute value are of lowest probability or zero then such initial attribute value becomes void for finding frequent items chain. Selection of such initial attribute value whose probability of determining other attribute values is zero, results in identifying infrequent items in a data set. This attribute value cannot be included in frequent item set. The algorithms, Apriori and FP Growth costs more, when compared with the algorithm Amoeba in terms of disk usage. The Table. 2. summaries the algorithms Apriori, FP Growth and Amoeba.

Table 2.Summary of algorithms Apriori, FP-Growth and Amoeba.

Parameters	Apriori	FP-Growth	Amoeba
Execution time	Slow	Fast	Moderate
Dimension of association rule	Required	Not required	Not required
Construction of transactional data set	Required	Not required	Not required
Support and Confidence values	Required	Not required	Not required
Construction of FP Tree	Not required	Required	Not required
Memory cost	High	Very high	Low
Input data set	Discrete	Discrete	Discrete
Size of frequent item pattern	Depends on iteration	Not limited	Not limited

References

1. T. Sudhir and Naga Sriharsha Mulugu, *AMOEBA, International Journal of Engineering Research and Technology*, 11th November 2013, Vol. 2 Issue
2. McGrath, Kimberley; Blachford, Stacey, *Gale Encyclopedia of Science Vol. 1: Aardvark-Catalyst* (2nd ed.), Gale Group, eds. 2001, ISBN 0-7876-4370-X. OCLC 46337140.
3. J. Han, H. Pei, and Y. Yin. Mining, *Frequent Patterns without Candidate Generation*. In: Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX). ACM Press, New York, NY, USA 2000.
4. Rakesh Agrawal and Ramakrishnan Srikant, *Fast algorithms for mining association rules in large databases*, Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, September 1994; . p. 487-499
5. Amoeba at *Dorland's Medical Dictionary*
6. Leidy, Joseph, Amoeba proteus, *The American Naturalist*12 (4): 235–8. doi:10.1086/272082. JSTOR 2463786.