Evaluating Categorical Association Rules from Large Database Using Genetic Algorithm

T. Priyadharsini, D. Vandhana, G. Akila

Abstract- Association rules are one of the most frequently used tools for finding relationships between different attributes in a database. Various techniques for obtaining the categorical association rules exist. However, when there is a need to relate attributes which are numeric and discrete, methods which generate quantitative association rules are supposedly used. When the database is extremely large, many of these tools cannot be used. In this paper, an efficient tool for retrieving the association rules from *large databases* will be implemented using matrix apriori algorithm and improvements in the speed at which the algorithm performs.

Index Terms- Association rules, Confidence, Data mining, Fitness Evaluation, Genetic Algorithm, Matrix Apriori Algorithm, Support

1 INTRODUCTION

Data mining (DM) is the extraction of implicit, valid, and potentially useful knowledge from large volumes of raw data (Han & Kamber, 2006). In recent years, data size has grown considerably, which has caused increased difficulty in extracting useful information. The use of DM to facilitate decision support can lead to improved decision-making performance and can enable us to tackle new types of problems that have not been addressed before (Alcala-Fdez, Flugy-Pape, Bonarini, & Herrera, 2010; Kamrunnahar & Urquidi-Macdonald, 2010; Mladenic´, Eddy, & Ziolko, 2001; Tsumoto, Matsuoka, & Yokoyama, 2008).

Association rules were introduced as a method to and relationships among the attributes of a database. By means of these techniques a very interesting qualitative information with which we can take later decisions can be obtained. In general terms, an association rule is a relationship between attributes in the way C1) C2, where C1 and C2 are pair conjunctions (attribute-value) in the way A = v if it is a discrete attribute or $A \in [v1; v2]$ if the attribute is continuous or numeric. Generally, the antecedent is formed by a conjunction of pairs, while the consequent usually is a unique attribute-value pair.

The most used measures to define the interest of the rules were described in [11]:

Support: It is a statistical measure that indicates the ratio of the population that satisfies both the antecedent and the consequent of the rule. A rule R: C1=>C2 has a support s, if an s% of the records of the database contain C1 and C2.

Confidence: This measure indicates the relative frequency of the rule, that is, the frequency with which the consequent is fulfilled when it is also fulfilled the antecedent. A rule R: C1 => C2 has a confidence c, if the c% of the records of the database that contain C1 also contains C2.

The goal of the techniques that search for association rules is to extract only those that exceed some minimum values of support and confidence that are defined by the user. The greater part of the algorithms that extract association rules work in two phases: in the first one they try to find the sets of attributes that exceed the minimum value of support and, in the second phase, departing from the sets discovered formerly, they extract the association rules that exceed the minimum value of confidence.

2 MATRIX APRIORI ALGORITHM

Resembling Apriori, algorithm Matrix Apriori consists of two steps. First, discover frequent patterns and second, generate association rules from the discovered patterns. As mentioned before, the first step determines the performance of the algorithm. For this reason, Matrix Apriori presents a different logic for the first step, but maintains the logic for the second step identical to that of Apriori, and so for the rest of this article, we will use the terms "Matrix Apriori" and "first step of Matrix Apriori" interchangeably. This section describes in detail the first step of algorithm Matrix Apriori. This algorithm was developed on the basis of a critical analysis carried out on both Apriori and FP growth. The structures used are presented first, and then follows the pseudo code of the algorithmic procedure.

- *T.Priyadharsini is currently pursuing bachelor's degree program in computer science and engineering in SASTRA University, India, PH-09566379523. E-mail: priyadharsini73@gmail.com*
- *D.Vandhana is currently pursuing bachelor's degree program in computer science and engineering in SASTRA University, India,PH-08220107347, E-mail: vandhana1192@gmail.com*
- *G.Akila is currently pursuing bachelor's degree program in computer science and engineering in SASTRA University, India,PH-09566494789,E-mail: akilaganesan.trichy@gmail.com*

2.1 Structures for Matrix Apriori

Let L = {i1, i2, …, im} be a set of items, D a repository containing a set of transactions, ξ a minimal support predefined by the user, T a transaction, where each transaction T ⊆ L. Algorithm Matrix Apriori employs two simple structures for generating frequent patterns: a matrix called MFI (matrix of frequent items) which holds the set of frequent items identified during the first traversal of repository D, and a vector called STE which stores the support of candidate sets. Frequent items will be represented by columns of MFI; however, they could also be represented by rows. In this article we take the first approach.

DEFINITION 1: The matrix of frequent items MFI is a structure such that

- Columns represent frequent items found during the first traversal of D, and rows (except the first row) represent the candidate sets.
- Element MFI [i, j] = 1 if item j belongs to the $(i - 1)^{th}$ candidate set; otherwise MFI [i, j] = 0.

DEFINITION 2: The conditional pattern of a k-set of items groups the set of transactions where the k-set of items is present.

```
Call MFI_STE

    {The 1-sets of items identified through this
procedure make part of the set of frequent patterns}

call Positions_MFI
    for j = NFI downto 2
    Call Frequent_Patterns (str (j))

    {str is a function that converts the numeric variable
in variable j into a string}

    Endfor
```

Figure 1: Procedure for Matrix Apriori

Next, we describe the three procedures that constitute the first step of Matrix Apriori.

2.2 Procedure for Matrix Apriori

◆ **Procedure MFI_STE**

This procedure builds matrix MFI and vector STE. The steps of this procedure are presented in figure 2.

Input: The set of transactions stored in D and an expected minimal support ξ.

Output: The rows corresponding to candidate sets in MFI, the support corresponding to each candidate set in vector STE, the total number of frequent items NFI and the total number of candidate sets.

1) Traverse repository D with the purpose of identifying the frequent items set. Firstly, compute the support for each item that appears in D. Then, identify all those items that have a support greater than or equal to the minimal support î; these conform the frequent items set FI. Sort the elements in FI by ascending support, and determine the value of NFI.

2) Initialize variable NCS to 0.

3) In the second traversal, for each transaction T in D do:

- Identify frequent items that are present in T and sort them by support value, placing the frequent item of greatest support in the first position. Each resulting set is a candidate set.
- For the transaction T currently being processed, represent its candidate set in matrix MFI as specified in Definition 1, given in section 2.1 Also, accumulate the value of support of the candidate set in vector STE.

Figure 2: Procedure MFI_STE

◆ **Procedure Positions_MFI**

This procedure turns matrix MFI into an index to speed up the traversal that identifies frequent patterns in the matrix. The pseudo code for the procedure is given below.

Input: Matrix MFI, where candidate sets are represented; the total number of frequent items NFI; and the total number of candidate sets NCS.

Output: Matrix MFI modified, now containing the rows where frequent items are present (an index).

```
procedure Positions_MFI()
for j ← 1 to NFI
        {traverse all frequent items}
prev ← 1
for i ← 2 to NCS +1
        {traverse all candidate sets}
if ( MFI[i, j] <> 0 ) then
        MFI[prev, j] ← i
prev ← i
endif
endfor
endfor
```

Figure 3: Procedure Postitons_MFI

◆ **Procedure Frequent_Patterns**

This procedure's objective is the generation of frequent patterns. This procedure employs the two structures MFI and STE, and applies the method of growing patterns to identify the item sets that are frequent patterns. The main operation of Frequent_Patterns consists of combining the conditional pattern cp with the frequent items found on its left hand side in MFI, and to calculate its corresponding support; then it checks whether the analyzed combination is a frequent pattern. To reach such conclusion, the following condition should be met: the calculated support for the analyzed combination must be greater than or equal to the minimal support î.

The main operation of Frequent_Patterns consists of combining the conditional pattern cp with the frequent items found on its left hand side in MFI, and to calculate its corresponding support; then it checks whether the analyzed combination is a frequent pattern. To reach such conclusion, the following condition should be met: the calculated support for the analyzed combination must be greater than or equal to the minimal support ξ.

Input: Matrix MFI already updated with the indexes; vector STE with the support for each candidate set stored; and the minimal support î.

Output: Frequent patterns associated to the conditional pattern cp.

3 GENETIC ALGORITHM

In the computer science field of artificial intelligence, a Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of Evolutionary Algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

3.1. Rank selection

Rank selection sorts the population first according to the fitness value and then ranks them. Then every chromosome is allocated selection probability with respect to its rank. Individuals are selected as per their selection probability. Rank selection is an explorative technique of selection. Rank selection prevents too quick convergence and differs from other selection methods in terms of selection pressure. Rank selection overcomes the scaling problems like stagnation or premature convergence. Ranking controls selective pressure by uniform method of scaling across the population. Rank selection behaves in a more robust manner than other methods.

3.2 Crossover

In genetic algorithm, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. Cross over is a process of taking more than one parent solutions and producing a child solution from them. Many crossover techniques exist for organisms which use different data structures to store themselves. Here, the crossover technique used is single point crossover. In single point crossover, both parents' organism string is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children.

3.3 Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

3.4 Fitness Evaluation

Although there are many measures of the fitness of a rule, the most representative, and the ones which provide most information on the quality of the rules, are support and confidence. In addition, other factors capable of establishing priority between individuals have been considered, as explained below. Hence, the evaluation function which should be maximized for each individual 'i' is given by the following equation:

$$F(i) = (\text{support} * w_s) + (\text{confidence} * w_c) + (n_{\text{attributes}} * w_{na})$$

As can be seen in (1), each parameter has an associated weighting factor. It is extremely difficult to establish fixed values that are valid for solving all the problems. For example, establishing high values for confidence in databases with a high level of noise (high degree of dispersion) (i.e., reward confidence in the rule over the other measures) can mean that the rules discovered have a very low value in terms of support, which is not always the most appropriate case.

The meaning of each of the parameters of the function f(i) is as follows:

Support: this measure can be affected by the modifier w_s. A high w_s value means that the rules obtained have a high support value, to the detriment of the other parameters. A low ws value finds rules that have only to meet a limited number of cases.

Confidence: this measure can be affected by the modifier w_c. High we values are used when we are interested in finding error-free rules, and low values when there is significant noise
in the dataset, and hence the integrity of the rule is not so important.

N Attributes: This measure indicates the number of attributes making up the rule. If rules with a high number of attributes are to be sought, then a high w_{na} value should be chosen. In many cases, the user is only interested in finding associations between a limited number of attributes and a low w_{na} value is then appropriate.

The steps involved in Genetic Algorithms are represented in the fig. 4.

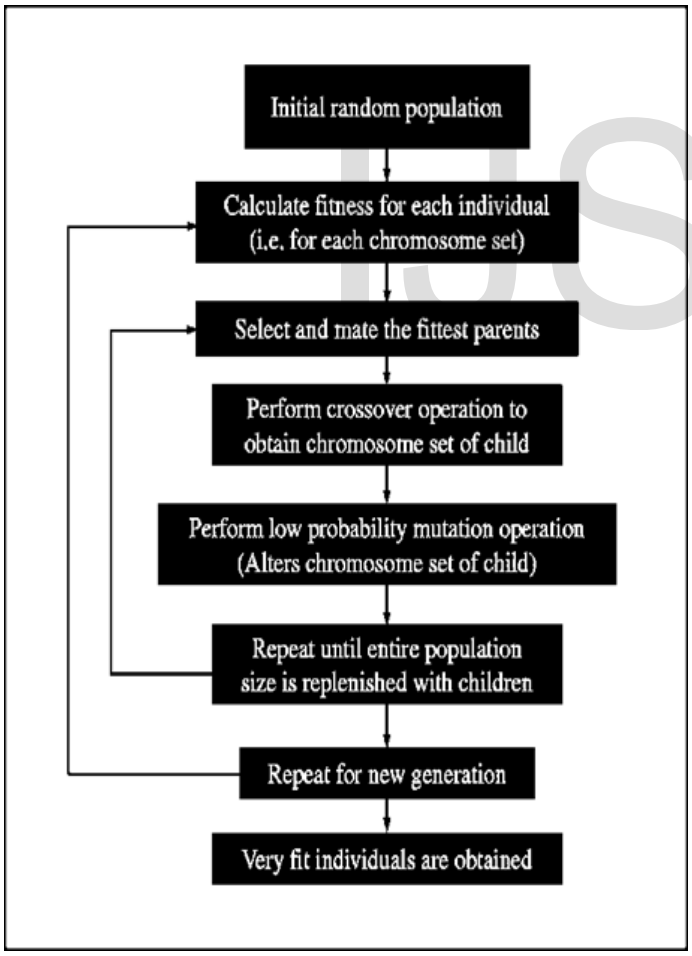

Figure 4: Steps in Genetic Algorithm

4 EXISTING SYSTEM

An evolutionary tool for finding optimized association rules in databases (both small and large) comprising quantitative and categorical attributes without the need for an a priori discretization of the domain of the numeric attributes. This genetic algorithm finds the best rules, on the basis of various parameters including support, confidence, interval amplitude, number of attributes and so on. Finally, the tool is evaluated using both real and synthetic databases.

Here, FP-Growth technique is used. When compared with Apriori algorithm, FP-Growth is efficient. Even then, it has some drawbacks. When the database is large, it is sometimes unrealistic to construct a main memory based FP-Tree. The tree is expensive to build. Time is wasted as the only pruning that can be done is on items. Support can only be calculated once the entire dataset is added to the FP-Tree.

Here, the selection methodology used is Roulette-wheel. In this selection method, it has higher probability to select the individuals with higher fitness value. There is a possibility that it may miss the best individuals. There is no guarantee that good individual will find their way into next generation.

5 PROPOSED SYSTEM

The association rules are generated form databases comprising of numerical attributes by using the matrix apriori algorithm. The resulting random association rules are taken as the initial population for the genetic algorithm. The optimized result set is achieved by the Genetic algorithm steps fitness, selection, crossover and mutation. When compared to the FP-Growth, Matrix Apriori algorithm is efficient as it speeds up the search pattern. The selection methodology adopted is the rank selection. Rank selection sorts the population first according to fitness value and ranks them. It prevents too early convergence and differs from roulette wheel convergence in terms of selection pressure. Rank Selection behaves in a more robust manner than other methods of selection strategies.

VANDHANA-PC.matap - dbo.mutation | VANDHANA-PC.matap - dbo.crossover | VANDHANA-PC

id	ante	con	...	sup	conf
107	86,76,85,34,43	43	...	13	80
108	85,59,34,93,52	43	...	13	80
109	67,86,59,34,55	43	...	13	80
111	76,90,85,76,65	43	...	20	86
112	90,2,36,23,68	43	...	20	86
129	52,85,76,86,73	111	...	14	82
134	59,34,93,52,79	111	...	12	78
135	76,86,59,34,23	111	...	12	78
136	90,34,93,52,55	111	...	12	78
139	59,85,76,86,16	111	...	12	78
140	76,85,59,34,94	111	...	12	78
144	76,86,52,34,65	111	...	12	78
145	90,59,34,93,78	111	...	12	78
147	86,85,76,93,46	111	...	12	78
149	67,86,59,93,32	111	...	12	78
151	76,85,76,86,27	111	...	26	90
152	90,59,85,76,86 ...	111	...	26	90

Figure 5: Optimized Association Rules

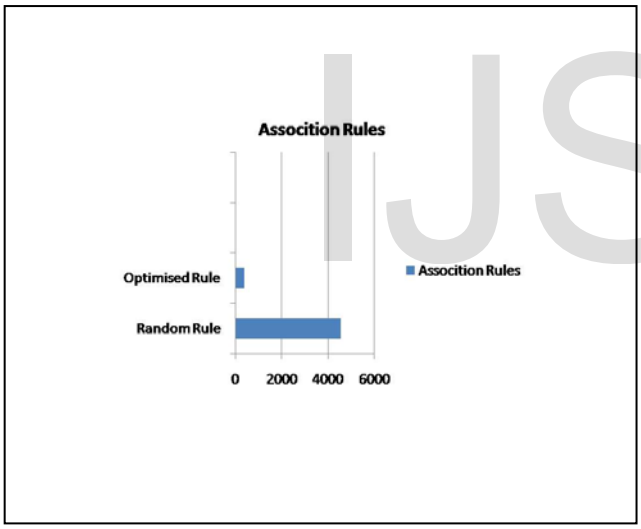

Figure 6: Comparison of Matrix Apriori Algorithm and Genetic Algorithm

6 CONCLUSION

We have established the association rules with the help of matrix apriori algorithm. The matrix apriori algorithm increases the efficiency by speeding up the search. The association rules are estimated by calculating the support count and confidence. The genetic algorithm methods fitness evaluation, selection, crossover and mutation are implemented which optimizes the random association rules generated. The fitness evaluation is carried out with the attributes support, confidence, and number of attributes. The rank selection technique improves the efficiency of fitness by selecting the best individuals. At last, the optimized association rules are established after the execution of genetic algorithm.

6 REFERENCES

[1] Victoria Pachón Álvarez & Jacinto Mata Vázquez,(2012). An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an a priori discretization

[2] Anubha Sharma, Nirupma Tivari(2012). A Survey Of Association Rule Mining Using Genetic Algorithm Rakesh Kumar, Jyotishree(2012).

[3] Mar´ýa J. del Jesus,1 Jose´ A. Ga´mez,2 Pedro Gonza´ lez1 and Jose´ M.Puerta2 (2011). On the discovery of association rules by means of evolutionary algorithms

[4] Han, J., & Kamber, M. (2006). Data mining: Concepts and techniques (2nd ed.)

[5] Blending Roulette Wheel Selection and Rank Selection in Genetic Algorithm

[6] ata, J., Alvarez, J. L., & Riquelme, J. C. (2002). An evolutionary algorithm to discover numeric association rules (pp. 590–594).

[7] Judith Pavón, Sidney Viana, Santiago Gómez(1996). Matrix Apriori: Speeding up the Search for Frequent Patterns

[8] Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. SIGMOD Record (ACM Special Interest Group on Management of Data), 25, 1–12.

[9] Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining (pp. 80 –86).

[10] Peter P. Wakabi-Waiswa and Dr. Venansius Baryamureeba, "Extraction of Interesting Association Rules Using Genetic Algorithms", Advances in Systems Modeling and ICT Applications, pp. 101-11

[11] R.Agrawal, T.Imielinski, and A.Swami. Mining association rules between sets of items in large databases. In Proc. ACM SIGMOD, pages 207{216, Washington, D.C., 1993