Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 3, Issue 5; July-September, 2016, pp. 444-447

http://www.krishisanskriti.org/Publication.html

A Genetic Algorithm to Optimize Association Rules

Munmun Kalita[1] and Chitvan Gupta[2]

[1]M.Tech Student, Department of CSE, Noida Institute of Engineering and Technology, Greater Noida, UP, Uttar Pradesh
[2]Department of CSE, Noida Institute of Engineering And Technology, Greater Noida, UP, Uttar Pradesh
E-mail: [1]kalita.munmun75@gmail.com, [2]chitvangupta@Gmail.Com

Abstract—*Data mining is synonymous with knowledge mining which means extraction of useful information from an existing dataset and transforms it into a flexible structure. Association rule mining is one of the most important tasks of data mining. It is the process of finding some relations among the attribute values of a large database. Genetic algorithms have found their strong base in mining Association Rules. Many researchers have proposed genetic algorithms for mining interesting rules from dataset. This paper provides an algorithm to optimized association rule using genetic algorithm.*

1. INTRODUCTION

The process of discovering interesting and unexpected rules from large data sets is known as association rule mining. An association rule is an *implication* or *if-then-rule* which is supported by data. Mining of association rules is a field of data mining that has received a lot of attention in recent years [6]. Most of the association rule algorithms are based on methods proposed by Agrawal, Imielinski, and Swami [1] and Agrawal and Srikant [2], Apriori [1], SETM [1], AIS [1] etc.[7]. However, these algorithms have their limitations. Genetic algorithm is used in mining association rule to remove some of the limitations of the existing approaches [3]. GA is relatively simple, easy to implement and easy to use. Furthermore, it follows a database-independent approach which does not rely upon the minimum support and the minimum confidence thresholds which are hard to determine for each database. [4]

The rest of this paper is organized as follows. In Section 2 an overview of the existing association rule mining techniques is provided. Section 3 provides an overview genetic algorithms. Section 4 covers the details of the proposed method. Implements and results are put in Section 5. Finally, Section 6 includes the concluding remarks.

2. ASSOCIATION RULE MINING (ARM).

Principle of association rule mining (ARM) lies in the market basket or transaction data analysis. The major aim of ARM is to find the set of all subsets of items or attributes that frequently occur in many database records or transactions, and additionally, to extract rules on how a subset of items influences the presence of another subset. ARM algorithms discover high-level prediction rules in the form: IF the condition of the values of the predicting attributes are true, THEN predict values for some goal attributes. The task of mining association rules over market basket data was first introduced by Agrawal et al. [1].

Let I= $\{i_1, i_2, i_3 \ldots, i_m\}$ be the set of database items and T= $\{t_1, t_2, \ldots, t_m\}$ be the set of transactions in the database, D, with each transaction t_i having a unique identifier and containing a set of items, called an itemset. An association rule is a conditional implication among itemsets, X→Y, where X and Y are itemsets and $X \cap Y = \emptyset$. An itemset can be a single item or a set of items. An itemset with k items is called a k-itemset. A subset of k elements is called a k-subset.

An association rule (AR) is called frequent if its support exceeds a minimum value *min sup*. The confidence of a rule $X \Rightarrow Y$ in T denotes the percentage of the transactions in T containing X that also contains Y. It is taken to be the conditional probability $P(X|Y)$.

That is, $confidence(X \Rightarrow Y, T) = \frac{support(X \cup Y, T)}{support(X, T)}$

A rule is called confident if its confidence value exceeds a threshold *min_conf*. . The ARM problem can be defined as follows. Find the set of all rules R of the form $X \Rightarrow Y$ such that

$R = \{X \Rightarrow Y | X, Y \subset I, X \cap Y = \emptyset, X \cup Y \subseteq f(T, min\ sup),$

$confidence(X \Rightarrow Y, T) > min\ conf\}$.

Generally, the ARM process consists of the following two steps

1) Find all frequent itemsets.

2) Generate strong ARs from the frequent itemsets.

The number of itemsets grows exponentially with the number of items |*I*|. A commonly used algorithm for generating frequent itemsets is the *apriori* algorithm.[9,7,6]

In the present work we have tried to optimize the association rule mining problem with a Pareto based genetic algorithm. At first the possible rules are represented as chromosomes, for which a suitable encoding/decoding scheme is required. For this, two approaches are available. In the Pittsburgh approach each chromosome represents a set of rules, and this approach is fit for classification rule mining, as we do not have to decode the consequent part and the length of the chromosome limits the number of rules generated. The other approach is known as the Michigan approach where each chromosome represents a separate rule. In the original Michigan approach we have to encode the antecedent and consequent parts separately; and thus this may be an efficient way from the point of space utilization since we have to store the empty conditions as we do not know from beginning which attributes will appear in which part. So a new approach will be followed where with each attribute we associate two extra tag bits. If these two bits are 00 then the attribute next to these two bits appears in the antecedent part and if it is 11 then the attribute appears in the consequent part. And the other two combinations, 01 and 10 will indicate the absence of the attribute in either of these parts. So the rule AEF->BC will look like 00A 11B 11C 01D 00E 00F. The next step is to find a suitable scheme for encoding/decoding the rules to/ from binary chromosomes. Since the positions of attributes are fixed, we need not store the name of the attributes. We have to encode the values of deferent attribute in the chromosome only. Another problem of the existing algorithms is that while generating the rules, the orders of the items also play an important role. In these algorithms, it is not possible to generate a rule of the following format, I1 I2 I6 I8 I10 I12 -> I4 I5 I9 (suffix indicates the order of appearance of the item in the binary database); though these relationships may be present inside the database. The proposed approach is free from this limitation. The next step is to find a suitable scheme for encoding/decoding the rules to/ from binary chromosomes. Since the positions of attributes are fixed, we need not store the name of the attributes. We have to encode the values of different attribute in the chromosome only. For encoding a categorical valued attribute, the market basket encoding scheme is used. This scheme is not suitable for numeric valued attributes. For a real valued attribute their binary representation can be used as the encoded value. The range of value of that attribute will control the number of bits used for it. Decoding will be simply the reverse of it. The length of the string will depend on the required accuracy of the value to be encoded.[3]

3. OVERVIEW OF GENETIC ALGORITHM.

The Genetic Algorithm was developed by John Holland in 1970. GA is stochastic search algorithm modeled on the process of natural selection, which underlines biological evolution. GA works in an iterative manner by generating new populations of strings from old ones. Every string is the encoded binary, real etc., version of a candidate solution. An evaluation function associates a fitness measure to every string indicating its fitness for the problem.

Chromosome*:* A chromosome (also sometimes called a genome) is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve. The chromosome is often represented as a simple string; although a wide variety of other data structures are also used.

Gene*:* A Gene is a part of chromosome. A gene contains a part of solution. For example if 162759 is a chromosome then 1, 6, 2, 7, 5 and 9 are its genes.

Fitness*:* Fitness (often denoted ω in population genetics models) is a central idea in evolutionary theory. It can be defined either with respect to a genotype or to a phenotype in a given environment. In either case, it describes the ability to both survive and reproduce, and is equal to the average contribution to the gene pool of the next generation that is made by an average individual of the specified genotype or phenotype. If differences between alleles at a given gene affect fitness, then the frequencies of the alleles will change over generations.

Here the chromosomes are selected (using standard selection scheme, e.g. roulette wheel selection) using the fitness value. Fitness value is calculated using their ranks, which are calculated from the non-dominance property of the chromosomes. The ranking step tries to find the non-dominated solutions, and those solutions are ranked as one. Among the rest of the chromosomes, if p_i individuals dominate a chromosome then its rank is assigned as $1 + p_i$. This process continues till all the chromosomes are ranked. Then fitness is assigned to the chromosomes such that the chromosomes having the smallest rank gets the highest fitness and the chromosomes having the same rank gets the same fitness. After assigning the fitness to the chromosomes, selection, replacement, crossover and mutation operators are applied to get a new set of chromosomes.

3.1 Outline of Basic Genetic Algorithm.

1. Generate random population Pn of n chromosomes (suitable solutions for the problem).
2. Evaluate the fitness f(x) of each chromosome x in the population.
3. Create a new population P_{n+1} by repeating following steps until the new population is completed.
4. Compute fitness f (i) of each individual i of the population P_n.

Fitness function is given by f(i)= S(A&C)/S(A)

Where S(A) is number of insistences satisfying all the conditions in antecedent A and S(A&C) is Number of examples satisfying both antecedent A and consequence C . The metric measures predictive accuracy in terms of how

many cases both antecedent and consequence part hold out of all the cases where antecedent hold.

5. Select two parent chromosomes from a population P_n according to their fitness (the better fitness, the bigger chance to be selected)

6. With a crossover probability Pc cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

7. With a mutation probability mutate new offspring at each locus (position in chromosome). Place new offspring in a population P_{n+1}.

8. Use new generated population for a further run of algorithm.

9. If the end condition is satisfied, stop, and return the best solution in current population

10. Go to step 3.

4. THE PROPOSED METHOD.

The association rules are generated from the frequent itemsets generated in each generation .Those rule which satisfies the minimum support and minimum confidence are added to our list and rest are discarded. This process continues until the desired no of generation is not completed the whole algorithm can be summarized as follows:

1. Select a suitable database that satisfies our requirements.

2. Load a sample of records from the database that fits in the memory.

3. Generate the set of frequent itemset by applying aprori algorithm on the selected record of the database based on minimum support and minimum confidence as specified by the user .Let I be the set of frequent itemset.

4. Set B is the output set, which contains the association rule.

5. For each chromosome i , compute P(i),where P(i) is probability of chromosome i .

6. Represent each frequent item set of A as binary string using the combination of representation specified in section 3.

7.Two members from the frequent item set are selected using Roulette Wheel sampling method based on their probability of selection ie P(i).

ROULETTEWHEELSELECTION(r,sum,i)

(1) let r = random number where $0 \leq r < 1$; sum := 0;

(2)for each individual i

(3)do sum := sum + P (choice = i);

(4) if r < sum;

(5)return i;

(6)else goto step (2)

8. Crossover and mutation operations are applied on the selected members to generate the association rules. 9. Find the fitness function for each rule A C and check the following condition.

I. if (> Cmin) ,where Cmin is minimum confidence.

II. Add the newly generated association rules to the set B

10. If the desired number of generations is not completed, then go to Step 3.

11. Stop

5. IMPLEMENTATION AND RESULTS.

Here we present the results on one supermarket transaction database having 16 attributes and 100 records. Crossover and mutation probabilities were taken respectively as 0.1 and 0.05; 1 point crossover operator was used and the population size was kept fixed as 2 ie. in each generation 2 parents are selected using roulettes wheel selection method. Number of generations was fixed as 10. The rules are selected based on their fitness value.16 attributes, namely are

Table 1: List of attributers of the data base

Attribute	ID
MILK	I1
BREAD	I2
BUTTER	I3
TEA	I4
SUGAR	I5
BEER	I6
JAM	I7
CORNFLAKES	I8
COFFEE	I9
CHEESE	I10
BROWNBRAD	I11
MEAT	I12
CHOCOLATE	I13
CAKE	I14
COKE	I15
FRUITS	I16

Frequent itemset is generated by apriori algorithm. Smin =10 size of the frequent itemset is= 34. Our main goal is optimization. Again we know that data mining technique (with genetic algorithm) does not give the best solution, it gives optimal solution. Here we have selected rules with Cmin=10%. We also implemented apriori algorithm Using the same. Result are compared in the table below:

Table 2: Comparison of rules

Apriori Algorithm	Genatic algorithm
{MILK=>BREAD ,SUGAR}/33.6 {BREAD=>MILK,SUGAR}/58.4 {SUGAR=>MILK,BREAD}/64 {BREAD,SUGAR=>MILK}/91 {MILK ,SUGAR=>BREAD}100 {MILK,BREAD=>SUGAR}/64	{ BREAD SUGAR } => { MILK } /(Confidence:91.66666666666666 %) { SUGAR } => { MILK TEA }/ (Confidence:38.235294117647065%) { MILK TEA } => { SUGAR }/ (Confidence:92.85714285714288 %) { MILK } => { BREAD SUGAR }/ (Confidence:22.448979591836732%) { MILK SUGAR } => { BREAD }/ (Confidence:36.666666666666664%) { BREAD } => { MILK SUGAR } /(Confidence:20.754716981132074%) { MILK } => { CORNFLAKES }/ (Confidence:36.734693877551024%) { MILK } => { TEA SUGAR }/ (Confidence:26.530612244897966 %) { MILK SUGAR } => { TEA }/ (Confidence:43.333333333333336%) { TEA SUGAR } => { MILK }/ (Confidence: 81.25%) { SUGAR } => { MILK BREAD }/ (Confidence:32.35294117647059 %) { TEA } => { MILK SUGAR } /(Confidence:72.22222222222223 %) { MILK BREAD } => { SUGAR }/ (Confidence:47.826086956521735%) { CORNFLAKES } => { MILK }/ (Confidence:94.73684210526316 %)

In the above table we have shown a comparison of rule generated by apriori algorithm and genetic algorithm using the same dataset. With genetic algorithm we observed some surprising rules and the no of rules with Cmin also increases. So we can say that the rules we obtained by the proposed algorithm are optimized one.

6. FUTURE SCOPE AND CONCLUSION.

The extracted rules showed good consistency for the testing, training and validation period. The method describe here is very simple and efficient. To improve the efficiency of this algorithm, some refinement may be required. For example, this algorithm works on a sample of the original database, and the sample may not truly reflect the actual database. A perfect sample will improve the correctness of the rules generated by the algorithm.

REFERENCES

[1] Agrawal, R., Imielinski, T., & Swami, A. (1993). "Mining association rules between sets of items in large databases." In Proceedings of ACM SIGMOD conference on management of data (pp. 207–216)

[2] Agrawal, R., & Srikant, R. (1994). "Fast algorithms for mining association rules". In Proceedings of the 20th international conference on very large databases, Santiago, Chile.

[3] A. Ghosh and B. Nath, "Multi-objective rule mining using genetic algorithms," *Inf. Sci.*, vol. 163, nos. 1–3, pp. 123–133, Jun. 2004.

[4] Alatas B, Akin E, Karci A (2008) "MODENAR: multi-objective differential evolution algorithm for mining numeric association rules". Appl Soft Comput 8(1):646–656

[5] Peter P. Wakabi-Waiswa,Venansius Baryamureeba, "Extraction of interestiong association rules using genetic algorithms, International Journal of Computing and ICT Research", Vol. 2 No. 1, June 2008.

[6] Soumadip Ghosh, Sushanta Biswas, Debasree Sarkar, Partha Pratim Sarkar, "International Journal of Artificial Intelligence & Applications (IJAIA)", Vol.1, No.4, October 2010.

[7] Hamid Reza Qodmanan, Mahdi Nasiri, Behrouz Minaei-Bidgoli, "Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence", 2010 Elsevier Ltd.

[8] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello, "Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part I", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 18, NO. 1, FEBRUARY 2014.

[9] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello, "Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part II", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 18, NO. 1, FEBRUARY 2014.

[10] M. Khabzaoui, C. Dhaenens, and E.-G. Talbi, "Combining evolutionary algorithms and exact approaches for multiobjective knowledge discovery", *RAIRO—Oper. Res.*, vol. 42, no. 1, pp. 69–83, 2008.

[11] M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, J. C. Riquelme1, "Improving a multi-objective evolutionary algorithm to discover quantitative association rules", Springer-Verlag London 2015

[12] M. Kaya and R. Alhajj, "Facilitating fuzzy association rules mining by using multiobjective genetic algorithms for automated clustering", in *Proc. 3rd IEEE ICDM*, 2003, pp. 561–564.

[13] S. G. Matthews, M. A. Gongora, and A. A. Hopgood, "Evolving temporal fuzzy association rules from quantitative data with a multiobjective evolutionary algorithm," in *Proc. Int. Conf. HAIS—Vol. Part I*, 2011, pp. 198–205.