International Journal of Soft Computing 2 (1): 75-79, 2007

Optimization of Association Rules Using Genetic Algorithms

[1]Sanjeev Sharma, [2]Vivek Badhe and [3]Sudhir Sharma
[1,2]School of Information Technology,
[3]Department of Mathematics, Rajiv Gandhi Technological University,
Bhopal (M.P.), India

Abstract: Association Rule Mining is a powerful method in data mining, which aims at finding latent knowledge. In other words, the objective of rule mining is the discovery of interesting patterns that exist in database but are unseen among massive volumes of data. Mining Association rules in not full of reward until it can be utilized to improve decision-making process of an organization. The main obstacle in this area is the number of rules grows exponentially with the number of items, so it arises problem of selecting interesting rules from set of rules. In this study, we attempt to optimize the rules generated by Apriori method using genetic algorithm. The generated rules may optimize using certain measures like support count, confidence factor, interestingness, comprehensibility and completeness. But for the sake of high-level rules we consider the interestingness and completeness as measures for optimization.

Key words: Genetic algorithms, data mining, association rule mining

INTRODUCTION

In the current years it has seen a dramatic increase in the amount of information or data being stored in database. The exponentially growth in size of on hand databases, mining for latent knowledge become essential to support decision-making. According to researchers two elementary goals of data mining[1] are prediction and description. Prediction makes use of existing variables in the database in order to predict unknown or features of interest. And description focuses on finding patterns describing the data and the subsequent presentation for user interpretation. There are several techniques satisfying these objectives. Some of these can be classified into the following categories: classification, clustering, association rule mining, sequential pattern discovery and analysis. In this study we considered Association Rule Mining for knowledge discovery and generate the rules by applying apriori implementation on our student database. And finally attempt to optimize the generated rules by applying genetic algorithm.

Association rule mining: Association rules identify relationships among sets of items in a transaction database. Ever since it's introduced[2], association rule discovery has been an active research area. Association rule mining finds interesting association or correlation relationships among a large set of data items. In the most general form, an association rule is an implication C1 related to C2 where C1 and C2 are conditions on tuples of the relation. Usually, they are conjunctions of simple conditions that is, conditions involving a single attribute, e.g., sex=female or age<30. Such simple conditions are often called items; conjunctions of simple conditions are called itemsets. Rule support and confidence are the two metrics that are used to prune uninteresting rules. The support of a rule is the fraction of tuples in the data that satisfy both of the rule's conditions (C1^C2). A rule's confidence is the ratio of the number of tuples that satisfy C1^C2 to the number of tuples that satisfy C1.

Then, the problem of discovering association rules can be stated as follows: find all association rules that hold with more than the minimum support and confidence as defined by the user. The task of discovering association rules can be performed in two steps. In the first step, the sets of items that have the minimum support are found. Such sets are called frequent item sets. In the second step, the discovered frequent itemsets are used to generate association rules. It turns out, that the first step is much more expensive than the second, since the number of item sets grows exponentially with the number of items. A large number of increasingly efficient algorithms to mine frequent item sets have been developed over the years.

Corresponding Author: Sanjeev Sharma, School of Information Technology, Rajiv Gandhi Technological University, Bhopal (M.P.), India

The strategies for efficient discovery of frequent item sets can be divided into two categories. The first is based on the candidate generation-and-test approach. Apriori[2] and its several variations belong to this category. They use the anti-monotone or Apriori property that any subset of a frequent item set must be a frequent item set. In this approach, a set of candidate item sets of length $n + 1$ is generated from the set of item sets of length n and then each candidate item set is checked to see if it meets the support threshold. The second approach of pattern-growth has been proposed more recently. It also uses the Apriori property, but instead of generating candidate item sets, it recursively mines patterns in the database counting the support for each pattern. Algorithms in this category include FP-Growth[3]. The candidate generation and test approach suffers from poor performance when mining dense datasets since the database has to be read many times to test the support of candidate item sets. The alternative pattern growth approach reduces the cost by combining pattern generation with support counting. In this study we consider the first approach i.e. apriori algorithm for generation of rules.

Rule discovery problem definition: The formal statement of association rule mining is largely based on the description of the problem in[2,4]. Formally, the problem can be stated as follows: Let $Z = \{i_1, i_2, \ldots, i_m\}$ be a set of m. distinct literals called items. D is a set of variable length transactions over I. Each transaction contains a set of items $i_i, i_j, , i_k \subset 1$. A transaction also has an associated unique identifier called TID. An association rule is an implication of the form X=>Y, where X, $Y \subset I$ and $X \cap Y = \phi$. X is called the antecedent and Y is called the consequent of the rule. There are two basic measures for association rules[2] support(s) and confidence(c). Each item set has an associated measure of statistical significance called support. A rule has a measure of its strength called confidence.

The X=>Y holds in transaction set D with support s, where s is the percentage of transaction in D that contain X U Y (i.e., both X and Y). This is taken to be probability, P (X U Y).

$$\text{Support(s)} = \text{support } (X=>Y) = P(A \cup B)$$

Thus, support(s) of an association rule is defined as the percentage/fraction of records that contain X U Y to the total number of records in the database.

The rule X=>Y has confidence c in the transaction set D if c is the percentage of transaction in D containing X that also contain Y. This is taken to be the conditional probability, P(Y\X). That is:

$$\text{Confidence(c)} = \text{confidence } (X=>Y)$$
$$P(Y\backslash X)=\text{sup\_count}(XUY)/\text{sup\_count}(X)$$

In other words, confidence of an association rule is defined as the percentage/fraction of number of transactions that contain X U Y to the total number of records that contains X, where if the percentage exceeds the threshold of confidence association rule X=>Y can be generated.

The problem of mining association rules[2,4] is to generate all rules that have support and confidence greater than some user specified minimum support and minimum confidence thresholds, respectively. This problem can be decomposed into the following sub-problems:

- All item sets that have support above the user specified minimum supports are generated. These item set are called the frequent item sets or large item sets.
- For each large item sets, all the rules that have minimum confidence are generated as follows: for a large item set X and any $Y \subset X$, if support (X)/support (X-Y) ≥ minimum-confidence, then the rule X - Y => Y is a valid rule.

Genetic algorithms: Genetic Algorithms[5] (GAs), first conceived by John Holland at the University of Michigan in 1975, are class of computational models that imitate natural evolution to solve problem in as wide verity of domains. Genetic Algorithms are particularly suitable for solving problems, which require optimization.

A genetic algorithm[6] is a type of searching algorithm. It searches a solution space for an optimal solution to a problem. The key characteristic of the genetic algorithm is how the searching is done. The algorithm creates a population of possible solutions to the problem and lets them evolve over multiple generations to find better and better solutions. The generic form of the genetic algorithm is found in Fig. 1. The items in bold in the algorithm are defined here. The population is the collection of candidate solutions that we are considering during the course of the algorithm. Over the generations of the algorithm, new members are born into the population, while others die out of the population. A single solution in the population is referred to as an individual. The fitness of an individual is a measure of how good the solution represented by the individual is. The better the solution, the higher the fitness-obviously, this is dependent on the problem to be solved. Representation of rules in encoded from is very important. The Genetic Algorithm[7] having two basic approaches. Pittsburg Approach: In this study each

0 Start: Create random population of n chromosomes
1 Fitness : Evaluate fitness f(x) of each chromosome in the population
2 New population
  0 Selection : Based on f(x)
  1 Recombination : Cross-over chromosomes
  2 Mutation : Mutate chromosomes
  3 Acceptation : Reject or accept new one
3 Replace : Replace old with new population: the new generation
4 Test : Test problem criterium
5 Loop : Continue step 1-4 until criterium is satisfied

Fig. 1: Generic form of the genetic algorithm

chromosome represents a set of rule and this study is more suitable for classification rule. Michigan Approach: Each chromosome represents a separate rule.

MATERIALS AND METHODS

Optimization does not mean maximization/ minimization. Optimization[7] means to get the most feasible solution or utilization of the available methodology for their best uses.

In this study the Genetic Algorithm is used to optimize (i.e. finding interesting rule) rules fetched from association rule mining. For demonstration of its utility, the database was generated synthetically. The database conations the choice of electives by students during their II and III semesters of course studies at School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal. Students have to opt four subjects (2-2 in each semesters) from eight (4-4 choices) based on their liking and area of interest. For optimizing Association Rule[8] first we implement Apriori and then take a Fitness (objective function) function and optimize it. And apply the final value in searching of interesting rules with completeness.

Fitness function: The success of a genetic algorithm is directly linked to the accuracy of the fitness function. The fitness function should be tailored to the specific search spaces. We take a fitness function that considers major issues in evaluating an individual against its search space. The fitness of a population is the sum of the individual fitness values of that population. The fitness function is the primary performance sink in a genetic algorithm, because this is the place that the underlying

		Predicted	
		Yes	No
Actual	Yes	TP	FN
	No	TP	FN

Fig. 2: Confusion matrix for rule

data must be accessed, so optimizations should be considered wherever possible.
Rule is generally defend as:

IF antecedent than consequent

The confusion matrix in the following Fig. 2 defines the performance of a rule:

The abbreviation and meaning of the labels used in the confusion matrix have the following meaning:

TP: True Positive ➜ Number of examples satisfying antecedent and consequent
FP: False Positive ➜ Number of examples satisfying antecedent but not consequent
FN: False Negative ➜ Number of examples satisfying consequent but not antecedent
TN: True Negative ➜ Number of examples not satisfying antecedent and consequent

Interestingness Factor (INF) = TP/(TP+FP)
Completeness Factor (CF)= TP/(TP+FN)
Fitness = INF x CF

We used the above fitness function, for the rule optimization but for the different dataset or application one may modify the fitness function as:

Fitness = W1x (INF x CF) + W2xS

Here W1 and W2 is the user-defined weight based on dataset. And S is the simplicity factor generally values lie between [0,1]

Implementation detail with result: For the implementation of the Apriori algorithm we have written the codes of Microsoft® Visual Basic 6.0 and developed a program, which handles the sequential file for input and output. Finding the optimum value of the fitness function[11], we used the MATLAB®[9], my goal is to find the best weight for every variable, which provide best result. Finally we have also develop a module in visual basic that take the final fitness values along with the individual variable values involved in fitness function and find the rules that

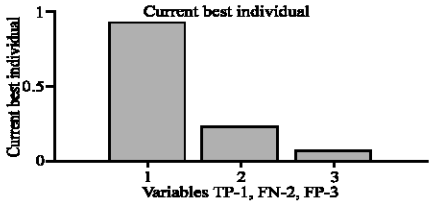

Fig. 3: Best individual

Table 1: Synthetic dataset

Sub-1	Sub-2	Sub-3	Sub-4
NS	AOS	MC	BI
CG&M	AI&N	BI	ERP
CG&M	AOS	MC	BI
AI&N	CG&M	BI	MC
NS	AI&N	DM	ERP
NS	AI&N	BI	DM
NS	AI&N	DM	ERP
CG&M	AI&N	ERP	DM
NS	CG&M	DM	BI
AOS	CG&M	MC	BI

Table 2: Rules

Association rule	Support confidence
DM <- AI&N	(44.4, 56.3)
DM <- NS	(47.2, 82.4)
AI&N <- ERP DM	(22.2, 50.0)
NS <- ERP DM	(22.2, 62.5)
DM <- ERP CG&M	(19.4, 57.1)
CG&M <- ERP DM	(22.2, 50.0)
AI&N <- NS MC	(16.7, 50.0)
DM <- AI&N NS	(19.4, 85.7)
NS <- AI&N DM	(25.0, 66.7)
NS <- BI DM	(22.2, 75.0)

Table 3: Fitness values obtained

TP	FN	FP	Fitness value
8	5	8	0.31
7	10	6	0.22
7	14	6	0.18
8	15	5	0.21
14	3	9	0.5
12	6	6	0.44
5	18	2	0.16
6	12	6	0.17
6	12	3	0.22
7	14	5	0.19

belongs the optimal fitness value with 10% tolerance and generate the final result. Following Table 1-5 and Fig. 3 shows the sample datasets and results.

The values for the basic Genetic Algorithm operators used in this study are as follows:

The association rule mining algorithm and other preprocessing module and interfaces were implemented in Visual Basic, for the task related to genetic algorithms we used MATLAB and tested all experiments on an IBM compatible PC with a Pentium IV Processor of 2.4 GHZ and 256 MB of main memory running the Microsoft XP professional operating system.

Table 4: Optimized rules

Association rule	Support confidence
CG&M <- ERP	(36.1, 53.8)
BI <- AOS MC	(33.3, 50.0)
CG&M <- AOS MC	(33.3, 58.3)
CG&M <- AOS BI	(25.0, 66.7)
DM <- AOS CG&M	(27.8, 60.0)

Table 5: Genetic algorithm parameters

Selection	Tournament, Size=4
Crossover	Single point Crossover (with probability 0.8)
Mutation	Uniform (with rate=0.01)
GA population	100

CONCLUSION

A data mining system has the capability to generate thousands or even millions of rules. So all the rules are not interesting, only small fraction of the rules potentially generated would actually be interest to any user. Generate only interesting rules is the optimization problem in data mining.

In this study to generate the association rules we have implemented Apriori algorithm, all rules generated by Apriori are not interesting. We have used Genetic Algorithm to optimize the generated rules; we take a fitness function for the task of optimization and find the optimum solutions that are interesting rules.

To improve the efficiency and accuracy of the optimization there is need of incorporating more measures like comprehensibility and some refinement and enhancement in simple Genetic Algorithm for such data mining problems.

REFERENCES

1. Han, J. and M. Kamber, 2001. Data Mining: Concepts and techniques, Morgan Kaufmann Publishers, Elsevier India.
2. Agrawal, R., T. Imielinski and A. Swami, 1993. Mining Association Rules Between sets of items in large Databases. In Proceedings of the 1993 ACM SIG- MOD International Conference on Management of Data, Washington, DC, pp: 207-216.
3. Han, J., J. Pei and Y. Yin, 2000. Mining Frequent Patterns Without Candidate Generation. In ACM SIGMOD Conf. Management of Data.
4. Agrawal, R. and R. Shrikanth, 1994. Fast Algorithm for Mining Association Rules. Proceedings Of VLDB conference, Santigo, Chile, pp: 487-449.
5. Melanie Mitchell, 1996. An Introduction to genetic Algorithms, PHI.

6. Deb, K., 2001. Multi-objective Optimization using Evolutionary Algorithms, Wiley, New York.
7. Ashish Ghosh and Bhabesh Nath, 2004. Multi-objective rule mining using genetic algorithms, Information Sci., 163: 123-133.
8. Ayahiko Niimi and Eiichiro Tazaki, 2000. Rule Discovery technique using genetic programming combined with apriori algorithm, springer-verlag Berlin Heidelberg, pp: 273-278.
9. Chipperfield, A.J. and P.J. Fleming, 1995. The MATLAB genetic algorithm toolbox, Department of Automatic Control and Systems Engineering, University of Sheffield, PO Box 600, Mappin Street, Sheffield, England. S1 4DU, From IEE Colloquium on Applied Control Techniques Using MATLAB, Digest.