OPTIMAL RULE SET GENERATION USING PSO ALGORITHM

Shampa sengupta[1], Asit Kumar Das[2]

[1]Department of Information Technology, MCKV Institute of Engineering, Liluah,
Howrah – 711 204, West Bengal, India
shampa2512@yahoo.co.in

[2]Department of Computer Science and Technology, Indian Institute of Engineering, Science and Technology, Shibpur, Howrah – 711 103, West Bengal, India
akdas@cs.becs.ac.in

ABSTRACT

Classification and Prediction is an important research area of data mining. Construction of classifier model for any decision system is an important job for many data mining applications. The objective of developing such a classifier is to classify unlabeled dataset into classes. Here we have applied a discrete Particle Swarm Optimization (PSO) algorithm for selecting optimal classification rule sets from huge number of rules possibly exist in a dataset. In the proposed DPSO algorithm, decision matrix approach was used for generation of initial possible classification rules from a dataset. Then the proposed algorithm discovers important or significant rules from all possible classification rules without sacrificing predictive accuracy. The proposed algorithm deals with discrete valued data, and its initial population of candidate solutions contains particles of different sizes. The experiment has been done on the task of optimal rule selection in the data sets collected from UCI repository. Experimental results show that the proposed algorithm can automatically evolve on average the small number of conditions per rule and a few rules per rule set, and achieved better classification performance of predictive accuracy for few classes.

KEYWORDS

Particle swarm optimization, Data Mining, Classifiers.

1. INTRODUCTION

Many particle swarm algorithms have been developed that deals only with continuous variables [1, 5, 10]. This is a significant limitation because many optimization problems are there which featuring discrete variables in the problem domain. Typical problems are there in the space which deals with the ordering, grouping or arranging of discrete variables such as scheduling or routing problems. Therefore, the developing of particle swarm algorithms that deals with discrete variables is important for such kind of problem. We propose a variant of Particle Swarm Optimization (PSO) algorithm applied to dicerete data for rule selection in Data Mining. We will refer to this algorithm as the discrete Particle Swarm Optimization (DPSO) algorithm. The DPSO deals with discrete valued data, and its population of candidate solutions contains particles of different sizes. Although the algorithm has been specifically designed for optimized rule selection task, it is by no means limited to this kind of application. The DPSO algorithm may be applied to other optimization problems with little modification. Construction of classifier model for any decision system is an important job for many data mining applications. The objective of developing such a classifier is to classify unlabeled dataset (object belongs to test set) into classes. Here we propose a discrete Particle Swarm Optimization (PSO) algorithm for selection of optimal or near optimal classification rules from huge number of rules possibly exists in a dataset. Primarily, all exhaustive rules are generated from the dataset using decision matrix approach [11, 12] which is based on rough set theory. Then the DPSO method identifies important or significant rules from all possible classification rules without sacrificing predictive accuracy. For the larger dataset, if all decision rules are considered for data analysis then time complexity will be very high. For this reason a minimum set of rules are generated.

The paper is organized as follows. Section 2 briefly introduces PSO algorithm. Section 3 introduces the DPSO algorithm proposed in this paper for the task of optimal rule selection. Section 4 and 5 reports experimental methodology, experiments and results respectively and finally Section 6 presents conclusions of the work.

2. A Brief Introduction To Particle Swarm Optimization

PSO [5, 6, 7] is an evolutionary optimization algorithm proposed by Kennedy and Eberhart in 1995. In PSO, a population, called a swarm, of candidate solutions are encoded as particles in the search space. PSO starts with the random initialization of a population of particles. The whole swarm move in the search space to search for the best solution by updating the position of each particle based on the experience of its own and its neighbouring particles. In PSO a potential solution to a problem is represented by a particle X(i) = ($x_{(i,1)}$, $x_{(i,2),}$ …,$x_{(i,n)}$) in an n-dimensional search space. The coordinates $x_{(i,d)}$ of these particles have a rate of change(velocity) $v_{(i,d)}$, d = 1, 2, ..., n. Every particle keeps a record of the best position that it has ever visited. Such a record is called the particle's previous best position and denoted by B_i. The global best position attained by any particle so far is also recorded and stored in a particle denoted by G. Iteration comprises evaluation of each particle with the adjustment of $v_{(i,\, d)}$ in the direction of particle X(i)'s previous best position and the previous best position of any particle in the neighbourhood.

Generally speaking, the set of rules that govern PSO are: evaluate, compare and evolve. The evaluation phase measures how well each particle (candidate solution) solves the problem. The comparison phase identifies the best particles. The evolve phase produces new particles based on some of the best particles previously found. These three phases are repeated until a given stopping criterion is matched. The objective is to find the best particle which solves the target problem. Important concepts in PSO are velocity and neighbourhood topology. Each particle, X(i), is associated with a velocity vector. This velocity vector is updated at every generation. The updated velocity vector is then used to generate a new particle X(i). The neighbourhood topology defines how other particles in the swarm, such as B (i) and G, interact with X (i) to modify its respective velocity vector and, consequently, its position as well.

3. The Proposed Discrete PSO Algorithm

The algorithm presented here is based on DPSO algorithm [6]. The proposed algorithm deals with generation of optimized classification rules from discrete valued dataset, which is typically a rule mining problem. Primarily, all exhaustive rules are generated from the dataset using decision matrix approach [11, 12] which is based on rough set theory. Every rule has two parts, conditional part and decision part, conditional part comprises of some conditional attributes with their values and decision part has decision attribute with the corresponding decision value or class. In a rule, each conditional attribute with the corresponding value is termed as rule component. So a rule is formed by some rule components. In PSO, population of candidate solutions contains particles of different sizes. Here population of candidate solutions is generated from initial rule set. Each particle represents the antecedent part of a rule by considering the conclusion part is fixed at the beginning of the execution and represents a target attribute value. In this case to discover the optimal rule set, predicting different target/decision values, the algorithm has to run several times, one for each decision value. There are N (No of initial rules) particles in a swarm. The length of each particle may vary from 1 to n, where n is the number of unique rule component present in the initial rule set, which is made by considering only the antecedent part of each rule. Each particle R_i keeps a record of the best position it has ever attained. This information is kept in a distinguished particle labeled as B_i. The swarm also keeps the information of the global best position ever attained by any particle in the swarm. This information is also kept in a separated particle labeled G. G is equal to the best Bi present in the swarm.

3.1 ENCODING OF THE PARTICLES FOR THE PROPOSED DPSO ALGORITHM

Each rule is formed by some rule components and each rule component is identified by a unique positive integer number or index. These indices, vary from 1 to n, where n is the number of unique rule components present in the initial rule set. Each particle is subsets of indices without repetition. For example, corresponding to the rules R_1, R_2 and R_3 given below, the rule components are (A=1), (A=2), (B=3), (B=4) and (C=5) which are indexed as 1, 2, 3, 4 and 5 respectively.

R_1= {A=1, B=3, C=5}
R_2= {A=2, B=4}
R_3= {A=2, C=5}
Where, A, B, C are the conditional attributes, N (Number of initial rules) = 3. Here initial swarm representing candidate solution could looks as follows:
R_1= {1, 3, 5}
R_2= {2, 4}
R_3= {2, 5}.

3.2 THE INITIAL POPULATION FOR THE PROPOSED DPSO ALGORITHM

The initial population of particles is generated as follows.

Population of candidate solutions is generated from initial rule set. Each particle represents the antecedent part of a rule by considering the conclusion part as fixed at the beginning of the execution and represents a target attribute value. Rule encoding process is described in section 3.1. By considering the rule encoding process the particles are formed for each rule generated using decision matrix based classification method.

3.3 VELOCITIES

The DPSO algorithm does not use a vector of velocities. It works with proportional likelihoods. Basically, the idea of proportional likelihood used in the DPSO algorithm is almost similar with the idea of velocity used in the standard PSO algorithm. Each particle is associated with a $2 \times n$ array of proportional likelihoods, where 2 and n represents number of rows and number of columns respectively. In this standard proportional likelihood array, each element in the first row of V_i represents the proportional likelihood based on which a rule component be selected. The second row of V_i has the indices of the rule components which is associated with the respective proportional likelihoods of the first row of the vector V_i. There is a one-to-one correspondence between the columns of this array. At the beginning, all elements in the first row of V_i are set to 1, e.g., V_i = {{1, 1, 1, 1, 1 }, {1,2,3,4,5}}.

After the initial population of particles is generated, this array is always updated before a new configuration for the particle associated to it is made. The updating process is based on R_i, B_i (particle's previous best position) and G (global best position) and works as follows.

In addition to R_i, Bi and G, three constant updating factors, namely, a, b and c are used to update the proportional likelihoods $v_{(i,d)}$. These factors determine the strength of the contribution of R_i, B_i and G to the adjustment of every coordinate $v_{(i,d)}$ € $V_{i.}$ Parameter values of a, b and c is chosen experimentally.

3.4 GENERATING NEW PARTICLES FOR THE PROPOSED DPSO ALGORITHM

The proportional likelihood array V_i is then used to sample a new configuration of particle R_i that is, the particle associated to it. First, for a particle with V_i, all indices present in Ri have their corresponding proportional likelihood increased by 'a'. Similarly, all indices present in B_i and G have their corresponding proportional likelihood increased by 'b' and 'c' respectively. Now each element of the first row of the array V_i is then multiplied by a uniform random number between 0 and 1. A new random number is generated for every single multiplication performed. The new particle is then defined by ranking the columns in V_i by the values in its first row. That is, the elements in the first row of the array are ranked in a decreasing order of value and the indices of the rule components (in the second row of V_i) follow their respective proportional likelihoods.

Thus for example, after all the steps if particle i has length 3, and the particle associated to the array

$$V_i = \begin{array}{ccccc} 0.74 & 0.57 & 0.50 & 0.42 & 0.20 \\ 5 & 4 & 3 & 1 & 2 \end{array}$$

Then first 3 indices from the second row of Vi would be selected to compose the new particle. That is, $R_i = \{*, *, *\}$ the indices (rule components) 5, 4 and 3 would be selected to compose the new particle, i.e., $R_i = \{5, 4, 3\}$. Note that indices that have a higher proportional likelihood are, on average, more likely to be selected. If indices are the rule components with same attribute are selected, then higher confidence rule component is selected. In that case new particle size is also changes in the generation. The updating of R_i, B_i and G is identical to what is described earlier. In this way new particles are formed generation by generation.

4. EXPERIMENTAL METHODOLOGY

The purpose of this experiment was to evaluate the classification accuracy and comprehensibility by the number of rules in the data set, and the average number of rule conditions per rule. The fitness function f (Ri) of any particle i is computed as follows. Optimized rule selection process can be performed by DPSO as a multi objective problem to maximize the confidence (association rule mining concept) of a rule to achieve higher classification accuracy as well as minimizing the length of a rule. The goal is to see whether DPSO can select optimized set of rules to achieve a higher classification accuracy rate from initial set of rules.

In this regard following fitness function is considered.

$F_i = \alpha * rule_i\_confidence + (1-\alpha)*1/(rule_i\_length)$

$rule_i$ _confidence: A rule like, $R_i \rightarrow (C_{i1} = a)^\wedge(C_{i2} = b)\ldots{}^\wedge(C_{in} = c) \rightarrow (D = d_i)$ has the condition part C_i where conditional attributes are associated with value, and the decision part D has the decision values d_i. So here the rule maps $C_i \rightarrow d_i$.

Then the confidence of the rule is conf (R_i) = number of rows in *dataset* that match C_i and have class label $\boldsymbol{d_i}$ **/** number of rows in *dataset* that match only C_i.

Here relative importance of the rule confidence and the length of the rule are considered. Rule confidence is set larger than length of a rule because the classification performance is assumed more important than the number of conditional attributes present in a rule i.e. rule length. The objective is to find the fittest rules with which it is possible to classify the data set as belonging to one of the classes with an acceptable accuracy. Here $\alpha=0.8$ is considered.

5. EXPERIMENTS AND RESULT

The computational experiments involved a 10-fold cross-validation method [13].To illustrates the method, wine dataset from UCI repository [14] of Machine Learning databases [10] is considered. First, the 178 records with 3 distinct classes (0, 1, 2) in the wine data set were divided into 10 almost equally sized folds. The folds were randomly generated but under the following regulation. The proportion of different classes in every single fold was similar to the one found in the original data set containing all the 178 records. Each of the 10 folds is used once as test set and the remaining of the data set is used as training set. The purpose of the experiment was to evaluate the generalization ability, measured as the classification accuracy on test set. Each training set was used for generation of initial classification rules. For each initial rule set, DPSO algorithm generates optimized set of classification rules according to their fitness which is used to classify the test set (each 10 test folds) accordingly. DPSO selects only the best particle in each run as the rule.

As the DPSO algorithm is stochastic algorithm, 20 independent runs for the algorithm were performed for every single fold and for every decision classes. The average number of rules by the rule selection algorithms has always been rounded to the nearest integer. The population size (initial rule set) used for the algorithm is on average 300 and the search stops in one run after 100 iterations. Other choices of parameter values were a = 0.10, b = 0.12 and c = 0.14. The results of experiment were as follows. For Wine on average 95 rules are selected as optimized rules from 300 rules. And using these 95 rules on test set 99.32% of average classification accuracy is achieved.Using fewer rules, DPSO algorithm obtained on average, a better predictive accuracy than the classification performed using all the initial classification rules for few classes. For the initial rule set average accuracy values on test set for 3 classes (0, 1, 2)were 99.05, 99.48, 99.77, while using optimized rule set corresponding accuracy values on test set were 98.38, 99.59, 100. The results also indicate that not only the predictive accuracy is good, but also the number of rule conditions or rule length is relatively short like 2- 4 rule components in a rule where the same was 3-5 in the initial rule set and there are a small number of rules in the rule set.

6. CONCLUSION

We proposed an efficient method for rule discovery using DPSO algorithm. The discovered rules are with of high accuracy and comprehensibility. Using fewer rules, DPSO algorithm obtained on average, a better predictive accuracy than the classification performed using all the initial classification rules for few classes. The results on the Wine data set show that our approach has good performance for rule discovery on discrete data. Few less coverage rules were also selected. Rule Coverage is an important parameter for selection of good quality rule. In the future by applying some rule pruning technique better quality rule set can be generated.

REFERENCES

[1] T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In Lecture Notes in Computer Science, volume 3005, pages 489{500. Springer-Verlag, 2004.

[2] E. S. Correa, M. T. Steiner, A. A. Freitas, and C. Carnieri. Using a genetic algorithm for solving a capacity p-median problem. Numerical Algorithms, 35:373{388, 2004.

[3] D. Freedman, R. Pisani, and R. Purves. Statistics. W. Norton & Company, 3rd edition, September 1997.

[4] A. A. Freitas. Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag, October 2002.

[5] G. Kendall and Y. Su. A particle swarm optimisation approach in the construction of optimal risky portfolios. In Proceedings of the 23rd IASTED International Multi-Conference on Applied Informatics, pages 140{145, 2005. Artificial intelligence and applications

[6] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 Conference on Systems, Man, and Cybernetics, pages 4104{4109, Piscataway, NJ, USA, 1997. IEEE.

[7] J. Kennedy and R. C. Eberhart. Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001

[8] T. M. Mitchell. Machine Learning. McGraw-Hill, August 1997.

[9] R. Poli, C. D. Chio, and W. B. Langdon. Exploring extended particle swarms: a genetic programming approach. In GECCO'05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pages 169{176, New York, NY, USA, 2005. ACM Press.

[10] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In EP'98: Proceedings of the 7th International Conference on Evolutionary Programming, pages 591{600, London, UK, 1998. Springer-Verlag

[11] I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2nd edition, 2005

[12] N.Zhong and A. Skowron, “A Rough Set-Based Knowledge Discovery Process”, Int. Journal of Applied Mathematics and Computer Science., 11(3), 603-619, 2001. BIME Journal, Volume (05),), 2005

[13] Ethem Alpaydin Introduction to Machine Learning.PHI, 2010.

[14] Murphy, P. and Aha, W.: UCI repository of machine learning databases (1996), http://www.ics.uci.edu/mlearn/MLRepository.html