Rule Discovery with Particle Swarm Optimization

Yu Liu[1], Zheng Qin[1,2], Zhewen Shi[1], and Junying Chen[1]

[1] Department of Computer Science, Xian JiaoTong University, Xian 710049, P.R. China
liuyu@mailst.xjtu.edu.cn
http://www.psodream.net

[2] School of Software, Tsinghua University, Beijing 100084, P.R. China

Abstract. This paper proposes Particle Swarm Optimization (PSO) algorithm to discover classification rules. The potential IF-THEN rules are encoded into real-valued particles that contain all types of attributes in data sets. Rule discovery task is formulized into an optimization problem with the objective to get the high accuracy, generalization performance, and comprehensibility, and then PSO algorithm is employed to resolve it. The advantage of the proposed approach is that it can be applied on both categorical data and continuous data. The experiments are conducted on two benchmark data sets: Zoo data set, in which all attributes are categorical, and Wine data set, in which all attributes except for the classification attribute are continuous. The results show that there is on average the small number of conditions per rule and a few rules per rule set, and also show that the rules have good performance of predictive accuracy and generalization ability.

1 Introduction

There has been a great interest in the area of data mining, in which the general goal is to discover knowledge that is not only correct, but also comprehensible and interesting for the user [1]. Hence, the user can understand the results produced by the system and combine them with their own knowledge to make a well-informed decision, rather than blindly trusting on results produced by system. Classification is an important topic in data mining research. The knowledge in classification is often expressed as a set of rules. IF-THEN rules are high-level symbolic knowledge representations and have the advantage of being intuitively comprehensible for users. Evolutionary approaches like genetic algorithms (GA) and genetic programming (GP) have been applied to discover classification rules. Examples of GA for rule discovery can be found in [2-3], and examples of GP for rule discovery can be found in [4-6]. Recently, Particle Swarm Optimizer (PSO) has attracted researchers in optimization field. But using Swarm Intelligence in data mining is a fairly new research area and needs much more work to do. So using PSO for rule discovery is a quite new and challenging research area. Tiago Sousa et al. in [7-8] proposed a binary-encoding way to discover classification rules with PSO for categorical data. However, our PSO algorithm of rule discovery adopts a real-encoding way, which can be applied to both categorical and continuous attributes, as demonstrated in the experiment on Zoo (categorical attributes) and Wine (continuous attributes) data sets.

2 Rule Discovery with PSO

Particle Swarm Optimization (PSO), a new population-based evolutionary computation technique inspired by social behavior simulation, was first introduced in 1995 by Eberhart and Kennedy [9]. PSO is an efficient and effective global optimization algorithm, which has been widely applied to nonlinear function optimization, neural network training, and pattern recognition. In PSO, a swarm consists of N particles moving around in a D-dimensional search space.The position of the i-th particle at the t-th iteration is represented by $X_i^{(t)} = (x_{i1}, x_{i2}, \ldots, x_{iD})$ that are used to evaluate the quality of the particle. During the search process the particle successively adjusts its position toward the global optimum according to the two factors: the best position encountered by itself (*pbest*) denoted as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and the best position encountered by the whole swarm (*gbest*) denoted as $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$. Its velocity at the t-th iteration is represented by $V_i^{(t)} = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The position at next iteration is calculated according to the following equations:

$$V_i^{(t)} = \lambda(\omega * V_i^{(t-1)} + c_1 * rand() * (P_i - X_i^{(t-1)}) + c_2 * rand() * (P_g - X_i^{(t-1)})) \quad (1)$$

$$X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)} \quad (2)$$

where c_1 and c_2 are two positive constants, called cognitive learning rate and social learning rate respectively; $rand()$ is a random function in the range $[0,1]$; w is $inertiafactor$; and λ is $constrictionfactor$. In addition, the velocities of the particles are confined within $[Vmin, Vmax]^D$. If an element of velocities exceeds the threshold $Vmin$ or $Vmax$, it is set equal to the corresponding threshold.

In this paper, our PSO algorithm for rule discovery follows Michigan approach where an individual encodes a single prediction rule. Each run only one rule can be discovered. In order to get a set of rules, the algorithm must be run several times. All possible values of classification attributes are assigned one by one to PSO algorithm. Each run of PSO algorithm has a fixed value of classification attribute. So to get a rule set, the algorithm must be run at least K times if we want to predict K different classes.

2.1 Rule Presentation

The particle is concatenation of real-valued elements in the range $[0,1]$, which is divided into three parts as shown in Figure 1. If there are m decision attributes, each part has m elements respectively. So the size of a particle is $3m$. In order to form a rule according to a particle, three parts are translated into the original

Attribute-existence-array	Operator-array	Attribute-array

Fig. 1. Structure of a particle

information: (1) presence of attributes, (2) operators between attributes and their values, and (3) original values of attributes based on their types in the data set. If the i-th elements in *Attribute-existence-array* is greater than 0, then the i-th attribute is present in rule antecedent, else the i-th attribute is absent. Attributes in data mining tasks are often of several types: categorical or continuous types. So PSO algorithm for rule discovery must provide a way to encode these two types of attributes. For *Operator-array*, first the types of attributes must be considered. When the attribute is continuous, if the i-th elements in *Operator-array* is great than 0, then the operator is '$\geq$' else it is '$<$'; when the attribute is categorical (integer or nominal), if the i-th element is greater than 0, then the operator is '$=$' else it is '$!=$'. While the translation of *Attribute-array* is relatively complex because the different types of the attributes must be considered.

– For integer type, the translation is as follows:

$$V_{org}[i] = ceil(v_i * (V_i max - V_i min) + V_i min) \tag{3}$$

– For real type, the translation is as follows:

$$V_{org}[i] = v_i * (V_i max - V_i min) + V_i min \tag{4}$$

– For nominal type, the translation is as follows:

$$V_{org}[i] = ValArr_i(ceil(v_i * Count_i)) \tag{5}$$

Where $V_{org}[i]$ means the value translated from the particle for the i-th attribute, v_i is the i-th value in the particle which is a real value. In Equations (3) and (4), the type of the i-th attribute is of integer or real, here $V_i max$ means the maximum of the i-th attribute, and $V_i min$ the minimum. In Equation (5), the type of the i-th attribute is nominal, here $ValArr_i$ means the array that stored every different nominal values of the i-th attribute and $Count_i$ means the total number of different values of the i-th attribute. $ceil()$ is a function that returns the value round towards plus infinity.

2.2 Rule Evaluation (Fitness Function Design)

Let a rule be of such form: IF A THEN C

Where A is a rule antecedent, which is a conjunction of conditions, and C is a rule consequent which is the prediction class. The accuracy of classification rule is defined in Equation (6) to measure the degree of confidence for rules. The coverage of a rule is defined in Equation (7) to interpret the proportion of the examples that satisfy the rule in all examples that are of C class.

$$Accuracy = TP/(TP + FP) \tag{6}$$

$$Coverage = TP/(TP + FN) \tag{7}$$

TP = Number of examples satisfying A and C
FP = Number of examples satisfying A but not C.
FN = Number of examples not satisfying A but satisfying C.
TN = Number of examples not satisfying A nor C.

The comprehensibility of a rule set is often presented by less number of conditions in the rule antecedent and less rules in the rule set. In the process of rule discovery, we use Equation (8) to assure the shortness of a rule at the same time.

$$Succinctness = 1 - (countAnt - 1)/attributeCount \tag{8}$$

Where *countAnt* means the number of the conditions in the rule antecedent, *attributeCount* means the number of decision attributes in the data set. To present the three criterions, as a result, we define our fitness function as follows:

$$Fitness = w_1 * (Accuracy * Coverage) + w_2 * Succinctness + w_3 * Interesting \tag{9}$$

Where w_1, w_2, and w_3 are constants used to balance the weights of the three criterions in the rule discovery process. Since *Interesting* is highly dependent on users, in this paper we did not consider this criterion. In our experiment, w_1,w_2,and w_3 were set to 0.8, 0.2, and 0, respectively.

3 Experiments

3.1 Data Sets and Experiments Setup

The data sets, Zoo and Wine, were obtained from UCI repository of Machine Learning databases [10]. The Zoo data set with 18 categorical attributes was divided into 7 classes. The Wine data set with 13 continuous attributes was divided into 3 classes. To evaluate both the accuracy and generalization of the discovered rules, two different kinds of experiments were conducted on each data set. In the first kind of experiment, the data set was divided into two parts: 2/3 as training set and 1/3 as test set. The division was randomly generated. The experiment was run 5 times, each time on a different division. Then the average results were generated based on the 5 independent runs. The purpose of the first kind of experiment was to evaluate the generalization ability, measured as the classification accuracy on test set. In the second kind of experiment, the full data set was used to discover the final rules reported to users. 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. Here, Linearly Decreasing Weight PSO (LDW-PSO) [11] was employed, where a weight w decreased linearly between 0.9 and 0.4; $\lambda = 1$; $Vmin = Xmin = 0$; $Vmax = Xmax = 1$. The learning rates were $c1 = c2 = 2$.

3.2 Results and Discussion

The results in the first kind of experiment were as follows. The accuracy values averaged over 5 runs on training set for 7 classes were 1, 1, 1, 1, 1, 1, 0.9131, while corresponding accuracy values on test set were 0.9548, 1, 0.6000, 0.9667, 0.7000, 0.7150, 0.8767. For Wine, the accuracy values on training set for 3 classes were 0.9705, 0.9448, 0.9707, while corresponding accuracy values on test set were 0.9378, 0.6559, 1. From the results, it seems that the algorithm has good performance of predictive accuracy and generalization ability not only on categorical attributes of Zoo data set but also continuous attributes of Wine data set. In the second kind of experiment, the final rules discovered from full Zoo and Wine data sets were listed in Tables 1 and 2 respectively. The best rules for each class, the number of examples covered by rule antecedent $\mid A \mid$, and the number of correctly predicted examples $\mid A\&C \mid$ were showed in the tables. The results indicate that not only the predictive accuracy is pretty good, but also the number of rule conditions is relatively short and there are a small number of rules in the rule set, so the rules are competitively comprehensible.

Table 1. The results of learning from the full Zoo data set (rules with less coverage are removed from the table)

Class	Rule	$\mid A\&C \mid$	$\mid A \mid$
1	if $milk = 1$ then $type = 1$	41	41
2	if $feathers = 1$ $toothed = 0$ then $type = 2$	20	20
3	if $hair = 0$ $feathers = 0$ $aquatic = 0$ $backbone = 1$ $legs! = 8$ then $type = 3$	4	4
4	if $milk = 0$ $fins = 1$ then $type = 4$	13	13
5	if $milk = 0$ $aquatic = 1$ $breathes = 1$ legs !=2 catsize =0 then $type = 5$	4	4
6	if $aquatic = 0$ $legs = 6$ then $type = 6$	8	8
7	if $airborne = 0$ $backbone = 0$ then $type = 7$	10	12

Table 2. The results of learning from the full Wine data set (rules with less coverage are removed from the table)

Class	Rule	$\mid A\&C \mid$	$\mid A \mid$
1	If $A7 \geq 2.29$ $A13 \geq 723$ then $Class = 1$	57	58
2	If $A1 < 12.81$ $A12 \geq 1.81$ then $Class = 2$	60	62
2	If $A2 < 2.35$ $A11 \geq 0.81$ $A13 < 758$ then $Class = 2$	8	8
3	If $A7 < 1.51$ $A10 \geq 3.97$ then $Class = 3$	46	47

4 Conclusions

We proposed a real-encoding way for rule discovery using PSO algorithm. The experiments show that this encoding way is effective and efficient. The rules discovered in the two datasets are generally with high accuracy, generalization and comprehensibility. Furthermore, the real-encoding way is competitive in discovering rules not only from categorical data, but also from continuous data. The results on the Wine data set show that our approach has good performance for rule discovery on continuous data.

References

1. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases: An overview. Advances in Knowledge Discovery and Data Mining (1996) 1–34
2. Noda, E., Freitas, A.A., Lopes, H.S.: Discovering interesting prediction rules with a genetic algorithm. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress on Evolutionary Computation. Volume 2., Mayflower Hotel, Washington D.C., USA, IEEE Press (1999) 1322–1329
3. Jong, K.A.D., Spears, W.M., Gordon, G.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
4. Bojarczuk, C.C., Lopes, H.S., Freitas, F.: Discovering comprehensible classification rules using genetic programming: a cas study in a medical domain. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99, San Francisco, CA, Morgan Kaufmann (1999) 953–958
5. Freitas, A.A.: A genetic programming framework for two data mining tasks: Classification and generalized rule induction. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L., eds.: Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford University, CA, USA, Morgan Kaufmann (1997) 96–101
6. De Falco, I., Della Cioppa, A., Tarantino, E.: Discovering interesting classification rules with genetic programming. Applied Soft Computing **1** (2001) 257–269
7. Sousa, T., Neves, A., Silva, A.: Swarm optimisation as a new tool for data mining. In: 17th International Parallel and Distributed Processing Symposium (IPDPS-2003), Los Alamitos, CA, IEEE Computer Society (2003) 144–144
8. Sousa, T., Neves, A., Silva, A.: A particle swarm data miner. In: 11th Portuguese Conference on Artificial Intelligence, Workshop on Artificial Life and Evolutionary Algorithms. (2003) 43–53
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceeding of IEEE International Conference on Neural Networks (ICNN'95). Volume 4., Perth, Western Australia, IEEE (1995) 1942–1947
10. Blake, C., Merz, C.J.: UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998)
11. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ, IEEE (1998) 69–73