Improving Tree-Based Classification Rules Using a Particle Swarm Optimization

Chi-Hyuck Jun*, Yun-Ju Cho, and Hyeseon Lee

Department of Industrial and Management Engineering
Pohang University of Science and Technology, Pohang, 790-784, Korea
{chjun,jyj0729,hyelee}@postech.ac.kr

Abstract. The main advantage of tree classifiers is to provide rules that are simple in form and are easily interpretable. Since decision tree is a top-down algorithm using a divide and conquer induction process, there is a risk of reaching a local optimal solution. This paper proposes a procedure of optimally determining the splitting variables and their thresholds for a decision tree using an adaptive particle swarm optimization. The proposed method consists of three phases – tree construction, threshold optimization and rule simplification. To validate the proposed algorithm, several artificial and real datasets are used. We compare our results with the original CART results and show that the proposed method is promising for improving prediction accuracy.

Keywords: Classification, Data mining, Decision tree, Particle swarm optimization.

1 Introduction

Decision tree is a popularly used data mining tool for classification and prediction. A decision tree usually handles one variable at a time and so it provides the classification rule that is easily interpreted. Decision tree has been widely used in many areas such as medical diagnosis, customer churn prediction, quality prediction and so on. Among many algorithms, CART (Breiman et al. 1984) and C4.5 (Quinlan, 1993) are the most famous and popularly used ones.

Over the years, however, some studies show that constructing the optimal decision tree belongs to a non-deterministic polynomial-time complete problem (Hyafil and Rivest, 1976; Naumov, 1991). Therefore, most algorithms are greedy top-down approach (Lior and Oded, 2005). When the algorithm searches for the splitting criteria, it considers only one variable at a time without considering the potential interaction between other attributes or variables. This approach has a risk of reaching a local optimal solution. To overcome this problem, multivariate splitting criteria are proposed by many researchers. However, the problem of finding the optimal linear split is more difficult and intractable than that of finding the optimal univariate split (Murthy, 1998; Lior and Oded, 2005). In the problem of finding the optimal univariate

* Corresponding author.

C. Emmanouilidis, M. Taisch, D. Kiritsis (Eds.): APMS 2012, Part II, IFIP AICT 398, pp. 9–16, 2013.

split, Athanasios and Dimitris (2001) proposed the use of a genetic algorithm to directly evolve classification decision trees. The computational burden for genetic algorithms is substantially bigger than that for other algorithms. Saher and Shaul (2007) introduced a framework for anytime induction of decision tree, which generates the best answer up to the allowed time. However, it also suffers from the computational burden. Recently, Cho et al. (2011) developed an optimization procedure of a decision tree using a particle swarm. But, they only optimize threshold values of variables chosen by the CART.

If we combine an existing efficient decision tree algorithm and an evolutionary optimization algorithm, computational burden can be reduced and more efficient decision tree can be obtained. This paper extends the work by Cho et al. (2011) by considering an iterative way of finding better combination of variables and thresholds. The proposed algorithm consists of three phases; constructing a decision tree, optimizing the decision tree and rule simplification. First phase is to construct a preliminary decision tree by using well-known CART (classification and regression tree) proposed by Breiman et al. (1984). Second phase is to optimize the preliminary decision tree by determining the optimal thresholds of splitting variables using an adaptive particle swarm optimization (PSO). The last phase is to simplify the rules by combining the regions having same class.

2 Adaptive Particle Swarm Optimization

Particle swarm optimization (PSO) is one of evolutionary computation techniques developed by Kennedy and Eberhart (1995). As birds fly through a three-dimensional space, this algorithm makes use of particles moving in n-dimensional space to search for the optimal solution. PSO is a population-based iterative algorithm. The population consists of many particles, where each particle represents a candidate solution and moves toward the optimal position by changing its position according to a velocity.

Let x_{ik}^{t} and v_{ik}^{t} be the position and the velocity, respectively, in the k-th variable of the i-th particle at the t-th iteration. Then, the updating formula for the position and the velocity of the i-th particle are as follows.

$$x_{ik}^{t+1} = x_{ik}^{t} + v_{ik}^{t+1} \tag{1}$$

$$v_{ik}^{t+1} = w \times v_{ik}^{t} + c_1 \times rand_1 \times (pbest_{ik}^{t} - x_{ik}^{t}) + c_2 \times rand_2 \times (gbest_{k}^{t} - x_{ik}^{t}) \tag{2}$$

where $rand_1$ and $rand_2$ are random numbers between 0 and 1. Here, $pbest_{ik}^{t}$ is the previous best position in the k-th variable of the i-th particle and $gbest_{k}^{t}$ is the previous best position in the k-th variable among the entire population. We may omit the subscripts and the superscripts later if there is no risk of confusion. The parameter w is the inertia weight, and c_1 and c_2 are called acceleration coefficients which in the directions of pbest and gbest, respectively.

All the particles are evaluated by the fitness function at every iteration. The fitness function is used to evaluate the quality of the candidate solution. In the classification

problem, accuracy, or mixture of sensitivity and specificity is usually used as the fitness function.

Like other evolutionary computation algorithms, the PSO requires some parameters as inputs. The size of the population, inertia weight (i.e., w), acceleration coefficients (i.e., c_1 and c_2), and the maximum number of iterations are parameters. To overcome the difficulty of parameter selection, Clerc (1999) first proposed an adaptive process. Several papers report that adaptive PSO enhances the performance of PSO in terms of convergence speed, global optimality, solution accuracy, and algorithm reliability (Clerc, 1999; Shi and Eberhart, 2001; Xie et al. 2002). In this paper, the algorithm developed by Zhan et al. (2009) is used for optimizing the decision tree. It identifies the current population state and proposes a control strategy of w, c_1 and c_2. The current population state is classified into four evolutionary state based on the evolutionary factor: exploration, exploitation, convergence, and jumping out. Before defining the evolutionary factor, the mean distance of the i-th particle to all other particles at each iteration is calculated by

$$d_i = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{k=1}^{D}\left(x_{ik}-x_{jk}\right)^2} \tag{3}$$

where D is the number of variables and N is the number of population. Then, the evolutionary factor is obtained by

$$f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \in [0,1] \tag{4}$$

where d_g is the mean distance of the globally best particle, d_{max} is the maximum and d_{min} is the minimum mean distance. The evolutionary state will be assigned by the fuzzy membership function and the control strategy for w, c_1 and c_2 is determined according to the rules of each evolutionary state in Zhan et al. (2009).

3 Proposed Algorithm

As mentioned before, most decision tree algorithms are greedy approach. They usually consider only one variable at a time when growing the tree. This approach may reach a local optimal solution. Most desirably, splitting criteria should be searched simultaneously. However, searching every possible combination requires a huge computation time. To avoid it, a three-phase hybrid procedure is proposed. First phase is to construct a preliminary decision tree by CART and to determine the splitting variables. But, the threshold values from CART are not used in the proposed method. Only the number of threshold values will be maintained and the threshold values of each splitting variable will be re-determined in the second phase.

Second phase is to optimize the preliminary decision tree by determining the optimal thresholds of splitting variables simultaneously using the adaptive PSO in Zhan et al. (2009). The positions of particles in the adaptive PSO represent thresholds of

splitting variables. However, there are some differences in the way of classification between CART and the proposed method using the adaptive PSO.

Suppose that there are K splitting variables involved in the preliminary tree and that there are J_k threshold values for the k-th splitting variable (k=1,…,K). Then, the number of variables to be considered in the proposed method is

$$D = \sum_{k=1}^{K} J_k \tag{5}$$

Also, the total number of cells to be partitioned in the adaptive PSO is $(J_1 + 1)$ $(J_2 + 1)$ ... $(J_K + 1)$. The training data should be partitioned accordingly and the class prediction should be made in each cell. The class of a cell is predicted as the one having the largest observations among the training data partitioned. If there is no training data partitioned in a cell, predicted class may be determined randomly.

Basically, we may draw a separate classification rule for each cell partitioned in the proposed method. But, the classification rules can be simplified if some adjacent cells have the same predicted class. The third phase performs this. Based on this simplified rule, a new decision tree can be drawn if needed. It should be noted that the new decision tree may not have the same structure as the preliminary tree. The proposed algorithm can be summarized as follows.

Phase 1. Construct a preliminary decision tree by CART. Breiman et al. (1984) provide the detailed algorithm of CART, which consists of two steps.

Step 1-1. Grow the tree by splitting variables based on Gini impurity function until every node has a single (pure) class.
Step 1-2. Prune the tree based on cost-complexity and select the best pruned tree to obtain the preliminary tree.

Phase 2. Optimize the preliminary decision tree by the adaptive PSO.

Step 2-0. Initialize the positions and the velocities of particles at random. Set the initial gbest to the thresholds of the preliminary decision tree.
Step 2-1. Evaluate particles according to the selected fitness function.
Step 2-2. If the current fitness value of each particle is better than pbest, then update pbest value. If the current fitness value of population's overall best is better than gbest, then update gbest value.
Step 2-3. Update the parameters adaptively.
Step 2-4. Change the velocity and position of the particles according to the equations (1) and (2).
Step 2-5. If the maximum number of iterations is reached, stop. Otherwise, go to Step 2-1.

Phase 3. Simplify the rules by combining the adjacent cells having the same class.

Step 3-1. Represent each partitioning cell in terms of binary D-digits. Sort all cells in the ascending order of digits.

Step 3-2. Starting from the smallest digit cell, find the adjacent cells having the same class.
Step 3-3. Combine the adjacent cells having the same class
Step 3-4. If there are no remaining cells for combining, stop. Otherwise, go to Step 3-1.

Note that the Phase 1 and Phase 2 may be repeated several times by selecting different variables to generate various trees.

4 Experiments and Results

To validate the proposed algorithm, numerical experiments are performed with an artificial data and a real data. Each data set is separated into two: the two thirds of the observations are used as a training set and the rest of one thirds are set aside as the test set. The number of particles and the maximum number of iterations are set to 100 and 500, respectively in the adaptive PSO. The initial values of (w, c_1, c_2) are set to (0.9, 2, 2). We compare our results against the original CART results using 5-fold cross validation to see the performance improvement through the proposed optimization method. In the proposed method, we use the classification accuracy as the fitness function.

4.1 Artificial Data Sets

The artificial data sets having three classes and three attributes are generated from 3-dimensional multivariate normal distributions. Three different mean vectors and variance-covariance matrices are chosen as in Table 1. Three cases having different number of observations and different value of σ^2 are considered. Case 1 includes 100 observations for class 1, 150 observations for class 2, and 50 observations for class 3 (total of 300 observations), whereas σ^2 is set to 0.1. Case 2 includes 500 for class 1, 750 for class 2, and 250 observations for class 3 (total of 1500), whereas σ^2 is again set to 0.1. Case 3 is composed of the same class observations as Case 1, where σ^2 is set to 0.2.

Table 1. Mean vectors and variance-covariance matrices for generating artificial data

	Class 1	Class 2	Class 3
Mean	$\begin{bmatrix} 0.1 \\ -1.0 \\ 0.3 \end{bmatrix}$	$\begin{bmatrix} 1.5 \\ 0.1 \\ 1.2 \end{bmatrix}$	$\begin{bmatrix} -0.5 \\ 1.5 \\ 0.5 \end{bmatrix}$
Variance-Co variance	$\sigma^2 * \begin{bmatrix} 2 & 0 & 2 \\ 0 & 4 & 1 \\ 2 & 1 & 3 \end{bmatrix}$	$\sigma^2 * \begin{bmatrix} 4 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 3 & 4 \end{bmatrix}$	$\sigma^2 * \begin{bmatrix} 5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}$

Table 2 compares the classification accuracies of the CART and the proposed procedure in the training and test sets for each of three cases. We randomly selected 2/3 of observations as the training set and the rest 1/3 as the test set. In fact, each number

in a cell represents the average over 50 repetitions of experiments. It shows that the proposed method generally improves the classification accuracy as compared with CART. When the number of observations is increased as in Case 2 or σ^2 is increased as in Case 3, the proposed method outperforms the CART. The results are not reported here, paired t-tests were performed, which showed significantly difference in accuracies between the proposed method and the CART.

Table 2. Comparison of Accuracy for Training and Test Data (in percentage)

	Case 1		Case 2		Case 3	
	train	test	train	test	train	test
CART	95.86	**93.41**	92.57	89.61	89.28	81.52
Proposed	**97.43**	92.59	**96.18**	**93.32**	**94.53**	**83.98**

4.2 Real Data Sets

Three real data sets, that is, Parkinson's, Pima Indians Diabetes (Diabetes), and Blood Transfusion Service Center (Blood), were chosen for the application, which are available from UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ ml/). Table 3 shows the outline of these data sets.

Table 3. Outline of Three Real Data Sets

	Parkinson's	Diabetes	Blood
# classes	2	2	2
# observations	195	393	748
in training set	130	262	499
in test set	65	131	249
# attributes	22	8	4

Table 4. Classification Accuracy in Three Data Sets

	Parkinson's	Diabetes	Blood
Proposed method			
Training set	**100**	**91.6**	**83.0**
Test set	**89.2**	**72.5**	**77.1**
CART			
Training set	96.9	86.6	81.2
Test set	84.6	67.9	76.7

Table 4 shows the classification accuracy for each of the above three data sets. Performance of the proposed method is compared with that of the CART. We clearly see that the proposed method outperform the CART in all data sets. Particularly for Parkinson's and Diabetes data sets the proposed method significantly outperforms the CART.

5 Concluding Remarks

In this paper, we proposed the three-phase procedure of optimizing the decision tree. Combining the CART algorithm and an adaptive PSO as the optimization tool, computational burden can be reduced and an improved decision tree can be obtained with the increased the classification performance. Particularly, we consider an iterative way of finding optimal combination of variables and thresholds as well as rule simplification. The performance was demonstrated through numerical experiments with artificial and some real data sets.

The proposed method can be applied to many quality classification problems in the area of production management. The quality of products from a manufacturing process can be classified into several grades and so a classification model can be built using the operation data from the manufacturing process. Using this type of model we can predict the quality of the final product in advance.

To improve the decision tree further, we may optimize the fully grown tree instead of a pruned if the computation cost is not severe. After optimizing the fully grown tree, we may eliminate redundant splitting criteria and prune the decision tree to get the user-specific size of tree. Furthermore, other variable selection method can be adopted when generating the preliminary tree instead of CART. If we can get a good set of variables critical to predict classes, then this will lead to a better classification rule. In this sense, combining random forest and the adaptive PSO may be a good alternative. Similarly, other optimization algorithm such as genetic algorithm, ant colony optimization, or simulated annealing, may be used. Optimization of decision tree having multiway splits is also an interesting future area for study.

Acknowledgement. This work was supported by Basic Science Research Program through the NRF funded by the MEST. (Project No. 2011-0012879)

References

1. Athanasios, P., Dimitris, K.: Breeding Decision Trees Using Evolutionary Techniques. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 393–400 (2001)
2. Breiman, L., Friedman, J.H., Olashen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC, London (1984)
3. Cho, Y.-J., Lee, H., Jun, C.-H.: Optimization of Decision Tree for Classification Using a Particle Swarm. Industrial Engineering & Management Systems 10, 272–278 (2011)
4. Clerc, M.: The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1951–1957 (1999)
5. Hyafil, L., Rivest, R.L.: Constructing Optimal Binary Decision Trees is NP-Complete. Information Processing Letters 5, 15–17 (1976)
6. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)

7. Lior, R., Oded, M.: Top-Down Induction of Decision Trees Classifiers-A Survey. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews 35, 476–487 (2005)
8. Murthy, S.K.: Automatic Construction of Decision Trees from Data: A Multidisciplinary Survey. Data Mining and Knowledge Discovery 2, 345–389 (1998)
9. Naumov, G.E.: NP-Completeness of Problems of Construction of Optimal Decision Trees. Soviet Physics 36, 270–271 (1991)
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
11. Saher, E., Shaul, M.: Anytime Learning of Decision Trees. Journal of Machine Learning Research 8, 891–933 (2007)
12. Shi, Y., Eberhart, R.C.: Fuzzy Adaptive Particle Swarm Optimization. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 101–106 (2001)
13. UCI (University of California - Irvine) data repository: University of California, Irvine. Center for Machine Learning and Intelligent Systems, http://archive.ics.uci.edu/ml/
14. Xie, X.F., Zhang, W.J., Yang, Z.L.: Adaptive Particle Swarm Optimization on Individual Level. In: International Conference of 2002 6th on Signal Processing, pp. 1215–1218 (2002)
15. Zhan, Z.H., Jun, Z., Yun, L., Chung, H.S.: Adaptive Particle Swarm Optimization. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 39, 1362–1381 (2009)