IEMS Vol. 10, No. 4, pp. 272-278, December 2011.

Optimization of Decision Tree for Classification Using a Particle Swarm

Yun-Ju Cho
Department of Industrial and Management Engineering
Pohang University of Science and Technology, Pohang, 790-784, Korea
Tel: +82-54-279-2717, E-mail: jyj0729@postech.ac.kr

Hyeseon Lee
Department of Industrial and Management Engineering
Pohang University of Science and Technology, Pohang, 790-784, Korea
Tel: +82-54-279-2717, E-mail: hyelee@postech.ac.kr

Chi-Hyuck Jun†
Department of Industrial and Management Engineering
Pohang University of Science and Technology, Pohang, 790-784, Korea
Tel: +82-54-279-2717, E-mail: chjun@postech.ac.kr

Received, September 4, 2011; Revised, October 20, 2011; Accepted, October 31, 2011

Abstract. Decision tree as a classification tool is being used successfully in many areas such as medical diagnosis, customer churn prediction, signal detection and so on. The main advantage of decision tree classifiers is their capability to break down a complex structure into a collection of simpler structures, thus providing a solution that is easy to interpret. 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 thresholds of the chosen variables for a decision tree using an adaptive particle swarm optimization (APSO). The proposed algorithm consists of two phases. First, we construct a decision tree and choose the relevant variables. Second, we find the optimum thresholds simultaneously using an APSO for those selected variables. 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 algorithm is promising for improving prediction accuracy.

Keywords: Classification, Data Mining, Decision Tree, Particle Swarm Optimization

1. INTRODUCTION

Decision tree is a data mining tool for classification and prediction. The main advantage of decision tree classifiers is their capability to break down a complex structure into a collection of simpler structures, thus providing a solution that is easy to interpret. The classifying rule is in forms of 'IF-THEN' rule, so it is useful to identify the key variables in the analysis. Decision tree has been widely used until now in many areas such as medical diagnosis, customer churn prediction, signal detection and so on. There are many algorithms to construct decision trees from the given data. CHAID (Kass, 1980), CART (Breiman *et al.*, 1984), C4.5 (Quinlan, 1993) are the most famous and widely used ones.

Over the years, however, theoretical studies show that constructing the optimal decision tree belongs to a non-deterministic polynomial-time complete problem (Hyafil and Rivest, 1976; Naumov, 1991). It means that the optimal decision tree cannot be obtained in polynomial time by any algorithm. 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 while ignoring 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. Since Bre-

† : Corresponding Author

iman *et al.* (1984) first suggested the use of linear combination of input attributes as the splitting criteria, most of multivariate splitting criteria are based on linear combination of attributes induced by linear discriminant analysis, linear programming, perceptron learning and others (Duda and Hart, 1973; Bennett and Mangasarian, 1994; Utgoff, 1989; Schuermann and Doster, 1984). Even though there are many efforts to find the optimal multivariate splitting criteria, 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 split, Athanasios and Dimitris (2001) proposed the use of a genetic algorithm to directly evolve classification decision trees. But genetic algorithm has computational complexity problem. The computational burden is substantially bigger than that of other algorithms. Saher and Shaul (2007) introduce a framework for anytime induction of decision tree. Anytime algorithm is a family of algorithm which can generates the best answer that has been searched up to the allowed time. It trades computation speed for quality and yields better decision trees when more time is available. The property of anytime algorithms is similar with that of genetic algorithm. Therefore, it also suffers from the computational burden.

However, 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. The proposed algorithm consists of two phases; constructing a decision tree and optimizing the decision tree. 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 (APSO).

In the next section, particle swarm optimization (PSO) and adaptive particle swarm optimization (APSO) are briefly reviewed. Then, in Section 3, the proposed algorithm is described in detail and it is validated via several experiments in Section 4. Finally, conclusions are drawn in Section 5.

2. PARTICLE SWARM OPTIMIZATION

2.1 Conventional Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart (1995). It simulates a collective behavior in birds flocking flying for cornfield. As birds fly through a three-dimensional space, searching for food or evading predators, algorithm make use of particles moving in n-dimensional space to search for solutions for an n-variable function optimization problem. Many research use PSO to find the optimal solutions for the given real-world optimization problems (Kennedy and Eberhart, 1995; Eberhart and Shi, 2001).

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. The velocity of a particle is calculated by three components; inertia, cognitive, and social components. The inertia component simulates the movement of a bird to fly from the previous direction. The cognitive component represents the memory of a bird about its previous best position. The social component models the communication of a bird about the best position among all birds. 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 of formula for the position and the velocity of the i-th particle is as follows.

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

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

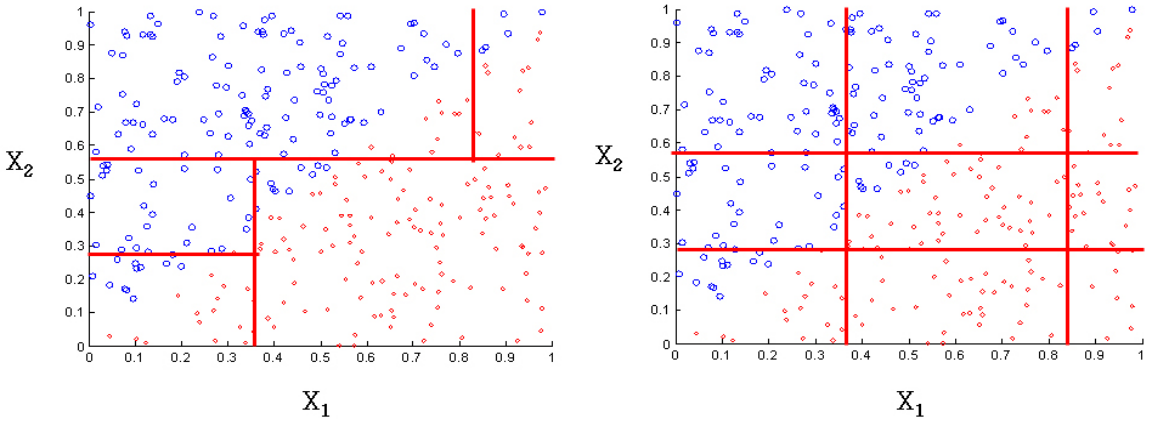
where $rand_1$ and $rand_2$ are random numbers between 0 and 1. Here, $pbest_{ik}^t$ is the previous best positionin the k-th variable of the i-th particle and $gbest_k^t$ is the previous best position in the k-th variable among the entire population. We may omit the subscripts and the superscripts later 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 *pbes*t and *gbest* respectively.

All of 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. Among these parameters, inertia weight and acceleration coefficients have a large impact on the algorithm performance. One variant version of PSO, adaptive particle swarm optimization (APSO), improves the search efficiency by adjusting w, c_1 and c_2 adaptively.

2.2 Adaptive Particle Swarm Optimization (APSO)

To overcome the difficulty of parameter selection, Clerc (1999) first proposed an adaptive process. Many researchers also have suggested other types of adaptive process. Several papers report that APSO enhances the performance of PSO in terms of convergence speed,



(a) Preliminary tree (CART) (b) Proposed method

Figure 1. Example of Cell Partitioning in Attributes-Space.

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

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

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. When the algorithm searches for the splitting criteria, it considers only one variable at a time. This approach has a risk of reaching a local optimal solution. Actually, splitting criteria should be searched simultaneously. However, searching every possible combination requires a huge computation time. To avoid it, a two-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 APSO in Zhan *et al.* (2009). The positions of particles in the APSO represent thresholds of splitting variables. However, there are some differences in the way of classification between CART and the proposed method using the APSO.

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 APSO is $(J_1+1)(J_2+1)\cdots(J_K+1)$. 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. Figure 1 shows an example, where two splitting variables (X_1 and X_2) are involved and there are two threshold values for each variable. As the result, the proposed method needs to predict the class for each of nine cells altogether.

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 APSO.
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.
2-3-1: At the current position, calculate the mean distance of each particle to all the other particlesusing the equation (3).
2-3-2: Calculate the evolutionary factor *f* by using the equation (4).
2-3-3: Classify *f* into one of the four sets; S_1, S_2,S_3 and S_4, which represent the state of exploration, exploitation, convergence, and jumping out, respectively.
2-3-4: According to the state, adjust *w,* c_1, and c_2 using the control strategies provided in Zhan *et al*. (2009).
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.

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. 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.

In addition, the total number of cells to be partitioned will be increased by using the adaptive PSO. This is because there are no splitting orders in the proposed method. Since the CART uses top-down approach, every splitting variable of the preliminary decision tree has the splitting order. On the other hand, the adaptive PSO treats every splitting variable equally in the second phase. This is why the new decision tree may not have the same structure as the preliminary tree.

4. EXPERIMENTS

To validate the proposed algorithm, numerical experiments are performed with artificial data and 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 50 and 500, respectively in the APSO. 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 Experiments with Artificial Data

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 chosen are shown 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	$\begin{bmatrix} 0.1 \\ -1.0 \\ 0.3 \end{bmatrix}$	$\begin{bmatrix} 1.5 \\ 0.1 \\ 1.2 \end{bmatrix}$	$\begin{bmatrix} 0.3 \\ 0.9 \\ 0.05 \end{bmatrix}$
Variance-Covariance	$\sigma^2 \times \begin{bmatrix} 5 & 1 & 2 \\ 1 & 9 & 3 \\ 2 & 3 & 6 \end{bmatrix}$	$\sigma^2 \times \begin{bmatrix} 5 & 3 & 1 \\ 3 & 4 & 5 \\ 1 & 5 & 4 \end{bmatrix}$	$\sigma^2 \times \begin{bmatrix} 9 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 6 \end{bmatrix}$

To illustrate how the threshold values become different by the proposed method, Table 2 shows the initial threshold values obtained from CART and the optimized valued by the proposed method for Case 1. Although the results are not reported here, the final classification rules are quite different from the initial ones.

Table 2. Threshold Values of Splitting Variables for the Artificial Data Set (Case 1).

	X1, 1st threshold	X1, 2nd threshold	X2	X3
CART	0.6748	1.3770	0.4859	-0.1579
Proposed	0.7425	1.3000	0.5309	1.6481

Table 3. Comparison of Accuracy for Training and Test Data (in percentage).

	Case 1		Case 2		Case 3	
	Train	Test	Train	Test	Train	Test
CART	86.0	79.0	86.1	82.8	75.0	66.0
Proposed	87.0	82.0	86.3	83.4	75.5	67.0

Table 4. Average Accuracy from 5-fold Cross Validation (in percentage).

	Case 1	Case 2	Case 3
CART	82.8	81.6	68.2
Proposed	83.8	83.9	70.4

Table 3 compares the classification accuracy of the CART and the proposed procedure in the training and test sets for each of three cases. It shows that the proposed method improves the classification accuracy. To compare the performance more precisely, 5-fold cross validation results are also reported in Table 4. It also shows that the proposed algorithm performs better than 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.

Even though the accuracy is close to each other, we can guarantee that the classification accuracy of the proposed procedure will be at least as good as CART because the initial *gbest* is set to the thresholds of the preliminary decision tree. However, as the iteration goes on, the adaptive PSO searches threshold values better than those of the preliminary decision tree.

4.2 Experiments with Real Data Set

The real data set under analysis is about the diagnosis of diabetes, which is available from UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/). The data (called diabetes data) include 393 observations having two classes (class 0: normal, class 1: diabetes) on 8 attributes (X1, ⋯, X8) after excluding some observations with missing values.

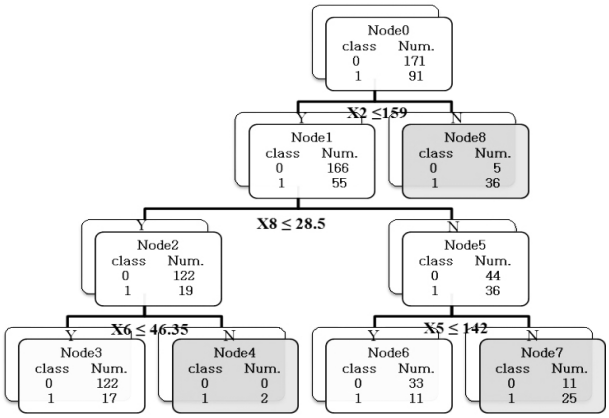


Figure 2. Preliminary Tree for the Diabetes Data.

Figure 2 is the preliminary tree from CART. As shown here, the splitting variables chosen are X2, X5, X6 and X8. After optimizing this decision tree using the APSO, threshold values of splitting variables remain the same. Even though these values are not changed, the classification rules are changed as show edin Table 5. From CART, there is only one rule if X2 > 159. But, based on the proposed method, this one rule is divided into three rules. So, the classification accuracy has been improved marginally as reported in Table 6.

Table 6. Comparison of Classification Accuracy for the Diabetes Data (in percentage).

	Training data	Test data	5-fold Cross validation
CART	83.21	77.86	76.03
Proposed	84.35	77.86	76.49

Table 5. Comparison of Classification Rules for the Diabetes Data.

Rules from CART	
(Node 3) If X2 ≤ 159 and X8 ≤ 28.5 and X6 ≤ 46.35,	then class0 (normal).
(Node 4) If X2 ≤ 159 and X8 ≤ 28.5 and X6 > 46.35,	then class1 (diabetes).
(Node 6) If X2 ≤ 159 and X8 > 28.5 and X5 ≤ 142,	then class0.
(Node 7) If X2 ≤ 159 and X8 > 28.5 and X5 > 142,	then class1.
(Node 8) If X2 > 159,	then class1.
Rules from the proposed method	
(Node 3) If X2 ≤ 159 and X8 ≤ 28.5 and X6 ≤ 46.35,	then class0.
(Node 4) If X2 ≤ 159 and X8 ≤ 28.5 and X6 > 46.35,	then class1.
(Node 6) If X2 ≤ 159 and X8>28.5 and X5 ≤ 142,	then class0.
(Node 7) If X2 ≤ 159 and X8>28.5 and X5 > 142,	then class1.
(Node 9) If X2 > 159 and X6 ≤ 46.35,	then class1.
(Node 11) If X2 > 159 and X6 > 46.35 and X5 ≤ 142,	then class1
(Node 12) If X2 > 159 and X6 > 46.35 and X5 > 142,	then class0.

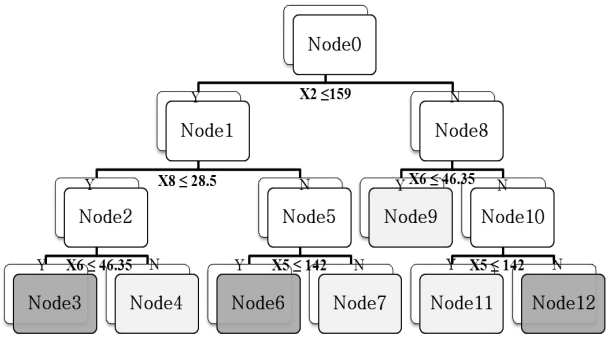


Figure 3. Optimized Decision Tree for the Diabetes Data.

We can express the rules from the proposed method shown in Table 5 in the form of decision tree as in Figure 3. It can be clearly seen that the Node 8 is divided into three terminal nodes (Node 9, Node 11, Node 12).As mentioned in the end of Section 3, the splitting orders in the preliminary decision tree are ignored in second phase. This is why the new decision tree does not have the same structure as the preliminary tree.

5. CONCLUSION

Decision tree is a widely used as data mining tool for classification and prediction. Since the classifying rule in forms of 'IF-THEN' rule is provided by decision tree, analysts can understand the result of decision tree easily. However, most decision tree algorithms consider one variable at a time to search splitting variables. This approach has a risk of reaching a local optimal solution. Splitting criteria should be searched simultaneously with more than one variable. In this paper, a new hybrid classification algorithm is proposed. Combining an existing decision tree algorithm and an APSO as the optimization tool, computational burden can be reduced and optimal decision tree can be obtained with the increased the classification performance.

Still, the APSO optimizes only threshold values of the selected variables. Once several variables are selected by CART, other variables cannot be considered in further analysis. It makes computational cost reduced, but the decision tree may be optimized locally. Nevertheless, the APSO can improve the prediction without the depth growing of decision tree. In this aspect, using fully grown tree instead of pruned tree can be a solution of this problem 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 feature selection method can be adopted when generating the preliminary tree instead of CART. If we can get a good set of features which are critical to predict classes, then thresholds of splitting variables can be achieved by the APSO. In this sense, combining random forest and the APSO may be a good alternative. Similarly, other optimization algorithm can be used instead of the APSO. Genetic algorithm, ant colony optimization, simulated annealing, or tabu search may be the candidates.

ACKNOWLEDGMENT

This research was supported with Basic Science Research Program through the National Research Foundation of Korea (NRF) from the Ministry of Education, Science and Technology (MEST) (Project No. 2011-00 12879). The work by Hyeseon Lee was supported with Basic Science Research Program through the NRF from the MEST (Project No. 2010-0003628).

REFERENCES

Athanasios, P. and Dimitris, K. (2001), Breeding Decision Trees Using Evolutionary Techniques,*Proceedings of the Eighteenth International Conference on Machine Learning*, 393-400.

Bennett, K. P. and Mangasarian O. L. (1994), Multicategory Discrimination via Linear Programming, *Optimization Methods and Software*, **3**, 29-39.

Breiman, L., Friedman, J. H., Olashen, R. A. and Stone, C. J. (1984), *Classification and Regression Trees*, Chapman and Hall/CRC, London, UK.

Clerc, M. (1999), TheSwarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization, *Proceedings of the 1999 Congress on Evolutionary Computation*, 1951-1957.

Duda, R. and Hart, P. (1973), *Pattern Classification and Scene Analysis*, A Wiley-Interscience Publication, New York.

Eberhart, R. C. and Shi, Y. (2001), Particle Swarm Optimization: Developments, Applications and Resources, *Proceedings of the 2001 Congress on Evolutionary Computation*, 81-86.

Frank, A. and Asuncion, A. (2010), UCI Machine Learning Repository (http://archive.ics.uci.edu/ml), Irvine, CA.

Hyafil, L. and Rivest, R. L. (1976), Constructing Optimal Binary Decision Trees is NP-Complete, *Information Processing Letters*, **5**, 15-17.

Kass, G. V. (1980), An Exploratory Technique for Investigating Large Quantities of Categorical Data, *Applied Statistics*, **29**, 119-127.

Kennedy, J. and Eberhart, R. C. (1995), Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, 1942-1948.

Lior, R. and Oded, M. (2005), Top-Down Induction of Decision Trees Classifiers-A Survey, *IEEE Transactions on Systems*, *Man*, *and Cybernetics-Part C:*

Applications and Reviews, **35**, 476-487.

Murthy, S. K. (1998), Automatic Construction of Decision Trees from Data: A Multidisciplinary Survey, *Data Mining and Knowledge Discovery*, **2**, 345-389.

Naumov, G. E. (1991), NP-Completeness of Problems of Construction of Optimal Decision Trees. *Soviet Physics*, **36**, 270-271.

Quinlan, J. R. (1993), *C4.5*: *Programs for Machine Learning*. Morgan Kaufmann Publishers Inc, San Francisco, CA.

Saher, E. and Shaul, M. (2007), Anytime Learning of Decision Trees, *Journal of Machine Learning Research*, **8**, 891-933.

Schuermann, J. and Doster, W. (1984), A Decision-theoretic Approach in Hierarchical Classifier Design, *Pattern Recognition*, **17**, 359-369.

Shi, Y. and Eberhart, R. C. (2001), Fuzzy Adaptive Particle Swarm Optimization, *Proceedings of the 2001 Congress on Evolutionary Computation*, 101-106.

Utgoff, P. E. (1989), Perceptron Trees: A Case Study in Hybrid Concept Representations, *Connection Science*, **1**, 377-391.

Xie, X. F., Zhang, W. J., and Yang, Z. L. (2002), Adaptive Particle Swarm Optimization on Individual Level, *International Conference of 2002 6th on Signal Processing*, 1215-1218.

Zhan, Z. H., Jun, Z., Yun, L. and Chung, H. S. (2009), Adaptive Particle Swarm Optimization, *IEEE Transactions on Systems*, *Man*, *and Cybernetics-Part B: Cybernetics*, **39**, 1362-1381.