Handling Continuous-Valued Attributes in Decision Tree with Neural Network Modeling

DaeEun Kim[1] and Jaeho Lee[2]

[1] Division of Informatics, University of Edinburgh
Edinburgh, EH1 2QL, United Kingdom
daeeun@dai.ed.ac.uk

[2] Department of Electrical Engineering, University of Seoul
Tongdaemoon-ku, Seoul, 151-011, Korea
jaeho@uofs.ac.kr

Abstract. Induction tree is useful to obtain a proper set of rules for a large amount of examples. However, it has difficulty in obtaining the relation between continuous-valued data points. Many data sets show significant correlations between input variables, and a large amount of useful information is hidden in the data as nonlinearities. It has been shown that neural network is better than direct application of induction tree in modeling nonlinear characteristics of sample data. It is proposed in this paper that we derive a compact set of rules to support data with input variable relations. Those relations as a set of linear classifiers can be obtained from neural network modeling based on back-propagation. This will also solve overgeneralization amd overspecialization problems often seen in induction tree. We have tested this scheme over several data sets to compare with decision tree results.

1 Introduction

Discovery of decision rules and recognition of patterns from data examples is one of challenging problems in machine learning. If data points contain numerical attributes, induction tree method needs a discretization of continuous-valued attributes with threshold values. Induction tree algorithms such as C4.5 build decision trees by recursively partitioning the input attribute space [14]. The tree traversal from the root node to each leaf leads to one conjunctive rule. Each internal node in decision tree has a splitting criterion or threshold for continuous-valued attributes to partition some part of the input space, and each leaf represents a class related to the conditions of each internal node.

Approaches based on decision tree involve the discretization of continuous-valued attributes in input space, making many rectangular divisions. As a result, it may have the inability to detect data trend or desirable classification surface. Even in the case of multivariate discretization methods which search at the same time for threshold values for more than one continuous attribute [4,12], the decision rules may not reflect data trend or the decision tree may build many rules with support of a small number of examples, often called over-specialization problem.

A possible way is suggested to catch the trend of data. It first tries to fit a given data set for the relationship between data points using a statistical technique, generates many data points on the response surface of the fitted curve, and then induces rules with induction tree. This method was introduced as an alternative measure against the problem of direct application of induction tree to raw data [9,10]. However, it still has a problem to need many induction rules to reflect the response surface.

In this paper we suggest to investigate a hybrid technique to combine neural networks and knowledge based systems for data classification. It has been shown that neural network is better than direct application of induction tree in modeling nonlinear charactersitics of sample data [3,13,15]. Neural networks have the advantage that they can deal with noisy, inconsistent and incomplete data. A method to extract symbolic rules from a neural network has been proposed to increase the performance of decision process [15]. They used in sequence a weight-decay back-propagation over a three-layer feedforward network, a pruning process to remove irrelevant connection weights, a clustering of hidden unit activations, and extraction of rules from discretized unit activations. Symbolic rules they derived from neural networks did not include input attribute relations. Also the direct conversion from neural networks to rules is related to exponential complexity when using search-based algorithm over incoming weights for each unit [5,16].

Our approach is to train a neural network with sigmoid functions and to use decision classifiers based on weight parameters of neural networks. Then induction tree selects the most relevant input variables and furthermore the desirable input variable relations for data classification. This algorithm is tested on various types of data and compared with the method based on decision tree alone.

2 Problem Statement

Induction tree is useful for a large number of examples, and it enables us to obtain proper rules from examples rapidly [14]. However, it has the difficulty in inferring relations between data points and cannot handle noisy data.

We can see a simple example of undesirable rule extraction discovered in induction tree application. Fig.1(a) displays a set of 29 original sample data with two classes. It appears that the set has four sections which have the boundaries of direction from upper-left to lower-right. A set of the dotted boundary lines is the result of multivariate classification by induction tree. It has six rules to classify data points. Even in C4.5 run, it has four rules with 6.9 % error, making divisions with attribute y. The rules do not catch data clustering completely in this example. Fig.1(b)-(c) show neural network fitting with back-propagation method. In Fig.1(b)-(c) neural networks have slopes $\alpha = 1.5, 4.0$ for sigmoids, respectively. After curve fitting, 900 points were generated uniformly on the response surface for the mapping from input space to class, and the response values of neural network were calculated as shown in Fig.1(d). The result of C4.5 application to those 900 points followed the classification curves, but produced 55 rules. The production of many rules results from the fact that decision tree makes piecewise rectangular division for each rule, even though the response surface for data clustering has correlation between input variables.

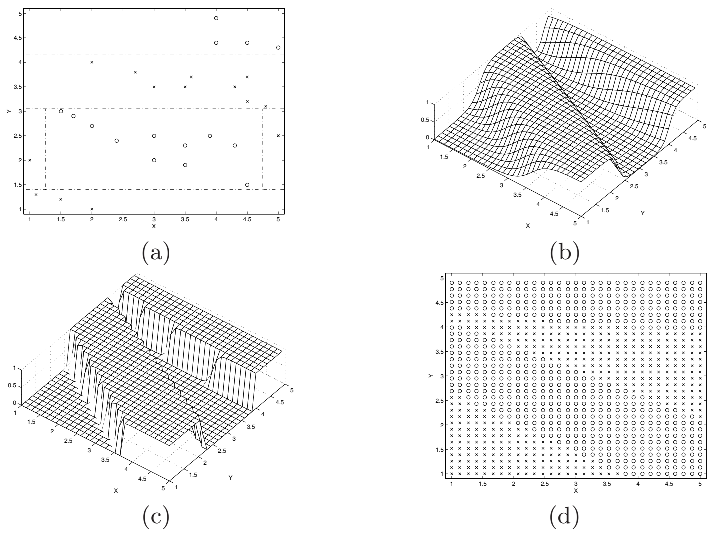

Fig. 1. Example (a) data set and decision boundary (O : class 1, X : class 0) (b)-(c) neural network fitting (d) data set with 900 points

As shown above, the decision tree has over-generalization problem for a small number of data and over-specialization problem for a large number of data. A possible suggestion is to consider or derive relations between input variables as another attribute for rule extraction. However, it is difficult to find input variable relations for classification directly in supervised learning, while unsupervised methods can use statistical methods such as principal component analysis [6].

3 Method

The goal for our approach is to generate rules following the shape and characteristics of response surface. Usually induction tree cannot trace the trend of data, and it determines data clustering only in terms of input variables, unless we apply other relation factors or attributes. In order to improve classification rules from a large training data set, we allow input variable relations for multi-attributes in a set of rules. We develop in this paper a two-phase method for rule extraction over continuous-valued attributes.

Given a large training set of data points, the first phase, as a feature extraction phase, is to train feed-forward neural networks with back-propagation and collect the weight set over input variables in the first hidden layer. A feature useful in inferring multi-attribute relations of data is found in the first hidden layer of neural networks. The extracted rules involving network weight values will reflect features of data examples and provide good classification boundaries. Also they may be more compact and comprehensible, compared to induction tree rules.

In the second phase, as a feature combination phase, each extracted feature for linear classification boundary is combined together using Boolean logic gates. In this paper, we use an induction tree to combine each linear classifier. From our results, it is shown that the two-phase method proposed is in general very effective and leads to solutions of high quality.

The highly nonlinear property of neural networks makes it difficult to describe how they reach predictions. Although their predictive accuracy is satisfactory for many applications, they have long been considered as a complex model in terms of analysis. By using expert rules derived from neural networks, the neural network representation can be more understandable. We use a neural network modeling with two hidden layers to obtain linear classification boundary. After training data patterns with neural network by back-propagation, we can have linear classifiers in the first hidden layer. To get desirable classifiers, we need to set sigmoid functions with high slope. It has been shown that a particular set of functions can be obtained with arbitrary accuracy by at most two hidden layers given enough nodes per layer [2]. Also one hidden layer is sufficient to represent any Boolean function [7]. Our neural network structure has two hidden layers, where the first hidden layer makes a local feature selection with linear classifiers and the second layer receives Boolean logic values from the first layer and maps any Boolean function. The second hidden layer and output layer can be thought of as a sum of product of Boolean logic gates. The n-th output of neural network for a set of data is $F_n = f(\sum_k^{N_2} W_{kn}^2 f(\sum_j^{N_1} W_{jk}^1 f(\sum_i^{N_0} W_{ij}^0 a_i)))$

For a node in the first hidden layer, the activation is defined as $f(\sum_i^{N_0} a_i W_{ik})$ for the k-th node where N_0 is the number of input attributes, a_i is an input, and $f(x) = 1.0/(1.0 + e^{-\alpha x})$ as a sigmoid function. When we train neural networks with back-propagation method, α, the slope of sigmoid function is increased as iteration continues. If we have a high value of α, the activation of each neuron is close to the property of digital logic gates, which has a binary value 0 or 1.

Except the first hidden layer, we can replace each neuron by logic gates if we assume we have a high slope on sigmoid function. Input to each neuron in the first hidden layer is represented as a linear combination of input attributes and weights, $\sum_i^N a_i W_{ik}$. This forms linear classifiers for data classification as a feature extraction over data distribution. When Fig.1(a) data is trained, we can introduce new attributes $aX + bY$ where a, b is a constant. We used two hidden layers with 4 nodes and 3 nodes, respectively, where every neuron node has a high slope of sigmoid to guarantee desirable linear classifiers as shown in Fig.1(c). Before applying data in Fig.1(d) to induction tree, we added four new attributes made from linear classifiers in the first hidden layer over 900 points and then we could obtain only four rules with C4.5, while a simple application of C4.5 for those data generated 55 rules. The rules are given as follows :

$$\begin{aligned}
&\text{rule 1 : if } (1.44x + 1.73y <= 5.98), \text{ then class 0} \\
&\text{rule 2 : if } (1.44x + 1.73y > 5.98) \\
&\qquad \text{and } (1.18x + 2.81y <= 12.37) \text{ then class 1} \\
&\text{rule 3 : if} (1.44x + 1.73y > 5.98) \\
&\qquad \text{and } (1.18x + 2.81y > 12.37) \\
&\qquad \text{and } (0.53x + 2.94y < 14.11), \text{ then class 0} \\
&\text{rule 4 : if} (1.44x + 1.73y > 5.98) \\
&\qquad \text{and } (1.18x + 2.81y > 12.37) \\
&\qquad \text{and } (0.53x + 2.94y > 14.11), \text{ then class 1}
\end{aligned}$$

These linear classifiers exactly match with boundaries shown in Fig.1(c), and they are more dominant for classification in terms of entropy maximization than a set of input attributes itself. Even if we include input attributes, the entropy measurement leads to a rule set with boundary equations. These rules are more meaningful than those of direct C4.5 application to raw data since their division shows the trend of data clustering and how each attribute is correlated.

Our approach can be applied to the data set which has both discrete and continuous values. If there is a set of input attributes, $Y = \{D_1, ..., D_m, C_1, ..., C_n\}$ where D_i is a discrete attribute and C_j is a continuous-valued attribute. For any discrete attribute D_x, it has a finite set of values available. For example, if there is a value set $\{d_{x1}, d_{x2}, d_{x3}, ..., d_{xp}\}$ for D_x, we can have a Boolean value for each value, using the conditional equation $D_x = d_{xj}$, for $j = 1, .., p$. We can put this state as a node in the first hidden layer, and then one of linear classifiers obtained with neural network is $L_k = \sum_i^{m+n} A_i W_{ik} = \sum_i^n C_i W_{ik} + \sum_i^m D_i W_{ik}$ where A_i is a member of the set Y. Since we have no interest in the relation of discrete attributes whose numeric conditions and coefficient values are not meaningful in this model, the value of linear classifier L_k only depends on a linear combination of continuous attributes and weights.

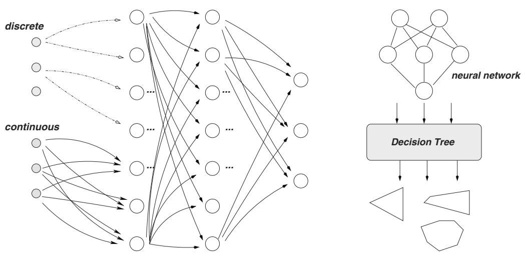

Fig. 2. Diagram for neural network and decision tree

The choice of discrete attributes in rules can be handled using induction tree algorithm more properly, without interfering the relations of continuous-valued attributes. The induction tree can do splitting any continuous value with selection of thresholds for given attributes, while it cannot derive the relation of input attributes directly. In our method, we can add to the data set of induction tree, new attributes $L_k = \sum_i^n C_i W_{ik}$ for $k = 1, .., r$, where r is the number of nodes in the first hidden layer for continuous-valued attributes. The new set of attributes for induction tree is $Y' = \{D_1, D_2, ..., D_m, C_1, C_2, ..., C_n, L_1, L_2, ..., L_r\}$. The entropy measurement will find out the most significant classification over the new set of attributes. Also we have tested another attribute set $Y'' = \{L_1, L_2, ..., L_r\}$ which consists of only linear classifiers generated by neural network.

4 Experiments

Our method has been tested on several sets of data in UCI depository [1]. Table 1 shows classification error rates for neural network and C4.5 [14] algorithm, and Table 2 shows error rates in our two methods. The first linear classifier $\{C + L\}$ method has a set of attributes for C4.5 classification, including both original input attributes and neural network linear classifiers together, while the second $\{L\}$ method only includes neural network linear classifiers.

Table 1. Data classification error rate result in neural network and C4.5

	neural network				*C4.5*	
data	pat / attr	training (%)	testing (%)	nodes	training (%)	testing (%)
wine	178 / 13	0 ± 0	1.6 ± 0.6	8 - 5	1.2 ± 0.1	6.6 ± 1.2
iris	150 / 4	0.6 ± 0.1	4.1 ± 1.3	5 - 4	1.9 ± 0.1	5.5 ± 1.1
breast-w	683 / 9	0.3 ± 0.1	4.9 ± 0.7	8 - 5	1.1 ± 0.1	4.7 ± 0.5
ion	351 / 34	0.8 ± 0.2	8.6 ± 0.8	10 - 7	1.6 ± 0.2	10.8 ± 1.3
pima	768 / 8	4.3 ± 0.5	24.7 ± 6.9	15 - 9	15.1 ± 7.6	24.7 ± 3.4
glass	214 / 9	6.5 ± 0.7	30.8 ± 2.0	15 - 8	7.0 ± 0.2	32.0 ± 2.5
bupa	345 / 6	5.9 ± 0.6	32.6 ± 1.5	10 - 7	12.9 ± 0.8	34.0 ± 1.4

Table 2. Data classification error result in our method using linear classifiers

	linear classifier$^{\{C+L\}}$		linear classifier$^{\{L\}}$	
data	training (%)	testing (%)	training (%)	testing (%)
wine	0.1 ± 0.1	3.7 ± 1.3	0.1 ± 0.1	3.2 ± 1.3
iris	0.7 ± 0.1	5.7 ± 1.1	0.7 ± 0.1	5.2 ± 1.7
breast-w	0.7 ± 0.1	4.4 ± 0.4	0.9 ± 0.2	4.4 ± 0.3
ion	0.8 ± 0.3	9.0 ± 0.9	1.2 ± 0.3	8.8 ± 0.9
pima	11.7 ± 1.3	27.0 ± 4.5	13.5 ± 1.2	26.6 ± 3.9
glass	5.8 ± 0.3	35.1 ± 1.7	6.7 ± 0.4	35.2 ± 2.2
bupa	10.5 ± 1.0	32.5 ± 2.6	15.3 ± 0.9	32.4 ± 2.9

The error rates were estimated by running the complete 10-fold cross-validation ten times, and the average and the standard deviation for ten runs are given in the table. Our method, adding linear classifiers into new attributes, is better than C4.5 in some sets and worse in data sets such as *glass*, *bupa* and *pima* which are hard to predict even in neural network. The result supports the fact that the method greatly depend on neural network training. If neural network fitting is not correct, then the result may mislead the result. Normally C4.5 application shows the error rate is very high for training data in Table 1. Table 3 says the number of rules using our method is smaller than that using conventional C4.5 in most of data sets. Especially when only linear classifiers from neural network are used, it is quite effective to reduce the number of rules. Most of data sets in UCI depository have a small number of data examples relative to the number of attributes. The significant difference between a simple C4.5 application and a combination of C4.5 application and neural network is not seen distinctively in UCI data unlike synthetic data in Fig.1. Information of data trend or input relations can be more definitely described when given many data examples relative to the number of attributes.

Table 3. Number of attributes and rules for C4.5 applications

	C4.5		linear classifier$^{\{C+L\}}$		linear classifier$^{\{L\}}$	
data	rules	attributes	rules	attributes	rules	attributes
wine	5.4 ± 0.2	13	3.0 ± 0.0	13 + 8	3.0 ± 0.0	8
iris	4.8 ± 0.2	4	3.9 ± 0.2	4+5	3.7 ± 0.2	5
breast-w	18.6 ± 0.7	9	10.5 ± 0.9	9 + 8	7.8 ± 0.8	8
ion	14.2 ± 0.6	34	7.7 ± 0.8	34 + 10	6.3 ± 0.5	10
pima	26.7 ± 2.8	8	33.7 ± 4.2	8 + 15	23.6 ± 2.9	15
glass	25.1 ± 0.6	10	24.6 ± 0.7	10 + 15	23.0 ± 0.8	15
bupa	29.1 ± 1.2	6	24.1 ± 2.0	6 + 10	15.3 ± 1.5	10

Table 1 and 2 says neural network classification is better than C4.5 applications. If we can derive easily Boolean gates directly from neural network, the combination of linear classifiers and Boolean logic gates will form a set of good rules. Each threshold logic based on neural weights is equivalent to a set of logic gates when it is applied to digital logic and will form a sum of product of Boolean logic [11]. We need to find an efficient or heuristic way to generate a set of Boolean logic gates from neural network function. If we apply a simple ID3 algorithm with linear classifiers to reduce error rate in the training sets instead of C4.5 algorithm, it may increase the performance up to the level of neural network.

5 Conclusions

This paper presents a hybrid method for constructing a decision tree from neural networks. Our method uses neural network modeling to find unseen data points and then induction tree is applied to data points for symbolic rules, using features from neural network. The combination of neural network and induction tree will compensate for the disadvantages of one approach alone. This method has several advantages over a simple decision tree method. First, we can obtain good features for classification boundary from neural networks by training input patterns. Second, because of feature extractions about input variable relations, we can obtain a compact set of rules to reflect input patterns.

We still need further work such as applying minimum description length principle to reduce the number of attributes over linear classifiers or comparing with other methods such as regression tree methods.

References

1. C. Blake, E. Keogh, and C.J. Merz. *UCI* repository of machine learning databases. In *Preceedings of the Fifth International Conference on Machine Learning*, http://www.ics.uci.edu/ mlearn, 1998. 216
2. G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Technical Report, Department of Computer Science, Tufts University, Medford, MA, 1988. 214
3. T.G. Dietterich, H. Hild, and G. Bakiri. A comparative study of id3 and backpropagation for english text-to-speech mapping. In *Proceedings of the 1990 Machine Learning Conference*, pages 24–31. Austin, TX, 1990. 212
4. U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of IJCAI'93*, pages 1022–1027. Morgan Kaufmann, 1993. 211
5. L. Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, New York, 1994. 212
6. S.S. Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall, Upper Saddle River, N.J., 2nd edition, 1999. 213
7. J. Hertz, R.G. Palmer, and A.S. Krogh. *Introduction to the Theory of Neural Computation*. Addision Wesley, Redwood City, Calif., 1991. 214
8. K. Hornik, M. Stinchrombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
9. K.B. Irani and Qian Z. Karsm : A combined response surface / knowledge acquisition approach for deriving rules for expert system. In *TECHCON'90 Conference*, pages 209–212, 1990. 212
10. DaeEun Kim. Knowledge acquisition based on neural network modeling. Technical report, Directed Study, The University of Michigan, Ann Arbor, 1991. 212
11. Zvi Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, London, 1970. 217
12. W. Kweldo and M. Kretowski. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 392–397. Springer, 1999. 211
13. J.R. Quinlan. Comparing connectionist and symbolic learning methods. In *Computational Learning Theory and Natural Learning Systems*, pages 445–456. MIT Press, 1994. 212
14. J.R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Approach*, (4):77–90, 1996. 211, 212, 216

15. R. Setiono and Huan Lie. Symbolic representation of neural networks. *Computer*, 29(3):71–77, 1996. 212
16. G.G. Towell and J.W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, Oct. 1993. 212