Rule Extraction from Neural Networks via Decision Tree Induction

Makoto Sato, Hiroshi Tsukimoto
Toshiba Corporation Research & Development Center
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, JAPAN
E-mail: satom@isl.rdc.toshiba.co.jp

Abstract

Rule extraction from neural networks is the task for obtaining comprehensible descriptions that approximate the predictive behavior of neural networks. Rule-extraction algorithms are used for both interpreting neural networks and mining the relationship between input and output variables in data. This paper describes a new rule extraction algorithm that extracts rules that contain both continuous (real-valued) and discrete literals. This algorithm decomposes a neural network using decision trees and obtains production rules by merging the rules extracted from each tree. Results tested on the databases in UCI repository are presented.

1 Introduction

Two important criteria for inductive learning are predictive accuracy and comprehensibility of learned models. While artificial neural network attains high predictive accuracy, the incomprehensibility of its predictive behavior prevents its application in some problems. For this reason, several techniques for extracting comprehensible knowledge from trained neural networks have been proposed [1]. The extracted knowledge enables an expert to inspect the predictive behavior of neural networks, and, for improving its predictive accuracy, enables to make feedback to the process of neural network training such as attribute selection.

In addition, in the field of KDD (knowledge discovery in databases and data mining) [4, 19], the interpretation of trained models is useful for gaining insight into the training data. By comparison with the methods directly extract knowledge from data, the model-mediate methods have two advantages. One is the noise removal property of inductive models such as neural networks and decision trees. The other is their compliment property to unknown data. Therefore, it is possible that a model-mediate method discovers the knowledge that is never discovered directly from data.

In this paper, we present a new algorithm, CRED, for interpreting the predictive behavior of trained neural networks and discovering knowledge in data by extracting rules from neural networks. For rule extraction, we use a trained neural network and the activation value of each unit in the network on training data.

Several methods have been proposed for rule extraction from neural networks. Those are characterized by:

- Type of allowed variable (i.e. continuous, discrete or both),
- Type of extracted rule (i.e. production rule, n-of-m rule, n-of-m tree),
- Approach to rule search (i.e. pedagogical or decompositional).

Since many of real-world data contain both continuous and discrete variables, the knowledge representation that allows both continuous and discrete variables is useful. Although quite a few algorithms deal with discrete variables [3, 9, 10, 12, 16, 17] or continuous variables [15, 11], only a few algorithms deal with both. TREPAN [2] extracts a n-of-m tree that contains both continuous and discrete variables by using the technique of decision tree induction. Decision tree induction makes it possible to deal with both continuous and discrete variables [8]. We also use decision tree induction for dealing with the both type of variables. Examples of the production rules that involve both continuous and discrete literals are as follows:

$$If(Age : 30..39)\&(Married : Y) \rightarrow (NumCars : 2..3),$$
$$If(Age : 40..45)\&(FastTrack : No) \rightarrow (Married : No).$$

We refer to them as c/d-rules.

The decompositional algorithms extract rules from each unit in a neural network and aggregate them into

0-7803-7044-9/01/$10.00 ©2001 IEEE

the rules for the network. In contrast, the pedagogical algorithms consider a neural network to be a black box and use only the activation value of input and output units in the network. The decompositional algorithms can show the role of each hidden unit and extract detail rules compare with the pedagogical algorithms.

CRED also employs decision tree for extracting c/d-rules. The primary difference between TREPAN and CRED is the approach for searching rules. TREPAN is a pedagogical algorithm, and it builds a decision tree based on the activation patterns of input and output units which are obtained using a given neural network and training data. On the other hand, CRED decomposes a neural network by building decision trees based on the activation patterns of hidden and output units and input and hidden units. The network decomposition generates relatively accurate rules and gives aspects of the role of each hidden unit.

In some networks, the simplification of c/d-rules is needed for comprehensibility. For example, the rules $If(Age : 30..39) \rightarrow (NumCars : 2..3)$ and $If(Age : 45..49) \rightarrow (NumCars : 2..3)$ can be simplified to the rule $If(Age : 30..49) \rightarrow (NumCars : 2..3)$. In some cases, rule simplification suffers from the tradeoff between accuracy and comprehensibility. In the above example, if the rule $If(Age : 40..42) \rightarrow (NumCars : 0..1)$ exists, the above simplification corrupts the accuracy of rules. Therefore, some metric is needed for evaluating each simplification for c/d-rules.

Recent research in KDD has been presented methods for integrating the production rules with continuous variables [5, 14, 18]. In this paper, we present a new c/d-rule simplification algorithm based on J-measure, which is a well known rule interestingness measure.

2 The CRED algorithm

CRED (a Continuous/discrete Rule Extractor via Decision tree induction) consists of the following five steps (see Figure 1).

- Step 1 : Set a target class corresponding to the consequence literal and set the target pattern of output units in a trained neural network.
- Step 2 : Build a hidden-output tree using the target pattern and activation pattern of hidden units, extract intermediate rules from the tree, and decompose the network into constitutive functions.
- Step 3 : For each function, build an input-hidden tree and extract input rules.
- Step 4 : Obtain total rules by substituting the input rules for the intermediate rules.
- Step 5 : Merge the total rules for simplification with some merging criterion.

Step 1 decides what to ask to a neural network. If the neural network is trained for a continuous class variable, An example of query is about: *what is the condition that the activation values of the output unit are within some range.* Clustering on samples of the activation values enables automatically to generate ranges by using boundaries of clusters. If the neural network is trained for a discrete class variable, examples of query are about: *what is the condition that the network classifies data into the class* or *what is the condition that the activation values of the output unit for the class are larger than the threshold.* According to the decided query, the continuous activation values of the output units are transfered to a discrete class value for decision tree induction.

Step 2 builds a decision tree (hidden-output tree) using the activation pattern of hidden units as attribute data, and the discretized pattern created in step1 as class data. Since each leaf of a decision tree corresponds to a production rule, exclusive production rules are extracted (intermediate rules) from decision tree [7]. In this stage, each intermediate rule is simplified by removing useless literals and eliminate rules that are included by another rule. Note that all of the attribute variables of the hidden-output tree are continuous. Here we pay attention to the boundaries used in the literals of the intermediate rules. In Figure 1, the corresponding boundaries are 0.6 for h1 (hidden unit 1) and 0.5 for h2 (hidden unit 2). In this case, we approximately consider one function of h1 is to distinguish the activation pattern of the previous layer by 0.6. So, this step generates a query about *what is the condition that the activation values of h2 are larger than 0.5* and discretizes the activation values of h2. This process is repeated for each boundary.

Step 3 builds decision trees (input-hidden trees) each of which corresponds to a query generated in Step 2. In this step, the input variables of given data are the attribute variables and the discritized pattern created in Step 2 is the class variable. Similarly, this step generates from each input-hidden tree production rules (input-rules) that comprehensibly describe the function of the hidden unit, simplifies each rule, and eliminates the redundant rules.

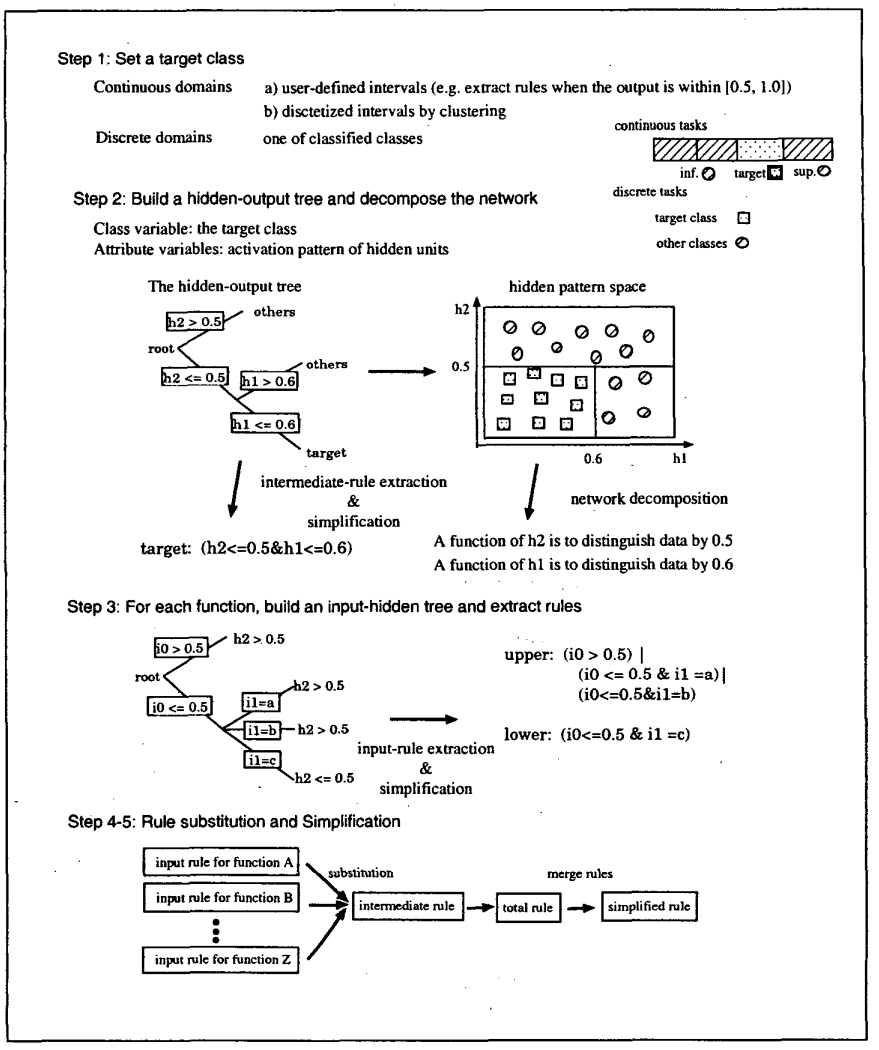

Figure 1: The CRED algorithm

By substituting the input rules for the intermediate rules, Step 4 generates the total rules, which describe the relationship between input pattern and the target query decided in Step 1. Then the redundant rules are eliminated after simplification of each total rule.

In Step 2, 3, and 4, a c/d-rule is separately simplified (generalized) by selecting the best rule by comparing its rule interestingness value with the values of the rules, where each is obtained by removing one antecedent literal. For simplification of continuous ranges described in Section 1, we present a new c/d-rule simplification method in the next section.

[10] proposed a rule extraction algorithm that discretizes the pattern of hidden units by clustering. Since clustering automatically decides boundaries with pattern data, this algorithm can generate redundant boundaries. Besides, it can classify the two data samples that have different classes and similar attribute patterns into the same cluster. So, it needs a special learning algorithm and network pruning before rule extraction. In contrast, since our algorithm discretizes the pattern of hidden units by decision tree induction, it generates only necessary boundaries for a given query and does not need special learning algorithms. Since our algorithm also discretizes the pattern of input units by decision tree induction, it deals with the problems contain both continuous and discrete variables.

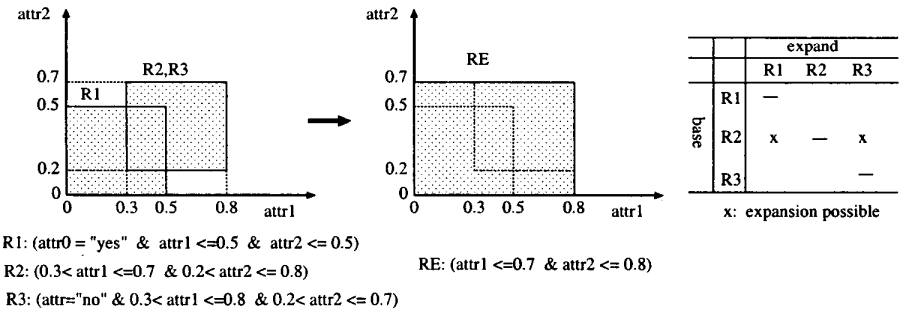

Figure 2: An example of c/d-rule expansion.

3 Expansion of c/d-rules

In addition to the elimination of antecedent literals, integration of continuous literals generalizes the production rule. We adopt hill climbing approach because finding the optimal integration, for a given production rule set, is awfully difficult.

If a rule set $\mathcal{R}$ is given, CRED repeats the following evaluation phase and integration phase. The evaluation phase calculates, with some integration measure, the integration losses for each combination of $R_i(\in \mathcal{R})$ and $R_j(\in \mathcal{R}, R_i \neq R_j)$. The integration phase generates a rule by adapting the most harmless integration as far as the minimum loss is not more than a threshold, and eliminates redundant rules from the rule set. This cycle continues while $\mathcal{R}$ is changed.

If R_i and R_j are given and the consequence literals of the two are the same, there are various candidates of the generalized rule. Here, let L_c and L_d denote the conjunction of continuous antecedent literals of R and the conjunction of discrete antecedent literals of R, respectively. And let $L \preceq L'$ denotes that all of data that satisfy L also satisfy L'. For simplicity of explanation, we define the principal and accessory to R_i and R_j, and refer *to expand R_i with R_j into R_e* as to find a generalized rule R_e based on R_i using R_j, where $L_{c,i}, L_{c,j} \preceq L_{c,e}$ and $L_{d,i}, L_{d,j} \preceq L_{d,e}$.

Figure 2 illustrates our strategy of expansion. That is:

> If $L_{d,j} \preceq L_{d,i}$, expand R_i with R_j into R_e, where, $L_{d,e} = L_{d,i}$ and $L_{d,e}$ is the maximum hyper-cube using the boundaries in $L_{c,i}$, $L_{c,j}$.

In Figure 2, only the $R2$ is able to expand with $R1$ and $R3$ because $L1_d \preceq L2_d$ and $L3_d \preceq L2_d$. Note that we do not execute the expansion of L_d (for example, the expansion of $R1$ with $R3$) because it can be achieved by the literal removing simplification that is described in Section 2.

In the field of knowledge discovery in databases and data mining, various methods have been proposed for evaluating production rules and integrating plural rules into a rule for comprehension. [19] has proposed a c/d-rule merging algorithm based on *J-measure*[13], which is a well known rule interestingness measure.

For rule $X \rightarrow Y$, let $p(X), p(Y), p(XY)$ denote the fractions of data satisfying condition X, Y, XY, respectively, and $P(Y|X) = P(XY)/p(X)$. *J-measure* is defined as:

$$J(X,Y) = p(X)p(Y|X)\log_2\frac{p(Y|X)}{p(Y)} + p(X)(1-p(Y|X))\log_2\frac{1-p(Y|X)}{1-p(Y)} \quad (1)$$

[19] uses only the first term of eq.(1) (refers to J_1), and evaluates the loss, $Loss(X_1, X_2)$, of the merge $(X_1, X_2 \rightarrow X_e)$ by:

$$Loss(X_1,X_2) = J_1(X_1,Y) + J_1(X_2,Y) - J_1(X_e,Y), \quad (2)$$

and uses this value for decision of merging.

We basically uses eq.(2). But our rules can contain an overlapping area as shown in Figure 2. Therefore we modify eq.(2) by:

$$Loss'(X_1,X_2) = Loss(X_1,X_2) - J_{min}\frac{p(X_1X_2)}{p(X_{min})}, \quad (3)$$

where, $J_{min} = min\{J_1(X_1,Y), J_1(X_2,Y)\}$, $p(X_{min})$ is the fraction of data satisfying the condition that takes the lower J_1, and $p(X_1X_2)$ denotes the fraction of data satisfying both X_1 and X_2. Eq.(3) means that we use the higher J_1 for the overlapping area.

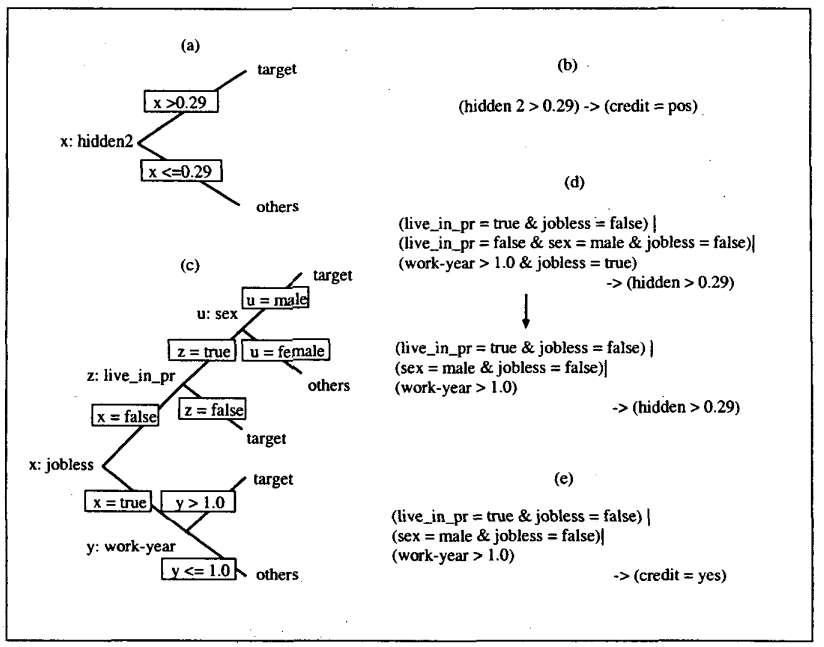

Figure 3: Results on The Credit Data. The hidden-output tree (a), the intermediate-rules (b), the input-hidden tree (c), the input-rules before and after simplification (d), and the total rules (e).

4 Experiments

This section shows the results of experiments through applying CRED to several benchmark databases in UCI repository. In this experiments, we used C4.5 [8] for decision tree induction and used the expected error rate [7] as the rule interestingness measure for the literal removing simplification in Section 2. Neural networks were trained by a normal back-propagation algorithm.

4.1 A Detailed Example - The Credit Data

We show in detail how CRED extracts rules in *credit* data, which consists of five continuous, four boolean, a discrete, and an ID variables and contains 125 samples. We eliminated the discrete variable and the ID variable form the data, and trained a three-layer feed-forward neural network with the data. The number of hidden units was two.

Figure 3 shows the detailed results of the rule extraction. We queried the network about the condition that the *credit* variable to be *yes*. With our network weights, we found that the first hidden unit was useless in this network and four variables (*work-year, live_in_pr, sex, and jobless*) were important for this classification problem. 99% of the samples satisfied the total rules were classified into *yes* by the network.

4.2 Results on Several Data

We applied CRED to various data in UCI repository (Table 1). For continuous class data, we executed clustering of the output variable by *k-means* algorithm [6], where the number of cluster was three and queried about the condition that the values of the output variable were within the medium cluster. For discrete class data, we queried about the condition that a sample was classified into the first class. The accuracy in Table 1 means the probability of the samples satisfied the total rules also satisfied the target query. These results showed the validity of CRED.

5 Conclusion

This paper has described a new rule-extraction algorithm for domains that involve both continuous and discrete algorithms. This algorithm does not depend on the structure of the networks nor on the training methods. Empirical results on UCI benchmark databases showed that our algorithm was useful for various type of databases.

database	attr. type	class type	num.of attr.	num.of data	accuracy
iris	continuous	discrete	4	150	0.98
machine	continuous	continuous	7	209	0.98
auto-mpg	both	continuous	7	398	0.89
credit	both	bool	10	125	0.99
brest canc.	continuous	bool	9	699	0.99
bupa	continuous	bool	6	345	0.83
monk1	discrete	bool	6	124	1.00
monk3	discrete	bool	6	122	0.99
university	both	continuous	15	285	0.78

Table 1: performances on UCI databases

References

[1] Andrews, R. , Diederich, J., & Tickle, A.B., *Survey and critique of techniques for extracting rules from trained artificial neural networks.*, Knowledge-Based Systems, Vol. 8, No. 6, pp.373-384, 1995.

[2] Craven, M.W. & Shavlik,J.W., *Extracting tree-structured representations of trained networks.*, Advances in Neural Information Processing Systems (vol. 8), MIT Press.

[3] Fu,L., *Rule learning by searching on adapted nets.*, In Proceedings of the Ninth National Conference on Artificial Intelligence, pp.590-595, 1991.

[4] Fayyad, U., Shapiro, G.P., Smyth, P., & Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining.*, AAAIPress/MIT-Press, Menlo Park, 1996.

[5] Fukuda, T., Morimoto, Y., & Morishita, S., *Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization.*, In Proceedings of SIGMOD, pp.13-23, 1996.

[6] MacQueen, J.B.: *Some methods for classification and analysis of multi variable observations.*, Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 1, University of California Press, 1967.

[7] Quinlan, J., *Generating production rules from decision trees.*, Proceedings of the 10th International Joint Conference on Artificial Intelligence, pp.304-307, 1987.

[8] Quinlan, J., *C4.5:Programs for Machine Learning.*, Morgan Kaufmann, San Mateo, CA, 1993.

[9] Saito,K. & Nakano,R., *Medical diagnostic expert system based on pdp model.*, In Proceedings of IEEE International Conference on Neural Networks, vol., 1, pp.255-262, 1998

[10] Setiono, R., & Liu, H., *Understanding neural networks via rule extraction.*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, pp.480-485, 1995.

[11] Setiono,R., *Extracting rules from neural networks by pruning and hidden-unit splitting.*, Neural Computation, vol.9, no.1, pp.205-225, 1997.

[12] Setiono,R. & Leow, W.,L., *FERNN:An algorithm for Fast Extraction of Rules from Neural Networks.*, Applied Intelligence, 2000, volume 12, No.1/2, pp.15-25.

[13] Smyth, P. & Goodman, R., *An information theoretic approach to rule induction from databases.*, IEEE Transactions on Knowledge and Data Engineering, vol.4, no.4, 1992.

[14] Srikant, R., & Agrawal, R., *Mining quantitative association rules in large relational tables.*, In Proceedings of SIGMOD, pp.1-12, 1996.

[15] Thrun,S., *Extracting rules from artificial neural networks with distributed representations.*, Advances in Neural Information Processing Systems, vol.7, MIT Press, Cambridge, MA, 1995.

[16] Towell,G. & Shavlik, J., *Extracting refined rules from knowledge-based neural networks.*, Machine Learning, vol.13, no.1, pp.71-101, 1993.

[17] Tukimoto, H., *Extracting rules from trained neural networks.*, IEEE Transactions on Neural Networks, Vol. 11, No. 2, March, pp.377-389, 2000.

[18] Wang, K., Tay,S., & Liu, B., *Interestingness-based interval merger for numeric association rules.*, Proceedings of International Conference on Knowledge Discovery and Data Mining, pp.121-128,1998.

[19] Weiss, S.M. & Indurkhya, N., *Predictive Data Mining: A Practical Guide* ., Morgan Kaufmann Publishers, Inc., San Francisco, 1998.