Rule Extraction from Neural Networks : Enhancing the Explanation Capability

Sang Chan Park · S. Monica Lam & Amit Gupta

ABSTRACT

This paper presents a rule extraction algorithm RE to acquire explicit rules from trained neural networks. The validity of extracted rules has been confirmed using 6 different data sets. Based on experimental results, we conclude that extracted rules from RE predict more accurately and robustly than neural networks themselves and rules obtained from an inductive learning algorithm do. Rule extraction algorithms for neural networks are important for incorporating knowledge obtained from trained networks into knowledge based systems. In lieu of this, the proposed RE algorithm contributes to the trend toward developing hybrid and versatile knowledge-based systems including expert systems and knowledge-based decision support systems.

Keyword : rule extraction, neural network, inductive learning, expert systems, knowledge based decision support systems

1. Introduction

Learning from examples, one branch of machine learning topology, is to acquire classification knowledge from given training examples. Given a set of examples in the formation of input attribute vector x and corresponding class y, the learning algorithm identifies the mapping function f(x)=y. The mapping function is considered as the knowledge obtained from the machine learning process. In order to facilitate storing,

* Department of Industrial Management, Korea Advanced Institute of Science and Technology
* Department of Management Information Science, School of Business Administration, California State University-Sacramento
* School of Business, University of Wisconsin-Madison

processing, and understanding the knowledge in rule-based systems, it is preferable to have knowledge represented in symbolic rules like IF ($x_1 < C_1$, $x_2 \geq C_2$, $x_n < C_n$) THEN (class y) where $C_{i,\ i=1,\ \ldots,\ n}$ is a constant.

Among various algorithms for learning from examples, neural networks have been proved to be robust and accurate (Barnard and Casasent, 1989 ; Decatur, 1989 ; Dutta and Shekhar, 1988 ; Tam and Kiang, 1992 ; Tang et al., 1990). However, neural networks lack in explaining the classification or prediction result. A trained neural network, consisting of a set of connection weights, is difficult to interpret. It is a well known fact that knowledge about the relative importance of input attributes and their relationship can provide valuable information for collecting sample data and for remedial actions in experimental research.

Especially when neural networks are to be used as a knowledge source for expert systems, the content of the knowledge in a trained network form needs to be reshaped to satisfy a fairly common capability in most current expert systems tools and shells which show the reasoning process at given points in the operation. The knowledge source has to be fairly easy to maintain, with the ability to incorporate new or changing concepts into itself whereas current neural network needs to be retrained whenever there exist additional training examples (cases or pairs of input/output vectors). Thus, the task of enhancing the explanation capability of neural network becomes an utmost research subjects. Perhaps, the enhancing the explanation capability of the neural network advocates in alleviating the criticism that it does not provide any insight to a philosophical understanding of human intelligence concepts because it simply works as a black-box.

To meet this need, various researchers put their efforts to design algorithms to extract classification rules from trained neural networks (Fu, 1991 ; Gallant, 1988 ; Saito et al., 1988 ; Towell et al., 1991). On the other hand, inductive learning algorithms like ID3, and C4.5 (Quinlan, 1993), which belong to learning from examples category, have been widely used to extract classification rules directly from training examples.

This paper introduces a new rule extraction algorithm RE for neural networks. The RE algorithm is developed to eliminate several problems of the existing rule extraction algorithms. To verify the validity of the extracted rule, the classification and prediction accuracy is compared to those of neural networks as well as rules from C4.5 using 6 different data sets, 3 of which have only nominal attributes (propositional value), and the remaining 3 have attributes having continuous values.

The remainder of this paper is organized as follows : Section 2 reviews 3 existing rule extraction algorithms for neural networks and the rule generation process of the inductive learning algorithm C4.5 ; Section 3 presents the RE algorithm ; Section 4 describes the experimental methodology ; Section 5 discusses the experiment results ; and the last section concludes the paper with future research directions.

2. Literature Review

Research on extracting rules from neural networks is closely related with the trend to apply neural networks in expert systems. In lieu of that, existing rule extraction algorithms tend to share 3 undesirable characteristics. First, algorithms are designed to use only attributes having propositional values (binary or ternary values). Second, algorithms tend to ignore attributes whose connection weight is weak. Third, as the number of input attributes or connection weights increases, the number of rules generated grows exponentially. We will review 3 existing rule extraction algorithms and 1 inductive learning algorithm in this section : connectionist expert systems approach, the subset method, the NofM method, and C4.5. The first three algorithms represent the major directions of rule extraction for neural network.

2.1 Rule Extraction for Connectionist Expert Systems

Gallant (1988) suggested a hybrid system integrating a feedforward neural network with an expert system. In the network he used, connection weights are either positive or negative integers. A connection weight (W_{ij}) of 0 indicates no link between two nodes. The activation value of a node (U_i) can be 1, -1, or 0 which corresponds to the logical meaning of true, false, or unknown. The calculation of the activation of a node Ui is as following :

$$U_i = \begin{cases} +1 & \text{if } S_i > 0 \\ -1 & \text{if } S_i < 0 \\ 0 & \text{if } S_i = 0 \end{cases}$$

where $S_i = \Sigma_j W_{ij} U_j$.

The network is trained using pocket algorithm which is a modification of the perceptron (Rosenblatt, 1962). The network does not have hidden layers. The rule extraction process for a connectionist expert system is a byproduct of the effort to exhibit the explanation capability. Figure 1 shows a trained network with 6 input nodes, and 1 output node determined to have a value of 1. The rule extraction process is as follows :

1) List all known inputs that contribute to the positivity of the output (U_7).

   *** U_1, U_2, and U_5 are selected. U_6 is ignored because the product of the input value of $U_6(-1)$ and its connection weight (5) is negative. U_3 and U_4 are excluded because their impacts are unknown.

2) Arrange the list in descending order based on the absolute values of the weights.

   *** The sequence becomes U_2, U_5, and U_1.

3) Generate clauses for an IF-THEN rule from the ordered list until the following condition is satisfied : { Σ_{ua} | W_i | }>{Σ_{rkn} | W_j | }
where "ua" means by Ui used as antecedents of the rule and "rkn" means by remaining known nodes Uj (including those known input nodes which are not selected in the ordered list). The generated rule will be

IF U_2 is false and U_5 is true THEN U_7 is true.

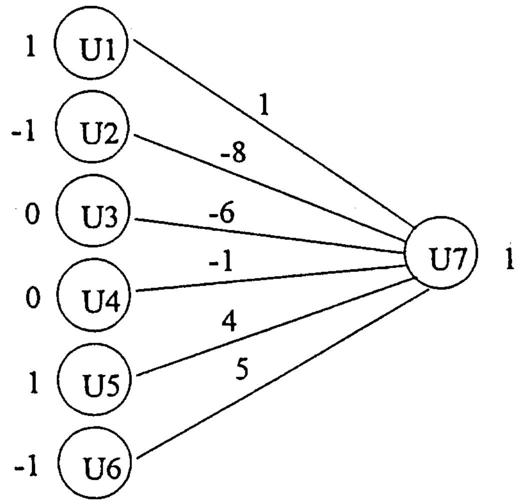


Figure 1. Neural Network used for Connectionist Expert Systems Rule Extraction

2.2 Subset Method for Rule Extraction

Saito (1988) and Fu (1991) came up with another variation of rule extraction algorithm. The network allows node to have a threshold value (more realistic than the one used for the connectionist expert systems). Still, the attributes have binary values (0 or 1) and ones having weak connection weights are ignored. Figure 2 provides an example network for the algorithm summarized below :

1) For each U ⊂ {all hidden and output nodes}, let T be the threshold of the node U.

2) Find up to P individual subsets of positively weighted connections to the node U such that

$$(T+\Sigma_{sp}\ W_i)>0,$$

where "sp" means by the individual subsets in the set of all input side nodes positively connected to the node U.

3) For each "sp", find up to N subsets of negatively weighted connections to the node U such that

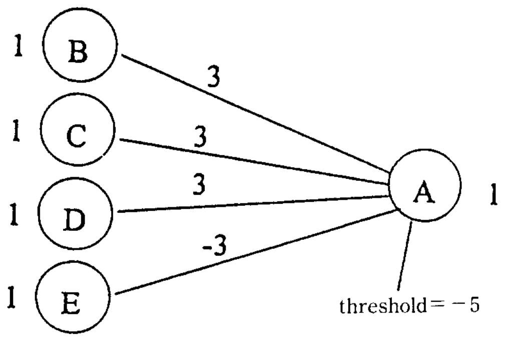
$(T+\Sigma_{sp} W_i + \Sigma_{sn} W_j) < 0$,

where "sn" means by the individual subsets in the set of all input side nodes negatively connected to the node U.

4) For each "sp", and for each "sn" in a given "sp", create a rule :

IF $\vee$nodes in "sp" and not $\vee$nodes in "sn" THEN U.

5) Simplify rules and remove any duplicate rules. For example, in Figure 2, the presence of the antecedent "not E" in the rule "*IF B, C, D, and not E THEN A*" does not influence the positivity of node A, so that the clause is removed.



Extracted rules are as follows :

IF B, C and not E THEN A
IF B, D and not E THEN A
IF C, D and not E THEN A
IF B, C and D THEN A <=
IF B, C, D, and not E THEN A

Figure 2 Neural Network used for Subset Rule Extraction Algorithm

2.3 NofM Algorithm for Rule Extraction

NofM (Towell, 1991) is computationally less expensive than the subset method in that it uses groups of nodes having similar connection weights as building blocks for rule antecedents instead of individual node. Figure 3 provides an example for the algorithm summarized as follows :

1) For each hidden and output node, collect input side nodes having similar connection weights into groups.

2) Set weights of the same group to the average weight of the group.

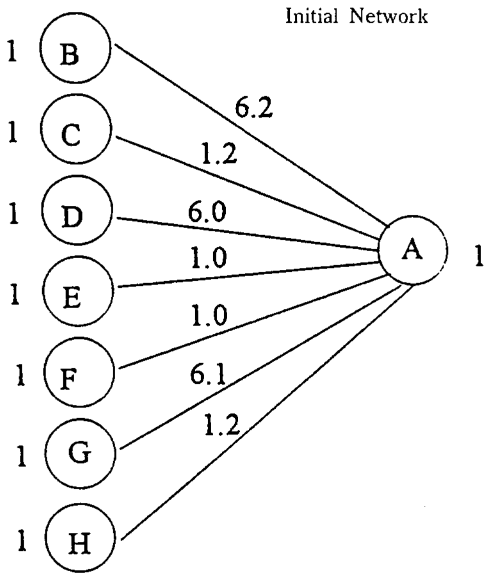
3) Eliminate any group that does not significantly affect the positivity of the output.

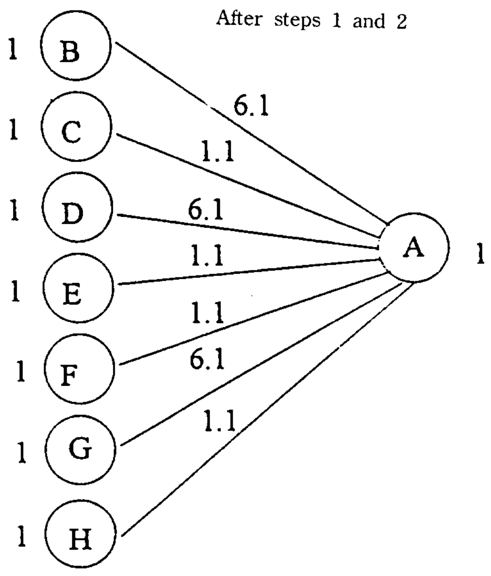
4) Holding all weights constant, retrain the network to optimize thresholds of all hidden and output nodes using backpropagation algorithm.

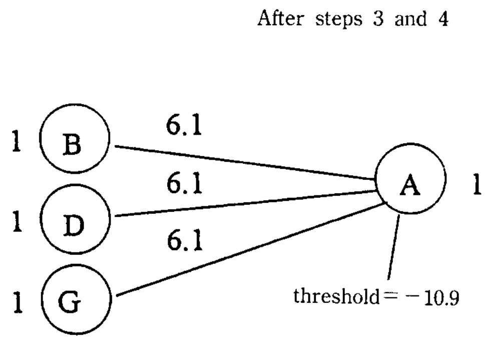
5) Form a single rule for each hidden and output unit.

6) Simplify rules to eliminate weights and thresholds.

NofM is a cubic algorithm whereas the subset method is an exponential algorithm. Experimental results show that the rule sets derived by the NofM method are more accurate than those derived by the subset method (Towell, 1991).







After step 5 :

IF 6.1 * NumberTrue(B, D, G)>10.9
THEN A

After step 6 :

IF 2 of {B, D, G} THEN A

note : NofM algorithm takes propositional value (true of false) inputs. Thus the NumberTrue(nodes) function returns the number of nodes having true values.

Figure 3. Neural Network used for NofM Algorithm

2.4 Rule Generation by Inductive Learning Algorithm

The C4.5 algorithm (Quinlan, 1993) is an advanced version of ID3 which has one module to generate decision trees and the other to extract rule from the decision tree. In order to enhance the prediction power of the extracted rules, C4.5 generalizes rules, called pruning process, by deleting antecedents from rules until prediction rates increase. Specifically, the program applies the minimum description length principle to determine the optimal length of antecedents in rules, and the simulated annealing technique to determine which antecedents to drop.

All the rules for the same class (output value) are collected in a rule set. The order of rule sets to be applied to a new testing case (a pair of input and output vector) is determined by the number of false positive cases (negative cases misclassified as positive) in the training set. The rule set with the fewest errors is applied first. If no rule in the first set can be applied to the new case, the rule set with the second fewest errors is then applied, and so on. If all rule sets are not applicable to the new case, the case will be assigned to the default class which is the most frequent class in the training set.

C4.5 obviously can handle attributes having continuous value ranges. C4.5 can generate more than one decision tree and select the best one for rule extraction based on predicted error rates for unseen cases. The predicted error rate is a probability determined by confidence limits for the binomial distribution. Each path (leaf or class) in a decision tree has a predicted error rate. The sum of the predicted errors from all leaves divided by the number of examples in the training set is the predicted error rate of the selected tree for unseen cases.

3. The RE Algorithm

One of the major criticisms of the neural network is its lack of explanation capability. Sets of connection weights resulted from training, hardly shed light to the importance of input attributes and their relationship. To address this issue, research efforts have concentrated on developing algorithms to extract classification rules from neural networks, yet several problems have remained unsolved. The suggested RE algorithm eliminates the following problems shared by existing rule extraction algorithms for neural networks : 1) the number of rules generated from a neural network grows exponentially as the number of connection weights in the network increases ; 2) generated rules only include attributes having propositional values (binary or ternary input values) instead of those having continuous value ranges and valid intervals ; 3) attributes having weak connection weights are totally ignored ; and 4) some algorithms wont allow neural networks with hidden layers. RE does not take all or nothing approach in selecting important attributes, rather it selects valid/influential value ranges of all the attributes to construct rules. In addition, RE can control the number and size of the rules by incrementally including attributes ranked by their relative importance. Obviously, RE does not have any limitation in handling networks with hidden layers.

The RE algorithm (Lam 1994) has several advantages over its counterparts in the literature. In terms of tractability, RE is better than its counterparts since only one composite rule is extracted for each class (output node) whereas subset method and NofM algorithm may generate several rules for each class. Consequently, RE is computationally much more efficient than existing algorithms. RE allows input attributes having continuous values as long as several intervals are defined. To serve this purpose, however, input nodes of the network needs to be expanded so that each node represents each interval (category) of the attribute.

Figure 4 shows a trained feedforward network. The sample network has 9 input nodes, 4 hidden nodes, and 3 output nodes. For the illustration purpose, 3 input nodes have 3 value categories (intervals), respectively. Node X_{ij} indicates the category j of attribute i. The sample network has 1 hidden layer. This should not be a severe handicap as it has been proved that one hidden layer with sufficient number of hidden nodes can approximate any Borel measurable function (Cybenko, 1989 ; Hornik et al., 1989). The following steps describe the RE algorithm for extracting classification rules :

1) For each hidden node, arrange the input side nodes based on absolute connection weights between input nodes to the hidden node in descending order. Call the above rankings R_i. Put a positive sign in front of the node if the connection weight is positive or a negative sign if weights are negative.

R1 : +X33 +X22 −X12 +X21 +X23 +X13 −X11 +X32 +X31
R2 : +X11 +X12 −X13 +X21 −X22 −X23 −X31 −X32 −X33
R3 : +X11 −X13 +X22 −X31 +X33 −X12 +X21 −X23 +X32
R4 : −X33 −X32 −X31 −X23 +X22 +X21 −X13 −X12 −X11

2) Decide the parameter N which is the number of categories in R_i to be used for the rule extraction. Suppose if N=2, then Ri is reduced as follows :

R1 : +X33 +X22
R2 : +X11 +X12
R3 : +X11 −X13
R4 : −X33 −X32

3) Rank R_i in descending order based on the connection weights between hidden nodes and the output node. Put a positive sign in front of the node if the connection weight is positive or a negative sign if weights are negative.

+R2 −R1 +R3 −R4

4) List X_{ij} in all ranked R_i. Change the sign of X_{ij} if R_i has a negative sign. Remove the duplicate X_{ij} (delete X_{ij} which appears later).

+X11 +X12 −X33 −X22 −X13 +X32

5) Create a rule for the output node using only positive X_{ij} in the list.

IF attribute 1 is in category 1 or 2 and attribute 3 is in category 2 THEN the C1 is 1

Step 3 to 5 will be repeated for each output node to create rules. The application order of rules to the

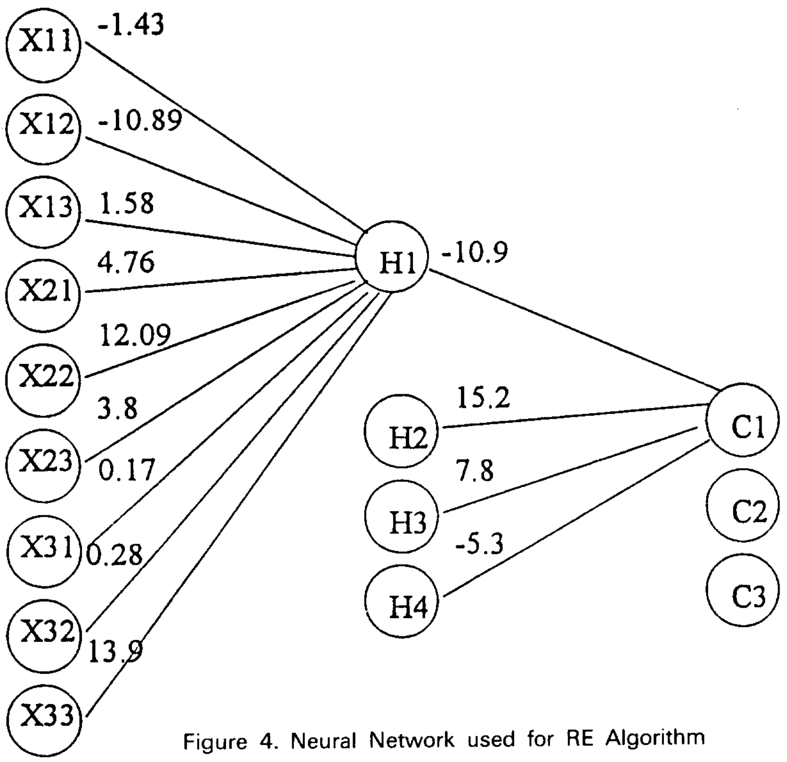


Figure 4. Neural Network used for RE Algorithm

testing case can be very important for the correct classification of the case. The current implementation is to apply the most restrictive rule first (rules that have more antecedents). For the testing cases to which no rule can be applied, the majority class in the training examples set is used as a default class.

To avoid noise, it may not be necessary to match all attributes specified by a rule in order for case to be labeled as a specified class. The RE algorithm has a parameter NMATTR that specifies the minimum number of attributes in a rule case must match. We will discuss the effect of the parameter NMATTR as well as N on the performance of RE algorithm.

4. Experimental Methodology

The objective of experiments in this study is to verify the validity of extracted rules obtained from RE algorithm for explanation and prediction purposes. We compare the performance to those of neural network itself and rules obtained from C4.5 algorithm. Existing rule extraction algorithms are not suitable candidates for comparison mainly because they are restricted in attribute value representation and in input network topology. Moreover, some algorithms require prior domain knowledge (mostly in rule forms) to construct initial input networks so that it is unfair to use rule extraction algorithms using networks built with prior knowledge on rules as a benchmark.

We conducted experiments on 6 different data sets from UCI machine learning databases (Murphy and Aha, 1995) : the Postoperative patient (henceforth referred to as Post), Balloon (the last data set out of the 4 available balloon data sets), Hepatitis, Bupa, Glass, and Iris data sets. The last 3 data sets have only continuous attributes whereas the first 3 data sets have only nominal attributes. For the data sets having continuous attributes, we use 5 even intervals to represent the value category. Tables 1 describes the details

Data Set	# Attribute	Training Set				Testing Set			
	: # Class	Class1	Class2	Class3	Total	Class1	Class2	Class3	Total
Post	8 : 2	42	16	-	58	20	8	-	28
Balloon	4 : 2	5	9	-	14	2	4	-	6
Hepatitis	11 : 2	22	22	-	44	7	7	-	14
Bupa	6 : 2	25	25	-	50	12	12	-	24
Glass	7 : 2	25	25	-	50	12	12	-	24
Iris	4 : 3	34	34	34	102	16	16	16	48

Table 1. Descriptions of Data Sets used for RE Validity Testing

for each data set. We randomly selected cases without missing values into training and testing example sets. To avoid missing values and uneven class distributions, we choose only a subset of available data for some data sets. For the Hepatitis data set, used are only attributes 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, and 20 from the UCI databases. For the Glass data set, used are only attributes 2, 3, 4, 5, 6, 7, and 8 as well as classes 1 and 2.

All experiments were performed on a 486 IBM PC. The C4.5 program is the version from Quinlan (1993) and the backpropagation neural network program is a modified version from Pao (1989). All Programs were written in C and were compiled and executed using Microsoft C. Table 2 records the parameters used for the neural network training and the RE algorithm. The parameters were selected after a series of trial and error runs. The training epochs, learning rate, and momentum rate for neural network training were set at 1000, 0.5, and 0.0 respectively.

Data Set	#Hidden Layers : #Hidden Nodes	NMATTR (Maxima) : N (Maxima)
Post	1 : 3	7 : 18 (8 : 23)
Balloon	1 : 2	2 : 2 (4 : 8)
Hepatitis	1 : 3	7 : 10 (11 : 22)
Bupa	1 : 2	5 : 20 (6 : 30)
Glass	1 : 2	6 : 15 (7 : 35)
Iris	1 : 4	3 : 16 (4 : 20)

Table 2. Neural Network Topology and Paramenters used for RE

5. Experimental Results and Discussions

Table 3 records all training and testing sets' correct prediction rates (henceforth referred to as CPRs).

Data Set	C4.5	Neural Network	RE
Post	71.43	64.29	71.43
Balloon	66.67	83.33	100.00
Hepatitis	64.29	71.43	85.71
Bupa	41.67	50.00	58.33
Glass	83.33	75.00	83.33
Iris	93.75	93.75	93.75

Table 3. Correct Prediction Rates (CPRs in %) for C4.5, Neural Network, and RE

This section describes and discusses the performance difference between neural networks and rules obtained from RE, rules from C4.5 and from RE, sensitivity of RE to parameters NMATTR and N. Since CPRs from testing examples sets are more reliable indicators of performance, we will use holdout method (Taha, Park, and Russell, 1995) in testing the validity of RE algorithm.

Except for Iris data set, rules from RE achieve better CPRs that neural network does. The best improvement from neural network to RE is achieved in the Balloon data set (83.33% to 100%). RE achieves the same CPR as the neural network in Iris testing set. Overall, rules obtained from RE predicts more accurately than neural network does. Compared to rules obtained from C4.5, RE achieves equivalent or better CPRs (achieves the same CPRs as C4.5 does in 3 data sets) .

Figure 5 illustrates sets of rules obtained from C4.5 and RE using Balloon data set. It is noteworthy that RE always generates the same number of rules as the number of classes whereas C4.5 does as many as the branches of its resulting decision tree. Since the size of the rule (as well as the number of clauses in the condition part of the rule) affects the inferencing time, the deterministic characteristic of RE in generating the fixed number of clauses controlled by the parameter N and in producing the same number of rules as the number of the final conclusion states works favorably to the inference efficiency. Moreover, the fact that the effectiveness of rules generated by RE is as good as and sometimes better than that of C4.5 is remarkable.

Both algorithms generate rules having the final conclusion (class) in THEN clause, which do not create inference chain. However, we could reassemble the set of rules by creating intermediate assertion variables in both IF and THEN clauses. This rearrangement can be justified if the resulting decision tree is bushy for C4.5. The trade-off will be the increased number of rules versus the number of argument reduction in the IF clauses. For RE, the rearrangement of rules is possible if the neural network has many hidden layers. In such a case, for every other hidden layer, set of rules can be generated which enables the inference chaining.

Figure 5. Sample Rules Obtained by C4.5 and RE Using Balloon Data Set

The following are the rules generated from C4.5 :

rule1 : IF var4=1 and var2=1, THEN class1
rule2 : IF var4=1 and var2=0 and var1=1, THEN class1
rule3 : IF var4=0 and var3=1, THEN class1

:

rule4 : IF var4=1 and var2=0 and var1=0, THEN class0
rule5 : IF var4=0 and var3=0, THEN class0 (erroneous)

The following are rules generated from the RE :

rule1 : IF var1=0 and var2=0, THEN class0
rule2 : IF var1 <>0 and var2<>0, THEN class1

Figure 6 is a line chart for CPRs from different NMATTR and N for Hepatitis test set. The upper bound of NMATTR and N are 11 and 22 respectively. the optimum CPR (85.71%) is achieved under several combinations of NMATTR and N : (7, 10), (7, 12), (7, 14), (7, 20),and (9, 2). The rule of thumb from experimental experience about these two parameters suggests the values between 50% to 80% of maxima. For the data set having nominal attributes (binary values), surprisingly good results may occur at small values of N. It seems that having information on one category of an attribute is enough to infer the value on the other category. In such a case, rules will have a better prediction power if we compress their size by applying smaller N.

6. Conclusions and Further Research Direction

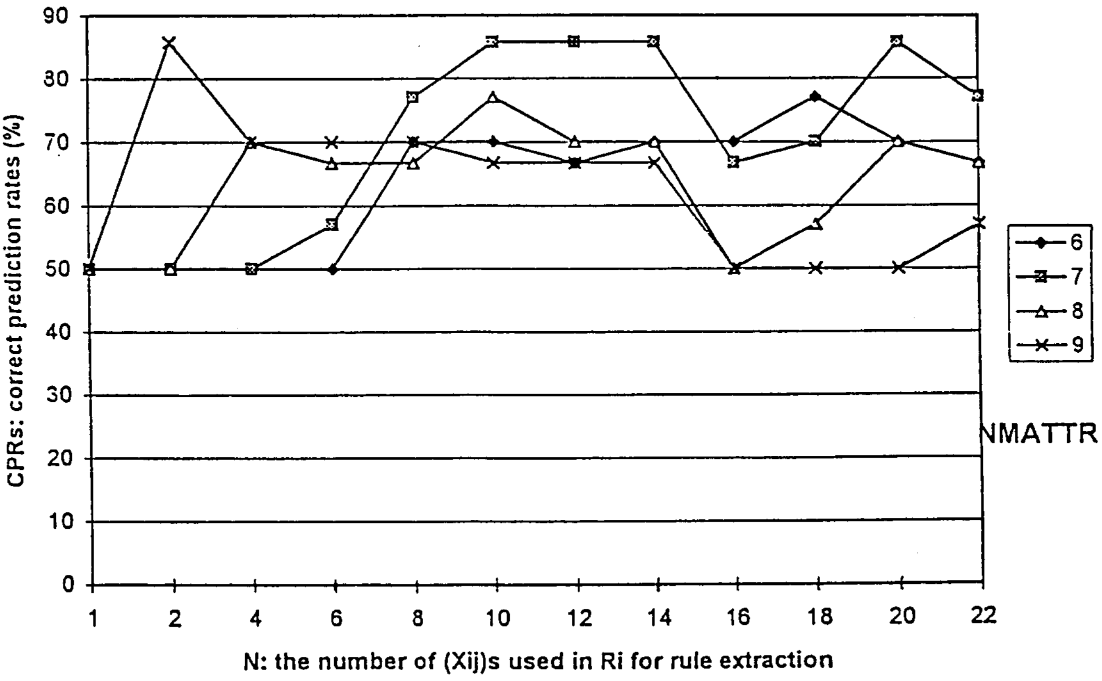
This paper introduces a rule extraction algorithm RE to acquire explicit rules from neural networks. The validity of extracted rules has been confirmed using 6 different data sets (Post, Balloon, Hepatitis, Bupa, Glass, and Iris data set from UCI repository of machine learning databases). The input to the RE algorithm is trained neural networks through backpropagation, and the output is extracted knowledge embodied in symbolic classification rules.

Experimental results from 6 different data sets confirm the validity and efficiency of the RE algorithm. Not to mention the enhanced explanation capability of the network, RE provides following additional advantages : 1) rules obtained from RE algorithm application predicts more accurately and robustly than neural networks themselves as well as rules from inductive learning algorithm C4.5 do, due to its generalized feature avoiding exact fitting through specialization ; and 2) for data sets with attributes having continuous values, RE is more adaptive than C4.5 toward different scaling methods for converting continuous attributes into nominal attributes.

Rule extraction algorithms for neural networks are important for incorporating knowledge obtained from trained networks into knowledge-based systems. In lieu of this, the proposed RE algorithm contributes to the trend toward developing hybrid and versatile knowledge-based systems including expert systems and knowledge-based decision support systems.

The purpose of this paper is an introduction of RE algorithm and its robustness verification. For further research direction, we are to conduct full-fledged level of comparative analysis on rule extraction algorithms of neural network. Moreover, we plan to extend the scope of validation by applying cross-validation method (Taha, Park, and Russell, 1995). Cross-validation method will provide a more robust estimate of accuracy on unseen cases (observations). Since, the holdout method we employed in this paper requires large test set, the validation results may be biased by the number and quality of the test set. Cross-validation can prevent this type of problem by randomly partitioning all available examples (cases or observations) into several train-and-test sets.

Figure 6.CPRs of extracted rules for the Hepatitis testing data set



References

1) Barnard, E. and D. Casasent, "A Comparison between Criterion functions for Linear classifiers, with an Application to Neural Nets", *IEEE Transactions on System, Man, and Cybernetics,* vol. 19, pp. 1030-1041, 1989.

2) Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems,* vol. 2, pp. 303-314, 1989.

3) Decatur, S.E., "Application of Neural Networks to Terrain Classification", *Proceedings of the International Joint Conference on Neural Networks,* vol. 1, pp. 283-288, 1989.

4) Dutta, S. and S. Shekhar, "Bond-Rating : A Non-Conservative Application of Neural Networks", *Proceedings of the IEEE International Conference on Neural Networks,* vol. 2, pp. 443-450, 1988.

5) Fu, L.M., "Rule Learning by Searching on Adapted Nets", *Proceedings of the 9th National Conference on Artificial Intelligence,* pp. 500-595, 1991.

6) Gallant, S.I., "Connectionist Expert Systems", *Communications of the ACM,* vol. 31, pp. 152-169, 1988.

7) Hornik, K., M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks,* vol. 2, pp. 359-366, 1989.

8) Lam, S.M., *Weight Decay Training, Fuzzy set Techniques, and rule Extraction for Backpropagation,* Ph.D. Thesis, School of Business, University of Wisconsin-Madison, 1994.

9) Murphy, P.M., and D.W. Aha, *UCI Repository of Machine Leaning Databases,* Department of Information and Computer Science, University of California-Irvine, 1995.

10) Pao, Y.H., *Adaptive Pattern Recognition and Neural Networks,* Addison-Wesley, 1989.

11) Quinlan, J.R., *C4.5 : Programs for Machine Learning,* Morgan Kaufmann, 1993.

12) Rosenblatt, F., *Principles of Neurodynamics,* Spartan, 1962.

13) Saito, K., and R. Nakano, "Medical Diagnostics Expert System based on PDP Model", *Proceedings of the IEEE International Conference on Neural Networks,* vol. 1, pp. 255-262, 1988.

14) Taha, M., S.C. Park, and J. Russell, "Knowledge Based Systems for Construction Contractor Prescreening", *European Journal of Operational Research,* vol. 84, pp. 35-46, 1995.

15) Tam, K.Y., and M.L. Kiang, "Managerial Applications of Neural Networks : the Case of Bank Failure Predictions", *Management Science,* vol. 38, pp. 926-947, 1992.

16) Tang, Z., C. de Almeida, and P. Fishwick, "Time series Forecasting using Neural Networks vs. Box-Jenkins methodology", *Proceedings of the first Workshop on Neural Networks : Academic/Industrial/NASA/Defense,* pp. 95-100, 1990.

17) Towell, G.G., *Symbolic Knowledge and Neural Networks : Insertion, Refinement, and Extraction,* Ph.D. Thesis, Department Computer Science, University of Wisconsin-Madison, 1991.