Concise Papers

Generalized Analytic Rule Extraction for Feedforward Neural Networks

Amit Gupta,
Sang Park, *Member, IEEE*, and
Siuwa M. Lam

Abstract—This paper suggests the "Input-Network-Training-Output-Extraction-Knowledge" framework to classify existing rule extraction algorithms for feedforward neural networks. Based on the suggested framework, we identify the major practices of existing algorithms as relying on the technique of generate and test, which leads to exponential complexity, relying on specialized network structure and training algorithms, which leads to limited applications and reliance on the interpretation of hidden nodes, which leads to proliferation of classification rules and their incomprehensibility. In order to generalize the applicability of rule extraction, we propose the rule extraction algorithm GeneraLized Analytic Rule Extraction (GLARE), and demonstrate its efficacy by comparing it with neural networks per se and the popular rule extraction program for decision trees, C4.5.

Index Terms—Classification, neural network, rule extraction.

1 Introduction

MANY recent studies have confirmed the effectiveness of neural networks for a variety of applications including classification problems, computer vision, time series analysis, and natural language recognition [4], [5], [9], [17], [18]. Despite the advantages of neural networks such as prediction accuracy, robustness, no requirements on data distribution assumptions, and model-free estimation procedure, there are still difficulties in determining the suitable network architecture, training parameters, and explaining the training results. This paper investigates rule extraction as a remedy for solving the problem of lack of explanation power in neural networks. The basic procedure of rule extraction involves submitting connection weights from a trained neural network into a rule extraction algorithm, and the algorithm generates output in the format of

$$If\ (x_1\ op\ v_{11}\ [or\ v_{12}\ or\ ...])$$

$$and/or\ (x_2\ op\ v_{21}\ [or\ v_{22}\ or\ ...])$$

$$and/or\ ...,\ then\ class_m$$

where x_i is the input attribute i, v_{ij} is the value j for attribute i, the expressions in square brackets are optional, and op is one of the relational operators $=$, $<$, $>$, $\leq$, $\geq$, or $<>$. Extracted rules can explain the classification procedure of the neural network, and shed light on the relative importance of input attributes as well as their relationship on determining a case's class affiliation. In addition to the explanation capability, extracted rules may also have the merit of predicting new cases more accurately than the neural network per se. The above conjecture is based on the principle of minimum description length for theory formulation. A "good" rule extraction algorithm should compress and transform the set of distributed connection weights from the neural network into a set of succinct, essential, and comprehensible rules for explanation and prediction purposes.

The next section of this paper presents an "Input-Network-Training-Output-Extraction-Knowledge" framework to classify existing rule extraction algorithms. The examination of existing rule extraction algorithms identifies several common practices, which motivates the development of a new rule extraction algorithm GeneraLized Analytic Rule Extraction (GLARE). Section 3 describes the specifications of GLARE using the classification framework presented in Section 2. An illustrative example for GLARE is given in Section 4. In order to demonstrate the efficacy of GLARE, experiments are carried out to compare the rule performance of GLARE with neural networks per se, as well as with C4.5, which is a popular rule extraction algorithm for decision trees. Section 5 explains the experimental methodology. Section 6 presents and discusses the experimental results. The last section concludes the paper by discussing some future directions for this research area.

2 A Classification Framework for Rule Extraction Algorithms

Because of the significance of the "lack of explanation" problem in neural networks, we have seen more and more rule extraction algorithms in the literature. There exists the need to organize rule extraction algorithms into perspectives using some classification schemes. Andrews et al. [1] are among the first to propose a scheme, which utilizes five factors to classify rule extraction algorithms including the expressive power of extracted rule, translucency of extraction technique, utilization of specialized training regimes, quality of extracted rules, and algorithmic complexity. However, the suggested scheme has overlapping and missing areas. For example, the expressive power of rules can be considered as a partial indicator of quality of rules, and there is no coverage on algorithms which require specialized network structure and domain knowledge. After examining different rule extraction algorithms, we propose the classification framework "Input-Network-Training-Output-Extraction-Knowledge." The remaining of this section will explain the framework and identify some major practices in existing rule extraction algorithms.

Fig. 1 summarizes the characteristics of some existing rule extraction algorithms in terms of the proposed classification framework. Network input, the first component of the framework, concerns about the input requirement to the network. Input attributes can be boolean, nominal, or continuous; and there can be the input requirement of domain knowledge. Network structure considers whether an algorithm requires specialized network architecture to facilitate the rule extraction process. Training algorithm considers whether a rule extraction algorithm requires specialized training methods. Network output considers whether the output from the trained network has to be preprocessed before submitting as input to the rule extraction process. Extraction process considers the methodology as well as the computational complexity of an algorithm. There are two main types of extraction methodologies: Generate and test, which is search based; and analytic, which is nonsearch based. The analytic approach extracts rules by directly interpreting the strengths of connection weights in

- *A. Gupta is with Andersen Consulting, 3773 Willow Road, Northbrook, IL 60062. E-mail: amit.gupta@ac.com.*
- *S. Park is with the Department of Industrial Management, KAIST, 373-1 Kusong-Dong, Yusong-Gu, Taejon, Korea 305-701. E-mail: sangpark@cais.kaist.ac.kr.*
- *S.M. Lam is with the Department of Management Information Science, California State University–Sacramento, 6000 J Street, Sacramento, CA 95819-6088. E-mail: lamsm@csus.edu.*

Manuscript received 12 June 1997; revised 4 Dec. 1998.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 105247.

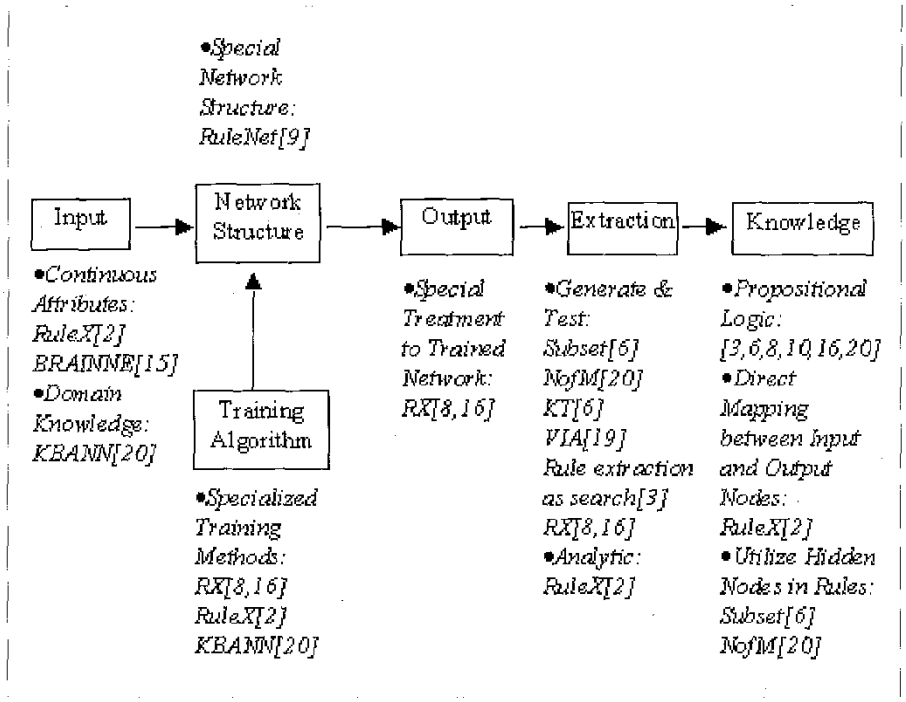

Fig. 1. The framework for classifying rule extraction algorithms.

a trained network. Search-based techniques usually imply high computation complexity of the rule extraction algorithms. The degree of complexity generally increases exponentially as a factor of the numbers of input and hidden nodes. The complexity problem can be alleviated by adopting heuristics [6] to constrain the search space. Extracted knowledge, the last component in the classification framework, considers the representation, quantity, and accuracy of extracted knowledge. The representation of extracted knowledge can be classified as: Symbolic rules vs. numeric function, conjunctive vs. disjunctive in rules, and including hidden nodes vs. direct mapping between input attributes and classes in extracted knowledge. In terms of comprehensibility, a direct mapping between input attributes and classes is easier to trace and apply than rules including hidden nodes. The quantity as a consideration factor for knowledge refers to the number of extracted rules or functions. Generally, without compromising the prediction accuracy, fewer rules and functions imply easier interpretation and application. The accuracy of knowledge, the last consideration factor for extracted knowledge, indicates the prediction accuracy of rules or functions as applied to new cases.

The examination of existing rule extraction algorithms using the "Input-Network-Training-Output-Extraction-Knowledge" framework reveals three major practices of rule extraction. The first practice is the computational complexity caused by the generate and test extraction technique. The general tactic in generate and test technique involves establishing relationships between input and hidden nodes, then relationships between hidden and output nodes, and finally relationships between input and output nodes by merging the first two sets of relationships. The generate and test approach is computationally expensive, which renders it unsuitable for real-time applications. We advocate the analytic approach, extracting knowledge by directly interpreting the strengths of connection weights, as a better approach than generate and test for rule extraction. The second practice concerns about the specialized network structure and training methods for extraction algorithms. There are two major reasons for adopting specialized schemes, either to customize the network to a specific problem domain [9], or to facilitate the extraction process [2], [16], [20]. For example, by modifying the activation function into a threshold function, NofM [20] can focus on interpreting the magnitude of connection weights, and ignore the strengths of nodes' output. While this tactic can simplify the extraction process, it deteriorates the learning power of a network by disabling partially activated hidden nodes. We prefer the utilization of standard network structure and training methods in rule extraction, so as to enhance the generalization power of the algorithms. The last practice involves the existence of hidden nodes in the final rule set. We prefer the approach of direct mapping between input and output nodes, so as to enhance the comprehensibility and applicability of rules. Before we turn to the rule extraction algorithm proposed in this paper, accuracy as a factor for evaluating extracted rules deserves a few words. Fidelity, which is defined as the exactness between the rules' and the network's classification behavior, has been suggested as a criterion of rule quality [2], [3]. Fidelity is a desirable trait for rule extraction algorithms to possess if the trained network predicts accurately. On the other hand, for an inaccurate network, we challenge the rule extraction algorithm to compress the knowledge by eliminating inaccurate, irrelevant, or nonessential information. To evaluate the quality of rules, we suggest that rules should predict as accurately as or more accurately than the network per se.

3 Specifications of GLARE

This section describes the proposed rule extraction algorithm GLARE. Fig. 2 summarizes the specifications of GLARE based on the "Input-Network-Training-Output-Extraction-Knowledge" framework. GLARE is designed for networks with only one hidden layer. The next section provides an illustrative example for the

Input:

- Continuous attributes have to be centered and converted into nominal, then boolean attributes.
- No domain knowledge required.

Network Structure:

- Standard feedforward neural network.
- One hidden layer.
- No restriction on number of hidden nodes.

Training Algorithm:

- Standard backpropagation algorithm.

Network Output as Input to Extraction:

- No special treatment to the set of connection weights from the trained network.

Rule Extraction Process:

For each output node m (each class in the data set), do the following

1. Create $RWIH_n$ which are rankings of all input nodes X_{ij} (category j of input attribute i, $\sum i$ is the total input attributes, $\sum j$ is the total category values for all input attributes) to hidden node n (n = 1, 2, ...N), based on the magnitude of connection weights from input nodes to hidden node n in descending order.
2. Create $RRWIH_n$ which are reduced $RWIH_n$ from step 1, based on the parameter NWIH = a.
3. Calculate $II(H_{mn})$ which are importance indexes for hidden node n to output node m.
4. Create $RWHO_m$ which is the ranking of importance of hidden nodes to output node m in descending order, based on $II(H_{mn})$ from step 3.
5. Create ATTR which is an $N \times a$ matrix consisting of $RRWIH_n$ from step 2, and adjusted based on $RWHO_m$ from step 4. Input nodes on the top rows and left columns in the ATTR are more influential for determining the output of the output node m.
6. Create RATTR which is an $\sum i \times \sum j$ matrix consisting of elements -1, 1 or 0 based on ATTR from step 5.
7. Create the classifcation rule for node m, based on RATTR from step 6.

Output Rules:

- Format: If attribute 0 = category 0 or 1 AND attribute 1 = category 2 or 3 AND ... Then class = 0.
- Symbolic, one composite rule for each class, and each rule as a conjunctive of disjunctives.
- The length of the conjunctive is the number of input attributes, and disjunctive i contains no more than the total categories in attribute i.

Fig. 2. Specifications of GeneraLized Analytic Rule Extraction (GLARE).

algorithm. Fig. 3 provides an example for a fully connected and feedforward neural network trained by backpropagation. For clarity purposes, only part of the connection weights are shown. X_{ij} represents category value j of input attribute i. X_{ij} equals to 1 (0) if attribute i has (does not have) category value j. Σi is the total number of input attributes, and Σj is the total number of category values for all attributes. Note that different attributes may have different numbers of category values. H_n is hidden node n where $n = 0, 1, ..., N$, and N is the total number of hidden nodes. C_m is output node m representing class m in the data set. For each output class m, GLARE performs the following seven steps for extracting the composite rule:

Step (1): Create $RWIH_n$.

For each hidden node n in the network, create ranking $RWIH_n$. $RWIH_n$ is the ranking of all input nodes based on the descending order of absolute values of connection weights between input nodes to hidden node n. A positive or negative sign is added to each input node in $RWIH_n$ to indicate whether the connection weight between the input node and hidden node is positive or negative. The output of step (1) is a set of rankings consisting of N different $RWIH_n$.

Step (2): Create $RRWIH_n$.

This step creates $RRWIH_n$, reduced $RWIH_n$, by reducing the length of $RWIH_n$ from step (1) as determined by the parameter NWIH. NWIH means the number of adopted weights between input and hidden layer. Set the parameter NWIH to a where $1 \leq a \leq \Sigma j$. Some heuristics for choosing NWIH will be discussed in Section 6. Then, the first a input nodes in $RWIH_n$ will be retained for further processing, and remaining input nodes in $RWIH_n$ will be deleted. The output of this step is a set of N different $RRWIH_n$ rankings.

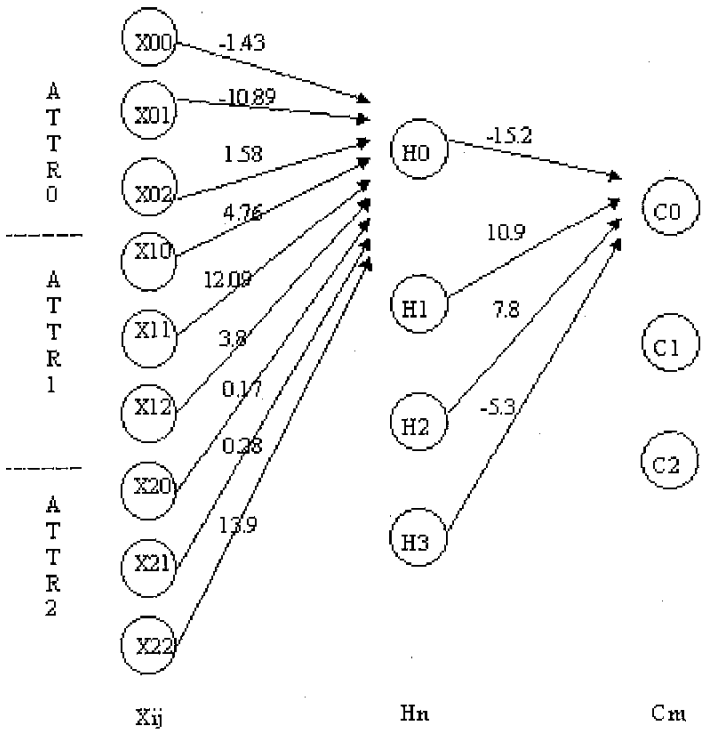

Fig. 3. A trained feedforward neural network.

The purpose of this step is to select several largest connection weights for rule extraction.

Step (3): Calculate importance indexes for hidden nodes.

Resubmit all training cases of class m to the trained network. Notice that the network must be trained before the resubmission. For each hidden node H_n, record the activation level of each resubmitted training case, calculate the average activation level, then calculate the importance index using the following equation:

$$\Pi(H_{mn}) = \mathrm{ABS}(AAL_{mn} * WHO_{mn}) \quad (1)$$

where:

- $\Pi(H_{mn})$ is the importance index of hidden node n to output node m,
- ABS(.) indicates absolute value,
- AAL_{mn} is the average activation level of hidden node n for training cases of class m, and
- WHO_{mn} is the connection weight from hidden node n to output node m.

The purpose of this step is to take into consideration the partial activation level of a hidden node, and thus preserve the learning power of partially activated hidden nodes. The output of step (3) is numeric values $\Pi(H_{mn})$ for all hidden nodes indicating the influential power of each hidden node in determining the output for output node m.

Step (4): Create $RWHO_m$.

Create the ranking $RWHO_m$. $RWHO_m$ is the ranking of all hidden nodes for output node C_m based on the descending order of the importance indexes from step (3). A positive or negative sign is added to each hidden node in $RWHO_m$ to indicate whether the connection weight between the hidden node and output node is positive or negative. The output of step (4) is a ranking of hidden nodes based on their importance on determining the output for output node m.

Step (5): Create ATTR.

$RWHO_m$ (one ranking) from step (4) and $RRWIH_n$ (N rankings) from step (2) are used to construct an $N \times a$ matrix ATTR. ATTR consists of $RRWIH_n$ reordered and adjusted based on $RWHO_m$. First, we reorder $RRWIH_n$ from step (2) according to the order of hidden nodes in $RWHO_m$. Second, for hidden nodes with negative signs in $RWHO_m$, we flip the sign of all input nodes in the corresponding $RRWIH_n$. The rationale of flipping the signs of Xs is explained as follows. For a hidden node which has a negative connection weight to an output node, the output of that hidden node must be low for the output node to generate a high output. Then, in order to have a low output from that hidden node, those input nodes which have negative connection weights to the hidden node must have the input value 1 and those input nodes which have positive connection weights to the hidden node must have the input value 0. This reasoning demands that the signs of all input nodes be reversed for hidden nodes with negative connection weights to the output node. The output of step (5) is a matrix ATTR in which the important input nodes for class m occupy the top rows and left columns. An $+X_{ij}(-X_{ij})$ in ATTR indicates that in order for output node C_m to have a high output (so that a certain case will be classified as class m), input node X_{ij} must have the input value 1 (0).

Step (6): Create RATTR.

Create an $\Sigma i \times \Sigma j$ matrix RATTR based on the ATTR from step (5). Following the directions of top to down and left to right, we use elements in ATTR to determine element values in RATTR. RATTR is initialized to 0. Then, an $+X_{ij}(-X_{ij})$ in ATTR will set category j of attribute i in RATTR to 1(−1). Once an element in RATTR is set, it will not be reset. In other words, less important elements in ATTR have only residual power to determine element values in RATTR. The output of step (6) is a matrix RATTR that can be used to construct the classification rule for class m.

Step (7): Create the classification rule for class m.

Based on element values in RATTR, we construct a composite rule for class m. An element of 1 (−1) in RATTR indicates that attribute i must have (must not have) category value j for class m. An element of 0 indicates that it does not matter whether attribute i has category value j. The current implementation of GLARE treats 0 as 1. The format of the extracted rule is:

If attribute 0 = category 0 or 1 AND
attribute 1 = category 2 or 3 AND ...
Then class = 0.

Each rule is a conjunctive of disjunctives. The length of the conjunctive is the number of input attributes, and disjunctive i contains no more than the total categories in attribute i.

The application order of rules to a new case can be very important for the correct classification of the case. The current implementation is to apply the most restrictive rule first, i.e., the rule with the most −1s in the RATTR matrix. For new cases to which no rule can be applied, the majority class in the training set is used as the default class. To avoid noise, it may not be necessary for a new case to match all attribute values in a rule in order to be labeled as the class indicated by the rule. The GLARE algorithm has a parameter NMATTR that specifies the minimum number of attributes a case must match in order to be classified as the class for the rule. In Section 6, we will discuss some heuristics for determining NMATTR.

4 An Illustrative Example

The network in Fig. 3 has nine input nodes (i = 0, 1, 2, and j = 0, 1, 2 for each i), four hidden nodes (n = 0, 1, 2, 3), and three output nodes (m = 0, 1, 2). To extract the composite rule for class 1, GLARE carries out the following steps:

Step (1): Create $RWIH_n$.

Note that the connection weights for H_1, H_2, and H_3 are not shown in Fig. 1, and are assumed to generate $RWIH_n$ as follows:

$$\begin{aligned}
RWIH_0 &: +X_{22} \; +X_{11} \; -X_{01} \; +X_{10} \; +X_{12} \\
&\quad +X_{02} \; -X_{00} \; +X_{21} \; +X_{20} \\
RWIH_1 &: +X_{00} \; +X_{01} \; +X_{02} \; +X_{10} \; -X_{11} \\
&\quad -X_{12} \; -X_{20} \; -X_{21} \; -X_{22} \\
RWIH_2 &: +X_{00} \; -X_{02} \; +X_{11} \; -X_{20} \; +X_{22} \\
&\quad -X_{01} \; +X_{10} \; -X_{12} \; +X_{21} \\
RWIH_3 &: -X_{22} \; -X_{21} \; -X_{20} \; -X_{12} \; +X_{11} \\
&\quad +X_{10} \; -X_{02} \; -X_{01} \; -X_{00}.
\end{aligned}$$

Step (2): Create $RRWIH_n$.

Suppose we set NWIH to 2. The following are the $RRWIH_n$ for output node 0:

$$\begin{aligned}
RRWIH_0 &: +X_{22} \; +X_{11} \\
RRWIH_1 &: +X_{00} \; +X_{01} \\
RRWIH_2 &: +X_{00} \; -X_{02} \\
RRWIH_3 &: -X_{22} \; -X_{21}.
\end{aligned}$$

Step (3): Calculate importance indexes for hidden nodes.

$\Pi(H_{00})$, $\Pi(H_{01})$, $\Pi(H_{02})$, and $\Pi(H_{03})$ are calculated using (1). Suppose the calculated values are $\Pi(H_{00}) = 10.5, \Pi(H_{01}) = 19.8$, $\Pi(H_{02}) = 9.8$, and $\Pi(H_{03}) = 4.5$.

Step (4): Create $RWHO_m$.

Based on the importance indexes from step (3), the following $RWHO_0$ is constructed:

$$RWHO_0 : +H_1 - H_0 + H_2 - H_3.$$

Note that we attach the positive sign to H_1 and H_2, and the negative sign to H_0 and H_3, based on the signs of connection weights from those hidden nodes to C_0. The above $RWHO_0$ indicates that H_1 is most important for determining the output value of C_0, followed by H_0, then H_2; and H_3 is the least important.

Step (5): Create ATTR.

Using $RWHO_0$ from step (4) and $RRWIH_0$, $RRWIH_1$, $RRWIH_2$, $RRWIH_3$ from step (2), ATTR is generated as:

		column 0	column 1
row	0	$+X_{00}$	$+X_{01}$
	1	$-X_{22}$	$-X_{11}$
	2	$+X_{00}$	$-X_{02}$
	3	$+X_{22}$	$+X_{21}$.

Notice that in the above ATTR, we put $RRWIH_1$ in row 0, $RRWIH_0$ in row 1, $RRWIH_2$ in row 2, and $RRWIH_3$ in row 3, as demanded by the order of hidden nodes in $RWHO_0$. We also flip the signs of the input nodes in $RRWIH_0$ and $RRWIH_3$ because H_0 and H_3 have negative signs in $RWHO_0$.

Step (6): Create RATTR.

Using ATTR from step (5), the following RATTR is constructed:

		Categories(j) 0	1	2
	0	1	1	−1
Attributes(i)	1	0	−1	0
	2	0	1	−1

The procedure of filling the above RATTR is as follows. We start with element 0 in row 0 from ATTR. Since that element is $+X_{00}$, we set category 0 of attribute 0 in RATTR to 1. Then, we use element 1 in row 0 from ATTR, and since that element is $+X_{01}$, we set category 1 of attribute 0 to 1. The filling procedure will go on until we exhaust all elements in ATTR. Notice that since element 0 in row 1 from ATTR has set category 2 of attribute 2 to −1, element 0 in row 3 cannot reset that to 1, according to the residual power principle for elements in ATTR.

Step (7): Create the classification rule for class m.

Using the RATTR from step (6), the following rule for class 0 is constructed:

If attribute 0 = 0 or 1 and
attribute 1 = 1 or 2 and
attribute 2 = 0 or 1
Then class = 0.

TABLE 1
Correct Classification Rates (in Percentages) for Training Sets and Test Sets

	Tree		Tree-Rule		Neural Network		GLARE	
Data Set*	Training	Test	Training	Test	Training	Test	Training	Test
Post - n	72.41	71.43	72.41	71.43	87.93	64.29	72.41	71.43
Balloon - n	64.29	66.67	85.71	50.00	78.57	66.67	78.57	100.00
Hepatitis - n	88.64	78.57	81.82	64.29	97.73	71.43	84.09	85.71
BUPA - c	82.00	58.33	82.00	58.33	98.00	75.00	--	--
Glass - c	96.00	75.00	96.00	66.67	98.00	79.17	--	--
Iris - c	97.06	97.92	97.06	97.92	100.00	95.83	--	--
BUPA - p	70.00	75.00	60.00	41.67	50.00	50.00	52.00	58.33
Glass - p	80.00	87.50	90.00	83.33	92.00	75.00	78.00	83.33
Iris - p	61.77	60.42	94.12	93.75	97.06	93.75	93.14	93.75
BUPA - q	90.00	62.50	72.00	58.33	74.00	58.33	54.00	66.67
Glass - q	78.00	62.50	86.00	58.33	100.00	70.83	94.00	79.17
Iris - q	60.78	62.50	62.75	58.33	98.04	97.92	88.24	93.75

* *n means data sets with nominal attributes.*
c means data sets with continuous attributes.
p means converting continuous attributes into nominal attributes using p scaling.
q means converting continuous attributes into nominal attributes using q scaling.

5 Experimental Methodology

The experiment adopted six data sets from the machine learning databases [10] at the University of California at Irvine (UCI). The Postoperative Patient (Post), Balloon (only the last data set in the repository), and Hepatitis data sets have only nominal attributes. The BUPA, Glass, and Iris data sets have only continuous attributes. To avoid missing values, we deleted attributes with missing values. For Hepatitis, only attributes 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, and 20 were used. For Glass, only attributes 2, 3, 4, 5, 6, 7, and 8 were used. To avoid uneven class distribution, we used only classes 1 and 2 in the Glass data set. Then, we randomly selected 74 cases from Hepatitis as well as from Glass for the experiment. The ratio of training to test cases is 7 : 3 in each data set. To apply GLARE, continuous attributes in the BUPA, Glass, and Iris data sets have to be centered and converted into nominal attributes. We developed two scaling methods, p scaling and q scaling, for the conversion. The method of p scaling is to assign attribute values into five intervals of equal length along the line between the minimum and maximum values for each attribute. The method of q scaling is to assign the first greatest 20 percent attribute values into category 0, the next greatest 20 percent into category 1, and so on. After the conversion, there are 12 different data sets including Post-n, Balloon-n, Hepatitis-n, BUPA-c, Glass-c, Iris-c, BUPA-p, Glass-p, Iris-p, BUPA-q, Glass-q, and Iris-q where n indicates nominal attributes, c indicates continuous attributes, p indicates nominal attributes from p scaling, and q indicates nominal attributes from q scaling. The following experimental procedure was applied to each of the n, p, and q data sets:

1. Use the C4.5 program [13] to build 10 decision trees from the training set. Decision trees are used to predict the test set. Record the best test set CCR (correct classification rate as the percentage of correctly classified cases).
2. Apply the rule extraction procedure from C4.5 to decision trees from step (1). C4.5 chooses the best tree from step (1) based on predicted error rates. Classification rules are generated from the chosen tree. Rules are applied to the training and test set. Record the test set CCR.
3. Convert nominal attributes into binary input attributes for backpropagation training. Apply backpropagation [12] to the training set, and predict test set. Repeat the training 10 times with a new random set of initial weights for each trial. Record the best test set CCR.
4. Apply GLARE to trained networks from step (3). Extracted rules are used to predict the training and test set. Record the best test set CCR.

The above experimental procedure was also applied to data sets with continuous attributes except that step (4) was not carried out since GLARE cannot be applied to continuous attributes. On the other hand, step (2) was performed on data sets with continuous attributes since C4.5 can perform threshold testing for continuous attributes, which will test each midpoint between two adjacent continuous attribute values to select the best threshold for grouping continuous attributes into nominal attributes. The number of hidden nodes is set at about 50 percent to 75 percent of the number of input nodes. All backpropagation training were executed in C, and have 1,000 epochs, 0.5 learning rate, and 0 momentum rate.

6 Experimental Results and Discussion

Table 1 records all training and test set CCRs for experimental steps (1) to (4) described in Section 5. The discussion focuses on test set CCRs. The test set results on nominal data sets (Post, Ballon, Hepatits) show that GLARE achieves the same or higher CCRs than the other methods. In Post data set, GLARE achieves a CCR of 71.43 percent, which is the same as Tree and Tree-Rule, and

TABLE 2
Best Performance on Continuous Data Sets

Data Set	Tree	Tree-Rule	Neural Network	GLARE
BUPA	75.00 - p	58.33 - c,q	75.00 - c	66.67 - q
Glass	87.50 - p	83.88 - p	79.17 - c	83.33 - p
Iris	97.92 - c	97.92 - c	97.92 - q	93.75 - p,q

is higher than Neural Network per se. In Ballon and Hepatitis data sets, GLARE achieves 100 percent and 85.71 percent respectively, which are higher than all other methods. Table 2 presents the best performance on continuous data sets (BUPA, Glass, and Iris). In BUPA, the best performer (75 percent) is Tree (p) and Neural Network (c). GLARE (q) achieves the medium result 66.67 percent. Tree-Rule has the lowest result 58.33 percent. In Glass, the best performer is Tree (p). GLARE (p) and Tree-Rule (p) achieve the same medium result of 83.88 percent. Neural network (c) has the lowest of 79.17 percent. In Iris, Tree (c), Tree-Rule (c), and Neural network (q) achieve the same result of 97.92 percent, and GLARE is 4.17 percent lower than the majority. Among the three continuous data sets, GLARE cannot achieve the best results, and tree is always among the best performers. This phenomenon may be due to the loss of information from converting continuous attributes into nominal attributes for GLARE. Overall, the experimental results show that GLARE outperforms other methods in nominal data sets, but not in continuous data sets. As for the parameters NMATTR and NWIH in GLARE, our experience suggests their values be set at 50 percent to 80 percent of the maxima for the two values.

7 Conclusion

This paper proposes an algorithm GLARE to extract classification rules from feedforward and fully connected neural networks trained by backpropagation. The major characteristics of GLARE include its analytic approach for rule extraction, being applicable to standard network structure and training method, and extracted rules as direct mapping between input and output nodes. Experimental results have shown GLARE's efficacy for prediction comparing with neural networks per se and C4.5 in nominal data sets. The "Input-Network-Training-Output-Extraction-Knowledge" framework also reveals that numeric analysis for extraction process and numeric function as extracted knowledge are underexplored. Recently, we have seen some efforts [11], [14] in these directions. As neural networks are inherently numeric, more research should be directed at designing algorithms for numeric analysis and numeric representation for knowledge in neural networks.

References

[1] R. Andrews, J. Diederich, and A.B. Tickle, "Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373-389, Dec. 1995.

[2] R. Andrews and S. Geva, "Rule Extraction from a Constrained Error Back Propagation MLP," *Proc. Fifth Australian Conf. Neural Networks*, pp. 9-12, 1994.

[3] M.W. Craven and J.W. Shavlik, "Using Sampling and Queries to Extract Rules from Trained Neural Networks," *Proc. 11th Int'l Conf. Machine Learning*, pp. 37-45, 1994.

[4] S.E. Decatur, "Application of Neural Networks to Terrain Classification," *Proc. Int'l Joint Conf. Neural Networks*, vol. 1, pp. 283-288, 1989.

[5] S. Dutta and S. Shekhar, "Bond-Rating: A Non-Conservative Application of Neural Networks," *Proc. IEEE Int'l Conf. Neural Networks*, vol. 2, pp. 443-450, 1988.

[6] L.M. Fu, "Rule Generation from Neural Networks," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1,114-1,124, Aug. 1994.

[7] S.I. Gallant, "Connectionist Expert Systems," *Comm. ACM*, vol. 31, pp. 152-169, 1988.

[8] H. Lu, R. Setiono, and H. Liu, "Effective Data Mining Using Neural Networks," *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 6, pp. 957-961, Dec. 1996.

[9] C. McMillan, M.C. Mozer, and P. Smolensky, "The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network," *Proc. 13th Ann. Conf. Cognitive Science Soc.*, pp. 424-430, 1991.

[10] P.M. Murphy and D.W. Aha, *UCI Repository of Machine Learning Databases*, Dept. of Information and Computer Science, Univ. of California–Irvine, 1997.

[11] H. Narazaki, T. Watanabe, and M. Yamamoto, "Reorganizing Knowledge in Neural Networks: An Explanatory Mechanism for Neural Networks in Data Classification Problems," *IEEE Trans. Systems, Man, and Cybernetics*, part B: cybernetics, vol. 26, no. 1, pp. 107-117, 1996.

[12] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, 1989.

[13] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann 1993.

[14] S. Ridella, G. Speroni, P. Trebina, and R. Zunino, "Pruning and Rule Extraction Using Class Entropy," *Proc. IEEE Int'l Conf. Neural Networks*, vol. 1, pp. 250-256, 1993.

[15] S. Sestito and T.S. Dillon, *Automated Knowledge Acquisition*, Prentice Hall, 1994.

[16] R. Setiono and H. Liu, "Understanding Neural Networks via Rule Extraction," *Proc. Int'l Joint Conf. Artificial Intelligence*, vol. 1, pp. 480-485, 1995.

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

[18] Z. Tang, C. de Almeida, and P. Fishwick, "Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology," *Proc. First Workshop Neural Networks: Academic/Industrial/NASA/Defense*, pp. 95-100, 1990.

[19] S.B. Thrun, "Extracting Provably Correct Rules from Artificial Neural Networks," Technical Report IAI-TR-93-5, Inst. for Informatik III, Universitat Bonn, Germany, 1994.

[20] G.G. Towell and J.W. Shavlik, "The Extraction of Refined Rules from Knowledge Based Neural Networks," *Machine Learning*, vol. 131, pp. 71-101, 1993.