# Recursive Neural Network Rule Extraction for Data With Mixed Attributes

Rudy Setiono, *Senior Member, IEEE*, Bart Baesens, and Christophe Mues

***Abstract*—In this paper, we present a recursive algorithm for extracting classification rules from feedforward neural networks (NNs) that have been trained on data sets having both discrete and continuous attributes. The novelty of this algorithm lies in the conditions of the extracted rules: the rule conditions involving discrete attributes are disjoint from those involving continuous attributes. The algorithm starts by first generating rules with discrete attributes only to explain the classification process of the NN. If the accuracy of a rule with only discrete attributes is not satisfactory, the algorithm refines this rule by recursively generating more rules with discrete attributes not already present in the rule condition, or by generating a hyperplane involving only the continuous attributes. We show that for three real-life credit scoring data sets, the algorithm generates rules that are not only more accurate but also more comprehensible than those generated by other NN rule extraction methods.**

***Index Terms*—Continuous attributes, credit scoring, discrete attributes, rule extraction.**

## I. Introduction

RECENT algorithms that extract rules from neural networks (NNs) have enabled researchers and practitioners to employ NNs as an alternative tool for data mining. Because of their universal approximation property, NNs often achieve accuracy rates that are higher than other methods for classification. Unlike the outputs of NNs that are computed as a complex mapping of the data input attributes, the NN rule extraction algorithms generate comprehensible rules that could be simpler than the rules generated from decision trees. Another possible benefit from using NNs to generate the rules is that in addition to preserving the high accuracy of the networks, these rules could also provide novel insights about the data.

Many rule extraction algorithms have been designed to generate classification rules from NNs that have been trained to distinguish data samples from different classes. These algorithms frequently assume that the input data attributes are discrete in order to make the rule extraction process more manageable. NeuroRule [22] is one such algorithm. A component of NeuroRule is an automatic rule generation method called rule reneration (RG). Each rule is generated by RG such that it covers as many samples from the same class as possible with the minimum number of attributes in the rule condition. RG is applied to generate rules that explain the network's output in terms of the discretized hidden unit activation values and rules that explain the discretized activation values in terms of the discretized attributes of the input data.

Rule eXtraction (RX) [20] is another NN rule extraction algorithm that works on discrete data. RX recursively generates rules by analyzing the discretized hidden unit activations of a pruned network with one hidden layer. When the number of input connections to a hidden unit is larger than a certain threshold, a new NN is created and trained with the discretized activation values as the target output. Otherwise, the rule generation method X2R [14] is applied to obtain rules that explain the hidden unit activation values in terms of the inputs.

Both NeuroRule and RX extract rules from NNs that have been pruned. Removing irrelevant network connections would simplify the rule extraction process and the resulting extracted rules could be expected to be more concise. The method of successive regularization [11] also requires network pruning. Hidden unit activation values in the pruned network are restricted to be binary to speed up rule extraction. The rules generated are hierarchical in the sense that a small number of more dominant rules are generated first to cover most of the data, followed by the less dominant rules.

The generalized analytic rule extraction (GLARE) algorithm [8] does not require network pruning. It does, however, require that the continuous attributes be first converted to nominal, and then to binary attributes before the algorithm can be applied. While the algorithm is shown to work well on discrete data sets, it does not perform as well as decision tree methods on data sets with continuous attributes.

The orthogonal search-based rule extraction (OSRE) method [6] assumes that the data has been 1-from-N encoded, hence its application is also limited to data with only nominal or ordinal attributes. The method was shown to perform well on the three Monks problems [29], however, its accuracy on the Wisconsin Breast Cancer data set is not as good as NeuroRule.

In order to make the rule extraction process more efficient, NN training can be specifically adapted. For example, instead of the usual sigmoid activation function, certainty factor network (CFNet) [7] computes the activation function based on the certainty factor that separates the positive and negative inputs. This property of CFNet allows the attributes that are necessarily present in the rule to be quickly identified. As a result, effective rule extraction can be achieved even from a large complex network.

By restricting both the network connection weights and the inputs to be either $+1$ or $-1$, M-of-N rules [30] can be easily obtained from the network by the M-of-N3 algorithm [23]. The

Manuscript received October 13, 2006; revised February 9, 2007 and May 3, 2007; accepted June 26, 2007.

R. Setiono is with the School of Computing, National University of Singapore, Singapore 117543, Republic of Singapore (e-mail: rudys@comp.nus.edu.sg).

B. Baesens is with the Faculty of Economics and Applied Economics, K. U. Leuven, B-3000 Leuven, Belgium and the School of Management, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: Bart.Baesens@econ.kuleuven.ac.be).

C. Mues is with the School of Management, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: C.Mues@soton.ac.uk).

Digital Object Identifier 10.1109/TNN.2007.908641

algorithm was shown to extract fewer rules than other methods that extract M-of-N rules on some publicly available data sets.

Trepan, an algorithm that was developed by Craven and Shavlik [5] also extracts M-of-N rules from an NN. It treats the NN as an oracle to generate additional data samples. This is an important step in trepan as it grows a decision tree by recursive partitioning. As the tree grows, fewer and fewer training samples are available for deciding if a node should be split further. Additional samples are generated by taking into account the distribution of the existing data samples and their labels are determined by the NN oracle. A node in the tree becomes a leaf node if it has sufficiently a high proportion of samples that belong to one class or if the number of internal nodes in the tree has reached the maximum.

There exist algorithms that generate fuzzy rules from NNs. Nefclass [16] is one such algorithm. It works as a three-layer fuzzy NN. The difference between this type of networks and the standard backpropagation NNs lies in the connection weights which represent fuzzy sets and in the activation functions which act as fuzzy set operators. Nefclass employs a fuzzy variant of the backpropagation algorithm to find the characteristic parameters of the membership functions.

Benitez *et al.* [3] proposed a method for translating the knowledge embedded in an NN into a fuzzy-rule-based system. They showed that building a fuzzy-rule-based system that calculates exactly the same function as an NN is trivial. In order to generate a rule set that is more comprehensible, they introduced the concept of *f-duality*. This concept was applied to the logistic activation function of the network to define the fuzzy logic operator *interactive*-OR. A simple fuzzy-rule-based system involving this operator was presented for the Iris data set. An extension of this work was presented by Castro *et al.* [4]. The improvement to the algorithm allows the user to extract rules from NNs with more than one hidden layer. It also ensures that the continuous attribute values in the fuzzy rules are within their input domains.

A number of algorithms that extract fuzzy rules can be found in the literature. Kolman and Margaliot [12] make use of mathematical equivalence between NNs and a specific fuzzy rule base to extract the knowledge embedded in the networks. The same mathematical equivalence had earlier been applied by the authors to extract rules for several toy problems [13]. An excellent survey of neurofuzzy approaches to rule generation is presented by Mitra and Hayashi [15].

Real-world classification problems usually involve both discrete and continuous input attributes. For such problems, all the continuous attributes must be discretized if algorithms such as NeuroRule, GLARE, and OSRE are to be used for rule extraction. The drawback of discretizing the continuous attributes is that the accuracy of the networks, and hence the accuracy of the rules extracted from the networks, may decrease. This is because discretization leads to a division of the input space into hyperrectangular regions. Each condition of the extracted rules corresponds to one of these hyperrectangular regions where all data samples are predicted to belong to one class. Clearly, a data preprocessing step that divides the input space into rectangular subregions may impose unnecessary restrictions on the NNs as classifiers. It is highly likely that the boundaries of the regions that contain data samples from the same class are nonrectangular given that some of the data attributes are continuous.

There exist rule extraction algorithms from NNs that do not require the discretization of the continuous input data attributes [24], [25]. The conditions of the extracted rules are expressed as linear combinations of the relevant input attributes, normally involving both the discrete and continuous attributes. Such rule conditions may not be very intuitive and do not facilitate understanding of the data.

Building on his earlier works on continuous boolean functions, Tsukimoto [31] developed an NN rule extraction algorithm that could be applied to continuous data directly. Instead of linear combinations of the inputs as the rule conditions, the rules are represented as continuous boolean function of the attributes. The algorithm has been tested only on the Iris data set and the accuracy obtained was not as good as the accuracy of the decision tree method C4.5 [18].

The algorithm full rule extraction (full-RE) [28] is able to extract accurate rules from the Iris data set without the need to binarize or normalize the continuous attributes of the data prior to network training. For each hidden node in the network, full-RE generates an intermediate rule that predicts its output according to the linear combinations of the input attributes. In order to obtain the final set of classification rules that do not involve network weights in their rule conditions, full-RE discretizes the input attributes and then solves a linear programming problem to select the relevant discretization boundary.

A recent rule extraction algorithm that works on discrete and continuous data was proposed by Rabuñal *et al.* [19]. The algorithm applies genetic programming to generate a syntactic tree representing a set of rules that mimics the functioning of the NN. By having randomly generated constants in the terminal and nonterminal nodes of the genetic-programming tree, it is possible to have logical or arithmetical rule conditions for boolean and continuous attributes, respectively. Each node in the tree is also assigned a type such that the generated rules have grammatically correct conditions. There is, however, no provision for generating rules with separate rule conditions involving discrete and continuous attributes.

In this paper, we propose a recursive algorithm for rule extraction from an NN that has been trained for solving a classification problem having mixed discrete and continuous input data attributes. This algorithm shares some similarities with other existing rule extraction algorithms. It assumes that the trained network has been pruned so that irrelevant and redundant network connections and units have been removed. It also makes use of the decision tree method C4.5 [18] to generate rules with only discrete attributes in their conditions. The novel feature of the proposed recursive algorithm lies in the rule set generated. The rules are hierarchical such that only those rules at the deepest level have rule conditions that involve linear combinations of the continuous attributes, while the conditions of all the other rules involve only the discrete attributes. We believe that such rule conditions would greatly increase the comprehensibility of the rules, and hence greatly pave the way to open the NN "black box" wider.

## II. Recursive Rule Extraction: The Re-RX Algorithm

Here, we consider only two-group classification problems, although the algorithm that we are proposing can be easily ex-

tended to handle multigroup problems. The outline of the algorithm is as follows.

---

**Algorithm Re-RX**$(\mathcal{S}, \mathcal{D}, \mathcal{C})$

---

**Input:** A set of data samples $\mathcal{S}$ having the discrete attributes $\mathcal{D}$ and continuous attributes $\mathcal{C}$.

**Output:** A set of classification rules.

1) Train and prune an NN using the data set $\mathcal{S}$ and all its attributes $\mathcal{D}$ and $\mathcal{C}$.

2) Let $\mathcal{D}'$ and $\mathcal{C}'$ be the sets of discrete and continuous attributes still present in the network, respectively. Let $\mathcal{S}'$ be the set of data samples that are correctly classified by the pruned network.

3) If $\mathcal{D}' = \emptyset$, then generate a hyperplane to split the samples in $\mathcal{S}'$ according to the values of their continuous attributes $\mathcal{C}'$ and stop.

Otherwise, using only the discrete attributes $\mathcal{D}'$, generate the set of classification rules $\mathcal{R}$ for the data set $\mathcal{S}'$.

4) For each rule $\mathcal{R}_i$ generated:

If support$(\mathcal{R}_i) > \delta_1$ and error$(\mathcal{R}_i) > \delta_2$, then

• Let $\mathcal{S}_i$ be the set of data samples that satisfy the condition of rule $\mathcal{R}_i$ and $\mathcal{D}_i$ be the set of discrete attributes that do not appear in rule condition of $\mathcal{R}_i$.

• If $\mathcal{D}_i = \emptyset$, then generate a hyperplane to split the samples in $\mathcal{S}_i$ according to the values of their continuous attributes $\mathcal{C}_i$ and stop.

• Otherwise, call Re-RX$(\mathcal{S}_i, \mathcal{D}_i, \mathcal{C}_i)$.

---

In step 1 of the algorithm, any NN training and pruning method can be employed. The algorithm Re-RX does not make any assumption on the NN architecture used, but we have restricted ourselves to backpropagation NNs with one hidden layer as such networks have been shown to possess the universal approximation property [10].

An effective NN pruning algorithm is a crucial component of any NN rule extraction algorithm. By removing the inputs that are not needed for solving the problem, the extracted rule set can be expected to be more concise. In addition, the pruned network also serves to filter noise that might be present in the data. Such noise could be data samples that are outliers or incorrectly labeled. Hence, in step 2 onward, the algorithm processes only those training data samples that have been correctly classified by the pruned network. We have earlier developed an algorithm for NN pruning that incorporates a penalty function during network training [21]. A positive penalty value is added to the sum-of-squared error function for each connection that has nonzero weight. As a result, when network training terminates, many of the network connections have weights that are very close to zero. Network connections with very small values can usually be removed without adverse effect on the accuracy of the network.

If all the discrete attributes are pruned from the network, then in step 3, the algorithm generates a hyperplane

$$\sum_{C_i \in \mathcal{C}'} w_i C_i = w_0$$

that separates the two groups of samples. The constant $w_0$ and the rest of the coefficients $w_i$ of the hyperplane can be obtained by statistical and machine learning methods such as logit regression or support vector machines. In our implementation, we employ an NN with one hidden unit.

When at least one discrete attribute remains in the pruned network, a set of classification rules involving only the discrete attributes is generated. This step effectively divides the input space into smaller subspaces according to the values of the discrete attributes. Each rule generated corresponds to a subspace, and when the accuracy of the rule is not satisfactory, the subspace is further subdivided by the algorithm Re-RX.

The support of a rule is the percentage of samples that are covered by that rule. The support and the corresponding error rate of each rule are checked in step 4. If the error exceeds the threshold $\delta_2$ and the support meets the minimum threshold $\delta_1$, then the subspace of this rule is further subdivided by either calling recursively Re-RX when there are still discrete attributes not present in the conditions of the rule or by generating a separating hyperplane involving only the continuous attributes of the data.

By handling discrete and continuous attributes separately, Re-RX generates a set of classification rules that are more comprehensible than rules that have both types of attributes in their conditions. We illustrate the working of Re-RX in detail in Section III.

## III. Illustrative Examples

To illustrate our algorithm, we use a credit approval data set that was used in a recent benchmarking study comparing different NN models [26]. The data set is publicly available as the CARD data set [17]. There are 690 samples in total, consisting of 345 training samples, 173 cross-validation samples, and 172 test samples. There are altogether 51 attributes, six of which are continuous and the rest binary. We label the continuous input attributes 4, 6, 41, 44, 49, and 51 as $C_4$, $C_6$, $C_{41}$, $C_{44}$, $C_{49}$, and $C_{51}$. The binary attributes are labeled $D_1, D_2, D_3, D_5, D_7, \ldots, D_{40}, D_{42}, D_{43}, D_{45}, \ldots, D_{48}$, and $D_{50}$.

There are three permutations of the data set available. We describe here how our algorithm performs on two of them, CARD2 and CARD3. We first present the working of Re-RX in details using the CARD3 data set for which the pruned NN has only one hidden unit left. We then present the rules extracted by the algorithm from a pruned network with two hidden units for the CARD2 data set.

### A. CARD3 Example

We trained an NN with one hidden layer using the available training and cross-validation samples. The number of input units was 51, and since there were two groups of samples, the number

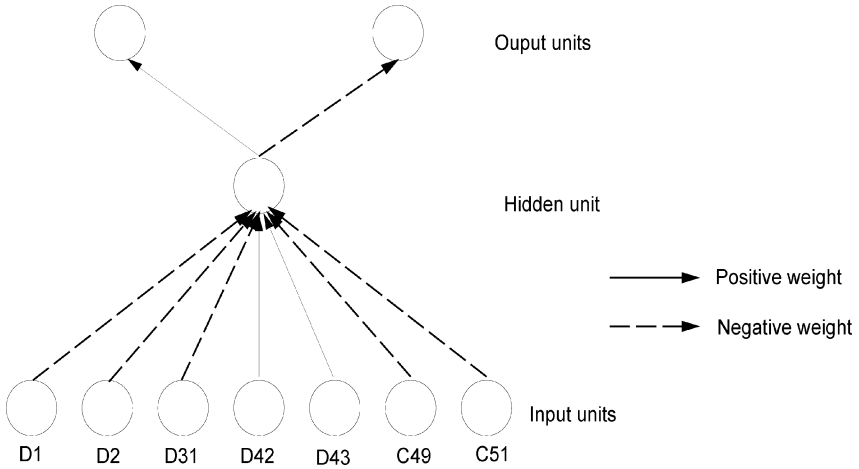


Fig. 1. Pruned NN for the CARD3 data set. It has only one hidden unit, and of the 51 original input units, only seven units remain. The accuracy rates on the training set and test set are 87.26% and 88.95%, respectively.

of output units was two. Once network training stopped, the network was pruned to remove redundant network units and connections. The resulting pruned network is depicted in Fig. 1. This pruned network has very few connections left and its accuracy rate is better than that achieved by other NNs reported by Sexton *et al.* [26]. Of the 518 samples used for training, 87.26% (452 samples) are correctly predicted by the network.

At the start of the Re-RX algorithm, the data set $\mathcal{S}$ consists of 518 samples, the continuous attribute set $\mathcal{C}$ is the set $\{C_4, C_6, C_{41}, C_{44}, C_{49}, C_{51}\}$, and the discrete attribute set $\mathcal{D}$ consists of all the binary attributes. After network training and pruning, we have in step 2 the set $\mathcal{S}'$ consisting of the 452 correctly predicted samples, the set $\mathcal{D}' = \{D_1, D_2, D_{31}, D_{42}, D_{43}\}$, and the set $\mathcal{C}' = \{C_{49}, C_{51}\}$. In step 3, the algorithm generates rules to separate the samples into two groups by applying the C4.5 decision tree method using only the binary attributes. The following set of rules is generated.

Rule $\mathcal{R}_1$: If $D_{41} = 1$ and $D_{43} = 1$, then predict Class 1.

Rule $\mathcal{R}_2$: If $D_{31} = 0$ and $D_{42} = 1$, then predict Class 1.

Rule $\mathcal{R}_3$: If $D_1 = 0$ and $D_{42} = 1$, then predict Class 1.

Rule $\mathcal{R}_4$: If $D_{42} = 0$, then predict Class 1.

Rule $\mathcal{R}_5$: If $D_1 = 1$ and $D_{31} = 1$ and $D_{43} = 0$, then predict Class 2.

Rule $\mathcal{R}_6$: Default rule, predict Class 2.

The number of samples classified by each rule and the corresponding error rates are summarized in Table I.

As rule $\mathcal{R}_5$ incorrectly classifies the highest number of training data samples, we describe how we refine this rule to improve its accuracy. First, the 23 samples classified by this rule are used to train a new NN. The input attributes for this network are $D_2$, $D_{42}$, $D_{43}$, $C_{49}$, and $C_{51}$. When the network is pruned, it turns out that only one hidden unit and two inputs $C_{49}$ and $C_{51}$ are still left unpruned. The coefficients of a hyperplane separating class 1 samples from class 2 samples can then be determined from the network connection weights between the input units and the hidden unit. Based on this, the two classes of samples are separated as follows:

- if $44.65\, C_{49} - 17.29\, C_{51} \leq 2.90$, then predict class 1;
- else predict class 2.

If the thresholds $\delta_1$ and $\delta_2$ were set to 0.05, the algorithm would terminate after generating this rule. For completeness, however, let us assume that these parameters are set to zero. This would force Re-RX to generate more rules to refine rules $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$, and when the algorithm finally terminates, the rules generated would correctly classify all the training samples that have been correctly classified by the original pruned NN in Fig. 1.

The final set of rules generated is as follows.

Rule $\mathcal{R}_1$: If $D_{41} = 1$ and $D_{43} = 1$, then predict class 1.

Rule $\mathcal{R}_2$: If $D_{31} = 0$ and $D_{42} = 1$, then the following rules hold.

- Rule $\mathcal{R}_{2a}$: If $C_{49} \leq 0.339$, then predict class 1.
- Rule $\mathcal{R}_{2b}$: Else predict class 2.

Rule $\mathcal{R}_3$: If $D_1 = 0$ and $D_{42} = 1$, then the following rules hold.

- Rule $\mathcal{R}_{3a}$: If $C_{49} \leq 0.14$, then predict class 1.
- Rule $\mathcal{R}_{3b}$: Else predict class 2.

Rule $\mathcal{R}_4$: If $D_{42} = 0$, then the following rules hold.

TABLE I
SUPPORT LEVEL AND ERROR RATE OF THE RULES GENERATED BY C4.5 FOR THE CARD3 DATA SET USING ONLY THE BINARY-VALUED ATTRIBUTES FOUND RELEVANT BY THE PRUNED NN IN FIG. 1

| Rule | # of samples | Correct classification | Wrong classification | Support (%) | Error (%) |
|---|---|---|---|---|---|
| $\mathcal{R}_1$ | 163 | 163 | 0 | 36.06 | 0 |
| $\mathcal{R}_2$ | 26 | 25 | 1 | 5.75 | 3.85 |
| $\mathcal{R}_3$ | 12 | 9 | 3 | 2.65 | 25 |
| $\mathcal{R}_4$ | 228 | 227 | 1 | 50.44 | 0.44 |
| $\mathcal{R}_5$ | 23 | 13 | 10 | 5.09 | 43.48 |
| $\mathcal{R}_6$ | 0 | 0 | 0 | 0 | - |
| All rules | 452 | 437 | 15 | 100 | 3.32 |

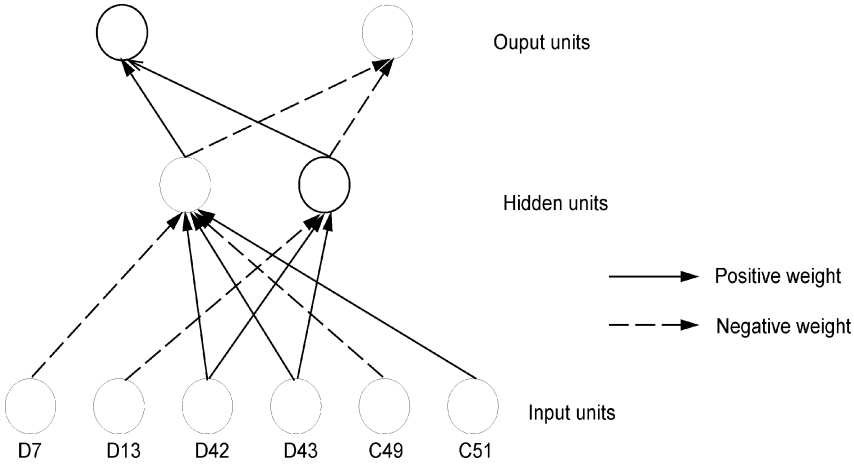


Fig. 2. PNN for the CARD2 data set. Of the initial three hidden units and 51 input units in the network, two hidden units and six input units remain. Its accuracy rates on the training set and test set are 89.38% and 86.05%, respectively.

TABLE II
ACCURACY OF THE PRUNED NETWORK AND OF THE RULES EXTRACTED BY Re-RX FOR THE CARD3 DATA SET

| | Neural network | Re-RX | |
|---|---|---|---|
| | | $\delta_1 = \delta_2 = 0.05$ | $\delta_1 = \delta_2 = 0$ |
| Training set | 87.26% | 86.29% | 87.26% |
| Test set | 88.95% | 88.95% | 88.37% |

TABLE III
COMPARISON OF TEST SET ACCURACY RATES OBTAINED FOR CARD3

| GA | NeuralWorks | NeuroShell | PNN | Re-RX1 | Re-RX2 |
|---|---|---|---|---|---|
| 88.37% | 87.79% | 84.88% | 88.95% | 88.95% | 88.37% |

Rule $\mathcal{R}_{4a}$: If $D_1 = 0$, $D_2 = 2$, $D_{43} = 1$, then predict class 1.

Rule $\mathcal{R}_{4b}$: Else predict class 2.

Rule $\mathcal{R}_5$: If $D_1 = 1$, $D_{31} = 1$, $D_{43} = 0$, then the following rules hold.

Rule $\mathcal{R}_{5a}$: If $44.65\ C_{49} - 17.29\ C_{51} \leq 2.90$, then predict class 1.

Rule $\mathcal{R}_{5b}$: Else predict class 2.

Rule $\mathcal{R}_6$: Default rule, predict class 2.

The accuracy rates of the previous rules and the pruned NN are summarized in Table II. Note that with smaller values for $\delta_1$ and $\delta_2$, the accuracy of the rules is exactly the same as the accuracy of the NN on the training data set. However, the accuracy on the test set is slightly lower as there is one sample that is correctly predicted by the network but not by the rules.

In Table III, we compare our accuracy rates on the test set with the best results reported by Sexton *et al.* [26]. They applied a genetic algorithm (GA)-based NN training technique and two commercial software packages, NeuralWorks and NeuroShell. PNN stands for the accuracy of our pruned NN, while the accuracy rates of Re-RX with $\delta_1 = \delta_2 = 0.05$ and $\delta_1 = \delta_2 = 0$ are shown under Re-RX1 and Re-RX2, respectively. From the figures in the table, we may conclude that our PNN achieves the highest predictive test set accuracy. The network has only one hidden unit and seven input units, and with that, Re-RX is able to generate a simple set of rules that preserves the accuracy of the network. In addition, we believe that the rules are easy to understand as the rule conditions involving discrete attributes are disjoint from those involving continuous attributes, thus allowing different types of conditional expressions to be used for discrete and continuous attributes.

### *B. CARD2 Example*

The PNN for the CARD2 data set depicted in Fig. 2 is selected to illustrate how Re-RX works on a network with more than one hidden unit. We first note that Re-RX falls under the category of pedagogical approaches to NN rule extraction [1]. A pedagogical algorithm does not explicitly analyze the activation values of the network's hidden units. Instead, it considers the network as a "black box" and directly extracts rules that explain the input–output relation of the network. Hence, Re-RX can be expected to perform efficiently even for more complex networks such as networks with many hidden units or more than one hidden layer.

The NN in Fig. 2 obtains accuracy rates of 89.38% and 86.05% on the training and test sets, respectively. Using the 463 correctly classified training samples and the values of the discrete attributes $D_7$, $D_{12}$, $D_{13}$, $D_{42}$, and $D_{43}$, the following rules are obtained.

Rule $\mathcal{R}_1$: If $D_7 = 1$ and $D_{42} = 0$, then predict class 2.

Rule $\mathcal{R}_2$: If $D_{13} = 1$ and $D_{42} = 0$, then predict class 2.

Rule $\mathcal{R}_3$: If $D_{42} = 1$ and $D_{43} = 1$, then predict class 1.

Rule $\mathcal{R}_4$: If $D_7 = 1$ and $D_{42} = 1$, then predict class 1.

Rule $\mathcal{R}_5$: If $D_7 = 0$ and $D_{13} = 0$, then predict class 1.

Rule $\mathcal{R}_6$: Default rule, predict class 2.

All the previous rules except rule $\mathcal{R}_4$ achieve 100% accuracy. Of the 57 samples with $D_7 = 1$ and $D_{42} = 1$, 17 samples actually belong to class 2. Applying Re-RX on this subset of 52 samples with input attributes $D_{12}$, $D_{13}$, $D_{43}$, $C_{49}$, and $C_{51}$, we obtain an NN with one hidden unit and with only two continuous inputs left unpruned. This network correctly separates all 40 class 1 samples from 17 class 2 samples. Using the connection weights of this network, we refine rule $\mathcal{R}_4$ and obtain the following complete set of rules.

Rule $\mathcal{R}_1$: If $D_7 = 1$ and $D_{42} = 0$, then predict class 2.

Rule $\mathcal{R}_2$: If $D_{13} = 1$ and $D_{42} = 0$, then predict class 2.

Rule $\mathcal{R}_3$: If $D_{42} = 1$ and $D_{43} = 1$, then predict Class 1.

Rule $\mathcal{R}_4$: If $D_7 = 1$ and $D_{42} = 1$, then the following rules hold.

Rule $\mathcal{R}_{4a}$: If $20.23C_{49} - 51.42C_{51} \leq 1.40$, then predict class 1.

Rule $\mathcal{R}_{4b}$: Else predict class 2.

Rule $\mathcal{R}_5$: If $D_7 = 0$ and $D_{13} = 0$, then predict Class 1.

Rule $\mathcal{R}_6$: Default rule, predict Class 2.

The accuracy of the rules on the training set is the same as the accuracy of the pruned NN, that is, 89.38%. On the test data set, the accuracy of the rules is slightly higher at 86.63% compared to the 86.05% rate obtained by the pruned network. In Table IV, we show the test set accuracy obtained by our pruned NN, Re-RX and the other NN methods as reported by Sexton *et al.* [26]. As in the case for the CARD3 data set, the rules generated by Re-RX achieve higher predictive accuracy rates than the other methods.

TABLE IV
COMPARISON OF TEST SET ACCURACY RATES OBTAINED FOR CARD2

| GA | Prechelt | NeuralWorks | NeuroShell | PNN | Re-RX |
|---|---|---|---|---|---|
| 84.88% | 81.98% | 81.98% | 81.98% | 86.05% | 86.63% |

## IV. EMPIRICAL EVALUATION ON REAL-LIFE CREDIT SCORING DATA

### A. Data Sets and Empirical Setup

Application scoring is a key financial decision making problem where the aim is to distinguish good payers from defaulters based on the application characteristics provided (e.g., income, employment status, age, etc.). We used three real-life application scoring data sets in the experiments to test the effectiveness of our algorithm. Table V displays the characteristics of these data sets. The German credit data set is an application scoring data set that is publicly available at the UCI repository.[1] The Bene1 and Bene2 data sets were obtained from major financial institutions in Benelux countries (Belgium, the Netherlands and Luxembourg). They contain application characteristics of customers who applied for credit. In accordance with common banking practice and regulations, for these two data sets, a bad customer is defined as someone who has been in payment arrears for more than 90 days at some point in the observed loan history.

Given the rather large number of observations in all data sets, we split them into training set and test set with approximately two to one proportion. The NNs were trained and rules were extracted with the training set. Each continuous attribute in the data required one input unit in the network. A discrete attribute value was converted into a string of binary inputs with the use of either the dummy variable or the thermometer encoding scheme [27]. In general, a discrete variable with $N$ possible values required $N - 1$ input units. As a result, the number of input units in the networks trained on the German, Bene1, and Bene2 data sets were 63, 57, and 76, respectively. As all three problems are binary classification problems, the number of output units was always two. The numbers of input units left after network pruning were 10, 9, and 11, respectively.

[1]www.ics.uci.edu/~mlearn/MLRepository.html

### B. Results

Table VI summarizes the results from applying the Re-RX algorithm. The results from Re-RX were obtained by setting $\delta_1 = \delta_2 = 0$. For comparison purposes, we also include the results from C5.0tree, C5.0rules, Trepan, Nefclass, NeuroLinear and NeuroRule. C5.0 is a more recent and improved version of C4.5. Performance was calculated as percentage of correctly classified (PCC) observations on the training and test data sets.

For each of the three application scoring data sets considered, Re-RX gave the best test set classification accuracy. The differences in accuracy with respect to the other algorithms were significant according to a one-tailed McNemar test at 1% significance level. Notably, unlike C5.0tree and C5.0rules, the difference between training set accuracy and test set accuracy was rather small for Re-RX, clearly illustrating that the algorithm was not too sensitive to overfitting. This could be attributed to the effective input pruning procedure that removed the redundant connections from the network, thereby preventing it from focusing on the noise and idiosyncrasies in the data. Due to its recursive nature, Re-RX tended to generate more rules than the other algorithms. However, since the rules were exhaustive, only one rule was triggered for a new credit applicant, so the higher number of rules would not hinder interpretation. In summary, Re-RX provides an almost ideal balance between classification performance and interpretability in classifying new credit applicants.

Although the performance differences in absolute terms between Re-RX and the other algorithms may seem small at first sight, they are statistically significant for all three application scoring data sets. Moreover, small but statistically significant differences in the discriminatory power of even a fraction of a percentage point may, in a risk management context, translate into significant future savings and/or profit [9].

In the following, we show some of the rules extracted from the various methods to highlight their differences.

1) **A hyperplane rule extracted by NeuroLinear for the German data**
   If [$-24.59$ (checking account) $+29.66$ (term) $-16.45$ (credit history) $-3.66$ (purpose) $-18.69$ (savings account) $+9.29$ (installment rate) $-18.74$ (personal status) $+6.19$ (property) $-10.03$ (age) $-9.36$ (other installment plans) $-11.51$ (housing) $+7.15$ (existing credits) $+16.68$ (job) $+2.046$ (number of dependents) $-4.54$ (telephone) $-8.29$ (foreign worker)] $\leq 0.15$, then customer $=$ good payer. Else customer $=$ defaulter.
2) **A propositional rule extracted by NeuroRule for the Bene1 data**
   If term $> 12$ months and purpose $=$ cash provisioning and savings account $\leq 12.40$ Euro and years client $\leq 3$, then customer $=$ defaulter.

TABLE V
CHARACTERISTICS OF THE DATA SETS. THE LETTERS C, D, AND B STAND FOR CONTINUOUS, DISCRETE, AND BINARY (ENCODED), RESPECTIVELY

| Data set | Inputs | Data set size | Training/Test set size | Goods/Bads (%) |
|---|---|---|---|---|
| German | 7 C, 13 D (56 B) | 1000 | 666/334 | 70/30 |
| Bene1 | 18 C, 9 D (39 B) | 3123 | 2082/1041 | 66.7/33.3 |
| Bene2 | 18 C, 9 D (58 B) | 7190 | 4793/2397 | 70/30 |

3) **A hierarchical rule extracted by Re-RX for the Bene1 data.**
If purpose $=$ cash provisioning and marital status $=$ not married and known client $=$ no, then the following rules hold.

Rule $\mathcal{R}_1$: If owns real estate $=$ yes, then the following rules hold.

Rule $\mathcal{R}_{1a}$: If term of loan $<$ 27 months, then customer $=$ good payer.

Rule $\mathcal{R}_{1b}$: Else customer $=$ defaulter.

Rule $\mathcal{R}_2$: Else customer $=$ defaulter.

4) **A hierarchical rule extracted by Re-RX for the Bene2 data**
If years client $<$ 5 and purpose $\neq$ private loan, then the following rules hold.

Rule $\mathcal{R}_1$: If number of applicants $\geq 2$ and owns real estate $=$ yes, then the following rules hold.

Rule $\mathcal{R}_{1a}$: If savings account $+1.11$ income $-38249.74$ insurance $-0.46$ debt $> -1939300.00$, then customer $=$ good payer.

Rule $\mathcal{R}_{1b}$: Else customer $=$ defaulter.

Rule $\mathcal{R}_2$: If number of applicants $\geq 2$ and owns real estate $=$ no, then the following rules hold.

Rule $\mathcal{R}_{2a}$: If savings account $+1.11$ income $-38249.74$ insurance $-0.46$ debt $> -1638720.0$, customer $=$ good payer.

Rule $\mathcal{R}_{2b}$: Else customer $=$ defaulter.

Rule $\mathcal{R}_3$: If number of applicants $= 1$ and owns real estate $=$ yes, then the following rules hold.

Rule $\mathcal{R}_{3a}$: If savings account $+1.11$ income $-38249.74$ insurance $-0.46$ debt $> -1698200.0$, customer $=$ good payer.

Rule $\mathcal{R}_{3b}$: Else customer $=$ defaulter.

Rule $\mathcal{R}_4$: If number of applicants $= 1$ and owns real estate $=$ no, then the following rules hold.

TABLE VI
ACCURACY AND COMPLEXITY OF DECISION TREES AND NN RULE EXTRACTION TECHNIQUES

| Data set | Method | $PCC_{train}$ | $PCC_{test}$ | Complexity |
|---|---|---|---|---|
| **German** | C5.0tree | 81.98 | 71.26 | 27 leaves |
| | C5.0rules | 80.78 | 70.06 | 14 propositional rules |
| | Trepan | 75.37 | 73.95 | 11 leaves |
| | Nefclass | 73.57 | 73.65 | 14 fuzzy rules |
| | NeuroLinear | 80.93 | 77.25 | 2 oblique rules |
| | NeuroRule | 75.83 | 77.25 | 4 propositional rules |
| | Re-RX | 80.48 | 80.54 | 41 propositional rules |
| **Bene1** | C5.0tree | 78.91 | 71.06 | 35 leaves |
| | C5.0rules | 78.43 | 71.37 | 15 propositional rules |
| | Trepan | 73.05 | 71.85 | 11 leaves |
| | Nefclass | 68.97 | 67.24 | 14 fuzzy rules |
| | NeuroLinear | 77.43 | 72.72 | 3 oblique rules |
| | NeuroRule | 73.05 | 71.85 | 6 propositional rules |
| | Re-RX | 75.07 | 73.10 | 39 propositional rules |
| **Bene2** | C5.0tree | 81.80 | 71.63 | 162 leaves |
| | C5.0rules | 78.70 | 73.43 | 48 propositional rules |
| | Trepan | 74.15 | 74.01 | 11 leaves |
| | Nefclass | 70.06 | 69.80 | 14 fuzzy rules |
| | NeuroLinear | 76.05 | 73.51 | 2 oblique rules |
| | NeuroRule | 74.27 | 74.13 | 7 propositional rules |
| | Re-RX | 75.65 | 75.26 | 67 propositional rules |

Rule $\mathcal{R}_{4a}$: If savings account $+1.11$ income $-38249.74$ insurance $-0.46$ debt $> -1256900.00$, customer $=$ good payer.

Rule $\mathcal{R}_{4b}$: Else customer $=$ defaulter.

The first example illustrates a hyperplane rule extracted by NeuroLinear for the German credit data set. The rule is basically a linear discriminant function extracted from an NN. Observe how the rule mixes both continuous and discrete inputs in its condition, making interpretation very difficult. The second example illustrates a propositional rule extracted by NeuroRule for Bene1. Note how the continuous attributes term, savings account, and years client have been discretized, clearly imposing an unnecessary rectangular restriction on the decision boundary. The last two examples are rules extracted by Re-RX for Bene1 and Bene2, respectively. In each example, the rules involving continuous attributes are disjoint from those involving discrete attributes, and they appear only in the lowest level in the rule hierarchy. As can be seen, both rules are highly interpretable, providing the financial analyst with an explanation facility as to why credit applicants should be rejected or accepted. If needed, the rules can be further transformed into other visualization formalisms such as decision tables and decision diagrams to further facilitate interpretation and application [2].

### *C. Fine-Tuning the Parameters of RE-RX*

The two threshold parameters $\delta_1$ and $\delta_2$ in Re-RX play a crucial role in determining the accuracy as well as the complexity of the rules extracted from NNs. We could expect fewer and simpler rules with larger values for these parameters; however,

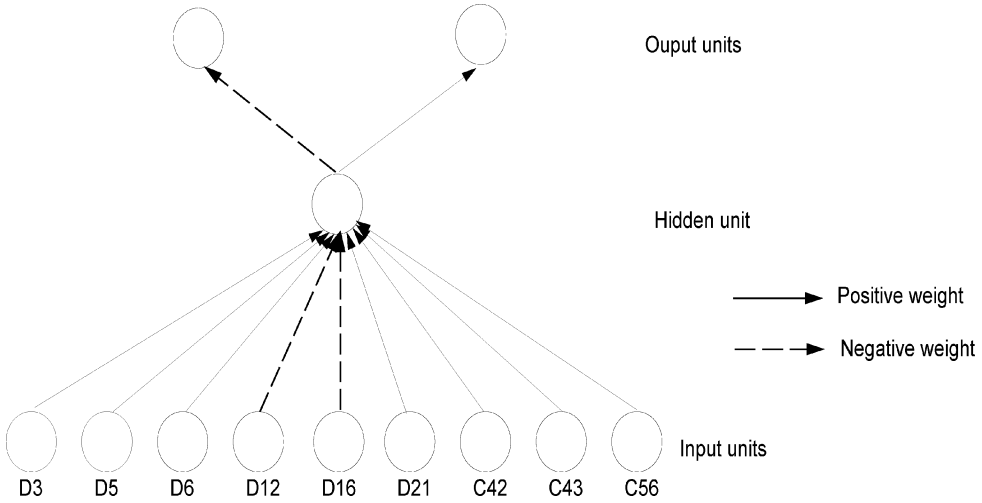


Fig. 3. PNN for the Bene1 data set. D3, D5, D6, D12, D16, and D21 are binary-valued inputs while C42, C43, and C56 are continuously valued inputs.

TABLE VII
SUPPORT LEVEL AND ERROR RATE OF THE RULES GENERATED BY C4.5 USING ONLY THE DISCRETE ATTRIBUTES FOUND RELEVANT BY THE NN

| Rule | Support (%) | Error rate (%) |
|---|---|---|
| $\mathcal{R}_1$ | 11.48 | 5.08 |
| $\mathcal{R}_2$ | 8.30 | 15.63 |
| $\mathcal{R}_3$ | 3.31 | 15.69 |
| $\mathcal{R}_4$ | 38.29 | 2.20 |
| $\mathcal{R}_5$ | 8.96 | 5.07 |
| $\mathcal{R}_6$ | 15.51 | 7.53 |
| $\mathcal{R}_7$ | 5.19 | 35.00 |
| $\mathcal{R}_8$ | 8.96 | 34.06 |

TABLE VIII
ACCURACY AND NUMBER OF RULES EXTRACTED BY Re-RX FOR BENE1 USING VARIOUS VALUES FOR PARAMETERS $\delta_1$ AND $\delta_2$

| $(\delta_1, \delta_2)$ | Accuracy rates | | |
|---|---|---|---|
| | Training set (%) | Test set (%) | # of rules |
| (0,0) | 75.07 | 73.10 | 39 |
| (0.05, 0.02) | 74.78 | 72.62 | 38 |
| (0.02, 0.05) | 75.36 | 73.68 | 33 |
| (0.05, 0.05) | 75.07 | 73.20 | 32 |
| (0.08, 0.08) | 73.49 | 71.57 | 10 |
| (0.10, 0.10) | 72.43 | 70.80 | 8 |

they may not be as accurate as the network from which the rules are extracted. We experimented with various values for these thresholds on the Bene1 data set to investigate the tradeoff between complexity and accuracy.

The PNN is shown in Fig. 3. Its accuracies on the training and test sets were 74.02% and 72.14%, respectively. Six binary-valued input attributes were found to be relevant by the network. When C4.5 was employed to generate classification rules using these attributes, eight rules were generated. The support levels and error rates for these rules are given in Table VII.

Table VIII shows the accuracy and number of rules extracted when different threshold values for $\delta_1$ and $\delta_2$ were used. The relatively large value of 0.08 reduces the number of rules rather significantly. Accuracy on the training and test sets, however, dropped by an average of 2.5%. The figures in the table also indicate that to gain a small incremental increase in accuracy, many more rules are needed. Note that the accuracy of the rules on the training set could be higher than 74.02% obtained by the pruned network, as there were some training data samples incorrectly classified by the network but correctly classified by the rules.

The highest accuracy was obtained when $\delta_1 = 0.02$ and $\delta_2 = 0.05$ with 33 rules generated. In general, to obtain an optimal set of rules, we would recommend the use of cross-validation data samples to find the best parameter values. Assuming that the Bene1 test data samples are employed as cross-validation samples, the rule set with 33 rules should be used for predicting the class label of a new data sample.

## V. DISCUSSION AND CONCLUSION

In this paper, we have developed a new NN rule extraction algorithm Re-RX that is capable of extracting classification rules from NNs trained using both discrete and continuous attributes. The novel characteristic of the algorithm lies in its hierarchical nature of considering discrete variables before continuous variables, in a recursive way. We have illustrated the working of the algorithm using three real-world data sets. More specifically, we have considered application scoring where classification models are built to distinguish good payers from defaulters based on the characteristics provided at the time of application. Two key requirements of application scoring models are performance and interpretability. Especially, the latter is becoming more and more important since financial regulators and law enforcement bodies require all risk management models of a financial institution to be validated. Using real-life application scoring data sets, we have demonstrated that Re-RX is able to extract accurate and interpretable classification rules.

There is an obvious limitation to the applicability of the proposed algorithm, however. When the class labels of the data are best described as a nonlinear function of the continuous inputs, the extracted rules will not be able to preserve the accuracy of the NN. This is a consequence of the property of Re-RX which generates only linear classifiers involving the continuous attributes in order to improve the comprehensibility of the overall rule set. It is possible to preserve the accuracy, for example, by having an NN or other nonlinear classifiers, in place of the linear hyperplane. However, such rules would not be as comprehensible as Re-RX generated rules. The issue of accuracy versus

complexity and comprehensibility of the rules generated from trained NNs certainly deserves further research.

## REFERENCES


[1] R. Andrews, J. Diederich, and A. B. Tickle, "A survey and critique of techniques for extracting rules from trained neural networks," *Knowl.-Based Syst.*, vol. 8, no. 6, pp. 373–389, 1995.

[2] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen, "Using neural network rule extraction and decision tables for credit-risk evaluation," *Manage. Sci.*, vol. 49, no. 3, pp. 312–329, 2003.

[3] J. M. Benitez, J. L. Castro, and I. Requena, "Are neural network black boxes?," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1156–1164, Sep. 1997.

[4] J. L. Castro, C. J. Mantas, and J. M. Benitez, "Interpretation of artificial neural networks by means of fuzzy rules," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 101–116, Jan. 1997.

[5] M. Craven and J. Shavlik, "Extracting tree-structured representations of trained networks," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 24–30.

[6] T. A. Etchells and J. P. G. Lisboa, "Orthogonal search-based rule extraction (OSRE) for trained neural-networks: A practical and efficient approach," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 374–384, Mar. 2006.

[7] L. Fu, "A neural-network model for learning domain rules based on its activation function characteristics," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 787–795, Sep. 1998.

[8] A. Gupta, S. Park, and S. M. Lam, "Generalized analytic rule extraction for feedforward neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 6, pp. 985–991, Nov./Dec. 1999.

[9] W. E. Henley and D. J. Hand, "Construction of a k-nearest neighbor credit-scoring system," *IMA J. Math. Appl. Bus. Ind.*, vol. 8, pp. 305–321, 1997.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1989.

[11] M. Ishikawa, "Rule extraction by successive regularization," *Neural Netw.*, vol. 13, pp. 1171–1183, 2000.

[12] E. Kolman and M. Margaliot, "Knowledge extraction from neural networks using the all-permutations fuzzy rule base: The LED display recognition problem," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 925–931, May 2007.

[13] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 844–852, Jul. 2005.

[14] H. Liu and S. T. Tan, "X2R: A fast rule generator," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, New York, 1995, pp. 1631–1635.

[15] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 748–768, May 2000.

[16] D. Nauck and R. Kruse, "Neuro-fuzzy methods in fuzzy rule generation," in *Fuzzy Sets in Approximate Reasoning and Information Systems*. Norwell, MA: Kluwer , 1999, pp. 305–334.

[17] L. Prechelt, "PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms," Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94.

[18] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufman, 1993.

[19] J. R. Rabuñal, J. Dorado, A. Pazos, J. Periera, and D. Rivero, "A new approach to the extraction of ANN rules and to their generalization capacity through GP," *Neural Comput.*, vol. 16, no. 7, pp. 1483–1523, 2004.

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

[21] R. Setiono, "A penalty-function approach for pruning feedforward neural networks," *Neural Comput.*, vol. 9, no. 1, pp. 185–204, 1997.

[22] R. Setiono and H. Liu, "Symbolic representation of neural networks," *IEEE Computer*, vol. 29, no. 3, pp. 71–77, Mar. 1996.

[23] R. Setiono, "Extracting M-of-N rules from trained neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 512–519, Mar. 2000.

[24] R. Setiono and H. Liu, "Neurolinear: From neural networks to oblique decision rules," *Neurocomputing*, vol. 17, no. 1, pp. 1–24, 1997.

[25] R. Setiono and H. Liu, "A connectionist approach to generating oblique decision trees," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 3, pp. 440–444, Jun. 1999.

[26] R. S. Sexton, S. McMurtrey, and D. J. Cleavenger, "Knowledge discovery using a neural network simultaneous optimization algorithm on a real world classification problem," *Eur. J. Operat. Res.*, vol. 168, pp. 1009–1018, 2006.

[27] M. Smith, *Neural Networks for Statistical Modeling*. New York: Van Nostrand Reinhold, 1993.

[28] I. A. Taha and J. Ghosh, "Symbolic interpretation of artificial neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 3, pp. 448–463, May/Jun. 1999.

[29] S. B. Thrun *et al.*, "The MONK's problems – A performance comparison of different learning algorithm," Carnegie Mellon Univ., Pittsburgh, PA, 1991, CMU-CS-91-197, to be published.

[30] G. G. Towell and J. W. Shavlik, "Extraction of refined rules from knowledge-based neural networks," *Mach. Learn.*, vol. 13, no. 1, pp. 71–101, 1993.

[31] H. Tsukimoto, "Extracting rules from trained neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 377–389, Mar. 2000.


**Rudy Setiono** (SM'01) received the B.S. degree from Eastern Michigan University, Ypsilanti, and the M.Sc. and Ph.D. degrees from the University of Wisconsin-Madison, in 1984, 1986, and 1990, respectively, all in computer science.

He has been with the National University of Singapore (NUS), Singapore, since 1990, where he is currently an Associate Professor. He served as Vice Dean (undergraduate affairs) from November 2001 to July 2005 at the School of Computing, NUS. His research interests include linear programming, nonlinear optimization, and neural networks.

Dr. Setiono was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2000 to 2005.

**Bart Baesens** received the M.Sc. and Ph.D. degrees in applied economic sciences from the K. U. Leuven, Leuven, Belgium, in 1998 and 2003, respectively.

Currently, he is an Assistant Professor at the Vlerick Leuven Ghent Management School, K. U. Leuven, and a Lecturer at the University of Southampton, Southampton, U.K. His research interests include classification, rule extraction, neural networks, support vector machines, data mining, and credit scoring.

**Christophe Mues** received the degree of Doctor in applied economics from the K. U. Leuven, Leuven, Belgium, in November 2002.

Currently, he is a Lecturer (Assistant Professor) at the School of Management of the University of Southampton, Southampton, U.K. He was a Researcher at K. U. Leuven, Leuven, Belgium. His key research interests are in the business intelligence domain, where he has investigated the use of decision table and diagram techniques in a variety of problem contexts, such as business rule modeling and validation, and knowledge discovery and data mining. In addition, he has developed a strong interest in applying data mining techniques to credit risk management and credit scoring in particular. His findings have been published in various international journals and conference proceedings.