NeuroLinear: A System for Extracting Oblique Decision Rules from Neural Networks

Rudy Setiono and Huan Liu

Department of Information Systems and Computer Science
National University of Singapore
Singapore 116290

Abstract. We present NeuroLinear, a system for extracting oblique decision rules from neural networks that have been trained for classification of patterns. Each condition of an oblique decision rule corresponds to a partition of the attribute space by a hyperplane that is not necessarily axis-parallel. Allowing a set of such hyperplanes to form the boundaries of the decision regions leads to a significant reduction in the number of rules generated while maintaining the accuracy rates of the networks. We describe the components of NeuroLinear in detail using a heart disease diagnosis problem. Our experimental results on real-world datasets show that the system is effective in extracting compact and comprehensible rules with high predictive accuracy from neural networks.

1 Introduction

Neural networks have been widely used to solve classification problems. Comparisons between neural networks and decision trees algorithms for these problems have shown that in general neural networks can produce better accuracy rates [1, 2, 3, 4]. Recent developments in algorithms that extract rules from neural networks have made neural network techniques even more attractive. The extracted rules allow one to explain the decision process of a neural network. It is not surprising that in the past few years great efforts have been devoted to finding efficient and effective algorithms for extracting rules from a trained neural network

Gallant's connectionist expert systems [5] and Saito and Nakano's RN method [6] are two early works that attempt to generate rules from neural networks. These systems, however, do not actually extract rules from the networks; instead, they try to generate an explanation for each particular outcome of the networks.

The KT algorithm developed by Fu [7] is an algorithm that extracts rules from a trained network. It searches for subsets of connections to a network's unit with summed weight exceeding the bias of that unit. It is assumed that the unit's activation value is close to either 0 or 1. By searching for the proper subsets of the input connections, sets of rules are generated to describe under what conditions the unit's activation will take one of the two values.

The MofN algorithm of Towell and Shavlik [8] clusters the weights of the trained network into equivalence classes. Clusters that do not significantly affect

the unit's activation are eliminated. The complexity of the network is further reduced by replacing the weights in all remaining clusters by the average weight of the individual cluster. The rules are generated in a similar way as by Fu's KT algorithm. However, because of the creation of clusters with averaged weights and the removal of unneeded clusters, it can be expected that the rules extracted will be simpler than those of KT algorithm.

An algorithm that extracts rules from networks that have been trained via the backpropagation method with penalty is proposed by Blassig [9]. A penalty term is added to the network's error function with the hope that a large number of the weights will have zero values after training and thus can be removed. In order to force the activation value of a unit to take on the value of either 0 or 1, the network's weights are restricted to be -6, 0, or 6, and the unit's bias to be either -3 or 3.

A simple rule extraction algorithm is presented by Setiono and Liu [10]. It is claimed that the rules extracted from neural networks are comparable to those generated by decision trees [11] in terms of accuracy and comprehensibility. The basic idea behind the algorithm is the fact that it is generally possible to replace the continuous activations of the hidden units by a small number of discrete ones. Rule extraction is realized in two steps. First, rules that describe the network outputs in terms of the discretized activation values of the hidden units are generated. Second, rules that describe each discretized hidden unit activation values in terms of the network inputs are constructed. By merging the rules obtained in these two steps, a set of rules that relates the inputs and outputs of the network is obtained.

While the algorithm can generate symbolic rules that mimic the predicted outcome of the original network, it works only for data with binary inputs (the implicit assumption, that the various hidden unit activation values are determined by only a small number of input values, excludes problems with continuous attributes where there can be infinitely many possible values taken by these attributes). Data with some continuous attributes need to be preprocessed before training the network. The preprocessing of the data entails dividing the values of the continuous attributes into intervals. This is achieved by the ChiMerge algorithm [12]. ChiMerge employs the χ^2 statistic to determine the division of the original intervals into their respective subintervals. All continuous attribute values that fall in the same subinterval are then represented by a unique discrete value for use as inputs to the neural network. Rules involving the discrete representations of the inputs are generated from the network. Given the boundaries of the subintervals furnished by ChiMerge, getting the rules in terms of the original continuous attributes is trivial.

An inherent problem introduced by the preprocessing of the data by ChiMerge is that each condition of a rule involving a continuous attribute determines an axis-parallel decision boundary. For many classification problems, it is often more natural to allow oblique hyperplanes to form the boundaries of the decision regions. In other words, instead of imposing the axis-parallel constraint, being able to generate oblique decision hyperplanes makes it possible to let the learning al-

gorithm determine what kind of hyperplanes is more suitable for the data in hand. Oblique hyperplanes are more general and they may substantially reduce the number of rule conditions needed to describe the decision region. As a result, a more compact and comprehensible set of rules can be obtained.

In this paper, we describe how a set of rules, where each rule condition is given in the form of the linear inequality

$$\sum_i c_i x_i < \eta \tag{1}$$

where c_i is a real coefficient, x_i is the value of the attribute i, and η is a threshold, can be extracted from a neural network. The neural network that we use is the standard feedforward neural network with a single hidden layer. In contrast to the tree growing algorithms which generate the rules in a level-by-level and top-down fashion, rules are extracted from a network in two steps: from the hidden layer to the output layer and from the input layer to the hidden layer. Classification rules are obtained by merging the rules from these two steps. Unlike the decision tree algorithms [13, 11] which consider smaller and smaller subsets of the data to improve the accuracy of the rules, the neural network approach for rule generation considers the entire training set as a whole. Our experimental results show that more compact sets of rules with high accuracy rates can be obtained by the network approach.

The organization of the paper is as follows. Section 2 describes NeuroLinear, a system that we develop for extracting oblique decision rules from a neural network. The steps are described in detail by way of an example using a real-world dataset. Section 3 analyzes the merits of generating a set of classification rules with NeuroLinear. It also highlights the differences between the rules generated from a network and those from the decision-tree method C4.5 ([11]). Section 4 presents our experimental results on several real-word problems. Section 5 gives a brief conclusion of the paper.

2 Rule extraction with NeuroLinear

The process of extracting oblique decision rules from a neural network can be summarized in the following steps:

1. Select and train a network to meet a prespecified accuracy requirement.
   Remove the redundant connections in the network by pruning while maintaining the accuracy.
2. Discretize the hidden unit activation values of the network.
3. Extract rules that describe the network outputs in terms of the discretized network activation values.
   For each discretized hidden unit activation value, generate a rule in terms of the network's inputs.
   Merge the two sets of rules obtained above.

We shall use the Cleveland heart disease dataset to show the workings of NeuroLinear in detail.

2.1 Neural network training and pruning

The Cleveland heart disease data set can be obtained via anonymous ftp from the repository at the University of California, Irvine [16]. It consists of 303 patterns. Because we have not implemented a way to handle patterns with missing attribute values, we discard such patterns and use only the remaining 297 patterns. Each pattern is described by 13 attributes, 5 of which are continuous and the remaining 8 discrete. The attributes are summarized in Table 1. Each of the pattern is labeled either positive (presence of heart disease) or negative (absence of heart disease).

Table 1. The attributes of the Cleveland heart disease data set and network input units assigned to them.

Attribute	Type	Possible values/Range	Network inputs
$\mathcal{C}_1$	Continuous	[29, 77]	$\mathcal{I}_1$
$\mathcal{D}_2$	Discrete	0 or 1	$\mathcal{I}_2$
$\mathcal{D}_3$	Discrete	1,2,3 or 4	$\mathcal{I}_3, \mathcal{I}_4, \mathcal{I}_5, \mathcal{I}_6$
$\mathcal{C}_4$	Continuous	[94, 200]	$\mathcal{I}_7$
$\mathcal{C}_5$	Continuous	[126, 564]	$\mathcal{I}_8$
$\mathcal{D}_6$	Discrete	0 or 1	$\mathcal{I}_9$
$\mathcal{D}_7$	Discrete	0,1 or 2	$\mathcal{I}_{10}, \mathcal{I}_{11}, \mathcal{I}_{12}$
$\mathcal{C}_8$	Continuous	[71, 202]	$\mathcal{I}_{13}$
$\mathcal{D}_9$	Discrete	0 or 1	$\mathcal{I}_{14}$
$\mathcal{C}_{10}$	Continuous	[0, 6.2]	$\mathcal{I}_{15}$
$\mathcal{D}_{11}$	Discrete	1,2 or 3	$\mathcal{I}_{16}, \mathcal{I}_{17}, \mathcal{I}_{18}$
$\mathcal{D}_{12}$	Discrete	0,1,2 or 3	$\mathcal{I}_{19}, \mathcal{I}_{20}, \mathcal{I}_{21}, \mathcal{I}_{22}$
$\mathcal{D}_{13}$	Discrete	3,6 or 7	$\mathcal{I}_{23}, \mathcal{I}_{24}, \mathcal{I}_{25}$

Since the problem is a binary classification problem, one output unit is sufficient for the network. Positive patterns are given the target 1 and negative patterns 0. A hidden layer of 4 units is found to be sufficient to give the network a comparable accuracy rate to those reported in the literature. One input unit is assigned to each continuous attribute. Values of the continuous attributes are normalized such that their range is $[0, 1]$. Each value of a discrete attribute with n possible values is represented by an n-bit binary string, except when $n = 2$, only 1 bit is used (see the last column of Table 1). With an additional input unit to represent the hidden unit bias values, the total number of the network's input units is 26.

Given an n-dimensional input pattern x_i, the predicted output of the network at output unit p is computed as

$$S_{pi} = \sigma\left(\sum_{j=1}^{h} \delta\left((x_i)^T w_j\right) v_{pj}\right) \tag{2}$$

where

- h is the number of hidden units in the network.
- w_j is an n-dimensional vector of weights for the arcs connecting the input layer and the j-th hidden unit, $j = 1, 2, \ldots, h$. The weight of the connection from the ℓ-th input unit to the j-th hidden unit is denoted by $w_{j\ell}$.
- v_j is a C-dimensional vector of weights for the arcs connecting the j-th hidden unit and the output layer. The weight of the connection from the j-th hidden unit to the p-th output unit is denoted by v_{pj}.
- C is the number of output units.

The activation function of the hidden units is the hyperbolic tangent function

$$\delta(y) = (e^y - e^{-y})/(e^y + e^{-y}), \tag{3}$$

while the activation function of the output units is the sigmoid function

$$\sigma(y) = 1/(1 + e^{-y}). \tag{4}$$

The hyperbolic tangent function allows the activation values of the hidden units to be in the interval $[-1, 1]$, while the range of the sigmoid function used at the output units is $[0, 1]$.

The network is trained by minimizing the cross-entropy error function with weight decay [14]:

$$\theta(w, v) = -\sum_{i=1}^{k}\sum_{p=1}^{C}(t_{pi}\log S_{pi} + (1 - t_{pi})\log(1 - S_{pi})) + P(w, v), \tag{5}$$

where k is the number of patterns in the training dataset, t_{pi} is the binary-valued target output for pattern x_i at output unit p, and $P(w, v)$ is the penalty term:

$$P(w, v) = \epsilon_1 \sum_{j=1}^{h}\left(\sum_{\ell=1}^{n}\frac{\beta w_{j\ell}^2}{1 + \beta w_{j\ell}^2} + \sum_{p=1}^{C}\frac{\beta v_{pj}^2}{1 + \beta v_{pj}^2}\right) + \epsilon_2 \sum_{j=1}^{h}\left(\sum_{\ell=1}^{n} w_{j\ell}^2 + \sum_{p=1}^{C} v_{pj}^2\right), \tag{6}$$

(ϵ_1 and ϵ_2 are two positive penalty parameters and $\beta > 0$).

The penalty term $P(w, v)$ is added to the cross-entropy error function so that relevant connections can be distinguished from those irrelevant ones by the their magnitude. After training, it can be expected that the irrelevant connections will have small magnitude and therefore, they can be removed from the network with little or no effect on the accuracy of the network. Pruning is achieved by removing those connections with small magnitude. We train and prune the network such that it achieves an accuracy rate of at least 85 % on the training set.

Figure 1 depicts a pruned network for the Cleveland heart disease problem. The network has been trained with 90% of the patterns randomly picked from the dataset. The remaining 10% of the patterns are used to test the network's generalization capability. The pruned network has only 2 hidden unit and 10 connections left. Only one input, $\mathcal{I}_2$, is connected to the first hidden unit. A total of seven inputs are connected to the second hidden unit. The accuracy rates of this network are 86.14 % on the training data and 80.00 % on the testing data.

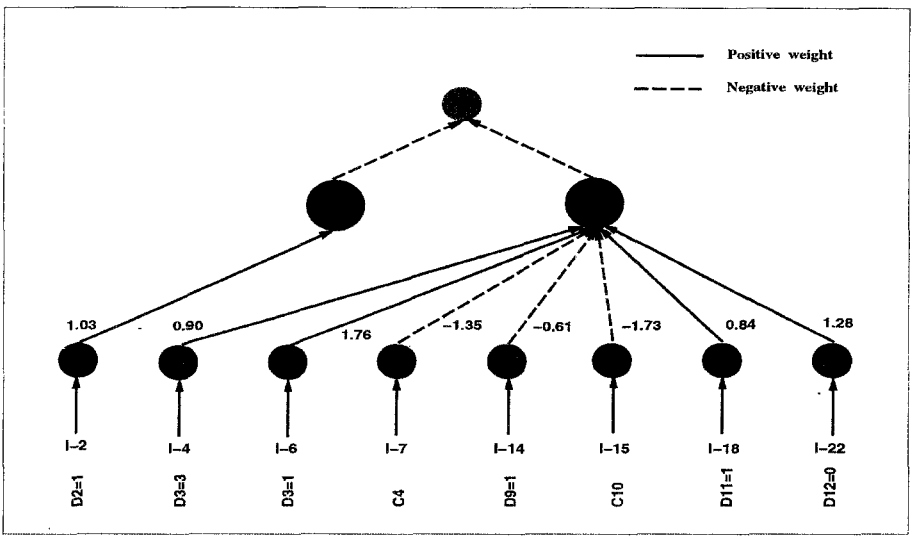

Fig. 1. A network with 10 connections and only 2 hidden units for the Cleveland heart disease dataset. The accuracy rates on the training set and the testing set are 86.14 and 80.00 %, respectively. The discrete input of the network is 0 or 1 depending on the original value of the corresponding nominal attributes (labeled with $\mathcal{D}$). For example, the second input $\mathcal{I}_4$ is equal to 1 if and only if the value of the attribute $\mathcal{D}_3$ is 3. The two inputs $\mathcal{I}_7$ and $\mathcal{I}_{15}$ correspond to the continuous attributes $\mathcal{C}_4$ and $\mathcal{C}_{10}$, respectively (see Table 1).

2.2 Chi2: Discretization of hidden unit activation values

The range of the activation values of the network's hidden units is the interval $[-1, 1]$, since they have been computed as the hyperbolic tangent of the weighted inputs (cf. Function 3). In order to extract rules from the network, it is better that these activation values be grouped into a small number clusters while at the same time preserving the accuracy of the network. We can expect to have a more concise set of rules if the number of clusters required is smaller. Chi2, an improved and automated version of ChiMerge, is the algorithm used for this purpose.

Given a dataset where each pattern is described by the values of the continuous attributes $\mathcal{A}_1, \mathcal{A}_2, \ldots$ and the class label of each pattern is known, Chi2 finds discrete representations of the patterns. Using the χ^2 statistic, Chi2 divides the range of the attributes into subintervals and assigns all values that fall in a subinterval a unique discrete value. The merging of adjoining subintervals takes place one at a time and attribute by attribute in a round robin fashion. This is intended to eliminate the ordering bias of attributes. The output of Chi2 is discretized data that preserve the discriminating power of the original data. Irrelevant attributes can be detected by their single intervals at the end of discretization and they can be removed. The outline of the algorithm is as follows:

The Chi2 algorithm

1. Let Chi-0 be an initial critical value.
2. For each attribute $\mathcal{A}_i$:
   (a) Sort the data according to the the input values of attribute i.
   (b) Form an initial set of intervals such that each interval contains only one unique value.
3. Initialize all attributes as "unmarked".
4. For each unmarked attribute $\mathcal{A}_i$:
   (a) For each adjoining pair of subintervals, compute the χ^2 values:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{7}$$

   where:
   k: the number of classes,
   A_{ij}: the number of samples in the ith interval, jth class,
   R_i: the number of samples in the ith interval $= \sum_{j=1}^{k} A_{ij}$.
   C_j: the number of samples in the jth class $= \sum_{i=1}^{2} A_{ij}$.
   E_{ij} : expected frequency of A_{ij}, $E_{ij} = R_i \times C_j / N$. If R_i or $C_j = 0$, E_{ij} is set to 0.05,
   N: the total number of samples.
   (b) Find two subintervals with the lowest χ^2 value. If this value is less than Chi-0 and if merging the subintervals does not introduce conflicting data[1], then merge these subintervals, and repeat from Step 4(a). Else if merging the subintervals will introduce conflicting data, mark attribute i.
5. If there is still an unmarked attribute, then increase Chi-0 and repeat Step 4.

Only one parameter is required for Chi2. This parameter is the initial critical value Chi-0. This value is used to determine if the null hypothesis that the subintervals and the class labels are independent can be rejected. If the test statistic (Eqn. 7) exceeds the critical value Chi-0, the null hypothesis is rejected. Otherwise, the null hypothesis is not rejected and the subintervals are merged. The critical value Chi-0 is determined by α, the significance level of the test. For example, if $\alpha = 0.50$ and the number of classes in the data is 3, the critical value is 1.386. If no inconsistency in the data is introduced after merging of the subintervals with the initial value of Chi-0, the critical value Chi-0 is increased, i.e., the significance level is reduced to check the possibility of further merging of subintervals.

For the activation values of the two hidden units of network depicted in Figure 1, the subintervals found by Chi2 are:

[1] Conflicting data occur when there are two or more patterns from different classes with the same discretized attribute values.

- Hidden unit 1: 2 subintervals $[0.0, 0.77), [0.77, 1]$.
- Hidden unit 2: 3 subintervals $[-1.0, 0.05), [0.05, 0.47), [0.47, 1]$.

A pattern can thus be represented by a pair of integers (i, j) where i and j denote the subintervals into which its first and second hidden unit activation values fall, respectively. If no conflicting data is allowed in Chi2, it is possible to obtain rules that preserve the accuracy rate of the network for the original dataset using their discrete representations.

2.3 Rule extraction

Rules are first generated to describe the network outputs in terms of the discretized hidden unit activation values. Since there are only 2 intervals in the first hidden unit and 3 intervals in the second hidden unit found by Chi2, a maximum of only 6 combinations are possible. A set of simple rules will be able to distinguish the positive patterns from the negative ones. The rules are obtained by X2R [15], an algorithm that generates a set of rules with 100 % accuracy (provided that there are no conflicting patterns) from data with discrete inputs. Using the notations i and j to denote the activation values at the hidden units, the rules are:

- Rule 1: If $j = 3$, then Negative.
- Rule 2: If $i = 1$ and $j = 2$, then Negative.
- Default rule: Positive.

Rules in terms of the inputs that describe each possible values of i and j can be obtained using the weights of the network (see Figure 1). For the first hidden unit, we have that

$$1.03\, \mathcal{I}_2 < \delta^{-1}(0.77) = 1.02 \Leftrightarrow i = 1$$
$$1.03\, \mathcal{I}_2 \geq \delta^{-1}(0.77) = 1.02 \Leftrightarrow i = 2$$

($\delta^{-1}(.)$ is the inverse of the hyperbolic tangent function (3)). Since $\mathcal{I}_2$ is binary-valued, the above two rules can be simplified as: $\mathcal{I}_2 = 0$ if and only if $i = 1$. For the second hidden unit, the conditions are:

$$0.90\, \mathcal{I}_4 + 1.76\, \mathcal{I}_6 - 1.35\, \mathcal{I}_7 - 0.61\, \mathcal{I}_{14} - 1.73\, \mathcal{I}_{15} + 0.84\, \mathcal{I}_{18} + 1.28\, \mathcal{I}_{22} < 0.05 \Leftrightarrow j = 1$$
$$0.05 \leq$$
$$0.90\, \mathcal{I}_4 + 1.76\, \mathcal{I}_6 - 1.35\, \mathcal{I}_7 - 0.61\, \mathcal{I}_{14} - 1.73\, \mathcal{I}_{15} + 0.84\, \mathcal{I}_{18} + 1.28\, \mathcal{I}_{22} < 0.51 \Leftrightarrow j = 2$$
$$0.90\, \mathcal{I}_4 + 1.76\, \mathcal{I}_6 - 1.35\, \mathcal{I}_7 - 0.61\, \mathcal{I}_{14} - 1.73\, \mathcal{I}_{15} + 0.84\, \mathcal{I}_{18} + 1.28\, \mathcal{I}_{22} \geq 0.51 \Leftrightarrow j = 3$$

The 2 continuous inputs $\mathcal{I}_7$ and $\mathcal{I}_{15}$ are the normalized values of the attributes $\mathcal{C}_4$ and $\mathcal{C}_{10}$, respectively (see Table 1). The equations that relate the normalized

and the original attributes are

$$\mathcal{I}_7 = \frac{\mathcal{C}_4 - 94}{106}$$
$$\mathcal{I}_{15} = \frac{\mathcal{C}_{10}}{6.2}$$

Substituting these relations for the conditions of Rules 1 and 2, we have

- Rule 1: If $71.02\,\mathcal{I}_4 + 138.20\,\mathcal{I}_6 - \mathcal{C}_4 - 47.83\,\mathcal{I}_{14} - 21.87\,\mathcal{C}_{10} + 65.93\,\mathcal{I}_{18} + 100.36\,\mathcal{I}_{22} \geq -53.95$, then Negative.
- Rule 2. If $\mathcal{I}_2 = 0$ and $-90.07 \leq 71.02\,\mathcal{I}_4 + 138.20\,\mathcal{I}_6 - \mathcal{C}_4 - 47.83\,\mathcal{I}_{14} - 21.87\,\mathcal{C}_{10} + 65.93\,\mathcal{I}_{18} + 100.36\,\mathcal{I}_{22} < -53.95$, then Negative.
- Default rule: Positive.

The accuracy of these rules on the training dataset is the same as that of the pruned network, i.e. 86.14 %. Although it cannot be guaranteed that the accuracy rate on the testing set will always be the same as the network's accuracy, for the set used in the experiment, the accuracy rates of the rules and the pruned network are identical.

3 Analysis

The result presented in the previous section indicates that NeuroLinear is effective in extracting oblique classification rules from a pruned neural network. When desired, it is possible to analyze the extracted rules further. For each possible combination of the discrete attribute values, a set of rules involving only the continuous attributes can be generated. The discrete inputs $\mathcal{I}_2, \mathcal{I}_{14}$ and $\mathcal{I}_{18}$ each can have an input value of 0 or 1. The inputs $\mathcal{I}_4$ and $\mathcal{I}_6$ can have the paired-values of $(0,0), (0,1)$, or $(1,0)$. Hence, the various combinations of the discrete inputs could produce a total of 24 rule sets involving only the continuous attributes. Let us consider some of these rule sets:

1. The case when $\mathcal{I}_2 = \mathcal{I}_4 = \mathcal{I}_6 = \mathcal{I}_{14} = \mathcal{I}_{18} = \mathcal{I}_{22} = 0$ (or equivalently, $\mathcal{D}_2 = 0, \mathcal{D}_3 = 2$ or $4, \mathcal{D}_9 = 0, \mathcal{D}_{11} = 2$ or 3, $\mathcal{D}_{12} = 1, 2$ or 3):
   - Rule 1-2: If $\mathcal{C}_4 + 21.87\,\mathcal{C}_{10} \leq 90.07$, then Negative.
   - Default rule: Positive.

   (Rule 1-2 above is the disjunction of Rules 1 and 2).
2. The case when $\mathcal{I}_2 = 1, \mathcal{I}_4 = \mathcal{I}_6 = \mathcal{I}_{14} = \mathcal{I}_{18} = \mathcal{I}_{22} = 0$ (or equivalently, $\mathcal{D}_2 = 1, \mathcal{D}_3 = 2$ or $4, \mathcal{D}_9 = 0, \mathcal{D}_{11} = 2$ or $3, \mathcal{D}_{12} = 1, 2$ or 3):
   - Rule 1: If $\mathcal{C}_4 + 21.87\,\mathcal{C}_{10} \leq 53.95$, then Negative.
   - Default rule: Positive.
3. The case when $\mathcal{I}_2 = \mathcal{I}_4 = 1, \mathcal{I}_6 = \mathcal{I}_{14} = \mathcal{I}_{18} = \mathcal{I}_{22} = 0$ (or equivalently, $\mathcal{D}_2 = 1, \mathcal{D}_3 = 3, \mathcal{D}_9 = 0, \mathcal{D}_{11} = 2$ or $3, \mathcal{D}_{12} = 1, 2$ or 3):
   - Rule 1: If $\mathcal{C}_4 + 21.87\,\mathcal{C}_{10} \leq 124.97$, then Negative.
   - Default rule: Positive.

4. The case when $\mathcal{I}_2 = \mathcal{I}_6 = 1, \mathcal{I}_4 = \mathcal{I}_{14} = \mathcal{I}_{18} = \mathcal{I}_{22} = 0$ (or equivalently, $\mathcal{D}_2 = 1, \mathcal{D}_3 = 1, \mathcal{D}_9 = 0, \mathcal{D}_{11} = 2$ or $3, \mathcal{D}_{12} = 1, 2$ or 3):
   - Rule 1: If $\mathcal{C}_4 + 21.87\ \mathcal{C}_{10} \leq 192.15$, then Negative.
   - Default rule: Positive.

The above rules clearly show that positive patterns and negative patterns are separated by the oblique line $\mathcal{C}_4 + 21.87\ \mathcal{C}_{10}$. For the different values of the discrete attributes, the threshold value is determined by the boundaries of the subintervals found by Chi2 and by the weights of the network between the discrete inputs and the hidden layer. Let us consider a pattern having its relevant discrete inputs all equal to 0 and label it the "base" case. According to the first set of rules above, such a pattern will be classified as negative if its continuous attributes satisfy the condition $\mathcal{C}_4 + 21.87\ \mathcal{C}_{10} \leq 90.07$. Let us now consider another pattern where the values of the relevant inputs are the same as the base pattern, expect for $\mathcal{I}_2 = 1$. The pattern actually corresponds to a female patient if $\mathcal{I}_2 = 0$, male otherwise. The second set of rules indicates that for a male patient having identical values for the other relevant discrete attributes, once the value of $\mathcal{C}_4 + 21.87\ \mathcal{C}_{10}$ exceeds 53.95, he will be diagnosed as positive. C4.5 rules obtained from a decision tree, where a discrete or a continuous attribute may be used as the branching attribute at any node in the tree, does not allow this kind of analysis.

4 Experimental results

We report our experiments with NeuroLinear in this section. Five datasets from the machine learning data repository at the University of California, Irvine are used to compare the performance of NeuroLinear to that of C4.5rules [11]. C4.5rules is chosen because the code is widely available and it has been commonly used by researchers in the machine learning community. It has been reported that C4.5rules performs well on many datasets. The datasets and the characteristics of their attributes are given in Table 2.

Table 2. Datasets used in the experiments.

Dataset	Size	Attributes	
		Discrete	Continuous
Australian Credit Approval	690	8	6
Boston Housing Data	506	1	12
Cleveland Heart Disease	297	5	8
Wisconsin Breast Cancer	699	0	9
Sonar Target	208	0	60

Following Towell and Shavlik [8], for each dataset, ten repetitions of ten-fold cross validation were performed using NeuroLinear. Each neural network was

given a set of initial weights randomly generated in the interval $[-1, 1]$. For all networks, the following values were fixed: the number of hidden units was 4, the number of output unit was 1. The penalty parameters ϵ_1 and ϵ_2 were set at 0.1 and 10^{-3}, respectively. The value of β was 10. We list the average predictive accuracy rates of 100 neural networks for the 5 test datasets in Table 3.

Table 3. Average predictive accuracy rates (%) and its standard deviation. Each average has been computed from 100 pruned neural networks.

Dataset	Accuracy (Std. Dev.)
Australian Credit Approval	83.84 (5.96)
Boston Housing Data	81.52 (8.82)
Cleveland Heart Disease	78.92 (5.79)
Wisconsin Breast Cancer	94.57 (4.84)
Sonar Target	88.63 (11.56)

C4.5rules was run to perform ten-fold cross validation with its default parameter values. The results from C4.5rules and NeuroLinear are summarized in Tables 4 and 5 below.

Table 4. Accuracy rates (%) of C4.5rules and NeuroLinear.

Dataset	C4.5rules	NeuroLinear	P-value
Australian Credit Approval	84.22 (2.93)	83.64 (5.74)	0.60
Boston Housing Data	83.81 (5.90)	80.60 (9.12)	0.28
Cleveland Heart Disease	75.45 (7.17)	78.15 (6.86)	0.24
Wisconsin Breast Cancer	95.28 (2.51)	95.73 (3.75)	0.71
Sonar Target	85.61 (8.64)	85.39 (12.77)	0.96

In the two tables, the average accuracy rates and the average number of rules obtained by C4.5rules and NeuroLinear are given. The figures in parentheses are the standard deviations. The P-values are computed for testing the null hypothesis that the means of two groups of observations are equal. The P-values for the accuracy rates in Table 4 show that there is no significant difference in the mean accuracy rates of C4.5rules and NeuroLinear. On the other hand, the large differences in the numbers of rules of C4.5rules and NeuroLinear in Table 5 clearly demonstrate the effect of using oblique hyperplanes as the rule conditions. Their corresponding small P-values verify the significance of these differences.

Table 5. Number of rules of C4.5rules and NeuroLinear.

Dataset	C4.5rules	NeuroLinear	P-value
Australian Credit Approval	14.60 (2.88)	6.60 (4.40)	0.0001
Boston Housing Data	15.20 (3.01)	3.05 (3.23)	0.0001
Cleveland Heart Disease	12.90 (2.85)	5.69 (4.25)	0.0001
Wisconsin Breast Cancer	8.90 (1.20)	2.89 (2.52)	0.0001
Sonar Target	9.70 (1.57)	7.03 (3.73)	0.0003

Comparing the predictive accuracy rates of NeuroLinear to those of the pruned networks from which they are extracted, we note that they are not identical. This is not unexpected, while the decision surface of the networks can be highly nonlinear, the decision boundaries produced by NeuroLinear are piece-wise linear. The *fidelity* of the extracted rules, however, is high. Towell and Shavlik [8] define fidelity as the ability of the extracted rules to mimic the behavior of the network from which they are extracted.

5 Conclusion

We have presented NeuroLinear, a system for generating classification rules using neural networks. The three components of NeuroLinear make it possible to generate oblique decision rules, these are

1. An efficient neural network training and pruning algorithm.
2. Chi2 algorithm for discretization of hidden unit activation values.
3. X2R algorithm for generating perfect rules from a small dataset with discrete attributes values.

Comparisons of experimental results using real-world datasets reveal that NeuroLinear can achieve similar accuracy rates as C4.5rules with far fewer rules. It can be expected that the smaller rule-set will enable one to explain the classification process in a more meaningful and comprehensible way.

We may refer to patterns having all relevant discrete attribute values equal to zero as the base level. Hence, a set of base-level rules that involves only the continuous attributes can be obtained. Similar sets of rules for other patterns with one or more of nonzero discrete attributes are generated by NeuroLinear. These rules differ from those of the base-level rules only in the threshold values. The different thresholds for the various combinations of the discrete attributes values provide potentially valuable information regarding the patterns in the dataset. This interesting feature of the rules extracted by NeuroLinear is not possessed by those of C4.5rules.

Acknowledgment

We wish to thank the reviewers for their valuable comments and suggestions and S. Murthy of Siemens Corporate Research for pointing out an error in an earlier version of this paper.

References

1. T.G. Dietterich, H. Hild, and G. Bakiri, "A comparative study of ID3 and backpropagation for English text-to-speech mapping," in *Machine Learning: Proceedings of the Seventh International Conference*, Austin, Texas, 1990.
2. D.H. Fisher and K.B. McKusick. "An empirical comparison of ID3 and backpropagation," in *Proceedings of 11th Int. Joint Conf. on AI*, pp. 788–793, 1989.
3. J.R. Quinlan. "Comparing connectionist and symbolic learning methods," in S.J. Hanson, G.A. Drastall, and R.L. Rivest, eds., *Computational Learning Theory and Natural Learning Systems*, Vol 1 (A Bradford Book, The MIT Press, 1994) pp. 445–456.
4. J.W. Shavlik, R.J. Mooney, and G.G. Towell, "Symbolic and neural learning algorithms: An experimental comparison", *Machine Learning*, vol. 6, no. 2, 111–143, 1991.
5. S. Gallant, "Connectionist expert systems," *Comm. of the ACM*, vol. 31, no. 2, pp. 152–169, Feb. 1988.
6. K. Saito and R. Nakano, "Medical diagnosis expert system based on PDP model," in *Proc. IEEE Intl. Conf. on Neural Networks*, IEEE Press, New York, pp. I255-I266, 1988.
7. L. Fu, "Rule learning by searching on adapted nets," in *Proc. of the Ninth National Conference on Artificial Intelligence*, (1991) pp. 590–595.
8. G.G. Towell and J.W. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Machine Learning*, vol. 13, no. 1, pp. 71-101, 1993.
9. R. Blassig. "GDS: Gradient descent generation of symbolic classification rules," in *Advances in Neural Information Processing*, Vol. 6, (Morgan Kaufmann, Los Angeles CA, 1994) pp. 1093–1100.
10. R. Setiono and H. Liu, "Symbolic representation of neural networks," *IEEE Computer*, March 1996, pp. 71–77.
11. J.R. Quinlan. *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.
12. R. Kerber, "ChiMerge: Discretization of numeric attributes," in *The Proc. of the Ninth National Conference on AI*, AAAI Press/The MIT Press, 1992, pp. 123–128.
13. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
14. J. Hertz, A. Krogh, and R.G. Palmer, "Introduction to the theory of neural computation," Redwood City, CA: Addison Wesley, 1991.
15. H. Liu and S.T. Tan, "X2R: A fast rule generator," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, IEEE Press, 1995.
16. P.M. Murphy and D.W. Aha, *UCI repository of machine learning databases [machine-readable data repository]*, Department of Information and Computer Science, University of California, Irvine, 1992.