Credit scoring using neural networks trained with augmented discretized inputs

Arnulfo Azcarraga
College of Computer Studies, De La Salle University
2401 Taft Avenue, Manila 1004, Philippines
Email: arnie.azcarraga@delasalle.ph

Yoichi Hayashi
Department of Computer Science, Meiji University
Tama-ku, Kawasaki 214-8571, Japan
Email: hayashiy@cs.meiji.ac.jp

Rudy Setiono
School of Computing, National University of Singapore
Computing 1, Computing Drive Singapore 117590
Email: disrudy@nus.edu.sg

Abstract

We apply neural networks that have been trained and pruned using augmented discretized input data for credit scoring. Credit scoring datasets normally contain input data attributes that are continuous (e.g. salary) and discrete (e.g. marital status). In order to improve the accuracy of the neural network prediction, we augment the input data by including the discretized values of the continuous attributes. Having both the original continuous attributes and their discretized values make it easier for the networks to form decision boundaries that could be axis-parallel or oblique in the input space defined by these attributes. Neural network

pruning is applied to determine which input attributes are actually useful. We present the results from neural networks trained with augmented discretized input attributes on an artificial dataset as well as on a widely used credit card scoring dataset. The results show that higher accuracy rates can be obtained from these networks. After pruning, simple networks with few connections are obtained. We present the classification rules that have been extracted from such networks.

1 Introduction

Neural networks have been very useful tool for many practical application problems. In particular, credit scoring is an area where neural networks have been shown to outperform more traditional methods in terms of accuracy. In a study of credit risk evaluation process in the Egyptian banking sector, for example, the authors found that the correct classification rates obtained by statistical methods such as discriminant analysis, probit analysis and logistic regression were significantly lower than the accuracy achieved by neural networks [2].

Neural networks were also shown to produce predictions with very low error rates when applied to a dataset of 76 small businesses from a bank in Italy [1]. The credit risk assessment was done to identify whether the small business was solvent or would default. The excellent result obtained by the neural networks was attributed to careful data pre-processing done prior to neural network training in addition to the complex non-linear mapping between the input and output of the networks.

The performance of neural networks with one layer of hidden units trained with different learning schemes was thoroughly investigated by Khashman using the publicly available German credit dataset [3]. The learning schemes differ in the way the data samples were distributed for training and cross-validation. It was discovered that the best results could be obtained when the number of hidden units was set to 27 and the 1000 available data samples were distributed as 40% and 60% for training and cross-validation, respectively.

A market comparative study that compared bond rating processes using data gathered from US and Taiwan markets showed that backpropagation neural networks achieved accuracy that was comparable to the more recent method based on statistical learning theory, namely support vector machines [5]. The Taiwan dataset consisted of 74 cases described by 21 financial variables and one of the five possible rating categories. The US data was larger with 265 cases. The

results from 10 fold cross-validation experiments on both datasets revealed that that support vector machines and neural networks achieved a maximum average accuracy of over 80%. On the other hand, the best result from logistic regression was 76.98%.

Municipal credit rating modeling by neural networks was shown to produce a predictive accuracy of more than 90% [4]. In this study, the author used municipal credit rating data collected from the state of Connecticut during the years 2003-2007. A total of 766 samples were available. There were 4 categories of input variables: economic, debt, financial, and administrative and a total of 52 variables. The municipalities were classified into 9 rating classes ranging from Aaa to Baa2. In this particular application, it was shown that probabilistic neural networks outperformed other methods including support vector machines, linear regression, multiple discriminant analysis, k-means and classification tree.

In this paper, we propose to improve the performance of neural networks for credit scoring by modifying the input data prior to network training. The modification involves adding strings of binary input which are generated according to the values of the continuous attributes originally present in the data. In credit scoring applications, datasets usually contain both continuous input attributes such as income and age, as well as discrete attributes such the purpose of the loan or the marital status. The partition of the data into different groups (e.g. good credit risk versus bad credit risk) is determined by the interactions among these input attributes. The continuous attributes contribute to the classification by forming decision boundaries that are oblique. When rules are extracted from a trained neural network by a rule extraction algorithm such as Re-RX [7], we obtain decision rule conditions of the form $\sum_i W_i x_i \geq b?$, where W_i is a connection weight, x_i is the value of the i-th continuous attribute value and b is a threshold. For some datasets, higher classification accuracy and/or more comprehensible rule sets may be obtained if the decision boundaries formed by the continuous inputs are not oblique but axis-parallel. For this purpose, we discretize the continuous attributes by simply dividing the range of each attribute into a number of equal sub-intervals. These discretized attribute values are then represented as binary string for input to the neural networks along with the values of the continuous attributes and the discrete attributes originally present in the data.

The outline of the paper is as follows. In Section 2 we describe how neural networks are trained with the addition of the discretized continuous variables as input. In Section 3, we present our experimental results. Two sets of data are

used in the experiments. The first is an artificial dataset created to demonstrate the usefulness of having the discretized inputs. The second dataset is the publicly available CARD dataset. There are 3 permutations of this credit scoring dataset [6]. On the 3 versions of the CARD data, we present and compare the performance of neural networks trained with and without the augmented discrete inputs. We also compare the results with those published in the literature. For each of the three datasets, we also extracted the rules that distinguish between good and bad credits. Finally, Section 4 concludes the paper.

2 Neural network training with augmented discrete input variables

In our earlier work [8], we proposed how a string of N binary variables was augmented to the input data for each continuous attribute prior to network training. By adding the binary input variables, it becomes easier for the network to achieve the minimum required accuracy rate on the training data. After network pruning, we also expect to extract accurate and concise classification rules which define sub-regions in the input space bounded by a combination of axis-parallel and oblique hyperplanes. The flexibity of having such a combination of hyperplanes in the rule conditions yields more concise and easier to understand rules.

Let c_i be the i-th input attribute in the data having continuous values, the steps to generate the augmented inputs for this continuous attribute are as follows:

1. Normalize the values of c_i so that for all data samples $s = 1, 2, \ldots S$, the values lie in the interval $[0, 1]$:

$$nc_i^s = \frac{c_i^s - c_{min,i}}{c_{max,i} - c_{min,i}}$$

   where $c_{min,i}$ and $c_{max,i}$ are the minimum and the maximum values of attributes c_i, respectively.

2. Set the number of sub-intervals equal to N, and let $\delta = \frac{1}{N}$.

3. Compute the index K:

$$K = \begin{cases} N & \text{if} \quad nc_i^s = 0 \\ = N + 1 - \lceil nc_i^s/\delta \rceil & \text{otherwise} \end{cases}$$

4. Generate N binary inputs I_k^s for c_i^s as follows:

$$I_k^s = \begin{cases} 0 & \text{if } \quad k < K \\ 1 & \text{otherwise} \end{cases}$$

for $k = 1, 2, \ldots N$.

Using the new augmented input data, we train and prune the neural networks for classification as follows:

Algorithm ADV-NN (Augmented Discrete Variable - Neural Networks)

Input: A labeled set of data samples $\{\mathbf{c_i^s}, \mathbf{d_j^s}, \ell^{\mathbf{s}}\}$, where

- $s = 1, 2, \ldots S$, and S is the number of data samples.
- $\mathbf{c_i}, i = 1, 2, \ldots I$ are input variables with continuous values.
- $\mathbf{d_j}, j = 1, 2, \ldots J$ are input variables with discrete values.
- ℓ^s is the class label of data sample s. For simplicity we assume $\ell^s \in [0, 1]$.

Output: A pruned neural network.

Step 1. For each continuous input variable $\mathbf{c_i}$, generate N additional columns of binary inputs. Let $\mathbf{x^s} \in \mathbb{R}^n$ be the augmented input data which include the original discrete and continuous input attributes plus the $I \times N$ new columns of binary inputs.

Step 2. Train a neural network with a single hidden layer of H hidden units to minimize an error function

$$F(\mathbf{W}, \mathbf{V}) = \sum_{s=1}^{S} \sum_{m=1}^{O} \left(d_m^s - y_m^s\right)^2 + P(\mathbf{W}, \mathbf{V}), \tag{1}$$

d_m^s and y_m^s are the m components of desired output and the predicted output from the neural network for data sample s. For a binary classification problem, the number of output units in the network O is set to 2 and $d_1^s = 0, d_2^s = 1$ if the class label $\ell^s = 0$ and $d_1^s = 1, d_2^s = 0$ if $\ell^s = 1$. $P(\mathbf{W}, \mathbf{V})$ is a penalty function that pushes unnecessary connections in the network to have small weights. A quadratic penalty function that is the sum of the squared weights of the network connections is used in our implementation.

Step 3. Prune the network:

- Hidden unit removal:
  i. Group the hidden units into two subsets: $\mathcal{P}$ and $\mathcal{Q}$ are the sets of hidden units still present in the network and those that have been checked for possible removal in the current stage of pruning, respectively. Initially, $\mathcal{P}$ corresponds to all the hidden units in the trained network and $\mathcal{Q}$ is the empty set.
  ii Save a copy of the weight values of all connections in the network.
  iii Find a set of connections from the input units to the hidden unit $h \in \mathcal{P}$ and $h \notin \mathcal{Q}$ such that when the weight values of the connections from the input units to h are set to 0, the accuracy of the network is least affected.
  iv. Set the weights for network connections from the input units to the hidden unit h to 0 and retrain the network.
  v. If the accuracy of the network is still satisfactory, then
    (a) Remove h, i.e. set $\mathcal{P} := \mathcal{P} - \{h\}$.
    (b) Reset $\mathcal{Q} := \emptyset$.
    (c) Go to Step (ii).
  vi. Otherwise,
    (a) Set $\mathcal{Q} := \mathcal{Q} \cup \{h\}$.
    (b) Restore the network weights with the values saved in Step (ii) above.
    (c) If $\mathcal{P} \neq \mathcal{Q}$, go to Step (ii). Otherwise, Stop.
- Input unit removal done in similar way as for hidden unit removal above.
- Network connection removal done in similar way as for hidden unit removal above.

The addition of discretized input variables requires the network to have more input units, and as a result there are more weights in the network that need to be optimized. The time for one iteration of network training will thus increase. In our experiments, however, having additional inputs actually allows the network training to be more likely to converge to a solution where the minimum accuracy requirement is met in fewer iterations.

3 Experimental results

We first generate an artificial dataset where the classification of the samples in 2-dimensional space are determined by a combination of axis-parallel and oblique decision hyperplanes. For the credit scoring application, we then apply the proposed method on the publicly available credit scoring CARD datasets [6, 10]. We show how the prediction of the data samples in these CARD datasets can be achieved with better accuracy when the input data are augmented with the discretized inputs of the original continuous attribute values.

3.1 An artificial dataset

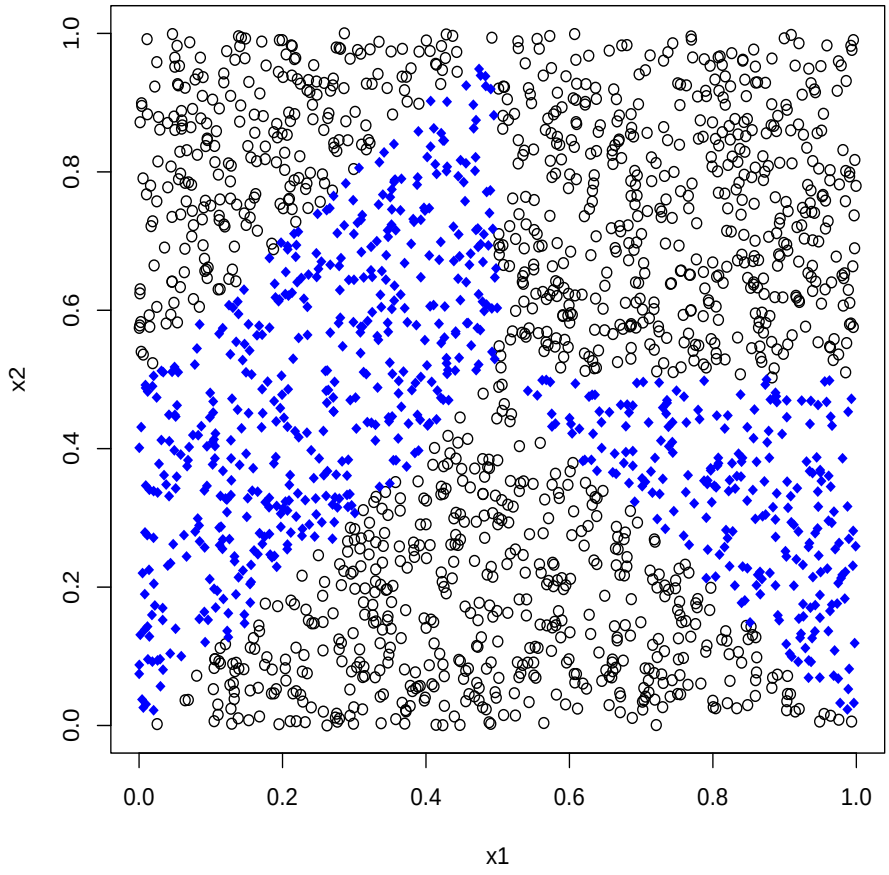

Figure 1: A randomly generated data samples with 2 classes separated by axis-parallel and oblique hyperplanes.

A dataset consisting a total of 3000 random samples were generated. The 4 input attribute values x_1, x_2, x_3 and x_4 were generated uniformly in $[0, 1]^4$. A

sample was assigned the class label 0 if one of the two conditions below was satisfied:

- $(x_1 \leq 0.5)$ and $(x_2 \geq x_1)$ and $(x_2 < x_1 + 0.5)$ then
- $(x_1 > 0.5)$ and $(x_2 \leq 0.5)$ and $(x_2 > 1 - x_1)$

Otherwise, the sample is labeled as class +1. The attributes x_3 and x_4 are noise and irrelevant for classification. The numbers of samples used for training and testing the network were 2000 and 1000, respectively. Figure 1 shows the plot of the 2000 samples in the training dataset.

Two groups of thirty neural networks were trained:

1. Group 1: Network training was done using the original dataset having 4 attributes. The number of input, hidden and output units in the networks were 4, 10 and 2, respectively. Networks connections were removed by pruning as long as the classification accuracy was at least 90%.
2. Group 2: Network training was done using the original dataset having 4 attributes plus 4 $\times N(= 40)$ discretized input variables. The number of input, hidden and output units in the networks were 164, 3 and 2, respectively. The networks were trained and pruned to maintain a classification accuracy of at least 95%.

Table 1: The results from neural network training with original and augmented discretized inputs.

Data set	# Hidden units	# Connections	Training Acc. (%)	Test Acc. (%)
Group 1	6.47 ± 1.92	14.97 ± 6.38	91.92 ± 1.18	92.56 ± 1.31
Group 2	2.90 ± 0.30	11.63 ± 4.66	95.90 ± 1.74	94.89 ± 2.01

With just 4 input attributes, it was more difficult to train a neural network to reach a preset minimum accuracy rate than when there are 164 inputs. This is because with more inputs, the large increase in the number of connection weights in the network makes it more likely for the training to reach a solution where the error function value is small. We had 10 hidden units and a minimum accuracy threshold of 90% when training the networks with original data. The averages and

standard deviations were computed from 30 pruned networks and are summarized Table 1. The number of hidden units in the table is the average number of hidden units left in the pruned networks. The number of connections is the average number of connections between the input units and the hidden units. Comparing the average number of connections for the two groups of neural networks, we can see that the networks trained with augmented discretized inputs actually have 3 fewer connections than the neural networks trained with the original data. At the same time, the accuracy rates on both the training and test data of the latter group of networks are actually higher.

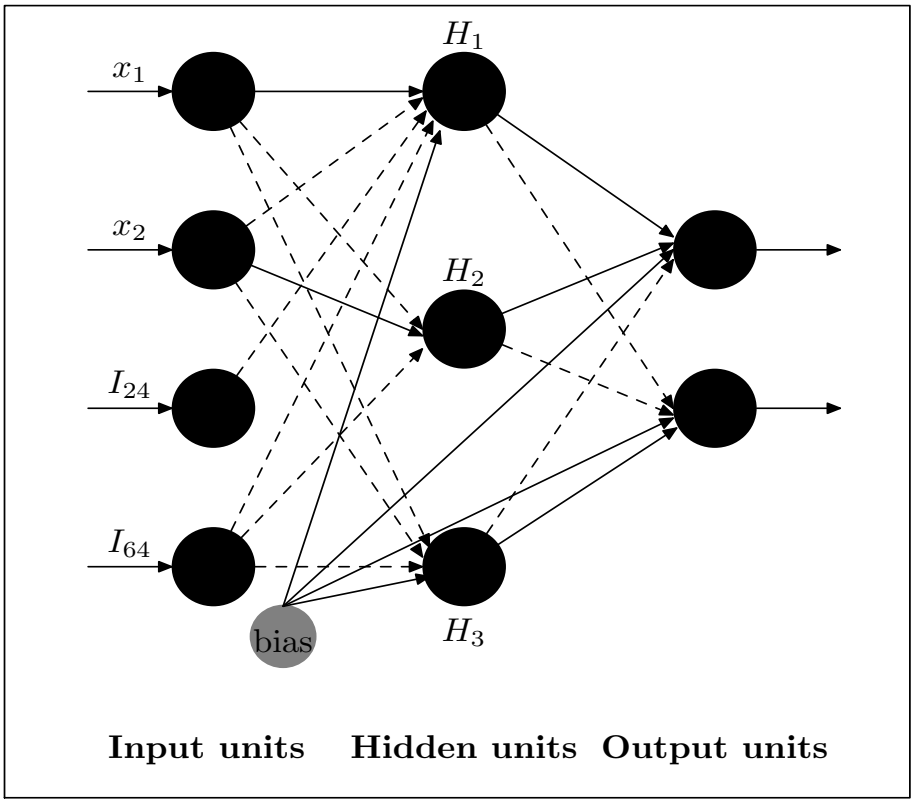

Figure 2: A pruned neural network for the artificial dataset.

A relatively small number of connections in the pruned network enable us to extract concise classification rules that explain how the distinction between the two classes in the data was made by the network. We apply the rule extraction algorithm Re-RX [7] on one of the pruned networks that has the best predictive performance (Figure 2). This network accuracy rates are 98.15% and 98.70% on the training and test datasets, respectively. The rule set below is the output from

Re-RX.

Rules for the artificial dataset:

Rule $\mathcal{R}_1$: If $I_{24} = 1$ and $I_{64} = 1$, then predict Class 1,

Rule $\mathcal{R}_2$: else if $I_{24} = 0$ and $I_{64} = 1$, then

Rule $\mathcal{R}_{2a}$: If $x_1 - 1.041x_2 > -0.524$, then predict Class 0, else

Rule $\mathcal{R}_{2b}$: predict Class 1,

Rule $\mathcal{R}_3$: else if $I_{24} = 1$ and $I_{64} = 0$, then

Rule $\mathcal{R}_{3a}$: If $x_1 + 1.025x_2 > 1.006$, then predict Class 0, else

Rule $\mathcal{R}_{3b}$: predict Class 1,

Rule $\mathcal{R}_4$: else if $x_1 - 0.980x_2 < 0.001$, then predict Class 0, else

Rule $\mathcal{R}_5$: predict Class 1,

The augmented binary variable I_{24} is 0 if and only if $x_1 < 0.5$. Similary, the augmented binary variable I_{64} is 0 if and only if $x_2 < 0.5$. It is evident that the rules extracted from the neural network are similar to the conditions under which the training data samples have been generated. The overall accuracy rates of these rules on the training and test datasets are 99.75% and 99.40%, respectively. Obtaining a concise and comprehensible set of classification rules with such high accuracy rates would not be possible without augmenting the input data with discretized inputs prior to neural network training.

3.2 The CARD datasets

The three variations of the CARD datasets have been used in benchmark studies for credit scoring [10, 9] and they are available publicly [6]. Table 2 shows the distribution of the data samples in the training and test sets. Of the 690 samples, 307 samples have target values of +1 (Class 1) and the remaining 383 have target values of 0 (Class 0).

Each data sample is described by 6 continuous attributes and 9 discrete attributes. The discrete attributes have been coded using binary representation. As a result, there is a total of 51 input attributes. As there is no detailed explanation on what each of the attributes represents, continuous input attributes 4, 6, 41, 44, 49 and 51 are simply labeled $C_4, C_6, C_{41}, C_{44}, C_{49}$, and

Table 2: The sample distribution in the CARD1, CARD2 and CARD3 credit approval datasets.

Data set	Training set		Test set	
	Class 0	Class 1	Class 0	Class 1
CARD1	291	227	92	80
CARD2	284	234	99	73
CARD3	290	228	93	79

C_{51}, respectively. The remaining binary-valued attributes are D_1, D_2, D_3, D_5, $D_7, \ldots, D_{40}, D_{42}, D_{43}, D_{45}, D_{46}, D_{47}$, D_{48}, and D_{50}. We normalized the continuous input attributes and discretized them by dividing the $[0, 1]$ normalized range into $N = 40$ subintervals of equal length. In total, the neural networks have $6 + 6 \times 40 + 45 = 291$ inputs.

The number of hidden units is set to one, as it has been shown that networks with just one hidden unit are able to provide good predictive accuracy [9]. Samples that correspond to credit card applications that have been approved or denied are assigned target values of +1 or 0, respectively.

Table 3: Comparison of the best predictive error rates on the test data samples from various neural network models.

Method	CARD1	CARD2	CARD3
GA	10.47	15.12	11.63
Prechelt	13.95	18.02	18.02
NeuralWorks	12.79	18.02	12.21
NeuroShell	11.63	18.02	15.12
PNN	10.47	12.79	11.63
NN-ADV (Rules)	9.30	12.21	11.05

The predictive error rates of the rules extracted from neural networks that have been trained using augmented discretized inputs are depicted in Table 3 as NN-ADV (Rules). These errors rates are lower than those obtained from other

methods. The PNN results were the error rates of the rules that have been obtained from neural networks trained with the original data [9]. The rest of the figures were reported by Sexton et al. [10]. PNN rules and NN-ADV rules have been extracted by applying the Re-RX algorithm. The accuracy obtained by NN-ADV rules is slightly better than the accuracy obtained by PNN rules. The improvement in accuracy is more evident when NN-ADV rules are compared with the other methods reported by Sexton et al.

The NN-ADV rules for the 3 datasets are given below.

Rules for CARD1 dataset:

Rule $\mathcal{R}_1$: If $D_8 = 1$ and $D_{26} = 0$ and $D_{42} = 0$, then predict Class 0,

Rule $\mathcal{R}_2$: else if $D_2 = 1$ and $D_{42} = 0$, then predict Class 0,

Rule $\mathcal{R}_3$: else if $D_8 = 1$ and $D_{29} = 1$, then predict Class 0,

Rule $\mathcal{R}_4$: else if $D_8 = 1$ and $D_{29} = 1$, then

Rule $\mathcal{R}_{4a}$: If $D_{26} = 0$, then

Rule $\mathcal{R}_{4a-i}$: If $C_{49} - 1.50C_{50} - 0.42C_{51} > 0.0550$, then predict Class 0, else

Rule $\mathcal{R}_{4a-ii}$: predict Class 1,

Rule $\mathcal{R}_{4b}$: else

Rule $\mathcal{R}_{4b-i}$: If $C_{49} - 1.50C_{50} - 0.42C_{51} > 0.2470$, then predict Class 0, else

Rule $\mathcal{R}_{4b-ii}$: predict Class 1,

Rule $\mathcal{R}_5$: else if $D_{29} = 0$ and $D_{42} = 1$, then predict Class 1,

Rule $\mathcal{R}_{5a}$: If $C_{49} + 4.18C_{50} + 0.58C_{51} > 0.3759$, then predict Class 0, else

Rule $\mathcal{R}_{5b}$: predict Class 1,

Rule $\mathcal{R}_6$: else if $D_8 = 0$ and $D_{12} = 0$, then predict Class 1,

Rule $\mathcal{R}_7$: else predict Class 0.

Rules for CARD2 dataset:

Rule $\mathcal{R}_1$: If $D_{12} = 1$ and $D_{42} = 0$, then predict Class 0,

Rule $\mathcal{R}_2$: else if $D_8 = 1$ and $D_{42} = 0$, then predict Class 0,

Rule $\mathcal{R}_3$: else if $D_8 = 1$ and $D_{26} = 0$, then

Rule $\mathcal{R}_{3a}$: if $C_{49} - 0.377C_{51} > 0.055$, then predict Class 0,

Rule $\mathcal{R}_{3b}$: else predict Class 1,

Rule $\mathcal{R}_4$: else if $D_8 = 0$ and $D_{42} = 1$, then

Rule $\mathcal{R}_{4a}$: if $C_{49} - 0.662C_{51} > 0.069$, then predict Class 0,

Rule $\mathcal{R}_{4b}$: else predict Class 1,

Rule $\mathcal{R}_5$: else if $D_{26} = 1$ and $D_{42} = 1$, then predict Class 1

Rule $\mathcal{R}_6$: else if $D_8 = 0$ and $D_{12} = 0$, then predict Class 1

Rule $\mathcal{R}_7$: else predict Class 0.

Rules CARD3 for dataset:

Rule $\mathcal{R}_1$: If $D_{42} = 0$, then predict Class 0

Rule $\mathcal{R}_2$: else if $D_{44} = 1$ and $C_{48} \leq 0.05$ and $C_{49} > 0.025$ and $C_{51} \leq 0.425$, then predict Class 0

Rule $\mathcal{R}_3$: else if $D_{43} = 1$ and $0.05 \leq C_{51} \leq 0.425$, then predict Class 0

Rule $\mathcal{R}_4$: else if $C_{48} \leq 0.05$ and $0.05 \leq C_{51} \leq 0.3$, then predict Class 0

Rule $\mathcal{R}_5$: else predict Class 1.

A closer inspection of the rules indicates that for the CARD1 and CARD2 datasets, the rule conditions do not include any of the discretized variables. That is, they all have been removed by network pruning. As a result, the decision boundaries that are represented by the rule conditions correspond to oblique hyperplanes in the input space. Only for the CARD3 dataset, six of the discretized inputs remained in the pruned network. These inputs are $I_{130}, I_{169}, I_{210}, I_{219}, I_{274}$ and I_{290}. Their values are either 0 or +1 depending on the original value of the corresponding continuous variable. For example, $I_{130} = 0$ if and only if $C_{48} < 0.05$ and $I_{290} = 0$ if and only if $C_{51} < 0.05$. In this case, the decision region formed by the continuous attributes is hyper-rectangular.

4 Conclusion

In this paper we have described how higher prediction accuracy can be achieved when discretized continuous input attributes are augmented to the input data prior to neural network training. We demonstrated that networks with such inputs obtained higher prediction accuracy on both the training and test data samples through a classification problem with synthetic data. In this artificial classification problem, the decision boundaries separating 2 classes of samples are a mix of axis-parallel and oblique hyperplanes. Not only higher accuracy rates, the networks trained with augmented discretized input also have simpler structure after pruning as more hidden units and connections were removed.

On a credict scoring application dataset, neural networks trained with augmented input data are also shown to be more accurate than neural networks trained with the original data. The few connections left in the networks after pruning make it possible for us to obtain very concise classification rules from the data. On two of the three CARD datasets, the continuous attributes formed decision boundaries that are oblique. On the third dataset, the network inputs corresponding to the continuous inputs were all pruned out. Some of the discretized continuous attributes were still relevant for classification and produced axis-parallel decision boundaries in the rules extracted from the network. In an application domain such as credit scoring, having such comprehensible rules sets helps us understand better how exactly the input data attributes interact and determine the class labels. Our plan for the near future is to apply the approach described in this paper on other credit scoring data sets as well as other datasets for business intelligence applications.

References

[1] E. Angelini, G. di Tollo and A. Roll, A neural network approach for credit risk evaluation. The Quarterly Review of Economics and Finance, **48(4)**, 733-755, 2008.

[2] H. Abdou, J. Pointon and A. El-Masry, Neural nets versus conventional techniques in credit scoring in Egyptian banking. Expert Systems with Applications, **35(3)**, 1275-1292, 2008.

[3] A. Khashman, Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. Expert Systems with Applications, **37(9)**, 6233-6239, 2010.

[4] P. Hájek, Municipal credit rating modeling by neural networks. Decision Support Systems, **51(1)**, 108-118, 2011.

[5] Z. Huang, H. Chen, C-J. Hsu, W-H. Chen and S. Wu, Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision Support Systems, **37(4)**, 543-558, 2004.

[6] L. Prechelt, "Proben1 - A set of benchmarks and benchmarking rules for neural network training algorithms," Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany. Anonymous ftp available from ftp://pub/papers/techreports/1994/1994-21.ps.gz on ftp.ira.uka.de (1994).

[7] R. Setiono, B. Baesens and C. Mues, Recursive neural network rule extraction for data with mixed attributes. IEEE Transactions on Neural Networks, **19(2)**, 299-307, 2008.

[8] R. Setiono and A. Seret, Discrete variable generation for improved neural network classification. In Proc. of the 24th International Conference on Tools with Artificial Intelligence, Athens, Greece, November 2012.

[9] R. Setiono, B. Baesens and C. Mues, Rule extraction from minimal neural networks for credit card screening. International Journal on Neural Systems, **21(4)**, 265–276, 2011.

[10] R.S. Sexton, S. McMurtrey and D.J. Cleavenger, Knowledge discovery using a neural network simultaneous optimization algorithm on a real world classification problem. European Journal of Operational Research, **168**, 1009-1018, 2006.