ABSTRACT
Credit scoring data contain discrete and continuous variables that are used in building models to differentiate between good and bad credit risks. Neural networks (NN) are known to produce good results when applied to analyze these data. We propose the introduction of interaction terms between the continuous variables to improve the NN accuracy and to simplify the classification rules extracted from the networks. By explicitly adding interaction terms, pruned NN with high accuracy and few connections are obtained from our test data. The few connections in the networks allow us to describe the classification process as concise sets of rules.

KEYWORDS: Neural networks, Rule extraction, Interaction terms, Credit scoring.

INTRODUCTION
In credit scoring, improving the predictive accuracy - even by a small margin, of the model which has been built to differentiate between good and bad credits can translate into a huge financial gain. Many machine learning techniques have been applied to build classification models that can differentiate between good and bad risks in a credit scoring data set with greater accuracy rates than those achieved by traditional statistical approaches. Popular machine learning techniques that have been shown to perform well in analyzing credit scoring data include decision trees, genetic programming, support vector machines and neural networks (Baesens et al., 2003; Bensic et al., 2006; Crook et al., 2007; Ince & Aktan, 2009; West, 2000). There are also hybrid methods that combine two or more of these techniques in the hope of obtaining even higher prediction accuracy (Barammirzaee et al., 2011; Huang et al, 2007).

In this paper, we propose the use of neural networks for building credit scoring models that are not only more accurate than other machine learning based models, but also provide more transparency about their decisions. While neural networks have been proven to be excellent tools for regression and classification, criticisms about their lack of capability to explain their outputs as comprehensible rules are often raised. The output of a neural network is computed as a complex nonlinear mapping of its input. Treating a neural network as a black-box that functions as an oracle churning out good predictions may be good enough in some application domains, but it is no longer sufficient in the case of credit scoring. The capability to generate accurate, concise and comprehensible rules would help make neural networks more acceptable as tools that build models for classification and regression in general, and for credit scoring in particular. The key components of our neural network based technique for credit scoring are (1)
neural networks that are trained with augmented inputs, and (2) an if-else-then rule generating algorithm from pruned neural networks. We add discretized values and interaction terms of the original continuous variables present in the data for neural network training. By having these additional inputs, smaller neural networks with good predictive accuracy are obtained. Smaller neural networks have fewer network connections than the fully connected networks. When rules are extracted from the smaller networks, it can be expected that more concise rule sets are obtained.

Many algorithms for rule extraction from neural networks can be found in the literature (Chorowski & Zurada, 2011; Diederich et al., 2010; Huynh & Reggia, 2011; Setiono et al., 2011). The application of the Re-RX algorithm on credit scoring data shows that neural networks do achieve good predictive accuracy and the classification rules extracted from the networks preserve the accuracy of the networks (Setiono et al., 2009). Moreover, the extracted rule sets consist of relatively few rules involving only a small number of the original input variables in the data. We attempt to improve the prediction accuracy of the neural network credit scoring models by adding 2 groups of new input information for neural network training. The first group of input consists of discretized values of the original continuous attributes in the data. The discretization is achieved by a simple division of the normalized range of the continuous attributes $[0,1]$ into a number of equal subintervals. The discretized values are then encoded as dummy binary variables for neural network training. The second group of input consists of interaction terms of the continuous attributes. For each pair of continuous attributes $(X_i, X_j)$, the interaction term $X_i \times X_j$ is added. By increasing the dimensionality of the input, we are able to achieve high predictive accuracy rates using neural networks with simpler structure. Original input attributes as well as newly added attributes that proved to be not useful for prediction are identified and removed by neural network pruning. Neural networks with few hidden units and connections left allow us to extract very concise rules that describe how a sample is classified as a good or bad credit risk.

The outline of this paper is as follows. In Section 2, we present related works on machine learning methods that have been applied for credit scoring. Besides neural networks, popular machine learning methods that have been shown to be effective include decision tree methods, genetic algorithm and genetic programming as well as support vector machines. In Section 3, we describe our proposed approach for neural network training, pruning and rule extraction with the additional discretized inputs and interaction terms. In Section 4, we present the results from our experiments using 3 publicly available benchmark credit scoring data sets. Finally Section 5 concludes the paper.

**LITERATURE REVIEW**

Sohn and Kim (2012) propose a decision tree based approach for evaluating start-up firms. The main purpose of their study is to build a decision tree credit scoring model that can be used to identify potential loan default of start-ups. The data used in the study comprise of 3347 start-up firms from various industries such as construction, mining, education, manufacturing and communication. The default rates of these start-ups vary from 0% in the real estate and rental industry to 56% in the public, repair and personal services. Three groups of input variables describe the data: technology scorecard attributes (16 variables), economic indicators (9 variables), and firm characteristics (10 variables). A number of decision trees are built using subsets of these variables and 70% of the available data. The performance of the resulting
decision trees is then cross-validated on the remaining 30% data samples. It is found that the
decision tree that has been built using all input variables is the most accurate with accuracy
rates of 73% and 74%, respectively on the training and validation data sets. These results are a
significant improvement over the 66% accuracy rates obtained from logistic regression.

Credit scorecard model, logistic regression and decision tree method are used to analyze the
payment history of members of a Malaysian recreational club in a study by Yap et al. (2011).
The data set is divided 70:30 for training and testing the models, respectively. Altogether there
are 977 defaulters and 1788 non-defaulters. The number of variables is 9, among which only 4
are selected based on their information values by the credit scorecard model. These variables
are number of dependents, number of cars, district address and age. A different set of variables
is found to be important in profiling the defaulters by the decision tree method. In addition to the
number of cars and the district address, marital status and the number of dependents are also
required to classify data samples using the decision tree. The classification accuracy rates of
the logistic regression model and decision tree model are found to be very similar and they are
slightly worse than the accuracy obtained by the credit scorecard model.

Decision tree method is shown to be superior predictor of business failures when compared to
discriminant analysis by Gepp et al. (2005). The size of the data set used in the experiments is
200, 142 correspond to successful businesses and the remaining 58 failed businesses. Each
data sample is described by 20 financial variables, most of which are financial ratios. Using the
5-fold cross-validation procedure, the performances of three decision tree methods See5 and
CART, Recursive Partitioning Analysis (RPA) and Multiple Discriminant Analysis (MDA) are
evaluated and compared. See5 is a very popular decision tree generating algorithm which is the
latest improved version of ID3 (Quinlan, 1993). CART is a decision tree generating algorithm for
classification and regression developed by Breiman et al. (1984). The third tree generating
method RPA was the first to be applied to analyze the data by Frydman et al. (1985). The study
by Gepp et al. (2005) concludes that less complex, more parsimonious decision tree models are
better predictors than more complex models.

Credit scoring models built using genetic programming are found to outperform other models
such as logistic regression, artificial neural networks and rough set methods by Ong et al.
(2005). Genetic programming is designed to extract the relationship in the data by representing
it as a tree. The tree-based structure composed of a function set and a terminal set. The
function set for the credit scoring data includes regular arithmetic operators such as (+, -, ×),
trigonometric operators {sin, cos}, and binary relation operators { ≥, =, ≤, and, or} as well unary
operator {not} and the conditional statement {if - then - else}. The terminal set comprises of all
inputs, constants and other zero-arguments in the GP tree. A single population consists of 40
individual trees, and the process of selection, cross-over and mutation is performed on these
trees to produce the next generation population. A maximum of up to 1000 generations may be
produced in order to find the fittest individual tree. Based on the results from the experiments on
two publicly available credit scoring data sets (Australian and German data sets from the UCI
repository), the authors conclude that genetic programming approach produces more flexible
and accurate credit scoring models.

A serious drawback of the models obtained by genetic programming is that the resulting trees
represent very complex relationship among the data variables which may involve not commonly
used functions such as sine and cosine functions. To improve the interpretability of the rules for
the decision maker, Huang et al. (2006) propose a two-stage genetic programming approach for
generating simpler if-then rules from credit scoring data. In the first stage, the depth of the tree is limited to 4. The function set is limited to contain only \( \{ \geq, =, \leq \} \) and the conditional statement \{if - then - else\}. The number of rules generated can be also reduced by excluding data samples that satisfy the conditions of a rule in further rule generation. In the second stage, a reduced data set consisting data samples that do not satisfy any rule or satisfy the conditions of more than one rule is employed to build a discriminant function. The authors find the accuracy of the 2 stage genetic programming models to be more accurate by 1% and 2% over the single stage genetic programming models on the Australian and German data sets, respectively.

Support vector machines (SVM) with selected input feature variables are employed to analyze the German credit scoring data set by Han et al. (2013). There are 13 categorical variables and 7 continuous variables in the data. Because SVM require numerical input, the categorical variables need to be encoded as dummy variables. Representing the dummy variables as binary input results in a relatively large increase in the total number of inputs. The final selection of dummy variables is done by the forward selection option in logistic regression, while the number of continuous input variables is reduced from 7 to 3 by Orthogonal Variable Reduction (ORD) or Principal Component Analysis (PCA). The performance of SVM with various ways of feature reduction is compared in terms of true positive, true negative and overall accuracy rates as well as the Area under the Curve (AUC). The authors summarize that feature reduction improves the accuracy of SVM, but not logistic regression. Based on the AUC values, it is concluded that the various ways of feature reduction do not produce significant performance difference in the accuracy of SVM.

Two data sets are used to analyze the effectiveness of SVM in building comprehensible and accurate credit scoring models by Niklis et al. (2014). The first set comprises 1314 samples obtained from listed non-financial firms in Greece over the period from 2005 to 2010. The second set consists of 10736 samples obtained from non-listed Greek firms in the financial sector over the period from 2007 to 2010. For both data sets, a sample is classified as either default or non-default according to the values of 7 financial ratios of the firm. Instead of just one support vector machine, a sequence of R support vector machines are generated in the additive SVM model described by the authors. The inputs for each SVM are generated by first dividing the range of the input intervals into a number of subintervals, and then transforming the original attribute value according to a linear interpolation function built using the cut-offs of the subintervals. The final class prediction of a sample is obtained by averaging the outputs from the R support vector machines. Performance measures in terms of area under the receiver operating characteristic curve (AUROC) and Kolmogorov-Smirnov distance are compared for the additive SVM, linear SVM, RBF SVM and linear regression models built using all or only a subset of the input variables. Using these measures computed on the data from the non-listed firms, it is shown that additive SVM models yield the best results.

Prediction based on more than one model could be more accurate than the prediction from a single model. Two linear regression (LR) models are combined to distinguish between good and bad mortgage accounts by Chi and Hsu (2012). A total of 86628 data samples have been obtained from a Taiwanese bank, of which only 758 are bad accounts. In order to build a useful model for prediction, the study uses only 758 bad accounts and 2242 good accounts for model building. Two sets of variables are available. The first set is the bank's internal data with information regarding the behavior of the borrowers such as their characteristics (e.g. gender, age, marital statues), their collateral characteristics (e.g. value of the property, mortgage...
amount), and payment characteristics (e.g. debt-to-income ratio, term to maturity of the mortgage). The second set of variables contains information obtained from external credit bureau. Included in this set are variables such as the amount of money owed (e.g. credit card balance), payment history (e.g. number of one or more days delinquent in last 6 months), new credit (e.g. number of inquiries), the number of months since obtaining a credit card, types of credit used and other account information. Due to the large number of variables, the authors perform feature selection by applying genetic algorithm. As a result, only 11 and 13 variables are selected from the bank internal data and the credit bureau data, respectively. Two linear regression models are fit separately using these two sets of data and their outputs are combined to provide final predictions in this dual scoring model. Outputs from each linear regression model are divided into 5 categories based on their degree of risk scores. By combining the outputs from two models, a 5 by 5 dual risk scoring matrix is generated. This 5 by 5 matrix effectively creates 5 segments of borrowers with different proportions of bad and good accounts. Overall, the prediction of the dual scoring model is shown to be more accurate than the predictions from each of its two components.

Data segmentation can also be achieved by clustering similar data samples into homogeneous groups. Tsai (2014) proposes the use of Self Organizing Maps (SOM) (Duda et al., 2001) to cluster credit scoring data prior to building the prediction models. Different map sizes ranging from 2 by 2 to 5 by 5 are tested, that is, the data are divided into 4, 9, 16 or 25 clusters by these maps. Some of these clusters are well defined clusters in the way that most of the samples they contain belong to only one class. The rest of the clusters that contain mix samples from good and bad credit risks are considered to be fuzzy clusters. Data samples used for building a classifier are determined by combining the outputs from 5-fold cross-validation run of SOM clustering. The best clustering result is obtained from 5 by 5 SOM. Out of 25 clusters on the map, only 2 clusters contained clear majority of good and bad credits. The other 23 are fuzzy clusters. Linear regression (LR), multiple linear perceptron (MLP) and decision tree (DT) method CART are employed to build the classification models. The training data for building these models are determined by the aggregating the clustering results from 5-fold cross-validation run. In addition to single classifiers, ensembles of classifiers are also used to evaluate the benefits of data clustering. Two types of classifier ensembles are tested. Homogeneous ensembles consist of a number of predictive models built using the same techniques (i.e. LR, MLP or DT), while heterogeneous ensembles consist of the best of LR, MLP and DT classifiers. The results from experiments on 5 credit scoring data sets show that combining SOM with homogeneous classifier ensembles with outputs computed by weighted voting produces the lowest Type I and Type II errors.

In credit scoring, comprehensibility of the decision is a key requirement and even a regulatory recommendation (Martens et al., 2007). In order to achieve a better understanding how a decision is derived, many rule extraction algorithms that translate complex mathematical relationship between the input and output in credit scoring models have been developed. Three algorithms that generate rules from support vector machines (SVM) are presented by Martens et al. (2007). These algorithms generate rules via the pedagogical approach. Using this approach, a trained support vector machine is treated as an oracle to predict the class labels of the data samples. Rule generating methods such as decision tree methods are then applied to actually produce classification rules. The results from experiments on several artificial data sets, credit scoring data sets and medical data sets show that SVM rule extraction techniques generate rules with accuracy rates that are almost as high as the accuracy rates of the SVM themselves.
In the next section, we present our proposed approach to improve the accuracy and simplicity of the rules extracted from neural networks that have been trained for credit scoring. We describe in details how discretized variables and interaction terms are added before training the neural networks. We also describe how the networks are trained and pruned, and how rules are then generated from the pruned networks.

CREDIT SCORING NEURAL NETWORKS WITH AUGMENTED INPUT

New inputs generation

Data samples of different classes are more likely to become linearly separable if they are transformed into higher dimensional space (Cover, 1965). In neural network learning, this transformation of the original data into higher dimensional space can be accomplished by having increasing number of hidden units. In support vector machines, it is normally achieved by employing complex kernel functions.

Instead of defining the transformation as a function of the original data input variables, we increase input data dimensionality by simply augmenting each data sample with two groups of new inputs generated from the values of the continuous variables originally present in the data. The first group consists of discretized values of the continuous variables. Assuming that the range of the original continuous variable has been normalized to interval [0, 1], we first divide this range into N equal sub-intervals. A corresponding set of N-1 inputs is then added to the neural network for each continuous variable. Depending on the original value of the continuous variable, the N-1 new inputs are assigned values of 0 or 1 according to the thermometer encoding scheme (Smith, 1996). An example of a continuous variable that is divided equally into N = 5 sub-intervals and the corresponding binary representation is shown in Table 1 below.

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,0.2)</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>[0.2,0.4)</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>[0.4,0.6)</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>[0.6,0.8)</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>[0.8,1.0]</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

The second group of augmented variables represents the interaction terms between two continuous variables $X_i$ and $X_j$. Let C be the number of continuous variables in the data, we will then have a total $C \times (C-1)/2$ interaction terms for $X_i$ and $X_j$ plus $C$ quadratic terms $(X_i)^2$.

Neural network training and pruning

Feedforward neural networks with one hidden layer (Fig. 1) are trained using the original continuous input variables in the data plus their discretized values as well as the interaction terms among the continuous variables. Let I, H, and O be the number of input, hidden and output units, respectively. The goal of network training is to obtain a set of weights $(W,V)$ that produces the correct output given the input $X$. Here $W$ is the $(I \times H)$ matrix of weights for the
connections between the input units and the hidden units, and $V$ is the $(H \times O)$ matrix of weights for the connections between the hidden units and the output units. The objective function to be minimized during network training is a function of the difference between the correct (actual) output $y_i$ and the neural network’s predicted output $p_i$. A widely used error function is the quadratic error function:

$$E(W,V) = \frac{1}{2} \sum (y_i - p_i)^2$$  \hspace{1cm} (1)

Figure 1: A feedforward neural network with 5 input units, 3 hidden units and 2 output units.

where the summation is over all the training data samples, and the norm is the Euclidean norm. The $O$-dimensional vector of predicted output $p_i$ is computed as a function of the hidden unit activation values:

$$p_i = F(\alpha_i^T V)$$  \hspace{1cm} (2)

where $\alpha_i$ is the $H$-dimensional vector of activation values at the network’s hidden layer given input $X_i$. The function $F$ computes its output by transforming each of the $O$ components of $(\alpha_i^T V)$ via the sigmoid function:

$$f(a) = \frac{1}{1 + e^{-a}}$$  \hspace{1cm} (3)

The computation of the outputs using the sigmoid function (3) results in predicted outputs having values between 0 and 1. For a data set where there are samples from $K$ classes, $K$ output units in the network are required and the target output $y_i$ for each sample is a string of 0’s except for one 1 indicating the class membership.
The vector of hidden unit activation values \( \alpha_i \) is similarly computed as the sigmoid of the product between the inputs and the weights from the input units to the hidden units:

\[
\alpha_i = F(X_i^T W)
\]  

(4)

The training process starts from an initial random weight \((W_0, V_0)\) and a sequence of weights \((W_k, V_k)\) is then computed to minimize the error function (1). Any unconstrained optimization methods can be applied. The error back-propagation method updates the weights by moving along the negative gradient of the error function (Haykin, 1994). It is the most popular method for neural network training; however, this method is quite slow to converge. Other optimization techniques to speed up convergence have been proposed including the quasi Newton method and conjugate-gradient method. We implemented the BFGS method (Dennis, Jr. & Schnabel, 1983) to find a local minimum of the error function (1).

Network pruning removes redundant and irrelevant network connections from a trained network. By removing such connections, the generalization capability of the network can be improved. That is, the network predictive accuracy on new data samples is expected to be better when data overfitting is reduced. In order to facilitate the pruning process, an error term that penalizes the use of large weights is added to the error function (1). We add the following penalty term:

\[
p(w) = \frac{1}{2} \gamma \left[ w^2 + \frac{w^2}{1 + w^2} \right] 
\]  

(5)

for each weight \( w \) in the neural network with the penalty parameter \( \gamma \) set to a small positive value. By adding this penalty, network connections that are redundant or irrelevant are identified by their small magnitude. Such connections can be removed without sacrificing too much the accuracy of the network on the training data set. Since we have intentionally included many additional inputs to our data, the majority of which are expected to be not useful for classification, it is important that such inputs can be identified and their network connections can be removed efficiently. Our iterative network pruning algorithm identifies a network connection for potential removal by checking the network predictive accuracy if the value of the connection weight is set to 0. Starting from the connection from the first input unit to the first hidden unit, we check the connection for possible removal. It is removed if the accuracy of the network is still above a preset minimum threshold. Otherwise, the next possible connection is checked and so on. When there is no more connection that can be removed without suffering too large a drop in the accuracy, the connection that cause the smallest drop in accuracy is removed, and the network is retrained. The process is repeated if the retrained network recovers its accuracy above the threshold, otherwise the next connection producing the second smallest accuracy drop is removed and the network is retrained. The network pruning process is terminated when there is no connection that can be removed without decreasing the accuracy below the threshold.

**Rule extraction from neural networks**

A classification output from a neural network is computed as a relatively complex nonlinear function of its input (Eqs. 2-4). In many applications, it may be beneficial, even required to explain how the output is obtained in terms of rules that can be easier understood by a layman. Many neural network rule extraction methods that extract if-then-else rules from neural networks
can be found in the literature. We use one such method, Re-RX (Setiono et al., 2008) to extract rules from neural networks that have been trained and pruned with additional inputs. The main advantage of this rule extraction algorithm over other methods is that it generates rules where the rule conditions involving discrete attributes are disjoint from the rule conditions involving continuous attributes. Rules will first be generated by the algorithm whereby the rule conditions contain only the discrete attributes. When the accuracy of a rule with such condition is not satisfactory, the rules will then be refined by either (a) calling the algorithm recursively, or (b) using the values of the continuous attributes to split the subspace defined by the discrete attributes into two half-spaces.

In the context of credit scoring, we may for example obtain the following rules: If the applicant is single, then if his combined average monthly saving and total saving account balance exceeds a fix amount, then he is considered to be an applicant with good credit, otherwise bad credit. On the other hand, if the applicant is married, then if his combined average monthly saving and total saving account balance exceeds another fix threshold, then only he is predicted as good credit. In this example, the neural network predicts an applicant as good or bad using the information from 3 attributes: marital status, monthly average saving and total saving. The Re-RX algorithm first generates rules that involve only the discrete attribute, that is, marital status. The first generated rule set consists of two rules. Rule 1: If the applicant is single, then bad credit. Rule 2: Else if the applicant is married, then good credit. The accuracy of both rules is computed. Assuming that the errors of the two rules exceed the allowable threshold, then both rules will be refined. For both Rule 1 and Rule 2, their accuracy can be improved (in this example) by checking the values of the continuous attributes average monthly saving and total saving account balance.

CREDIT SCORING NEURAL NETWORKS WITH AUGMENTED INPUT

<table>
<thead>
<tr>
<th>Data set</th>
<th>Training set (TR)</th>
<th>Test set (TS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 0</td>
<td>Class 1</td>
</tr>
<tr>
<td>CARD1</td>
<td>291</td>
<td>227</td>
</tr>
<tr>
<td>CARD2</td>
<td>284</td>
<td>234</td>
</tr>
<tr>
<td>CARD3</td>
<td>290</td>
<td>228</td>
</tr>
</tbody>
</table>

The three variations of the CARD data set have been used in previous studies for credit scoring (Setiono et al., 2009; Setiono et al., 2011; Sexton et al. 2006) and they are available publicly (Prechelt, 1994). Table 2 shows the distribution of the data samples in the training and test sets. Each data sample represents a real credit card application and the class label describes whether the bank approves the application for credit card or not. Of the 690 samples, 307 samples have class label equals to 1 (good credit) and the remaining 383 have class label equals to 0 (bad credit).

A data samples 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 these attributes represents, continuous input attributes 4, 6, 41, 44, 49 and 51 are simply labeled C4, C6, C41, C44, C49, and
Section: Relevant Variables and Interactions in Credit Scoring

We used the 51 attributes $C_{51}$, respectively. The remaining binary-valued attributes are $D_1$, $D_2$, $D_3$, $D_5$, $D_7$, ..., $D_{40}$, $D_{42}$, $D_{43}$, $D_{46}$, $D_{47}$, $D_{48}$, and $D_{50}$. We normalized the continuous input attributes and discretized them by dividing the $[0; 1]$ normalized range into $N = 20$ subintervals of equal length. In total, the neural networks have $6 + 6 \times 20 + 6 \times 5/2 + 6 + 45 = 192$ inputs. The number of subintervals $N = 20$ was selected after experimentations with other values and it was found that $N = 20$ produced better results.

The number of hidden units was set to one, as it has been shown that networks with just one hidden unit are able to provide good predictive accuracy (Setiono et al., 2011). The number of output unit in the network was also set to one for this binary classification problem.

<table>
<thead>
<tr>
<th>Data</th>
<th>Original data set</th>
<th>Augmented data set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>AUC</td>
</tr>
<tr>
<td>CARD1(TR)</td>
<td>88.26 ± 0.49</td>
<td>88.16 ± 0.83</td>
</tr>
<tr>
<td>CARD1(TS)</td>
<td>88.72 ± 0.59</td>
<td>88.80 ± 0.46</td>
</tr>
<tr>
<td>CARD2(TR)</td>
<td>87.37 ± 1.22</td>
<td>86.53 ± 1.49</td>
</tr>
<tr>
<td>CARD2(TS)</td>
<td>85.64 ± 0.57</td>
<td>84.89 ± 0.73</td>
</tr>
<tr>
<td>CARD3(TR)</td>
<td>85.82 ± 0.57</td>
<td>86.85 ± 0.48</td>
</tr>
<tr>
<td>CARD3(TS)</td>
<td>86.94 ± 0.59</td>
<td>87.22 ± 0.63</td>
</tr>
</tbody>
</table>

The performance comparison of pruned neural networks that have been trained using the original data sets and with the augmented data sets is shown in Table 3. In addition to correct classification, we also measure the performance of the neural networks in terms of the area under the ROC curve (AUC). The Receiver Operating Characteristic (ROC) curve is a plot of the true positive (tp) rate on the Y axis against the false positive (fp) rate on the X axis. The two rates are computed as follows:

\[
fp = \frac{(# \text{ Class 0 samples classified as Class 1})}{(# \text{ Class 0})}
\]

\[
tp = \frac{(# \text{ Class 1 samples classified as Class 1})}{(# \text{ Class 1})}
\]

A random classifier will have a ROC curve that is a straight line connecting the origin to (1, 1) giving an AUC of 0.5. In contrast, a perfect classifier, having 100% true positive rate and 0% false positive rate, has an AUC of 1. From the figures in Table 3, we can see that the accuracy and the AUC results show small decrease from the networks trained with augmented data for CARD1 data set. A small improvement in accuracy and AUC is observed for CARD3 data set. A significant improvement in both accuracy and AUC is achieved by augmenting the training data set for the CARD2 data set. A possible explanation for these results is that predictive accuracy of the pruned neural networks on CARD1 and CARD3 data sets are already maximum or near maximum that it is not possible to gain further improvement given the available samples for network training. On the other hand, the accuracy of the networks trained with the original CARD2 data set is significantly lower than the comparable accuracy rates for the other two data sets that adding new discretized input and interaction terms really improves the overall

Table 4 compares the number of connections in the neural networks that have been trained using the original data sets and those that have been trained with the augmented data. We note that the average number of connections left in the pruned networks for CARD1 and CARD3 drops when the networks are trained with augmented inputs. In the case of CARD1, the drop is significant, from 16 to fewer than 12 connections. For CARD2 data set, the average number is increased by 1.56. This increase in the number of connections may also explain the higher predictive accuracy obtained on this data set.

We select one pruned network with good predictive accuracy rate to extract classification rules for each of the three data sets and compare the accuracy of the rules with the accuracy obtained by other methods. The predictive error rates of the rules extracted from neural networks that have been trained using augmented inputs are depicted in Table 5 as NN-ADVIT (Rules). It is clear that the classification rules extracted from pruned neural networks with the augmented inputs have lower error rates than the errors from the other methods. The NN (Rule) results are the error rates of the rules that have been extracted from neural networks trained with the original data (Setiono et al., 2011). The rest of the figures are taken from Sexton et al. (2006).

We select one pruned network with good predictive accuracy rate to extract classification rules for each of the three data sets and compare the accuracy of the rules with the accuracy obtained by other methods. The predictive error rates of the rules extracted from neural networks that have been trained using augmented inputs are depicted in Table 5 as NN-ADVIT (Rules). It is clear that the classification rules extracted from pruned neural networks with the augmented inputs have lower error rates than the errors from the other methods. The NN (Rule) results are the error rates of the rules that have been extracted from neural networks trained with the original data (Setiono et al., 2011). The rest of the figures are taken from Sexton et al. (2006).

<table>
<thead>
<tr>
<th>Data</th>
<th>Original data</th>
<th>Augmented data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD1</td>
<td>16.03 ± 1.13</td>
<td>11.67 ± 1.79</td>
</tr>
<tr>
<td>CARD2</td>
<td>9.07 ± 0.37</td>
<td>10.63 ± 0.89</td>
</tr>
<tr>
<td>CARD3</td>
<td>8.03 ± 0.53</td>
<td>7.83 ± 0.79</td>
</tr>
</tbody>
</table>

The extracted rules for the three data sets are shown below. We note that all three rule sets are more concise than those reported for the same data sets in (Setiono et al., 2011). We may
conclude that increasing the dimensionality of the training data by adding discretized input and interaction terms results in the pruned neural networks that have fewer input connections left. This in turn, results in rule sets that are more concise. A closer inspection of the rule sets indicates that none of the added discretized inputs is useful, while one or two of the interaction terms are useful. For the CARD1 data, the interaction term between $C_6$ and $C_{49}$ and the quadratic term $C_{51}^2$ appear in the rule condition. For the CARD2 and CARD3 data, only the quadratic term $C_{51}^2$ is found to be useful. Despite having majority of the newly added binary variables and interactions terms removed during network pruning, we can say that explicitly adding the interaction terms for network training allows us to obtain smaller networks with fewer connections and more concise rule sets while at the same time maintaining or increasing the overall predictive accuracy.

Rules for CARD1 data set:

Rule R_1: If $D_{42} = 1$ and $D_{43} = 1$, then predict Class 1,

Rule R_2: else if $D_7 = 0$ and $D_8 = 0$, then predict Class 1,

Rule R_3: else if $D_7 = 1$ and $D_{42} = 0$, then predict Class 0,

Rule R_4: else if $D_8 = 1$ and $D_{42} = 0$, then predict Class 0,

Rule R_5: else if $D_{43} = 0$, then

Rule R_{5a}: If $D_8 = 0$, then

Rule R_{5a-i}: If $C_{51} + 3.98C_6C_{49} - 2.35C_{51}^2 > 0$, then predict Class 0, else predict Class 1,

Rule R_{5a-ii}: predict Class 1,

Rule R_{5b}: else

Rule R_{5b-i}: If $C_{51} + 3.98C_6C_{49} - 2.35C_{51}^2 > 0.2$, then predict Class 0, else predict Class 1,

Rule R_6: else predict Class 0.

Rules for CARD2 data set:

Rule R_1: If $D_{12} = 1$ and $D_{42} = 0$, then predict Class 0,

Rule R_2: else if $D_8 = 1$ and $D_{42} = 0$, then predict Class 0,

Rule R_3: else if $D_{42} = 1$, then

Rule R_{3a}: if $C_{44} - 0.23C_{49} + 0.29C_{51}^2 > 0.012$, then predict Class 1,

Rule R_{3b}: else predict Class 0,

Rule R_4: else if $D_7 = 0$ and $D_8 = 0$, then predict Class 1,

Rule R_5: else predict Class 0.
Rules CARD3 for data set:

Rule R1: If D_{42} = 1, then

   Rule R_{1a}: If D_{43} = 0, then
   
   Rule R_{1a i}: If C_{49} + 7.75C_{51}^2 - 18.60C_{51}^2 < 0.19, then predict Class 1,
   
   Rule R_{1a ii}: else predict Class 0,
   
   Rule R_{1b}: else
   
   Rule R_{1b i}: If C_{49} + 7.75C_{51}^2 - 18.60C_{51}^2 < 2.30, then
   
   Rule R_{1b ii}: else predict Class 1,
   
   Rule R_2: else predict Class 1.

DISCUSSION AND CONCLUSIONS

Building a classification model that improves the accuracy of existing models is a challenging task as many researchers have spent much effort building the current models. However, it is always exciting to attempt building better models through development of new training algorithms, modifying learning parameters, or experimenting with data preprocessing. In this paper, we improve the overall accuracy and simplicity of the rules extracted from neural networks for credit scoring by modifying the neural network inputs. This modification is done by adding two types of new inputs generated from the values of the continuous input attributes in the data. The first type of new input is a set of discretized values of the continuous input attributes. For each continuous attribute, we add N-1 dummy (binary) variables with values 0 or 1 that are determined by the subinterval containing the continuous value. The subintervals are obtained by dividing the normalized range of the continuous attribute into N subintervals of equal length. The second type of new input is added so that it is easier to model interactions among the continuous variables as well as quadratic terms by the neural networks. This addition of new inputs leads eventually to simpler neural networks and more concise extracted rules.

Our experiments on benchmark credit scoring data produce interesting results. For the data set CARD1, a large drop in the average number of connections in the pruned networks is observed. The average accuracy of these networks is marginally lower than the accuracy of networks trained using the original data. From the second data set CARD2, an average of one more connection in the pruned networks is obtained. It gives more than 2% increase in the average accuracy and more than 3% increase in the average AUC. From the third data set CARD3, a small increase in the average accuracy and AUC is obtained despite a small decrease in the number of pruned network connections. The most notable result is that for all data sets, we are able to find networks that have better accuracy than what have been reported in the literature previously. We obtain rules that preserve the high accuracy of the networks. We believe that the presence of the interaction terms and the quadratic terms in pruned networks, and thus in the rule conditions enable us to obtain compact sets of rules with high predictive accuracy.

Our plan for the near future is to extract classification rules on other credit scoring and business analytics data sets. We are optimistic that good results in terms of accuracy, simplicity and novelty of the rules can be obtained from the data in these problem domains.
REFERENCES


Han, L., Han, L., & Zhao, H. (2013). Orthogonal support vector machine for credit scoring. Engineering Applications of Artificial Intelligence, 26, 848–862.


