12-31-2006

Risk Management and Regulatory Compliance: A Data Mining Framework Based on Neural Network Rule Extraction

Rudy Setiono  
National University of Singapore

Christophe Mues  
University of Southampton

Bart Baesens  
Katholieke Universiteit Leuven

Follow this and additional works at: http://aisel.aisnet.org/icis2006

Recommended Citation
http://aisel.aisnet.org/icis2006/7
RISK MANAGEMENT AND REGULATORY COMPLIANCE: A DATA MINING FRAMEWORK BASED ON NEURAL NETWORK RULE EXTRACTION

Design Science

Rudy Setiono
National University of Singapore
School of Computing
Singapore
disrudy@nus.edu.sg

Christophe Mues
University of Southampton
School of Management
United Kingdom
c.mues@soton.ac.uk

Bart Baesens
Katholieke Universiteit Leuven
Department of Decision Science and Information Management
Belgium
Bart.Baesens@econ.kuleuven.ac.be

Abstract

The recent introduction of various regulatory standards such as Basel II, Sarbanes-Oxley, and IFRS stimulates the need to develop new types of information systems based on data mining that will help improve the quality and automation of the decisions that need to be taken. Although neural networks have been frequently adopted in data mining applications, their opacity and black box character prevents them from being used to develop white box, comprehensible information systems for decision support in a financial context. In this paper, we introduce a new neural network rule extraction algorithm, Re-RX, that can be efficiently adopted to develop a data mining system for risk management in a Basel II context. The novelty of the algorithm lies in its new way of simultaneously working with discrete and continuous attributes without a need for discretization. Having extracted the Re-RX rules, we discuss how they can be used to build Basel II-compliant ICT systems taking into account the operational and regulatory requirements.

Keywords: data mining, regulatory compliance, neural networks, rule extraction

Introduction

Over the past decades, financial institutions have seen an ever-growing need for quantitative analysis techniques to optimize and monitor decisions related to such issues as risk and investment management, financial planning, trading, hedging, pricing and asset valuation, credit risk, and fraud detection. For some time now, the gradual adoption of data warehousing and knowledge discovery in data (KDD) technology is allowing these institutions to analyze ever-larger amounts of data, using a range of powerful techniques from various disciplines such as conventional statistics, machine learning, neurocomputing, and operations research. This process is only being further accelerated by the recent implementation of several international financial and accounting standards (such as Basel II, Sarbanes-Oxley, IFRS). For example, by allowing banks to use their internal credit risk assessment models as inputs to the minimum regulatory capital calculations, the Basel II framework is providing financial institutions with additional incentives to refine existing models, or develop new models, in compliance with certain specified standards. Hence, there has been a growing interest throughout the financial world in research on novel data mining techniques and information technologies to support the implementation of these compliance frameworks. As the result of a longstanding interest from the research community, a plethora of techniques has been proposed for many of the aforementioned problems, in particular for classification problems such as credit scoring or fraud detection.
However, not all of these approaches have proven readily transferable from the academic domain to financial practice. Many of the representations applied by the suggested algorithms cannot be easily interpreted and validated by humans. For example, neural networks are considered a black box technique, since the reasoning behind how they reach their conclusions cannot easily be obtained from their structure. This has not only hindered their acceptance by practitioners, but also fails to address the increasing need for transparency under various regulatory frameworks. Credit risk analysts are unlikely to accept black box techniques such as neural networks to make credit decisions, since under the Basel II accord, they are now required to demonstrate and periodically validate their models, and present reports to the national regulator for approval. Therefore, recent research proposed the use of neural network rule extraction to generate a powerful yet more intuitive and transparent rule set from an estimated neural network.

Baesens et al. (2001) analyzed credit risk data using Neurorule, a neural network rule extraction algorithm that works on data with discrete attributes. However, many real-world classification problems such as application scoring usually involve both discrete and continuous input attributes. For such problems, continuous attributes must be discretized before neural network training and rule extraction can commence. The drawback of discretizing continuous attributes is that the accuracy of the networks, and hence the accuracy of the rules extracted from the networks, may deteriorate. The reason for this is that discretization leads to a division of the input space into hyper-rectangular regions. Each condition of the extracted rules corresponds to one of the hyper-rectangular regions in which all data samples are predicted to belong to one class. Clearly, a data preprocessing step that divides the input space into rectangular subregions may impose unnecessary restrictions on neural networks as classifiers. It is highly likely that the boundaries of the regions that contain data samples from the same class are non-rectangular, given that some of the data attributes are continuous. Some rule extraction algorithms from neural networks do not require the discretization of continuous input data attributes. Setiono and Liu (1997) proposed Neurolinear, a neural network rule extraction algorithm that works with continuous data and produces hyperplane based rules. The conditions of the extracted rules are then expressed as linear combinations of the relevant input attributes, normally involving both discrete and continuous attributes. Such rule conditions may not be intuitive and do not facilitate understanding of the data.

In this paper, we introduce a new rule extraction algorithm, Re-RX, that is able to deal with both discrete and continuous variables, thus combining the advantages of Neurolinear and Neurorule, and allowing one to cope effectively with large datasets having a mix of both types of variables. Its effectiveness will be demonstrated on three real-life credit risk datasets for application scoring, to develop Basel II-compliant models predicting whether a new credit customer is likely to default or not (Baesens et al., 2003; Thomas, 2000). Once a satisfactory Re-RX rule set has been obtained, Basel II-compliant ICT systems need to be developed taking into account the operational and regulatory requirements. While in the KDD literature much attention has been paid to the former stages, relatively few guidelines are supplied with regard to the latter. In this paper, we show how the Re-RX rules can be used to build Basel II-compliant IT systems for verification and validation, decision support, performance monitoring, and Basel II regulatory capital calculations.

**Neural Networks for Risk Management**

Application scoring is a key risk management technique in the context of Basel II that tries to classify customers as good or bad payers based on their application characteristics (age, marital status, income, education level, etc.). Numerous classification techniques have already been adopted for application scoring. These techniques include traditional statistical methods (e.g. discriminant analysis and logistic regression (Steenackers & Goovaerts, 1989; Yobas et al., 2000), nonparametric statistical models (e.g., k-nearest neighbor (Henley & Hand, 1997; West, 2000) and decision trees (Yobas et al., 2000) and neural networks (Desai et al., 1996; West, 2000; Yobas et al., 2000). Often conflicts may be found when the conclusions of some of these studies are compared. One large-scale benchmarking study (Baesens et al. 2003) compares the classification performance of various state-of-the art classification techniques on eight real-life application scoring data sets. It concludes that neural networks perform very well in terms of classification accuracy.

A neural network is typically composed of an input layer, one or more hidden layers, and an output layer, each consisting of several units. Each unit processes its inputs and generates one output value that is transmitted to the units in the subsequent layer. Figure 1 shows an example of a neural network with four units in the input layer, three units in the hidden layer, and two units in the output layer for a binary classification problem. The weight of
the connection from input unit $m$ to hidden unit $l$ and the weight of the connection from hidden unit $l$ to output unit $p$

are denoted by $w_{ml}$ and $v_{lp}$, respectively.

![Diagram of a neural network](image)

**Figure 1. Example of a neural network.**

The output of a hidden unit and an output unit of the neural network are then computed as follows:

$$h^k_l = f^{(1)}(b^{(1)}_l + \sum_{m=1}^M w_{ml} x^k_m), \quad \text{and} \quad z^k_p = f^{(2)}(b^{(2)}_p + \sum_{l=1}^H v_{lp} h^k_l),$$

where $x^k_m$ is the $m$-th attribute value of an $M$-dimensional input sample $x^k$, $w_{ml}$ is the weight connecting input unit $m$ to hidden unit $l$, $b^{(1)}_l$ is the bias term for hidden unit $l$, $h^k_l$ is the output of hidden unit $l$ for sample $x^k$, $v_{lp}$ is the weight connecting hidden unit $l$ to output unit $p$, $b^{(2)}_p$ is the bias term for output unit $p$, $H$ is the number of hidden units, and $z^k_p$ is the output of output unit $p$ for sample $x^k$. The bias terms play a role analogous to that of the intercept term in a classical linear regression model. The class is then assigned according to the output unit with the highest activation value (winner-takes-all learning). The transfer functions $f^{(1)}$ and $f^{(2)}$ allow the network to model non-linear relationships in the data. Examples of transfer functions that are commonly used are the sigmoid function, the hyperbolic tangent function, and the linear function,

$$f(x) = \frac{1}{1 + \exp(-x)}, \quad f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}, \quad \text{and} \quad f(x) = x,$$

respectively.

The network is trained with the adjustment of its weights $(w, v)$ such that the sum of squared error function,

$$E(w, v) = \sum_{k=1}^K \sum_{p=1}^P (z^k_p - d^k_p)^2,$$

is minimized. Here $d^k_p$ denotes the desired output for sample $x^k$ at output unit $p$, $K$ denotes the number of samples used for training, and $P$ is the number of output units. The number of input units $M$ corresponds to the dimensionality of the input data, i.e., the number of attributes present in the data. To achieve reasonable accuracy on training data samples, a widely adopted practice in selecting the initial number of hidden units is to have more than the required hidden units. The network training process is terminated when a minimum of the error function is reached. To remove redundant input and hidden units, the network must be pruned. There are two benefits to network pruning. First, complex networks with many units and connections tend to overfit the data. They can predict training samples very accurately, but not unseen or new data samples. Second, by simplifying the network as much as possible, the rule extraction process can be done more efficiently, and the extracted rules can be more concise and easier to comprehend. We identify network connections for possible pruning by the magnitude of their weights. Connections with small weights are more likely to be redundant and can be removed without affecting network accuracy too much. When one or more connections are removed, the network is retrained. This iterative process is applied until there are no more network connections that can be removed without increasing the error rate of the network beyond the acceptable level (Setiono, 1997).
Recursive Rule Extraction Using Re-RX

Many techniques have been proposed for the extraction of rules from neural networks. They could be broadly categorized into two groups according to their translucency: decompositional approach and pedagogical approach (Tickle et al., 1998). The more translucent decompositional approach extracts the rules by analyzing the activation values of the individual hidden units and output units in the neural network. Rules are first generated to explain the individual unit's activation values and are later merged to obtain a rule set that explains the network's outputs in terms of its inputs. There are numerous examples of decompositional neural network rule extraction methods (Fu, 1994; Setiono & Liu, 1996; Setiono & Liu 1997; Tsukimoto, 2000; d'Avila Garcez et al., 2001; Andrews & Geva, 2002). On the other hand, the pedagogical approach is less translucent in the sense that the hidden unit activation values are not explicitly analyzed; instead, the rule extraction process is achieved by directly mapping the network inputs to its outputs. Numerous rule extraction techniques that fall under this category can also be found in the literature (Elalfi et al., 2004; Markowska-Kaczmar & Trelak, 2005; Etchells & Lisboa, 2006).

The key distinguishing feature of the rule extraction algorithm proposed here is its recursive nature that allows us to extract a set of classification rules with conditions involving discrete attributes and those involving continuous attributes that are disjointed. We believe that having such rules increases the comprehensibility of the rules extracted from the data sets collected in the particular domain problem we are addressing, i.e., risk management.

In the discussion below, we consider only two-group classification problems, although our proposed algorithm can be easily extended to handle multi-group problems. The outline of the algorithm is as follows:

**Algorithm Re-RX(S, D, C)**

**Input:** A set of samples $S$ having discrete attributes $D$ and continuous attributes $C$

**Output:** A set of classification rules

1. Train and prune a neural network using the data set $S$ and all its attributes $D$ and $C$.
2. Let $D'$ and $C'$ be the sets of discrete and continuous attributes still present in the network, respectively. Let $S'$ be the set of data samples that are correctly classified by the pruned network.
3. If $D' = \emptyset$, then generate a hyperplane to split the samples in $S'$ according to the values of their continuous attributes $C'$ and stop. Otherwise, using only discrete attributes $D'$, generate the set of classification rules $R$ for the data set $S'$.
4. For each rule $R_i$ generated:
   - If $\text{support}(R_i) > \delta_1$ and $\text{error}(R_i) > \delta_2$, then:
     - Let $S_i$ be the set of data samples that satisfy the condition of rule $R_i$, and $D_i$ be the set of discrete attributes that do not appear in the rule condition of $R_i$.
     - If $D_i = \emptyset$, then generate a hyperplane to split the samples in $S_i$ according to the values of their continuous attributes $C_i$ and stop.
   - Otherwise, call Re-RX($S_i$, $D_i$, $C_i$).

In Step 1 of the algorithm, the neural network is trained and then pruned to reduce data overfitting and to improve its generalization capability. Any neural network training and pruning method can be employed. The Re-RX algorithm does not make any assumption about the neural network architecture used, but we restrict ourselves to backpropagation neural networks with one hidden layer, as such networks have been shown to possess the universal approximation property (Bishop, 1995). An effective neural network pruning algorithm is a crucial component of any neural network rule extraction algorithm. By removing the inputs that are not needed for solving the problem, the extracted rule set can be expected to be more concise. In addition, the pruned network also serves to filter noise that might be present in the data. Such noise could be data samples that are outliers or incorrectly labeled. Hence, from Step 2 onward, the algorithm processes only those training data samples that have been correctly classified by the pruned network. If all the discrete attributes are pruned from the network, then in Step 3, the algorithm generates a hyperplane:

$$\sum_{C_i \in C} w_i C_i = w_0,$$
that separates the two groups of samples. The constant $w_0$ and the other coefficients $w_i$ of the hyperplane can be obtained by statistical or machine learning methods such as logistic regression or support vector machines. In our implementation, we employ a neural network with one hidden unit. Note that logistic regression, support vector machines, and neural networks can be easily applied to handle data sets having more than two classes.

When at least one discrete attribute remains in the pruned network, a set of classification rules involving only discrete attributes is generated. This step divides the input space into smaller subspaces according to the values of the discrete attributes. Each rule generated corresponds to a subspace, and when the accuracy of the rule is not satisfactory, the subspace is further subdivided by Re-RX. The support of a rule is the number of samples that are covered by that rule. The support level and the corresponding error rate of each rule are checked in Step 4. If the error exceeds the threshold $\delta_2$ and the support meets the minimum threshold $\delta_1$, the subspace of this rule is further subdivided by either calling Re-RX recursively when there are still discrete attributes not present in the conditions of the rule, or by generating a separating hyperplane involving only the continuous attributes of the data otherwise. By handling discrete and continuous attributes separately, Re-RX generates a set of classification rules that are more comprehensible than rules that have both types of attributes in their conditions. We illustrate the working of Re-RX in detail in the next section.

An Illustrative Example

To illustrate our algorithm, we use a credit approval data set that was used in a recent benchmarking study comparing different neural network models (Sexton et al., 2006). The data set is publicly available as the CARD3 data set (Prechelt, 1994). There are 690 samples in total, consisting of 345 training samples, 173 cross-validation samples, and 172 test samples. There are altogether 51 attributes, of which six are continuous, and the rest binary. As there is no detailed explanation on what each of the attribute 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 $C_{51}$, respectively. The remaining binary-valued attributes are $D_1$, $D_2$, $D_3$, $D_5$, $D_7$, ..., $D_{40}$, $D_{42}$, $D_{43}$, $D_{45}$, $D_{46}$, $D_{47}$, $D_{48}$ and $D_{50}$.

![Figure 2. The pruned neural network for the CARD3 data set has only 1 hidden unit, and of the 51 inputs, only 7 remain. The accuracy rates on the training set and test set are 87.26% and 88.95%, respectively.](image)

We train a neural network with one hidden layer using the available training and cross-validation samples. The number of input units is 51, and since there are two groups of samples, the number of output units is two. Once network training stops, the network is pruned to remove redundant network units and connections. The resulting pruned network is depicted in Figure 2. This pruned network has very few connections left, and its accuracy is better than those of other neural networks reported by Sexton et al. (2006). Of the 518 samples used for training, 87.26% (452 samples) are correctly predicted by the network.

We now explain the working of the Re-RX algorithm in detail. At the start of the algorithm, the data set $S$ consists of 518 samples, the continuous attribute set $C = \{C_4, C_6, C_{41}, C_{44}, C_{49}, C_{51}\}$, and the discrete attribute set $D$ contains all binary attributes. After network training and pruning, we have in Step 2 the set $S'$ consisting of 452 correctly predicted samples, the set $D' = \{D_1, D_2, D_{31}, D_{42}, D_{43}\}$ and $C' = \{C_{40}, C_{51}\}$. In Step 3, the algorithm generates rules
to separate the samples into two groups by applying the C4.5 decision tree method (Quinlan, 1993) using only the binary attributes. The following set of rules is generated:

**Rule R1:** If \( D_{41} = 1 \) and \( D_{43} = 1 \), then predict Class 1.

**Rule R2:** If \( D_{31} = 0 \) and \( D_{42} = 1 \), then predict Class 1.

**Rule R3:** If \( D_{1} = 0 \) and \( D_{42} = 1 \), then predict Class 1.

**Rule R4:** If \( D_{42} = 0 \), then predict Class 1.

**Rule R5:** If \( D_{1} = 1 \) and \( D_{31} = 1 \) and \( D_{43} = 0 \), then predict Class 2.

**Rule R6:** Default rule, predict Class 2.

The number of samples classified by each rule and the corresponding error rates are summarized in Table 1.

<table>
<thead>
<tr>
<th>Rule</th>
<th># Samples</th>
<th># Correct</th>
<th># Wrong</th>
<th>Support (%)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>163</td>
<td>163</td>
<td>0</td>
<td>36.06</td>
<td>0</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>26</td>
<td>25</td>
<td>1</td>
<td>5.75</td>
<td>3.85</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>12</td>
<td>9</td>
<td>3</td>
<td>2.65</td>
<td>25</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>228</td>
<td>227</td>
<td>1</td>
<td>50.44</td>
<td>0.44</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>23</td>
<td>13</td>
<td>10</td>
<td>5.09</td>
<td>43.48</td>
</tr>
<tr>
<td>( R_6 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>All rules</td>
<td>452</td>
<td>437</td>
<td>15</td>
<td>100</td>
<td>3.32</td>
</tr>
</tbody>
</table>

As rule \( R_5 \) misclassifies the highest number of training data samples, we describe how we refine this rule to improve its accuracy. First, the 23 samples classified by the rule are used to train a new neural network. The input attributes for this network are \( D_2, D_{42}, C_{49}, \) and \( C_{51} \). When the network is pruned, it turns out that only one hidden unit and two inputs, \( C_{49} \) and \( C_{51} \), are still left unpruned. The coefficients of a hyperplane that separates the samples from class 1 from those from class 2 can then be determined from the network connection weights from the input units to the hidden unit. The samples are separated as follows:

If 44.65 \( C_{49} \) - 17.29 \( C_{51} \) ≤ 2.90, then predict Class 1, otherwise predict Class 2.

If the thresholds \( \delta_1 \) and \( \delta_2 \) were set to 0.05, the algorithm would terminate after generating this rule. For completeness, however, let us assume that these parameters are set to zero. This would force Re-RX to generate rules to refine rules \( R_2, R_3, \) and \( R_4 \). When the algorithm finally terminates, the rules generated would correctly predict all the training samples that have been correctly classified by the original pruned neural network in Figure 2. The final set of rules generated is as follows:

**Rule R1:** If \( D_{41} = 1 \) and \( D_{43} = 1 \), then predict Class 1.

**Rule R2:** If \( D_{31} = 0 \) and \( D_{42} = 1 \), then

\[ \text{Rule R2a: If } C_{49} \leq 0.339, \text{ then predict Class 1,} \]

\[ \text{Rule R2b: Otherwise, predict Class 2.} \]

**Rule R3:** If \( D_{1} = 0 \) and \( D_{42} = 1 \), then

\[ \text{Rule R3a: If } C_{49} \leq 0.14, \text{ then predict Class 1,} \]

\[ \text{Rule R3b: Otherwise predict Class 2.} \]

**Rule R4:** If \( D_{42} = 0 \), then

\[ \text{Rule R4a: If } C_{49} \leq 0.339, \text{ then predict Class 1,} \]

\[ \text{Rule R4b: Otherwise predict Class 2.} \]
Rule R₄a: If \( D₁ = 0 \) and \( D₂ = 0 \) and \( D₄₃ = 1 \), then predict Class 1.

Rule R₄b: Otherwise predict Class 2.

Rule R₅: If \( D₁ = 1 \) and \( D₃₁ = 1 \) and \( D₄₃ = 0 \), then

Rule R₅a: If \( 44.65 \cdot C₄₉ - 17.29 \cdot C₅₁ \leq 2.90 \) then predict Class 1,

Rule R₅b: Otherwise predict Class 2.

Rule R₆: Default rule, predict Class 2

The accuracy of the above rules and the pruned neural network are summarized in Table 2.

| Table 2. Accuracy of the pruned neural network and the rules extracted by Re-RX for the CARD3 data set |
|-----------------------------------------------|---------------|---------------|
| Neural Network                               | Re-RX         |
| δ₁=δ₂=0.05                                   | δ₁=δ₂=0      |
| Training set                                 | 87.26%        | 86.29%        | 87.26%        |
| Test set                                     | 88.95%        | 88.95%        | 88.37%        |

Note that with smaller values for \( δ₁ \) and \( δ₂ \), the accuracy of the rules is exactly the same as the accuracy of the neural network on the training data set. However, the accuracy on the test set is slightly lower, as there is one sample that is correctly predicted by the network but not by the rules. Sexton et al. (2006) applied a genetic algorithm (GA) based neural network training technique and two commercial software packages, NeuralWorks and NeuroShell. They also included the results reported by Prechelt (1994). The accuracy rates on the test set are: GA: 88.37%, Prechelt: 81.98%, NeuralWorks: 87.79%, and NeuroShell: 84.88%. The C4.5 algorithm (Quinlan, 1993), a well-known decision tree classifier achieves 81.98% test set performance. From these numbers, we may conclude that our pruned neural network provides the highest predictive test set accuracy. The network has only one hidden unit and seven input units, and with that, Re-RX is able to generate a simple set of rules that preserves the accuracy of the network. In addition, we believe the rules are easy to understand, as the rule conditions involve discrete attributes and continuous attributes that are disjointed.

### Application Scoring Experiments

#### Data Sets and Empirical Setup

We used three real-life application scoring data sets in the experiments. Table 3 displays the characteristics of these data sets. The Bene1 and Bene2 data sets were obtained from two major financial institutions in Belgium, the Netherlands, and Luxembourg (Benelux). Both data sets are from the retail area and contain application characteristics of customers who applied for credit. For both data sets, a bad customer is defined as someone who has been in payment arrears for more than 90 days, which is consistent with the Basel II definition of default. The German credit data set is publicly available at the UCI repository (www.ics.uci.edu/~mlearn/MLRepository.html). Given the rather large number of observations of all three data sets, we split them into 2/3 training set and 1/3 test set. The neural networks were trained and rules were extracted with the training set. In line with the Basel II regulation, performance was estimated on independent hold-out test data.

Each continuous attribute in the data required one input unit in the network. A discrete attribute value was converted into a string of binary inputs with the use of either the dummy variable or the thermometer encoding scheme (Baesens et al., 2001). In general, a discrete variable with \( N \) possible values required \( N-1 \) input units. As a result, the numbers of input units in the networks trained on the German, Bene1, and Bene2 data sets were 63, 57, and 76, respectively. As all three problems were binary classification problems; that is, the number of output units was always two. We experimented with networks having a varying number of hidden units from one to five. At the end
of pruning, however, we discovered that networks with one hidden unit could provide good accuracy. This is consistent with earlier findings reported by Baesens et al. (2003). For the results reported here, all the rules were extracted from networks with only one hidden unit. The numbers of input units in the pruned networks for the three problems were 10, 9, and 11, respectively.

Table 3. Characteristics of data sets. C, D, and B stand for continuous, discrete, and binary, respectively

<table>
<thead>
<tr>
<th>Data set</th>
<th>Inputs</th>
<th>Data set size</th>
<th>Training set size</th>
<th>Test set size</th>
<th>Goods/Bads</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>7 C, 13 D (56 B)</td>
<td>1000</td>
<td>666</td>
<td>334</td>
<td>70/30</td>
</tr>
<tr>
<td>Bene1</td>
<td>18 C, 9 D (39 B)</td>
<td>3123</td>
<td>2082</td>
<td>1041</td>
<td>66.7/33.3</td>
</tr>
<tr>
<td>Bene2</td>
<td>18 C, 9 D (58 B)</td>
<td>7190</td>
<td>4793</td>
<td>2397</td>
<td>70/30</td>
</tr>
</tbody>
</table>

Results

Table 4 reports the results of applying the Re-RX algorithm to the three application scoring data sets. The results for Re-RX were obtained by setting $\delta_1 = \delta_2 = 0$. For comparison purposes, we also included results C5.0, Neurorule and Neurolinear. C5.0 is a successor of C4.5 (Quinlan, 1993). In addition to decision trees, C5.0 also generates classification rules. Neurorule is a neural network rule extraction method that assumes discrete inputs and generates propositional if-then rules (Setiono, 1997). Neurolinear is also a neural network rule extraction method, but it assumes continuous inputs and generates one or more hyperplanes as rule conditions (Setiono & Liu, 1997). Note that the data was split into exactly the same training and test sets as by Baesens et al. (2001), making the comparison fair. Performance was calculated as percentage of correctly classified (PCC) observations on the training and test sets.

Table 4. Accuracy and complexity of decision trees and neural network rule extraction techniques

<table>
<thead>
<tr>
<th>Data set</th>
<th>Method</th>
<th>PCC$_\text{train}$</th>
<th>PCC$_\text{test}$</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>C5.0 - decision tree</td>
<td>81.98</td>
<td>71.26</td>
<td>27 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>80.78</td>
<td>70.06</td>
<td>14 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>80.93</td>
<td>77.25</td>
<td>2 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>80.48</td>
<td>80.54</td>
<td>41 propositional rules</td>
</tr>
<tr>
<td>Bene1</td>
<td>C5.0 - decision tree</td>
<td>78.91</td>
<td>71.06</td>
<td>35 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>78.43</td>
<td>71.37</td>
<td>15 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>77.43</td>
<td>72.72</td>
<td>3 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>75.07</td>
<td>73.10</td>
<td>39 propositional rules</td>
</tr>
<tr>
<td>Bene2</td>
<td>C5.0 - decision tree</td>
<td>81.80</td>
<td>71.63</td>
<td>162 leaves</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>78.70</td>
<td>73.43</td>
<td>48 propositional rules</td>
</tr>
<tr>
<td></td>
<td>Neurorule</td>
<td>76.05</td>
<td>73.51</td>
<td>2 oblique rules</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>76.65</td>
<td>75.26</td>
<td>67 propositional rules</td>
</tr>
</tbody>
</table>

For each of the three application scoring data sets considered, Re-RX gave the best test set classification accuracy. The differences in accuracy with respect to the other algorithms were significant according to a one-tailed
McNemar’s test at the 1% significance level. Notably, unlike C5.0, the difference between training set accuracy and test set accuracy was rather small for Re-RX, clearly illustrating that the algorithm was not too sensitive to overfitting. This could be attributed to the effective pruning procedure, which removed the redundant connections from the network, thereby preventing it from focusing on the noise and idiosyncrasies in the data.

Although Re-RX tended to generate more rules than the other algorithms, that definitely added to the classification accuracy, contributing to improved discriminatory power. The higher number is caused by the algorithm that splits the rule condition according to whether the attributes are discrete or continuous. A typical rule set generated by the algorithm has the following structure:

If (discrete condition #1) then
    if (continuous condition #1) then predict class 1
    else predict class 2.
        ..........
Else if (discrete condition #K) then
    if (continuous condition #K) then predict class 1
    else predict class 2.
Else predict class 2.

A total of 2K+1 rules are generated. This number could be reduced to K+1 if the above rules are compressed and expressed as the following equivalent rule set:

If (discrete condition #1) and (continuous condition #1) then predict class 1
        ..........
Else if (discrete condition #K) and (continuous condition #K) then predict class 1
Else predict class 2.

The number of rules for Re-RX in Table 4 represents the number of uncompressed rules. Note that since the rules were exhaustive, only one rule was triggered for a new credit applicant, so the higher number of rules would not hinder interpretation. Summarizing, Re-RX provides an ideal balance between classification performance and interpretability in classifying new credit applicants.

In Baesens et al. (2003), the German, Bene1 and Bene2 data sets were used in a large-scale benchmarking experiment, which compares the performance of various state-of-the-art classification techniques for application scoring. Table 5 reports the best test set classification accuracy obtained as well as the performance of logistic regression, which is the industry standard for application scoring in the context of Basel II.1 Note that the test sets used were the same as in this paper.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Best performance and technique</th>
<th>Logistic regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>German</td>
<td>74.6 Linear discriminant analysis</td>
<td>74.6</td>
</tr>
<tr>
<td>Bene1</td>
<td>73.1 Support Vector Machine</td>
<td>72.0</td>
</tr>
<tr>
<td>Bene2</td>
<td>75.1 Neural network</td>
<td>74.4</td>
</tr>
</tbody>
</table>

The performance of a classification method is sometimes also measured in terms of sensitivity and specificity, especially when the distribution of the classes in the data set is imbalanced. Sensitivity refers to the proportion of

---

1 Logistic regression was also the technique used by the Bene1 and Bene2 institutions.
positive (or “good”) data samples that are correctly classified, while specificity refers to the proportion of negative (or “bad”) data samples that are correctly classified. In Table 6, we show the sensitivity and specificity of C5.0 tree and rules, as well as Neurolinear and Re-RX. The average of sensitivity and specificity is also computed. The figures for the average sensitivity and specificity indicate that the neural network extraction methods perform better than C5.0 on the test data.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Method</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training set</td>
<td>Test set</td>
<td>Training set</td>
<td>Test set</td>
</tr>
<tr>
<td>German</td>
<td>C5.0 - tree</td>
<td>0.8981</td>
<td>0.8821</td>
<td>0.6308</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>0.9236</td>
<td>0.8996</td>
<td>0.5282</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>0.8556</td>
<td>0.8515</td>
<td>0.6974</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>0.8854</td>
<td>0.9214</td>
<td>0.6103</td>
</tr>
<tr>
<td>Bene1</td>
<td>C5.0 - tree</td>
<td>0.8814</td>
<td>0.8065</td>
<td>0.5997</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>0.8821</td>
<td>0.8138</td>
<td>0.5836</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>0.8121</td>
<td>0.7639</td>
<td>0.6965</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>0.8043</td>
<td>0.7874</td>
<td>0.6408</td>
</tr>
<tr>
<td>Bene2</td>
<td>C5.0 - tree</td>
<td>0.9213</td>
<td>0.8564</td>
<td>0.5772</td>
</tr>
<tr>
<td></td>
<td>C5.0 - rules</td>
<td>0.8921</td>
<td>0.8600</td>
<td>0.5417</td>
</tr>
<tr>
<td></td>
<td>Neurolinear</td>
<td>0.8763</td>
<td>0.8468</td>
<td>0.4903</td>
</tr>
<tr>
<td></td>
<td>Re-RX</td>
<td>0.8897</td>
<td>0.8880</td>
<td>0.4458</td>
</tr>
</tbody>
</table>

We may conclude that the proposed Re-RX algorithm compares very favorably to the best classification algorithms as well as to the industry practice. Re-RX provides interpretable classification rules whereas linear discriminant analysis basically provides a hyperplane-based classification rule, and support vector machines, and neural networks are essentially black box models. Although the performance differences in absolute terms between Re-RX and the other algorithms may seem small at first sight, they are statistically significant for all three application scoring data sets. Moreover, small but statistically significant differences in the discriminatory power of even a fraction of a percentage point may, in the risk management context, translate into significant future savings and/or profit. From a Basel II perspective, models with good discriminatory power allow better quantification of risk-weighted assets, and thus also allow better determination of the regulatory safety capital for recovery from unexpected losses.

Figure 3 shows examples of rules generated for the German credit, Bene1, and Bene2 data sets by Neurolinear, Neurorule, and Re-RX. The first example illustrates a hyperplane rule extracted by Neurolinear on the German credit dataset. The rule is basically a linear discriminant function extracted from a neural network. Observe how the rule mixes both continuous and discrete inputs in its condition, making interpretation very difficult. Note that if there are N discrete attributes in a Neurolinear rule, and if we assume each one of them is binary-valued, then this rule actually represents up to $2^N$ parallel hyperplanes in the space defined by the continuous attributes. The second example in Figure 3 illustrates a propositional rule extracted by Neurorule for Bene1. In this rule, the attributes Term, Savings Account, and Years Client have been discretized, clearly imposing an unnecessary rectangular restriction on the classifier. The final two examples are hierarchical rules extracted by Re-RX for Bene1 and Bene2, respectively. In each example, the rules involving continuous attributes are disjointed from those involving discrete attributes, and they appear only in the lowest level in the rule hierarchy. As can be seen, both Re-RX rules are highly interpretable, providing the financial analyst with an explanation facility as to why credit applicants are rejected or accepted.
A hyperplane rule extracted by Neurolinear for German credit:
If \[-24.59(\text{Checking account}) + 29.66(\text{Term}) - 16.45(\text{Credit history}) - 3.66(\text{Purpose}) - 18.69(\text{Savings amount}) + 9.29(\text{Installment rate}) - 18.74(\text{Personal status}) + 6.19(\text{Property}) - 10.03(\text{Age}) - 9.36(\text{Other installment plans}) - 11.51(\text{Housing}) + 7.15(\text{Existing credits}) + 16.68(\text{Job}) + 2.046(\text{Number of dependents}) - 4.54(\text{Telephone}) - 8.29(\text{Foreign worker})\] \leq 0.15, then Customer = good payer. Else customer = defaulter.

A propositional rule extracted by Neurorule for Bene1:
If Term > 12 months and Purpose = cash provisioning and Savings amount \leq 12.40 Euro and Years client \leq 3, then customer = defaulter.

A hierarchical rule extracted by Re-RX for Bene1:
\text{Rule R}: If Purpose=cash and Marital status=not married and Known client=no, then
\text{Rule R1}: If Owns real estate=yes, then
\text{Rule R1a}: If Term of loan < 27 months, then customer=good payer.
\text{Rule R1b}: Else customer=defaulter.
\text{Rule R2}: Else Customer=defaulter.

A hierarchical rule extracted by Re-RX for Bene2:
\text{Rule R}: If Years Clients < 5 and Purpose \neq \text{Private Loan}, then
\text{Rule R1}: If Number of applicants \geq 2 and Owns real estate=yes, then
\text{Rule R1a}: If Savings amount + 1.11 Income -38249 Insurance -0.46 Debt > -1939300, then customer=good payer
\text{Rule R1b}: Else customer=defaulter
\text{Rule R2}: Else if number of applicants \geq 2 and Owns real estate=no then
\text{Rule R2a}: If Savings amount + 1.11 Income -38249 Insurance -0.46 Debt > -1638720, then customer=good payer
\text{Rule R2b}: Else customer=defaulter
\text{Rule R3}: Else if number of applicants=1 and Owns real estate=yes then
\text{Rule R3a}: If Savings amount + 1.11 Income -38249 Insurance -0.46 Debt > -1698200, then customer=good payer
\text{Rule R3b}: Else customer=defaulter
\text{Rule R4}: Else if Number of applicants=1 and Owns real estate=no then
\text{Rule R4a}: If Savings amount + 1.11 Income -38249 Insurance -0.46 Debt >-1256900, then customer=good payer
\text{Rule R4b}: Else customer=defaulter

Figure 3. Example rules generated by Neurolinear, Neurorule, and Re-RX.

Building Basel II Systems Using the Re-RX Rule Set

Up until now, we have largely focused on extracting a comprehensible set of rules to do risk management in a Basel II context. These rules now need to be further analyzed and used in various activities so as to arrive at a full-fledged, integrated, Basel II risk decision and management application. First, the rules can be transformed into PMML, an XML-based rule expression format, which will allow easy import and export between the various applications involved (http://www.dmg.org/). Figure 4 provides an example of a PMML specification of a Re-RX rule for Bene1, which was also reported in Figure 3.
A first set of tools can be used to verify and validate (V&V) the extracted rule set. Verification will attempt to look for syntax-based anomalies in the rule set. Whether the rule set is exhaustive (all cases being covered) and exclusive (a case only covered by one rule) will be investigated in this step. In the validation step, it will be investigated whether the rules adequately model the risk involved from a human interpretation viewpoint. The financial credit expert will be also consulted and asked to interpret the rule set in this step. Problems that may arise here are unexpected signs in the hyperplane part of the Re-RX rules, which may be due to spurious correlations in the data but do not represent the actual risk relationship. Note that this V&V step can be facilitated by transforming the rules into other representation formats such as decision tables or decision diagrams. An example tool that can be used to facilitate this step is Prologa (Vanthienen et al., 1998).

Once the rule set has been verified and validated, it needs to be implemented as a decision support system (DSS) that can be used by the loan officers so as to make the actual credit decision: accept or reject. The DSS can be implemented using a traffic light indicator approach that gives three possible outcomes: a green light, an orange light, or a red light (Tasche, 2003). A green light indicates that the rule set is confident enough to classify a customer as a good payer and credit should be accepted. An orange light indicates a doubt case for which human intervention is needed. This can be due to for example, low confidence of the rule set, external information obtained from a credit reference agency (e.g. Experian), a customer who is rejected as borderline by the rule set but is very profitable on other financial products, and/or a new marketing campaign in which the financial institution decides to grant credit to some of the more risky customers. The orange light can allow for model overrides by the credit expert. A low side override means that a customer rejected by the rule set is accepted, and a high side override vice
versa. A red light indicates that the rule set is confident enough to classify a customer as a bad payer and credit should be rejected. Note that this traffic light indicator approach can also be implemented using four colors (green, yellow, orange, red) or gauges in a dashboard application. An implementation of a traffic light indicator approach using four colors could be as follows. Red when Re-RX rule predicts bad customer and this is confirmed by credit bureau information; Orange when Re-RX rule predicts bad customer but credit bureau says customer is good risk; Yellow when Re-RX rule predicts bad customer but confidence is very low and credit bureau says customer is good risk; and Green when Re-RX rule says good customer and credit bureau says customer is good risk. Note that the financial institutions can decide for themselves on the number of colors and their meaning. Once this is done, the DSS can then typically be implemented as a Web Service into a Service Oriented Architecture (SOA) ICT environment.

The Basel II Capital Accord requires credit scoring systems to be validated at least annually. The accord distinguishes between backtesting, which is comparing the predicted outcome by the rule set with the realized outcome, and benchmarking, which is comparing the predicted outcome of the rule set with the outcomes of models of other parties in the industry (such as credit reference agencies, other financial institutions, or financial regulators). From a backtesting perspective, the performance of the rule set needs to be monitored. Again, a traffic light indicator approach can be adopted with three outcomes: green light, orange light, red light (Tasche, 2003). The decision about which light to switch on can be determined based on the outcome of a test statistic which monitors the classification accuracy (e.g. McNemar's test [Baesens et al., 2003]). A green light indicates that the rule set performance is stable, e.g. no significant differences at the 5% level are reported. It means the rule set can continue to be used. An orange light may indicate a difference at the 5% level but not at the 1% level of significance. It indicates a performance difference that requires no immediate action but needs to be closely monitored in the future. A red light then indicates a significant performance difference at the 1% level. It indicates that the model is no longer appropriate for the current data, which could possibly be due to a change of the population (often referred to as population drift) or a new strategy of the financial institution. In other words, the model needs to be rebuilt, which in our context would mean training and pruning a new neural network and extracting a new rule set using Re-RX. From a benchmarking perspective, a similar process can be conducted whereby the traffic lights now indicate how much the two parties agree or disagree on their credit decisions.

Finally, the rule set must also interface with a Basel II calculation engine that will use the rule outputs to calculate the regulatory capital that a financial institution needs to set aside in order to cover unexpected losses in its credit business. Note that this engine will then typically also use the LGD (Loss Given Default) and EAD (Exposure at Default) as additional inputs to its calculation. Figure 5 provides an outline of the different steps discussed above.

Figure 5. Building Basel II systems using Re-RX rules.

Conclusion

In this paper, we developed a new neural network rule extraction algorithm, Re-RX, which is capable of extracting hierarchical rules from trained neural networks. The novelty of the algorithm lies in its new way of simultaneously dealing with discrete and continuous attributes. Without a need for discretization of the continuous attributes prior to neural network training and pruning, the trained neural networks achieve very good predictive accuracy rates. The
rules extracted by Re-RX preserve the accuracy of the network, and they also provide a comprehensible explanation for the network’s decision-making process.

The recent introduction of the Basel II regulation has increased the need for financial institutions to build quantitative models for risk management, estimating, amongst others, the likelihood of a credit customer defaulting or not at the time of the application. Ideally, these models should be both effective and understandable at the same time, since they will be subject to supervisory review and evaluation. Since Re-RX starts from trained neural networks, which are very powerful because of their universal approximation property, and then subsequently extracts a set of if-then rules, both of these goals can be simultaneously achieved. Empirical evaluation using three real-life application scoring data sets illustrated the superior performance and interpretability of Re-RX. Furthermore, we also discussed and illustrated how the rules can be used to build the necessary Basel II systems to perform verification and validation, decision support, performance monitoring, and the Basel II regulatory capital calculations.

Acknowledgements

We wish to thank the financial institutions Bene1 and Bene2 for providing us with the application scoring data sets.

References

Bishop C., Neural Networks for Pattern Recognition, Oxford University Press, 1995.


