December 2001

Building Credit-Risk Evaluation Expert Systems Using Neural Network Rule Extraction and Decision Tables

Bart Baesens
Catholic University of Leuven

Rudy Setiono
National University of Singapore

Christophe Mues
Catholic University of Leuven

Stijn Viaene
Catholic University of Leuven

Jan Vanthienen
Catholic University of Leuven

BUILDING CREDIT-RISK EVALUATION EXPERT SYSTEMS USING NEURAL NETWORK RULE EXTRACTION AND DECISION TABLES

Bart Baesens
Department of Applied Economic Sciences
Catholic University of Leuven
Belgium
Bart.Baesens@econ.kuleuven.ac.be

Rudy Setiono
Department of Information Systems
National University of Singapore
Republic of Singapore
Rudys@comp.nus.edu.sg

Christophe Mues
Department of Applied Economic Sciences
Catholic University of Leuven
Belgium
Christophe.Mues@econ.kuleuven.ac.be

Stijn Viaene
Department of Applied Economic Sciences
Catholic University of Leuven
Belgium
Stijn.Viaene@econ.kuleuven.ac.be

Jan Vanthienen
Department of Applied Economic Sciences
Catholic University of Leuven
Belgium
Jan.Vanthienen@econ.kuleuven.ac.be

Abstract

The problem of credit-risk evaluation is a very challenging and important financial analysis problem. Recently, researchers have found that neural networks perform very well for this complex and unstructured problem when compared to more traditional statistical approaches. A major drawback associated with the use of neural networks for decision making is their lack of explanation capability. While they can achieve a high predictive accuracy rate, the reasoning behind how they reach their decisions is not readily available. In this paper, we present the results from analyzing two real life credit-risk evaluation data sets using neural network rule extraction techniques. Clarifying the neural network decisions by explanatory rules that capture the learned knowledge embedded in the networks can help the human experts in explaining why a particular decision is made. Furthermore, we also discuss how these rules can be visualized as a decision table in a compact and intuitive graphical format. Hence, extracting rules from trained neural networks and representing these rules as a decision table may offer a viable and valuable alternative for building credit-risk evaluation expert systems.

Keywords: Credit-risk evaluation, neural networks, rule extraction, decision tables.

INTRODUCTION

Neural networks have shown to be very powerful pattern recognition techniques for classification in a variety of domains. Their universal approximation property combined with their parallel processing ability is one of the major drivers behind their success. Unfortunately, an often mentioned drawback associated with using neural networks is their opacity. This refers to the fact that they do not allow formalization of the relationship between the outputs and the inputs in a user-friendly, comprehensible way. Neural networks are commonly described as black box techniques because they generate complex mathematical models which relate the outputs to the inputs using a set of weights, biases, and non-linear activation functions that are hard for humans to interpret.

In many application domains, it would be very useful to have a set of rules that explains why a particular classification is made. The domain of our current exploratory research is financial credit-risk evaluation. The importance of having a good model that aids the credit expert in decision making has been emphasized by many researchers. Since the pioneering work of Altman (1968), numerous methods have been proposed to develop models for financial decision making. These models include traditional statistical methods (e.g., discriminant analysis [Collins and Green 1982]), nonparametric statistical models (e.g., k-nearest neighbor [Henley and Hand 1996] and classification trees [David et al. 1992]), and neural network models. Especially the latter have proven to be very successful for credit-risk evaluation (West 2000). When applied to examine financial data, neural networks often outperform the other models in terms of predictive accuracy. However, most of these applications primarily focus at developing networks with high predictive accuracy without paying any attention to explaining how the classifications are being made. Clearly, this plays a very pivotal role in credit-risk evaluation as the evaluator may be required to give a justification why a certain credit application is approved or rejected.

Recent developments in algorithms that extract rules from trained neural networks enable us to generate explanatory classification rules that are as accurate as the networks. The purpose of our research is to investigate if these neural network rule extraction techniques can generate meaningful and accurate rule sets for the credit-risk evaluation problem. We conduct experiments on two real life credit-risk evaluation data sets. Our main interest lies in finding the distinguishing characteristics between good customers and bad customers. In this context, three popular neural network rule extraction techniques, Neurorule (Setiono and Liu 1996), Trepan (Craven and Shavlik 1996), and Nefclass (Nauck et al. 1997) are evaluated and contrasted. The performance of these methods is compared with the widely used C4.5 and C4.5rules algorithms (Quinlan 1993) in terms of predictive accuracy and conciseness of the generated rule sets or decision trees.

In a subsequent step of the expert system development process, the extracted rules are represented as a decision table (Vanthienen and Wets 1994; Wets et al. 1997). This is motivated by the fact that research in knowledge representation suggests that graphical representation formalisms can be more readily interpreted by humans than symbolic rules (Santos-Gomez and Darnell 1992). Representing the knowledge learned by the neural networks as a decision table allows the visualization of the rules in a format that is easily comprehensible and verifiable by the credit expert. Hence, in this paper, we investigate the usefulness of decision tables (DTs) as an intuitive graphical visualization aid for credit-risk evaluation purposes.

This paper is organised as follows. In the next section, we provide a brief overview of neural network rule extraction and discuss the Neurorule, Trepan, and Nefclass algorithms. The third section presents the empirical set up and the rule extraction results. Decision tables are then discussed. Conclusions are drawn in the final section.

NEURAL NETWORKS AND RULE EXTRACTION

Overview

Neural networks are very robust supervised learning tools for modelling complex non-linear relationships. They are basically mathematical representations inspired by the functioning of the human brain. A network with one hidden layer and one output layer performs the following non-linear function mapping:

$$\mathbf{y} = f_2(\mathbf{w_2} f_1(\mathbf{w_1 x})),$$

where **x** is the input vector and **y** is the output vector produced by the network. The transfer functions, f_1 at the hidden layer and f_2 at the output layer, allow the network to perform complex non-linear function mappings. Examples of transfer functions that are commonly used include the logistic and hyperbolic tangent transfer functions. The weight vectors, $\mathbf{w_1}$ and $\mathbf{w_2}$, need to be estimated during a training process. As universal approximators, neural networks can achieve significantly better predictive accuracy compared to models that are linear in the input variables. However, their complex mathematical internal workings prevent them from being used in real life situations (e.g., credit-risk evaluation) where besides having accurate models, explanation of the predictions being made is essential. In the literature, the problem of explaining neural network predictions has been tackled by techniques that extract symbolic rules from the trained networks. These neural network rule extraction techniques attempt to open up the neural network black box and generate symbolic rules with the same predictive power as the neural network itself. An advantage of using neural network rule extraction methods is that the neural network considers the contribution of the inputs toward classification as a group, while decision tree algorithms like C4.5 measure the individual contribution of the inputs one at a time as the tree is grown. In Duch et al. (2001), various rule extraction algorithms were compared by looking at the performance and the complexity of the generated rule sets on a number of publicly available benchmark data sets.

Andrews et al. (1995) propose a classification scheme for neural network rule extraction techniques based on various criteria. In this paper, we will mainly focus on two dimensions when discussing the algorithms: the translucency of the rule extraction algorithm and the expressive power of the extracted rules.

The translucency criterion considers the technique's perception of the neural network. A decompositional approach starts extracting rules at the level of the individual hidden and output units by analysing the activation values, weights and biases. Decompositional approaches then typically approximate the hidden units as threshold units. On the other hand, a pedagogical algorithm considers the trained neural network as a black box. Instead of looking at the internal structure of the network, these algorithms directly extract rules that relate the inputs and outputs of the network. These techniques typically use the trained network to classify examples and to generate additional artificial examples which are then used by a symbolic learning algorithm to infer the rules.

The expressive power of the extracted rules depends on the language used to express the rules. Propositional if-then rules are implications of the form *if X = a and Y = b, then Class = 1*. An example of a fuzzy classification rule is*: if X is low and Y is medium, then Class = 1*, whereby low and medium are fuzzy sets with corresponding membership functions. The M-of-N rule *if 2 of {X = a, Y = b, Z = c}, then Class = 1* is logically equivalent to if *((X = a and Y = b) or (X = a and Z = c) or (Y = b and Z = c)), then Class = 1.*

Neurorule

Neurorule is a decompositional algorithm that extracts propositional rules from trained three-layered feedforward neural networks (Setiono and Liu 1996). It consists of the following steps:

1. Train a neural network to meet the prespecified accuracy requirement.
2. Remove the redundant connections in the network by pruning while maintaining its accuracy.
3. Discretize the hidden unit activation values of the pruned network by clustering.
4. Extract rules that describe the network outputs in terms of the discretized hidden unit activation values.
5. Generate rules that describe the discretized hidden unit activation values in terms of the network inputs.
6. Merge the two sets of rules generated in steps 4 and 5 to obtain a set of rules that relates the inputs and outputs of the network.

Neurorule assumes the data are discretized and represented as binary inputs using the thermometer encoding (Smith 1993). The latter facilitates the process of generating propositional if-then rules.

The networks are trained to minimise the sum of squared errors of the predictions. They are subsequently pruned by inspecting the magnitude of the weights as discussed by Setiono (1997). Using pruned neural networks to extract rules results in more concise rule sets which are easier to interpret.

Trepan

Trepan was first introduced by Craven and Shavlik (1996). It is a pedagogical algorithm extracting decision trees from trained neural networks with arbitrary architecture. Like many decision tree induction algorithms, Trepan considers neural network rule extraction as an inductive learning task. It works by querying trained neural networks to induce a decision tree which represents the concept learned by the neural network. Trepan maintains a queue of leaves, which are expanded into sub-trees as they are removed from the queue. The trees are grown using a best-first expansion. The node at which there is greatest potential to increase the fidelity of the extracted tree with respect to the trained network will be the preferred node.

One major drawback of conventional decision tree induction algorithms like C4.5 (Quinlan 1993) is that the number of training examples available at a tree-node decreases with the depth of the tree. Hence, splits near the leaves of the tree may often be poorly chosen due to the insufficient number of training examples. Trepan overcomes this by enriching the original training data with additional, artificial training examples. In fact, these artificial training examples are generated for every tree node in which the number of original training examples is less then S_{min}, where S_{min} is a user-specified parameter of the algorithm. The artificial examples are generated taking into account all previously selected splits that lie on the path from the root of the tree to the current node. Furthermore, the generation process also takes into account the distributions of the individual features which are modeled using frequency counts for discrete-valued features and a kernel density estimation method for continuous features. The trained neural network is then used as an oracle which answers queries about class membership of the original and the artificial training examples.

Trepan induces both binary and M-of-N splits. The M-of-N splits are constructed using a hill-climbing search process. First, the best binary split is selected according to the gain ratio criterion which is also the criterion used in C4.5. This binary split is then used as a seed to construct M-of-N splits using again the gain ratio criterion and two operators discussed in Craven and Shavlik. Tree expansion is stopped using a statistical test to decide whether or not a node covers only instances of a single class or when a prespecified limit on the number of internal tree nodes is reached.

Nefclass

Nefclass (Neuro Fuzzy Classification) is a decompositional rule extraction algorithm that aims at extracting interpretable fuzzy rules from data (Nauck et al. 1997). Nefclass is based on a three-layer feedforward fuzzy perception whereby the first layer represents input variables, the hidden layer units represent fuzzy rules, and the third layer represents (crisp) output classes. Nefclass starts with an empty network having as many input neurons as there are inputs in the data set, zero hidden neurons, and as many output neurons as there are classes in the data set. For each input, fuzzy sets are defined to model linguistic concepts (e.g., small, medium and large). These fuzzy sets may have different types of membership functions (e.g., trapezoidal, triangular, bell-shaped).

A Nefclass system can be created with or without insertion of prior knowledge. When the classifier is created from scratch, a three-phase learning mechanism is used. In the first step, an initial rule base is constructed whereby hidden units are added until all training patterns have been covered by at least one rule. The best rules (hidden units) are then retained using a heuristic (e.g., the best k created rules). In the second step, the membership functions are trained using a variant of the well-known error backpropagation algorithm. In the third step, Nefclass offers the possibility to prune the rule base by removing rules and variables, based on a simple greedy algorithm which uses several heuristics (e.g., correlation and redundancy). This pruning step is fully automated without the need for user interaction. The goal of this pruning is to improve the comprehensibility of the created classifier. Note that the second and third steps are typically executed iteratively: after each pruning step, the membership functions are trained again.

NEURAL NETWORK RULE EXTRACTION EXPERIMENTS

Data Sets and Experimental Set Up

All neural network rule extraction techniques were applied to two real life credit-risk evaluation data sets. The first data set was obtained from a major Benelux financial institution. The other data set is the Statlog German credit data set which is publicly available at the Statlog repository (http://www.ncc.up.pt/liacc/ML/statlog/). The inputs include socio-demographic variables as well as loan specific information such as the purpose of the loan and its duration. For the Benelux data set, a bad loan was defined as a loan whereby the customer had missed three consecutive months of payments. Each data set was randomly split into two-thirds training set and one-third test set. The neural networks were trained and rules were extracted using the training set and evaluated on the test set. Table 1 displays the characteristics of both data sets. Table 2 presents the attributes for the German credit data set.

For the Benelux data set, we can not list the details of the attributes or the extracted rules due to our confidentiality agreement.

Table 1. Characteristics of Data Sets

	Number of Inputs	Data Set Size	Training Set Size	Test Set Size
German credit	20	1000	666	334
Benelux	33	3123	2082	1041

Table 2. Attributes for German Credit

Attribute	Type	Values
Checking account	qualitative	1: < 0 DM; 2: ≥ 0 and < 200 DM; 3: ≥ 200 DM/salary assignments for at least one year; 4: no checking account
Duration (in months)	numerical	
Credit history	qualitative	0: no credits taken/all credits paid back duly; 1: all credits at this bank paid back duly; 2: existing credits paid back duly till now; 3: delay in paying off in the past; 4: critical account/ other credits existing (not at this bank)
Purpose	qualitative	0: car (new); 1: car (old); 2: furniture/equipment; 3: radio/television; 4: domestic appliances; 5: repairs; 6: education; 7: vacation; 8: retraining; 9: business; 10: others
Credit amount	numerical	
Savings account	qualitative	1: < 100 DM; 2: ≥100 and < 500 DM; 3: ≥500 and < 1000 DM; 4: ≥1000 DM; 5: unknown/no savings account
Present employment since	qualitative	1: unemployed; 2: <1 year; 3: ≥1 and < 4 years; 4: ≥4 and < 7 years; 5: ≥7 years
Installment rate in percentage of disposable income	numerical	
Personal status and sex	qualitative	1: male, divorced/separated; 2: female, divorced/separated/married; 3: male, single; 4: male, married/widowed; 5: female, single
Other parties	qualitative	1: none; 2: co-applicant; 3: guarantor
Present residence since	numerical	
Property	qualitative	1: real estate; 2: if not 1: building society savings agreement/ life insurance; 3: if not ½ : car or other; 4: unknown/ no property
Age in years	numerical	
Other installment plans	qualitative	1: bank; 2: stores; 3: none
Housing	qualitative	1: rent; 2: own; 3: for free
Number of existing credits at this bank	numerical	
Job	qualitative	1: unemployed/unskilled-non-resident; 2: unskilled-resident; 3: skilled employee/official; 4: management/self-employed/ highly qualified employee/officer
Number of people being liable to provide maintenance for	numerical	
Telephone	qualitative	1: none; 2: yes, registered under the customers name
Foreign worker	qualitative	1: yes; 2: no

Both data sets are discretized using the discretization algorithm of Fayyad and Irani (1993). We include C4.5 and C4.5rules as a benchmark to compare the results of the rule extraction algorithms. Both algorithms use an error-based pruning strategy to prune conditions and/or rules with a default confidence level of 25% (Quinlan 1993). The classification accuracy and the complexity of the generated trees or rules are compared and discussed. Complexity for the trees is measured by the number of leaf nodes and the total number of nodes.

We start with extracting rules using Neurorule. All neural networks have hyperbolic tangent hidden units (f_1) and linear output units (f_2). We use two output units and the class is assigned to the output unit with the highest activation value (winner take all learning). The neural networks are trained and pruned according to the set up discussed earlier. The same pruned networks are also used to extract decision trees using Trepan. Since Trepan is a pedagogical tree extraction algorithm, we can apply it to any trained neural network with arbitrary architecture. This allows us to make a fair comparison between a decompositional approach (Neurorule) and a pedagogical approach (Trepan). Following Craven and Shavlik (1996), we set the parameters for the Trepan analyses as follows: S_{min} = 1000 and the maximum tree size to 15 internal nodes.

Finally, we also included Nefclass as an example of a neurofuzzy rule extraction system. We defined three fuzzy sets for each variable which were modeled using triangular or bell-shaped membership functions.

Neural Network Rule Extraction Results

Table 3 presents the results of applying C4.5, C4.5rules, Neurorule, Trepan, and Nefclass to the discretized data sets. Both the classification accuracy as measured by the percentage correctly classified (PCC) samples and the complexity of the generated rules or trees are depicted.

Table 3. Results of the Extraction Algorithms

Data Set		PCC_{train}	PCC_{test}	Complexity
German credit	C4.5	80.63	71.56	38 leaves, 54 nodes
	C4.5rules	81.38	74.25	17 propositional rules
	Neurorule	76.13	75.15	7 propositional rules
	Trepan	75.38	73.95	3 leaves, 5 nodes
	Nefclass	73.57	73.65	14 fuzzy rules
Benelux	C4.5	78.29	68.68	57 leaves, 98 nodes
	C4.5rules	77.23	70.12	19 propositional rules
	Neurorule	72.38	71.47	7 propositional rules
	Trepan	72.38	71.47	6 leaves, 11 nodes
	Nefclass	69.84	69.07	3 fuzzy rules

It may be concluded from Table 3 that Neurorule is able to extract very concise rule sets with a high classification accuracy on the test set for both discretized data sets. Although the performance differences between Neurorule and C4.5rules are not significant according to the chi-squared based McNemar's test (using a significance level of 5%) (Fleis 1981), the rule sets extracted by the former are a lot more compact than those extracted by the latter. Nefclass was never able to infer compact and powerful fuzzy rule sets. Also notice that for both data sets, Trepan has a significantly better performance than C4.5 with significantly fewer nodes and leaves (again at the 5% significance level).

The pruned neural networks that were used by both Neurorule and Trepan had one hidden unit. After binarization using the thermometer encoding and pruning, six inputs remained for the German credit data set and five inputs for the Benelux data set. For the German credit data set, the rules extracted by Neurorule performed somewhat better than the network itself. For the Benelux data set, Neurorule obtained the same performance as the neural network. Trepan obtained the same performance as the neural network on both data sets. This clearly indicates that both Neurorule and Trepan are able to accurately approximate the decision process of the trained neural networks.

```
5 of {Other parties <3, Other installment plans = 1, Savings account < 3, Duration = 2,
Credit history < 4, Checking account < 3}
    Checking account < 3: Applicant = bad
    Checking account ≥ 3: Applicant = good
NOT 5 of {Other parties < 3, Other installment plans = 1, Savings account < 3, Duration = 2,
Credit history < 4, Checking account < 3}: Applicant = good
```

Figure 1. Trepan Tree for German Credit Data

```
if (Checking account ≥ 3) then
    Applicant = good

if (Duration = 1) and (Other installment plans≥2) then
    Applicant = good

if (Other installment plans≥2) and (Credit history = 4) then
    Applicant = good

if (Duration = 1) and (Credit history = 4) then
    Applicant = good

if (Savings accoun t≥3) then
    Applicant = good

if (Other parties = 3) then
    Applicant = good

Default class: Applicant = bad
```

Figure 2. Rules Extracted by Neurorule for German Credit

In summary, Neurorule extracted concise rule sets with a very good predictive accuracy on both discretized data sets. Also Trepan gave good results in terms of classification accuracy and tree complexity. Figures 1 and 2 present the tree and rules extracted by Trepan and Neurorule for the German credit data set. We note that the duration attribute was discretized as follows: 1 if duration less than or equal to 15 months, 2 otherwise. For the meaning of the other attributes, we refer to Table 2.

VISUALIZING THE EXTRACTED RULE SETS USING DECISION TABLES

Decision tables (DTs) are a tabular representation used to describe and analyze decision situations, where the state of a number of conditions jointly determines the execution of a set of actions (Vanthienen and Wets 1994). A DT consists of four quadrants, separated by double-lines, both horizontally and vertically (cf. Figure 3). The horizontal line divides the table into a condition part (above) and an action part (below). The vertical line separates subjects (left) from entries (right).

condition subjects	condition entries
action subjects	action entries

Figure 3. DT Quadrants

The *condition subjects* are the criteria that are relevant to the decision making process. They represent the attributes about which information is needed to classify a given case. The *action subjects* describe the possible outcomes of the decision making process (i.e., the classes of the classification problem). Each *condition entry* describes a relevant subset of values (called a *state*) for a given condition subject (attribute), or contains a dash symbol (–) if its value is irrelevant within the context of that column.

Subsequently, every *action entry* holds a value assigned to the corresponding action subject (class). *True*, *false*, and *unknown* action values are typically abbreviated by "x," "–," and ".," respectively. Every column in the entry part of the DT thus comprises a classification rule, indicating what actions apply to a certain combination of condition states. If each column only contains simple states (no contracted or irrelevant entries), the table is called an *expanded* DT, whereas otherwise the table is called a *contracted* DT.

Several kinds of DTs have been proposed. We will require that the condition entry part of a DT satisfies the following two criteria:

❑ *completeness*: all possible combinations of condition values are included;

❑ *exclusivity*: no combination is covered by more than one column.

As such, we deliberately restrict ourselves to *single-hit* tables, wherein columns have to be mutually exclusive, because of their advantages with respect to verification and validation (Vanthienen et al. 1998). It is this type of DT that can be easily checked for potential anomalies, such as inconsistencies (a particular case being assigned to more than one class) or incompleteness (no class assigned). Additionally, for ease of legibility, the columns are arranged in lexicographical order, in which entries in lower rows alternate first. As a result, a tree structure emerges in the condition entry part of the DT, which lends itself very well to a top-down evaluation procedure: starting at the first row, and then working one's way down the table by choosing from the relevant condition states, one safely arrives at the prescribed action (class) for a given case. This condition-oriented inspection approach often proves more intuitive, faster, and less prone to human error than evaluating a set of rules one by one. Once the decision table has been approved by the expert, it can, in a final stage, be incorporated into a deployable expert system (Vanthienen and Wets 1994).

Figure 4 depicts the decision table generated from the rules extracted by Neurorule for the German credit data set. It was built using the PROLOGA workbench (http://www.econ.kuleuven.ac.be/tew/academic/infosys/research/prologa.htm).

Note that the fully expanded table contained 1,800 columns, which is the product of the number of attribute values (= 4 × 5 × 3 × 5 × 2 × 3). This could be reduced to nine columns by means of table contraction, which is a sufficiently small number for easy use. Table 4 presents the decision table properties for both Neurorule and Trepan on the credit-risk evaluation data sets.

In all cases, the contracted tables were satisfactorily concise and did not contain any anomalies, thus demonstrating the completeness and the consistency of the extracted rules. We also constructed the decision tables for the rules induced by C4.5rules and found that these tables were huge and impossible to handle because of the large number of generated rules and unpruned inputs.

German Credit									
1. Checking account	< 3								≥ 3
2. Savings account	< 3							≥ 3	–
3. Other parties	< 3						3	–	–
4. Credit history	< 4			= 4			–	–	–
5. Duration	1		2	1	2		–	–	–
6. Other installment plans	< 2	≥ 2	–	–	< 2	≥ 2	–	–	–
1. Applicant = Good	–	x	–	x	–	x	x	x	x
2. Applicant = Bad	x	–	x	–	x	–	–	–	–
	1	2	3	4	5	6	7	8	9

Figure 4. Decision Table for Neurorule on German Credit

Table 4 . Decision Table Properties of Extraction Methods on Credit-Risk Evaluation Data Sets

Data Set	Extraction Method	# Columns in Expanded Decision Table	# Columns in Contracted Decision Table
German credit	Neurorule	1800	9
German credit	Trepan	1800	16
Benelux	Neurorule	144	6
Benelux	Trepan	144	6

CONCLUSION

Many recent papers reporting on the application of neural networks for financial analysis suggest that neural networks can outperform other methods for classification but they do not provide an explicit explanation on how a classification is made. In application domains such as credit-risk evaluation, having a set of concise and comprehensible rules is essential. In this paper, we evaluated and contrasted three neural network rule extraction techniques, Neurorule, Trepan, and Nefclass for credit-risk evaluation. The experiments were conducted on two real life financial credit-risk evaluation data sets. The results were compared to the decision trees and rules induced by the popular C4.5 algorithm. Both the conciseness and the predictive power of the rules and trees were investigated. It was shown that both Trepan and Neurorule were able to extract compact trees and rules for both data sets with good predictive accuracy. We also described how decision tables could be used to represent the extracted rules. Decision tables represent the rules in an intuitive graphical format that can be easily verified by a human expert. Hence, the use of neural network rule extraction, in combination with decision tables, forms a viable and valuable alternative for building advanced credit-risk evaluation expert systems.

References

Altman, E. L. "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy," *Journal of Finance* (23), 1968, pp. 589-609.

Andrews, R., Diederich, J., and Tickle, A. B. "A Survey and Critique of Techniques for Extracting Rules from Trained Neural Networks," *Knowledge-Based Systems* (8:6), 1995, pp. 373-389.

Collins, R. A., and Green, R. D. "Statistical Methods for Bankruptcy Forecasting," *Journal of Economics and Business* (32), 1982, pp. 349-354.

Craven, M. W., and Shavlik, J. W. "Extracting Tree-Structured Representations of Trained Networks," in *Advances in Neural Information Processing Systems,* D. Touretzky, M. Mozer, and H. Hasselmo (eds.), MIT Press, Cambridge, MA, 1996, pp. 24-30.

David, R. H., Edelman, D. B., and Gammerman, A. J. "Machine Learning Algorithms for Credit-Card Applications," *IMA Journal of Mathematics Applied in Business and Industry* (4), 1992, pp. 43-51.

Duch, W., Adamczak, R., and Grabczewski, K. "A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules," *IEEE Transactions on Neural Networks* (12), 2001, pp. 277-306.

Fayyad, U. M., and Irani, K. B. "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France, 1993, pp. 1022-1029.

Fleis, J. L. *Statistical Methods for Rates and Proportions,* Wiley, New York, 1981.

Henley, W. E., and Hand, D. J. "A k-Nearest Neighbor Classifier for Assessing Consumer Credit Risk," *Statistician* (44), 1996, pp. 77-95.

Nauck, D., Klawonn, F., and Kruse, R. *Foundations of Neuro-Fuzzy Systems,* Wiley, New York, 1997.

Quinlan, J. R. *C4.5 Programs for Machine Learning,* Morgan Kaufmann Publishers, San Francisco, 1993.

Santos-Gomez, L., and Darnell, M. J. "Empirical Evaluation of Decision Tables for Constructing and Comprehending Expert System Rules," *Knowledge Acquisition* (4), 1992, pp. 427-444.

Setiono, R. "A Penalty Function Approach for Pruning Feedforward Neural Networks," *Neural Computation* (9:1), 1997, pp. 185-204.

Setiono, R., and Liu, H. "Symbolic Representation of Neural Networks," *IEEE Computer* (29:3), 1996, pp. 71-77.

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

Vanthienen, J., Mues, C., and Aerts, A. "An Illustration of Verification and Validation in the Modeling Phase of KBS Development," *Data & Knowledge Engineering* (27), 1998, pp. 337-352.

Vanthienen, J., and Wets, G. "From Decision Tables to Expert System Shells," *Data & Knowledge Engineering* (13:3), 1994, pp. 265-282.

West, D. "Neural Network Credit Scoring Models," *Computers & Operations Research* (27:11-12), 2000, pp. 1131-1152.

Wets, G., Vanthienen, J., and Piramuthu, S. "Extending a Tabular Knowledge Based Framework with Feature Selection," *Expert Systems with Applications* (13), 1997, pp. 109-119.