Routledge
Taylor & Francis Group

TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
ISSN 2029-4913 print/ISSN 2029-4921 online

2011 Volume 17(3): 519–534
doi:10.3846/20294913.2011.606339

NEURAL NETWORKS AND THEIR APPLICATION IN CREDIT RISK ASSESSMENT. EVIDENCE FROM THE ROMANIAN MARKET

Smaranda Stoenescu Cimpoeru

Academy of Economic Studies, Faculty of Cybernetics, Statistics and Informatics Economics, Department Cybernetic Economics, Romana Square, nr. 6, 010374 Bucharest, Romania
E-mail: smarandacimpoeru@yahoo.com

Received 14 March 2011; accepted 21 June 2011

Abstract. The purpose of this article is to see how neural networks are used in credit risk assessment problems. For this, we firstly introduce the main theoretical concepts of the neural calculus, as well as the fundaments for the main training algorithm: the error back-propagation algorithm. We review the specialty literature and find that the neural networks yield better results than other classification techniques, like multivariate discriminant analysis or logistic regression, when applying them in credit risk assessment problems. We focus on a few types of networks: feed-forward networks with multiple layers, fuzzy adaptive networks, support vector machines. We develop an analysis on Romanian Small and Medium Enterprises (financial ratios) and the results are in line with the findings from the literature: the neural networks give better results than the logistic regression. The study can be developed by analyzing a support vector machine or a fuzzy adaptive network.

Keywords: neural networks, credit risk, network training algorithms.

Reference to this paper should be made as follows: Stoenescu Cimpoeru, S. 2011. Neural networks and their application in credit risk assessment. Evidence from the Romanian market, *Technological and Economic Development of Economy* 17(3): 519–534.

JEL Classification: C45, G32.

1. Introduction

Starting with the 80s, the artificial intelligence (AI) techniques have become more and more used in predicting defaults and in discriminating clients on risk classes. Neural networks are part of the AI techniques. They have been used in a wide variety of financial and economic problems, like: discrimination / classification problems (assessing credit risk, securities classification, bankruptcy prediction), problems of series prediction like predicting the price of stock index, problems of predicting the profit from an investment project.

Vellido *et al.* (1999) make a comparative study that addresses the utilization of neural networks in different financial and economic problems. The study refers to the period 1992–1998, period in which, according to the authors, the statistic context of applying neural networks has evolved towards a practical, principle based methodology. The cited article mainly analysis decision taking problems, which also include the assessing of credit risk and the prediction of the bankruptcy event.

From Vellido *et al.* (1999) we find the main advantages and disadvantages of the neural networks. Firstly, we note that in the reviewed studies, the advantages of the neural networks are cited more often than the disadvantages. The main advantages mentioned in the study are: successfully applying neural networks for incomplete data sets; neural networks consist in a non-parametric method that doesn't need a prior hypothesis on the variables probability distribution; the ability of the neural networks of transforming any non-linearity, no matter how complex they are. Refenes (1994) argues that neural network techniques know a growing popularity and they cannot be seen as a temporary step or phase in the credit risk modeling techniques evolution.

The sole disadvantage, common to all studies, is the "black-box" approach of the neural networks, that doesn't permit the knowledge of the decision rules leading to a certain output of the network. As consequences of the "black-box" system, one cannot create a hierarchy of the input variables based on their relevance, nor can one choose between different models based from the set of rules that determines the mentioned models.

The traditional approach for assessing the companies' credit risk is mostly based on the credit risk officers' experience. This implies a certain subjectivity of the crediting process. Considering the aggressive dynamicity of the business environment, the task of the credit officers has become more and more complex. In this context, the need of using computerized programs for assessing the credit risk has became imperious. Since the late 90s, the neural network technique has been used in computer based programs for assessing the credit risk for portfolios of financial institutions. As programs used by banks that involve neural calculus, we mention CreditView implemented by former Chase Manhattan bank in USA (actual J. P. Morgan) and AQUARIUS, used by Foster Ousley Conley, a company specialized in mortgage loans.

The structure of the paper is as follows: in Section 2 we introduce the main concepts of the neural calculus (elements of the neural networks, types of network architecture); in Section 3 we review the learning rules and training algorithms, while in Section 4 we explain the concept of perceptron and the main idea of the error back-propagation network; in Section 5 review the specialty literature of applying the neural networks in credit risk assessment problems; we present multiple types of networks: feed-forward networks with multiple layers, fuzzy adaptive networks, support vector machines. Eventually, in Section 6 we develop an analysis on Romanian Small and Medium Enterprises (financial ratios) – the results show that the neural networks give better results than the logistic regression.

2. Main concepts used in neural calculus. Network architecture

Neural networks are learning system-machines, made of simplified neuron models, that after going under the learning process, modify their internal parameters, with the aim of performing a certain task.

The aim of a neural network is to transpose a certain input configuration into an output configuration. The outputs of the network may correspond to an action to be done by the network. The main tasks fulfilled by neural networks are: classification, pattern recognition, prediction. In general, the output can be any transformation of the input space, thus the network is the result of a projection, of a transformation of the input vectors into the output space. A comprehensive overview of neural networks can be found in Michie *et al.* (1994).

The elements that make a neuron are the following: the set of synaptic connections and the activation function. Each synaptic connection is characterized by a weight; the signal x_j associated with the neuron k is multiplied with the weight $w_{k,j}$; the weights determine the type and the intensity of the information changed. Actually, in the set of weights resides the information the networks uses to fulfill its task. The linear combination of inputs and weights is denoted with:

$$u_k = \sum_{j=1}^{m} x_j w_{k,j} \tag{1}$$

and represents the assessment of the input signals as well as the input value for the activation function.

The activation function is denoted with $a(\cdot)$. By applying this function, we limit the possible values for u_k. After applying the activation function, we obtain the output of the neuron, y_k .There are multiple types of activation functions: the linear function, the sigmoid function (like the hyperbolic tangent function), the signum function, etc.

As per Haykin (1999), there are three possible type of architecture for a neural network. Firstly, there are the feed-forward networks with a single layer of neurons. The feed-forward networks assume that the connections from one layer of neurons to another are permitted only in a single direction. This kind of network is also called an a-cyclical network, due to the fact that there are no feedback loops. The single layered network is the simplest form of layered network: there is only one layer of input nodes and the layer of output nodes. The second type of network architecture is the multi-layered feed-forward network. In this case, we have one or multiple hidden layers of neurons. The purpose of adding hidden layers of neurons is the increase in the network accuracy for fulfilling its task. The source nodes in the input layer serve as input values for the activation function – the result of the activation function represents the input for the neurons of the first hidden layer and so on. This scheme is continued until the final layer of neurons. The last type of network architecture is the recurrent network, which allows the presence of back and forward connections between layers.

3. Training the network. Types of learning

Neural networks learning rules represent the foundation of learning algorithms. We present briefly three of the most popular training rules. We start with the error back propagation learning rule, the most common rule used in developing the supervised training algorithms for the feed-forwards networks

In the context of a network, the output of the network, $y_k(n)$, is compared with a desired or target output, denoted $d_k(n)$. With these notations, the error signal will be:

$$e_k(n) = d_k(n) - y_k(n). \tag{2}$$

Our objective is to minimize the errors, or mathematically speaking, minimizing the function:

$$E(n) = 1/2e_k^2(n). \tag{3}$$

This training rule is better known as the delta rule or the Widrow-Hoff rule, after its authors (1960). The delta rule can be written as:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n). \tag{4}$$

Another training rule is the one based on memory. All or almost all past experiences are kept in the memory of correct classifications. All training algorithms based on this rule start from two fundamental elements: the criterion used in defining the neighbors for the test vector and the training rule applied to the training set for the local neighbors of the test vector (for example: the k-nearest neighbor rule, etc.).

In 1985, Rumellhart and Zipser propose o new training rule, competitive learning. As suggested by its name, when applying this rule, the neurons "compete" between each other in order to become active. Only one neuron it's activated at time. Thus, each neuron of the network specializes in the recognition of similar objects. This learning concept is used in the unsupervised training algorithms, also known as cluster algorithms.

For the network's training algorithm, we have two important classes of algorithms: training "with teacher" and "without a teacher". In the first category we have the supervised learning, while in the second we have: unsupervised learning and reinforcement learning. These three wide categories of algorithms assume:

- Supervised learning – this type of learning is mainly used in classification problems; the classification groups are previously known and there is a set of correctly classified objects used for training the network (thus the idea of learning with a "teacher");
- Unsupervised learning – in this case the classification classes are not priorly known and there is no set of correct classified cases ("no teacher" approach);
- Reinforcement learning – this is a special type of learning, which involves the use of incentives / rewards as functions for the network's feedback.

4. The perceptron. The error-back propagation algorithm

Supervised learning assumes that the network receives a sufficient number of examples of input-output pairs. The network uses these correctly classified object to adapt its synaptic weights, and therefore the training of the network takes place. After the training is completed, the network must be able to correctly discriminate and classify new objects, for which information about the classification class is not available (Twala 2010).

The single layer perceptron is based on the McCulloch-Pitts neuron, that has the following form:

$$y(k) = \begin{cases} 1, \; if \sum_{j=1}^{n} w_j(k) x_j(k) + b \geq 0 \\ -1, \qquad\qquad otherwise. \end{cases} \tag{5}$$

We remind the notations: the inputs of a neuron k are denoted by: $x_1(k)$, $x_2(k)$, …, $x_m(k)$, the synaptic weights: $w_1(k)$, $w_2(k)$, …, $w_m(k)$, and the bias – b will be included in the synaptic weights with input +1. The output of the neuron k is denoted with $y(k)$.

The purpose of the single layer perceptron is the correct classification of a set of input vectors, in two groups: C_1, C_2. The output of the perceptron represents the classification of the input vector or, otherwise said, the decision rule: if $y(k) = +1$, then $x(k) \in C_1$; if $y(k) = -1$, then $x(k) \in C_2$.

From a geometrical point of view, the two classes represent two regions of the m-dimensional space, separated by a hyperplane, whose equation is:

$$\sum_{i=1}^{m} w_i x_i + b = 0. \tag{6}$$

The perceptron yields good results only if the two discriminatory classes are linearly separable. If they are not linearly separable, that is if there are some points in which they are very close to one another and there is no hyperplane to separate them, then the perceptron doesn't hold the capacity to discriminate between the two classes.

The training of the perceptron is done by adjusting the synaptic weights by an iterative algorithm, based on the error back-propagation rule.

We assume that the input vectors come from two linearly separable classes: C_1, C_2. The training set which contains vectors belonging to the class C_1 is denoted with $T_1 : x_1(1), x_1(2), \ldots \in C_1$, while the set of test vectors belonging to C_2 is denoted by: $T_2 : x_2(1), x_2(2), \ldots \in C_2$.

The problem we have to solve is finding the weight vector, w so that the hyperplane determined by the equation $w^T x = 0$ will perfectly separate the two classes, C_1, C_2 (Cover's theorem, 1965) Given the training test sets, T_1, T_2, we have to find the weight vector, w, so that:

$$w^T x > 0, \forall x \in C_1, \tag{7}$$

$$w^T x \leq 0, \forall x \in C_2. \tag{8}$$

The existence of the vector w that would satisfy the above inequalities, is assured by the fact that the two classes are linearly separable.

The algorithm for adjusting w has the following form (the most known introduced by Lipmann 1987) :

1. If at iteration n, the element of the training set, $x(n)$ is correctly classified by the weight vector $w(n)$ calculated at this step, then no correction of the weight vector is made at this iteration:

$$w(n+1) = w(n) \text{ if } w^T x(n) > 0 \text{ and } x(n) \in C_1,$$
$$w^T x(n) \leq 0 \text{ and } x(n) \in C_2. \tag{9}$$

2. Else, if at iteration n, the training vector $x(n)$ is not correctly classified by the weight vector $w(n)$, then this must be modified by the following formulas:

$$w(n+1) = w(n) - \eta(n)x(n), \text{ if } w^T x(n) > 0 \text{ and } x(n) \in C_2, \tag{10}$$

$$w(n+1) = w(n) + \eta(n)x(n), \text{ if } w^T x(n) \leq 0 \text{ and } x(n) \in C_1, \tag{11}$$

where $\eta(n)$ represents the training rate, parameter that allows the adjustment of the weight vector.

The multi-layered feedforward networks we have afore mentioned are also called multi-layered perceptrons, because they represent a generalization of the single layer perceptron. Their training is made by applying the error back-propagation algorithm. This algorithm is based on the error-correction rule and has been introduced the first time in 1985, by Rumelhart and McClelland.

This type of networks have one or more hidden layers of neurons. The signals produced by the neurons can be functional or error signals. The functional signals propagate forward, from one layer of neurons to another, until the output layer. The error signals start from the output layer and are back propagated through the network. These signals determine the "back-ward" step of the network, for which the computation of a gradient is needed; in order to make the forward step, we have to apply the activation function on the different inputs.

The back-propagation algorithm which we will present in what follows is the approach proposed by Haykin (1999).

At iteration *n*, the error signal calculated at the output of neuron *j* is:

$$e_j(n) = d_j(n) - y_j(n), \tag{12}$$

where *j* is an output node or a node situated in a hidden layer.

We define the instant value of the error for neuron $j: 1/2e_j^2(n)$, and the instant value of the total error is:

$$E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n), \tag{13}$$

where *C* is the set of neurons from the output layer of the network, those neurons for which we can calculate directly the signal errors (thus not the "hidden" neurons).

The objective of the learning process is adjusting the weights so that the general average error would be minim. The minimum classification error is equivalent with a minimization of the costs. This weight adjustment is made upon the calculated error for each vector of the network.

The synaptic weight, $w_{ji}(n)$, is corrected with $\Delta w_{ji}(n)$, correction proportional with the derivative $\frac{\partial E(n)}{\partial w_{ji}(n)}$ – this is a sensitivity factor called gradient, a multiplier for the changes in the synaptic weights. In order to have a computational form, we conduct the following calculus:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{14}$$

Each factor in the right part of the above equation can be rewritten:

$$E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \xrightarrow{d(e_j(n))} \frac{\partial E(n)}{\partial e_j(n)} = e_j(n), \tag{15}$$

$$e_j(n) = d_j(n) - y_j(n) \xrightarrow{d(y_j(n))} \frac{\partial e_j(n)}{\partial y_j(n)} = -1, \tag{16}$$

$$y_j(n) = a_j(v_j(n)) \xrightarrow{d(v_j(n))} \frac{\partial y_j(n)}{\partial v_j(n)} = a_j'(v_j(n)), \tag{17}$$

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \xrightarrow{d(w_{ji}n))} \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n). \tag{18}$$

In this way we obtain an equivalent formula for the gradient:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) a_j'(v_j(n)) y_i(n). \tag{19}$$

The adjustment of the weight is made by applying the sensitivity factor on the learning rate, η, the so-called delta rule:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \cdot e_j(n) \cdot a_j'(v_j(n)) \cdot y_i(n) = \eta \delta_j(n) y_i(n), \tag{20}$$

where for the second equality we have replaced the expression of the sensitivity factor obtained above, while for the third equality we have defined:

$$\delta_j(n) = e_j(n) \cdot a_j'(v_j(n)). \tag{21}$$

We can write $\delta_j(n)$ by applying in a back-ward sense the formulas for the derivatives mentioned above:

$$\delta_j(n) = e_j(n) \cdot a_j'(v_j(n)) = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial v_j(n)}. \tag{22}$$

We have two possible scenarios for the neuron *j*: either it's a neuron from the output layer, either it's a neuron form a hidden layer. If the neuron is from the output layer, then we know $d_j(n)$ – the desired output, thus we can easily calculate the associated error and furthermore the gradient in order to apply the delta rule. However, if we have a neuron from a hidden layer, we don't know $d_j(n)$ – and in this case we have the back-propagation error. The problem appears when we have to calculate $\delta_j(n)$, which, starting from the last formula found, we can write as:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \cdot a_j'(v_j(n)). \tag{23}$$

In what follows we will make the following differentiation between the neurons' indexes: the index *j* will be used for the hidden layers neurons, while the index *k* will be used for the output neurons.

The problems lasts in computing $\frac{\partial E(n)}{\partial y_j(n)}$. We start from:

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \xrightarrow{d(y_j(n))} \frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)}. \tag{24}$$

And rewrite:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}. \tag{25}$$

We have the equation of error written for the neuron *k*:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - a(v_k(n)) \Rightarrow \frac{\partial e_k(n)}{\partial v_k(n)} = -a_k'(v_k(n)). \quad (26)$$

Moreover:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \Rightarrow \frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \quad (27)$$

By replacing the two results above in the expression of $\frac{\partial E(n)}{\partial y_j(n)}$, we get:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) \cdot a_k'(v_k(n)) \cdot w_{kj})n) = -\sum_k \delta_k(n) \cdot w_{kj}(n), \quad (28)$$

where in the second equality we used the definition introduced earlier for $\delta_j(n)$, but for the neuron k. By replacing the expression above in the formula of $\delta_j(n)$ we obtain:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \cdot a_j'v_j(n)) = a_j'v_j(n)) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n). \quad (29)$$

What we have written above represents the error back-propagation formula and the core of the back-propagation algorithm for feed-forward networks with multiple hidden layers.

5. Neural networks in credit risk assessment

Khashman (2010) investigates 9 different learning schemes for a feed forward network, trained with the error back-propagation algorithm. The difference between the applied schemes consists in the number of observations that make the training set and the validation set, respectively. Moreover, Khashman (2010) draws the attention on the normalization method chosen for the input data. If the method is chosen without consideration to the data distribution, it can lead to significant error in the result interpretation.

As our case study is based on Small and Medium Enterprises (SMEs) financial data, we will focus the specialty literature review to the application of neural networks in credit risk assessment associated with SMEs.

Although applied intensively in the financial and business analysis, the neural networks have been used for the first in assessing SMEs credit risk in a study by Wu, Wang (2000). They conduct a comparative study between the performance of the neural network and that of classic statistical techniques. The analysis is based on credit data for SMEs.

Wu, Wang (2000) expose a number of arguments for the suitability of neural networks in the assessment of SMEs credit risk. The process of crediting SMEs is particularly resource consuming (time, work), thus involving a supplementary cost. In general, the process of data collection is much more difficult if it involves smaller companies and the data set is incomplete. These issues, as well as the diversity of the applicants and the heterogeneity specific to the segment, are all arguments for which the problem can be approached with the neural networks techniques.

The authors find that a method based on neural calculus offers a good instrument in the credit risk analysis. The accuracy of classification is better than when using classic statistical models. The authors' proposal is using a neural network instrument as a decision support element in the process of credit risk evaluation. By using this instrument, one can discriminate between good debtors, that have a high probability of obtaining a credit and bad debtors, that need special attention from the risk officers. In their article, Wu and Wang used data

collected from a top bank in New York (for the year 1996) and choose as input variables 19 financial ratios and creditworthiness information. They build a network of 19 input nodes, 6 nodes in the hidden layer and one output node. The output node takes the value 1 if the application has been accepted for crediting and 0 if not.

The results show an accuracy of 100% for the classification of accepted credit, but an error rate of 57% for the rejected cases. The presented method is compared to discriminant analysis and with a "k-nearest neighbor" algorithm. The results of the neural network are better than the ones obtained with the other two mentioned techniques.

A more recent study, Angelini *et al.* (2008) also addresses the problem of crediting SMEs with a neural network approach. In this article two types of neural networks are proposed – feed-forward networks but with two hidden layers of neurons. The first network has a classic architecture, while the second, called an "ad-hoc" network has the neurons in the input layer, grouped by three; moreover, each group of three neurons is connected with a neuron in the next layer. The reason for choosing this architecture resides in the caracheristics of the sample data.

The authors use a sample of 76 SMEs, clients of a bank from Italy. Unlike Wu and Wang's paper (2000), where only data for a single financial period were used, Angelini *et al.* (2008) study covers a period of three years: 2001–2003. For every company and for each of the three financial years, we have 11 input variables: 8 financial ratios and 3 variables that include information regarding the credit line utilization. Thus, for the "ad-hoc" network, the input data are grouped on three, according to the three years considered.

Another difference from the study aforementioned is the interpretation of the output variable. The two groups of companies are: "non-default" – entities that fulfill their obligations towards the bank and "defaulted" cases, that don't fulfill their obligations towards the bank at the end of the observation period. If the output variable has a value above the threshold 0.5, then the company is considered "defaulted". We note that this classification for the entities is much more suggestive and interesting from the bank's point of view, as it offers previsions regarding the possible problematic cases.

The results are very good – a classification error of approximately 10%, conservative errors – corresponding to inputs wrongly classified in the "defaulted" group, while in reality they are part of the "non-defaulted" class. Still, the network correctly classifies all defaulted objects, thus the ones with an increased risk degree. The disadvantage of the study is the lack of a comparative analysis with other classification methods, like logistic regression or MDA. The authors conclude that the success of implementing a neural network resided mainly in the preliminary analysis and pre-processing of the training data.

5.1. Self-organizing maps

A self organizing map is a neural network which is trained by an unsupervised algorithm (the network knows only the input signals, but not the associated output). Self-organizing map were developed by Kohonen in 1989 and their applicability was proven in multiple domains like: pattern recognition, feature extraction, etc. The purpose of the network is the projection of an input space into an output space, so that the objects with similar characteristics would be represented on the map close to one another. Each neuron is trained to recognize a certain input pattern. The neurons close to each other on the map will recognize similar input

patterns, thus their images will be represented close to one another on the created map. The algorithm behind this network is a competitive one, the "winner takes all" type of neuron.

The self-organizing maps have many applications in the economic life. In predicting corporations' insolvency, we note the model based on self-organizing maps, developed by Serrano-Cinca (1996), starting from four financial ratios, the same ratios used in the Altman Z-Score model. The results are similar to the ones obtained if applying the multiple discriminant analysis and multi-layer perceptron networks. The author emphasizes that the classification errors are not due to the model, but mostly to the input data, meaning that the bankruptcy takes place (for the misclassified cases) for reasons that are not comprised in the input variables. A distinct advantage of the method is the visual representation, easy to interpret, of the influence factors and the easy in determining the weight of each input variable in the discrimination decision.

5.2. Fuzzy adaptive networks

Although neural networks are strong analytical instruments, it has been noticed (Vellido *et al.* 1999) that their performance grows by associating complementary techniques, resulting in the so-called hybrid models. The complementary techniques can be: fuzzy sets, genetic algorithms.

For example, Williamson (1995) proposes the simulation of a genetic process for an automatic configuration and training of a multi-layered perceptron, network used for credit risk assessment.

The idea of hybrid fuzzy models has appeared by putting together fuzzy sets (Bojadziev, G.; Bojadziev, M. 2003) and neural networks, in what we call adaptive fuzzy networks. Fuzzy adaptive networks were first introduced by Cheng, Lee (1999), starting from the idea of combining a fuzzy inference system and neural networks, in order to develop the nonparametric regression.

From the literature, we find out that this is a relatively new category of neural networks: the first references for this type of models date from the late 90's. However, the first study where the fuzzy adaptive networks are used in modeling credit rating is the one by Jiao, Syau and Lee, in 2007.

The reasons why Jiao *et al.* (2007) choose using fuzzy adaptive networks in modeling companies' rating are influenced especially on the input variables' nature. They sustain that the difficulty in defining precision, deciding the importance of the variables and the variables' definition itself determine the imprecision, the ambiguity and even the contradiction between determinant variables for a company's rating. Moreover the dynamicity of the factors induces rather often changes in a certain time horizon. These issues make difficult transposing the variables in a machine-code and the solution adopted by the financial institutions is the inclusion of the judgmental analysis in the rating process. To overcome these disadvantages, Jiao et al. sustain, in order to use this approach, the necessity of three elements: the ability of the correct representation of the imprecise information, the ability of manipulating or aggregating the information in order to obtain results and pertinent conclusions, and in the third place the ability of improving the representation of the reality once more data become available. According to the authors, fuzzy adaptive networks satisfy these requirements and build the arguments for which the mentioned networks were applied in the problem of establishing the rating.

We note the study of Jiao *et al.* (2007) as the first in the specialty literature that applies the fuzzy adaptive networks in the companies' credit risk assessment. However, we note that the approach is limited by the simulated data, not real-life information, thus the conclusions about the efficiency of the method in real-life situation remains a discussion subject. In spite of this disadvantage, the article remains a reference point for the presented methodology and for the innovative approach of the credit risk.

The fuzzy inference system represents a computation methodology based on the concepts of the fuzzy set theory, if-then rules, fuzzy logic. A fuzzy inference system assumes three conceptual components: a set of rules, the database that defines the membership functions used in the if-then rules, the reasoning based on which the inference on all fuzzy rules takes place to result in the final outcome. We also mention Piramuthu (1999) as a study of neuro-fuzzy systems.

Another type of generalized neural networks are the support vector machines, applied in credit risk modeling problems by Huang *et al.* (2004), Lee (2007), Kim and Sohn (2010). As the subject is very wide, we do not detail the fundamentals of this technique, but the results it yields are similar (and even beetter) with the ones obtained with neural networks.

6. Case study

Our aim is to test the efficiency of the methods presented in the first part of the paper.The common sample is a set of financial data collected from 105 small companies in Romania, with a turnover between EUR 700.000 and EUR 3.755.000 at the end of 2009. 75 of the entities in the sample are non-defaulted companies and 30 are defaulted cases. A firm is considered defaulted if the insolvency procedure was started in 2010. Data are collected from the financial statement at the end of 2009 (public available data). So, the default is observed in the next 12 months from the ending of the input data observation period. The chosen sample represents about 2% of the total population of companies with the turnover in the mentioned interval.

Firstly, we have to set up the input variables – the accounting ratios to be used. Our choice of financial ratios is based on the set of ratios usually used in the literature combined with data availability. The initial set of financial ratios is listed in the Table 1.

Table 1. The initial set of accounting ratios

R1	Debt to Equity
R2	Return on Assests
R3	Return on Equity
R4	Receivables Rotation
R5	Sales on Total Assets
R6	Sales to Equity
R7	Debt to Assets
R8	Stock to Total Assets
R9	Working Capital / Total Assets
R10	Retained Earnings / Total Assets
R11	Earnings before interest and taxes / Total Assets
R12	Equity / Total Liabilities
R13	Current liabilities / Current Assets
R14	Return / Sales

In order to depict the most relevant ratios, we first conduct a factor analysis on the set of ratios. We decide to settle with five clusters. These are depicted in Table 2.

We retain only one ratio from each cluster: R10 from the first cluster, R1 from the second one, R5 from the third, R7 from the fourth and R8, the only ratio in the last cluster. Thus, we have limited the analysis to five ratios: R1, R5, R7, R8, R10.

For the consistency of the analysis, we will conduct a correlation

analysis. We calculate the Spearman correlation coefficients between each two ratios. We cannot reject the null hypothesis for the correlation coefficient in three cases (when the calculated probability associated with the null hypothesis is less than 10% or 0.1): R10 and R5 – with a calculated correlation coefficient of 0.436; R10 and R7 – with a calculated correlation coefficient of 0.453; R1 and R7 – 0.346. We consider for elimination only the ratio R10, as the value of the calculate correlation coefficient between R1 and R7 is less than 0.4. Thus we have a remaining set of four ratios: R1, R5, R7 and R8.

Table 2. Factor analysis on the set of 14 ratios (SAS output table)

Total variation explained = 9,945827 Proportion = 0,7104

5 Clusters		R-squared with			
Cluster	Variable	Own Cluster	Next Closest	1-R**2 Ratio	Variable Label
Cluster 1	R2	0,6659	0,1280	0,3832	R2
	R9	0,6326	0,3184	0,5391	R9
	R10	0,6659	0,1280	0,3832	R10
	R11	0,7639	0,3245	0,3496	R11
	R12	0,2024	0,1239	0,9104	R12
	R14	0,5030	0,4253	0,8648	R14
Cluster 2	R1	0,8060	0,0088	0,1957	R1
	R3	0,6889	0,2321	0,4052	R3
	R6	0,8843	0,2611	0,1566	R6
Cluster 3	R5	1,0000	0,1050	0,0000	R5
	R13	1,0000	0,1050	0,0000	R13
Cluster 4	R4	0,6285	0,0763	0,4022	R4
	R7	0,6285	0,5349	0,7988	R7
Cluster 5	R8	1,0000	0,0012	0,0000	R8

Table 3. Output of the logistic regression (SAS output table)

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-5,0036	1,2142	16,9816	<,0001
R1	1	0,0236	0,0138	2,9255	0,0872
R3	1	-0,0474	0,0231	4,2056	0,0403
R5	1	0,1074	0,0530	4,1004	0,0429
R7	1	5,3504	1,4288	14,0219	0,0002
R8	1	-0,00457	0,00601	0,5779	0,4471

Association of Predicted Probabilities and Observed Responses

Percent Concordant	87,2	Somers' D	0,747
Percent Discordant	12,5	Gamma	0,749
Percent Tied	0,2	Tau-a	0,308
Pairs	2250	c	0,874

With the remaining ratios we first conduct a logistic regression, whose output is listed in Table 3. We also include ratio R3 in the analysis, as after comparing the Spearman correlation coefficients of all the ratios, we have observed it yields a good performance (Table 4).

We can see that the estimated parameter for R8 is not statistically significant: from the last column of the table, the probability to accept the null hypothesis for this estimate is 44.7%, above the threshold of 10% (we consider the significance level of 90% for the null hypothesis tests).

In the second output table, we observe the accuracy ratio for the entities' classification when applying this logistic regression. The accuracy ratio of the model (Somers D') is 74.7%.

Secondly, we will apply two types of neural networks to the data set analyzed so far. For a start, we use a neural network with a single hidden layer (with three neurons), trained with the error back-propagation algorithm. The output variable is the class which the entity belongs to. As input variables, we will consider: R1, R3, R4, R5, R7, R8, R9. The results of training this kind a network are represented in Table 5.

Table 4. The results of the discriminant analysis

Number of Observations and Percent Classified into Y

From Y	0	1	Total
0	71 94,67	4 5,33	75 100,00
1	15 50,00	15 50,00	30 100,00
Total	86 81,90	19 18,10	105 100,00

In the first box from the results above, we have the estimations for the weights associated with each input. The smallest weights are associated with the variables R4, R8, R9 and the strongest influences are from R7 (indebtness) and R3 (leverage). In the second box, we can see the convergence of the average error. We observe the average error starts from 0.6, in the first iteration and reaches 0.37 after 100 iterations. In the end, in the third box, we have the final value odefaulted cases is of only 50%. Thus, we conclude that the neural network model we have built yields better results than both logistic regression and discriminant analysis.

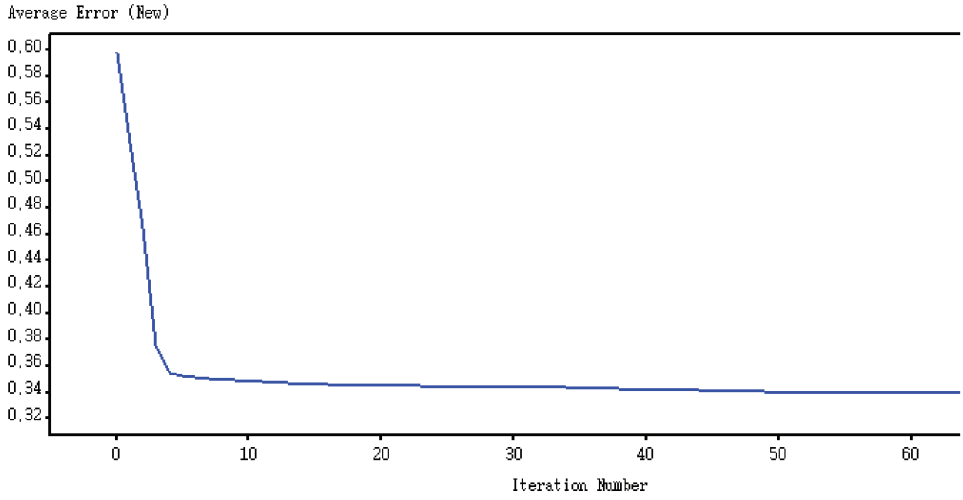
Table 5. The results of training a feed-forward network with the error back-propagation error

The NEURAL Procedure

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	R1_H11	-0,016245	0
2	R3_H11	-0,037792	0
3	R4_H11	-0,092032	0
4	R5_H11	-0,084007	0
5	R7_H11	0,027793	0
6	R8_H11	0,188742	0
7	R9_H11	-0,574955	0
8	BIAS_H11	0,791802	0
9	H11_Y1	0	-0,052312
10	BIAS_Y1	-0,916291	-1,4803E-17

Value of Objective Function = 0,5982695886



Average Profit	0.2857142857
Misclassification Rate	0.1428571429
Average Error	0.3397002754
Average Squared Error	0.1076176728
Sum of Squared Errors	22.599711291

7. Conclusions

In the present paper we have seen how neural networks are used in credit risk assessment problems. In the past 10 years, there has been an increase in the number of studies concerning neural networks and their application in economic and business problems. After introducing the main theoretical concepts of the neural calculus, we have put the fundaments for the main training algorithm: the error back-propagation algorithm.

From reviewing the specialty literature we find that the neural networks yield better results than other classification techniques, like multivariate discriminant analysis or logistic regression, when applying them in credit risk assessment problems. We presented multiple types of networks: feed-forward networks with multiple layers, fuzzy adaptive networks, support vector machines.

The conclusion we find from the specialty literature is verified in practice by the case study we undertake. We have developed an analysis on Romanian Small and Medium Enterprises (financial ratios) and the results show that the neural networks give better results than the logistic regression. The study can be developed by analyzing a support vector machine or a fuzzy adaptive network. Also, we can add a comparison between credit risk modeling techniques for different countries. However, this kind of analysis is difficult to develop due to lack of integrated micro data series.

Acknowledgements

This paper is a result of the project "Doctoral Program and PhD Students in the education research and innovation triangle". This project is co funded by European Social Fund through The Sectorial Operational Program for Human Resources Development 2007–2013, coordinated by The Bucharest Academy of Economic Studies.

References

Angelini, E.; Di Tollo, G.; Roli, A. 2008. A neural network approach for credit risk evaluation, *The Quarterly Review of Economics and Finance* 48: 733–755. doi:10.1016/j.qref.2007.04.001

Bojadziev, G.; Bojadziev, M. 2003. *Fuzzy logic for business, finance and management*. World Scientific Press. 232 p.

Cheng, C. B.; Lee, E. S. 1999. Applying adaptive network to fuzzy regression analysis, *Computers and Mathematics with Applications* 38: 123–140. doi:10.1016/S0898-1221(99)00187-X

Cover, T. M. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Transactions on Electronic Computers EC* 14: 326–334. doi:10.1016/S0898-1221(99)00187-X

Haykin, S. 1999. *Neural Networks: A comprehensive foundation*. Prentice Hall International.

Huang, Z.; Chen, H.; Hsu, C.-J.; Chen, W.-H.; Wu, S. 2004. Credit rating analysis with support vector machines and neural networks: A market comparative study, *Decision Support Systems* 37: 543–558. doi:10.1016/S0167-9236(03)00086-1

Jiao, Y.; Syau, Y.-R.; Lee, E. S. 2007. Modelling credit rating by fuzzy adaptive network, *Mathematical and Computer Modelling* 45: 717–731. doi:10.1016/j.mcm.2005.11.016

Khashman, A. 2010. Neural networks for credit risk evaluation: investigation of different neural models and learning schemes, *Expert Systems with Applications* 37: 6233–6239. doi:10.1016/j.eswa.2010.02.101

Kim, H. S.; Sohn, S. Y. 2010. Support vector machines for default prediction of SMEs based on technology credit, *European Journal of Operational Research* 201: 838–846. doi:10.1016/j.ejor.2009.03.036

Lee, Y.-C. 2007. Application of support vector machines to corporate credit rating prediction, *Expert Systems with Applications* 22: 67–74. doi:10.1016/j.eswa.2006.04.018

Lipmann, R. P. 1987. An introduction to computing with neural nets, *IEEE* 3(4): 4–22.

Michie, D.; Spiegelhalter, D. J.; Taylor, C. C. 1994. *Machine learning, neural and statistical classification*. Prentice Hall.

Piramuthu, S. 1999. Financial credit-risk evaluation with neural and neurofuzzy systems, *European Journal of Operational Research* 112: 310–321. doi:10.1016/S0377-2217(97)00398-6

Refenes, A.-P. N. 1994. Comments on 'Neural networks: Forecasting breakthrough or passing fad' by C. Chatfield, *International Journal of Forecasting* 10: 43–46. doi:10.1016/0169-2070(94)90048-5

Serrano-Cinca, C. 1996. Self organizing neural networks for financial diagnonsis, *Decision Support Systems* 17: 227–238. doi:10.1016/0167-9236(95)00033-X

Twala, B. 2010. Multiple classifier application to credit risk assessment, *Expert Systems with Applications* 37: 3326–3336. doi:10.1016/j.eswa.2009.10.018

Vellido, A.; Lisboa, P. J. G.; Vaughan, J. 1999. Neural networks in business: a survey of appliations (1992–1998), *Expert System with Applications* 17: 51–70. doi:10.1016/S0957-4174(99)00016-0

Williamson, A. G. 1995. Refining a neural network credit application vetting system with a genetic algorithm, *Journal of Microcomputer Applications* 18: 261–277. doi:10.1006/jmca.1995.0018

Wu, C.; Wang, X.-M. 2000. A neural network approach for analyzing small business lending decisions, *Review of Quantitative Finance and Accounting* 15: 259–276. doi:10.1023/A:1008324023422

NEURONINIAI TINKLAI IR JŲ TAIKYMAS KREDITO RIZIKAI VERTINTI RUMUNIJOS RINKOS PAVYZDŽIU

S. Stoenescu Cimpoeru

Santrauka. Šio straipsnio tikslas – parodyti, kaip neuroniniai tinklai yra naudojami kredito rizikos vertinimo problemoms spręsti. Iš pradžių pristatoma pagrindinė teorinė koncepcija, paskui – pagrindinis algoritmas. Literatūros analizė atskleidė, kad sprendžiant kredito rizikos vertinimo problemas neuroniniai tinklai duoda objektyvesnius rezultatus už kitus klasifikacijos metodus, t. y. daugiamatę diskriminantinę analizę arba logistinę regresiją. Dėmesys sutelkiamas į kelių tipų neuroninius tinklus: daugiasluoksnius, prisitaikančius ir vektorinius. Atlikta Rumunijos mažų ir vidutinių įmonių finansinių rodiklių analizė ir gauti rezultatai patvirtino prielaidą, kad neuroniniai tinklai duoda objektyvesnį rezultatą už logistinę regresiją.

Reikšminiai žodžiai: neuroniniai tinklai, kredito rizika, neuroninių tinklų algoritmai.

Smaranda STOENESCU CIMPOERU. PhD Student, Academy of Economic Studies, Faculty of Cybernetics, Statistics and Economic Informatics, Department of Cybernetics Economics, Bucharest. Graduate of Cybernetics Economics, Academy of Economic Studies Bucharest as valedictorian (2008). Research scholarship at Paris Dauphine University, France (2010–2011). Author of 5 scientific articles. Research interests: credit risk modelling techniques, neural networks, assessing companies' bankruptcy probability models.