Credit Risk: A Neural Net Approach

Giacomo di Tollo
Dipartimento di Scienze
Università di Chieti–Pescara
ditollo@sci.unich.it

SOMMARIO/*ABSTRACT*

In this this paper, we apply neural nets to credit risk: We define two architectures and we use them to classify the insolvency. We use real-world data about small businesses. Our results show that Neural Nets can be effectively applied to credit risk classification.

In questo lavoro proponiamo l'applicazione delle reti neurali al rischio di credito: abbiamo definito due architetture e le abbiamo usate per classificare l'insolvenza. Abbiamo usato dati reali relativi a piccole imprese, ed i nostri risultati mostrano che le reti neurali possono essere applicate con successo alla determinazione del rischio di credito.

Introduction

In banks and other financial institutions daily work, credit risk is the likelihood that a debtor will not fulfil his/her obligation. Assessing the insolvency plays an important role since a good estimate (related to a borrower) can help to decide whether granting the requested loans or not. Several methods and techniques can be used to tackle these decisions; the most commonly used approaches are linear scoring models. Nowadays, neural networks are gaining success and importance in this field mainly because of their capability of learning non-linear relationships amongst variables; moreover, they show good performances when data are noisy or incorrect. Neural network and linear-scoring approaches rely on analyzing balance-sheets and other financial ratios to assess the *default* likelihood of firms and classify them in order to determine which businesses will be safe and which will be not repaying.

In this work we apply neural nets to the insolvency classification. We used real-world data, after operating a series of pre–processing operations in order to make them suitable for our purposes. The neural nets parameters were set using an optimization procedure analogous to the gradient descent.

In our experiments, we used a feedforward neural network –in the classical topology– and a feedforward neural network with *ad hoc* connections.

1 Data set

For our experiments, we used data of 76 small businesses from a bank in Italy, across three years (2001-2003). For each business we have 11 indices: 8 of them are financial ratios drawn from the balance sheet of the firms, 3 are calculated analyzing the credit-positions-history with the supplying bank and with the overall Italian banking system[1]. The latter group (Credit-positions-history ratios) represents a novelty in the set of attributes, as no work in the literature uses those ratios (see [1]).

The sample businesses are categorized in two groups: the *in bonis* group (composed of firms repaying the loan obligation at the end of the analysing period) and the *default* group (conversely, firms not repaying at the end of the period).

A crucial phase in the deployment of a neural network-based application is data pre-processing, which is surprisingly too often performed without a systematic methodology. Therefore, in our work we devised a simple yet complete data pre-processing procedure, that we detail in the following. This aspect had proved to be a relevant contribution to the good performance of our approach.

Missing and wrong values Some values were missing or incorrect in our database. The usual way of overcoming this problem is to discard from the data set all the entries

[1]Financial ratios are: Cash flow/Total debt; Turnover/Inventory; Current Liability/Turnover; Equity/Total assets; Financial costs/Total debts; Net working capital/Total assets; Trade accounts receivables/Turnover; Value added/Total assets.
Credit-positions-history ratios are: Utilized credit line/Accorded credit line; Transpassing medium-long term/Accorded credit line medium-long term; Utilized credit line medium-long term/Accorded credit line medium-long term.

of the corresponding firm, with loss of a significant amount of information. In order to preserve as much information as possible, we replaced missing values with 'meaningful' ones: If the value was missing or clearly wrong we decided to substitute the empty space with the arithmetical mean of the indice, being the mean calculated as the mean of the existing values belonging to that indice for all the businesses in the overall period of collecting; if otherwise the value was missing because of a computational error (i.e. the variables we are working on are mathematical ratios, and in our sample some ratios can derive by a division by zero) we decided to replace the missing value with the upper limit of the normalization interval (see below)[2].

Data Normalization Data normalization must be performed in order to feed the net with data ranging in the same interval for each input node. We used the interval $[0, 1]$ for each input node. As the use of standard normalization techniques would have caused loosing lots of useful information, we decided to use the logarithmic formula to normalize data. This formula is the most flexible because it can be defined by the user[3].

$$\overline{x} = \log_u(x + 1) \tag{1}$$

.

Correlation analysis In order to decide which ratios to use as input variables we performed a correlation analysis: with this operation we want to find out the most strongly correlated variables and remove them from our further experiments. The results of this analysis show that there is no strong correlation between pairs of variables, so we use the 11 indices examined so far.

2 Experimental Results

In our experiments we relied on the *supervised learning* paradigm. This is characterized by a *training set* (which is a set of correct examples used to train the network, composed of pairs of inputs and corresponding desired outputs) and a *test set* (used to test and evaluate performances). The training set is composed of 70% of the overall sample, and the remaining 30% is used for the test set (this ratios are widely used, see [1]).

The *feedforward* network architecture is composed of an input layer, one or two *hidden* layers and an output layer (composed of only one neuron); The feed-forward network

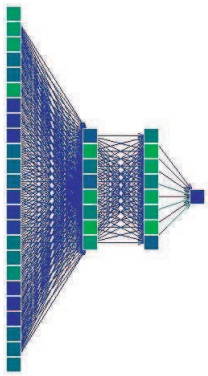

Figure 1: Feedforward network

with *ad hoc* connections (thereinafter referred to as ad *hoc network*) is a four layers feed-forward network with the input neurons grouped by three[4]. Each group is connected to one neuron of the following layer. The reasons for choosing this topology are to be found in the actual data we used (see section 1).

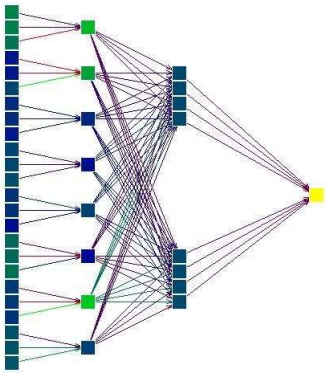

Figure 2: Ad hoc network

The inputs of the networks are the eleven (normalized) attributes. As we dispose of data across three years, networks are supplied with 33 input values (one for each input neuron). In the classical feedforward network, they are simply given as an ordered array, while in the ad hoc network they are first grouped by three, each group corresponding to the values of an attribute over three years. The output y of the network, a real value in the range $[0, 1]$, is interpreted as follows: if $y < 0.5$ then the firm is classified as **in bonis**, otherwise the firm is classified as **default**.

The two network architectures previously introduced have been trained with back-propagation[3] in a long series of experiments, of which we report only the best ones. To tune the neural network parameters (hidden neurons and

[2]This choice can be easily understood: as this occurrence is given by a division by zero, we can imagine the result of this ratio being ∞. If we normalize the corresponding indice, this occurrence will be replaced by the maximum value in the range.

[3]In the formula $\overline{x}$ represents the normalized value, x represents the value before being normalized and u is chosen adding the value 1 to the maximum value belonging to the indice. The formula must be defined taking care the argument being always ≥ 1, so we add 1 to the indice value, assuming its lower bound being 0 without loss of generality.

[4]The *feed-forward* network is widely used in credit risk classification but it is generally used with set of indices referring to one only year (see [4], [5]). The *ad hoc* network was never introduced before and represents a variant of the feed-forward suitable to handle historical data.

learning parameters) we used an optimization procedure analogous to the gradient descent. The best standard network produce null error on the training set and an error of 8.6%, corresponding to only wrong *in bonis* classifications. We remark the fact that this network was able to correctly classifying all the *default* cases, that are considerably riskier than *in bonis* ones. In fact if a *default* firm is wrongly classified as *in bonis*, the bank will grant the loan to an unsafe firm, so loosing the whole amount of money; Conversely, if an *in bonis* firm is wrongly classified as *default*, the bank will not grant the loan to a safe firm, just loosing hypothetical active interests. As reported in table 1, the network performance is very robust with respect to the number of hidden neurons, as errors doesn't change modifying the number of hidden nodes. In the tables, we report the number of neurons in the hidden layer, the wrong *in bonis* (*misbo*, indicating that the net wrongly classifies as in *default* firms actually *in bonis*) and *default* (*misdef* indicating that the net wrongly classifies as *in bonis* firms actually in *default*) classifications and the global error (i.e., the overall error on the training/test set), in the training set and the test set respectively.

All errors are reported in percentage. For completeness, we report also the values of the algorithm parameters: $\eta = 0.2$ (learning rate), $\beta = 0$ (momentum), $\delta = 0.1$(minimum error).

Table 1: Classical feedforward network

# of hidden neurons (1 layer)	tr. set error	test set misbo	test set misdef	test set error
25	0%	13.3%	0%	8.6%
27	0%	13.3%	0%	8.6%
28	0%	13.3%	0%	8.6%
33	0%	13.3%	0%	8.6%

The ad hoc network performs better in terms of over-all errors, but all errors are related to *misdef* cases[5] (see table 2). For this reason, we may consider this network as complementary to the first one: the first returns *safe* answers, while having a – rather very small – error; the second has a very high performance in terms of overall error on the test set, but it wrongly classifies positive cases. The results we got are comparable to those obtained by state-of-the-art applications.

3 Conclusion

In this paper we have presented an application of artificial neural network to credit risk assessment. We defined two architectures and applied them to real-world data. Both architectures were able to classify firms in the *default* and *in bonis* set with a low error ratio. These results are comparable to those obtained by state-of-the-art applications [1]. One of the elements for the good performance of our system has to be found in the data pre-processing procedure, performed with the aim of preserving as much information as possible in the available real-world data.

[5]The parameters used for training the ad hoc network are the following: $\eta = 0.8$, $\beta = 0.2$, $\delta = 0$. We performed also experiments with different numbers of neurons in the second hidden layer.

Table 2: Best results achieved with the ad hoc feedforward network.

# of hidden neurons	tr. set misbo	tr. set misdef	tr. set error	test set misbo	test set misdef	test set error
$11 + 11$ (2 layers)	0%	5%	1.8%	0%	12.5%	4.3%

REFERENCES

[1] A.F. Atiya. Bankruptcy Prediction for Credit Risk Using Neural Networks: A survey and New Results. *IEEE Transactions on Neural Networks, 2001, vol. 12 n.4*

[2] S.L. Pang and Y.M. Wang and Y.H. Bai. Credit scoring model based on neural network. *Proc. of the First International Conference on Machine Learning and Cybernetics*, Beijing, 4-5 November 2002.

[3] D.E. Rumelhart and G.E. Hinton and R.J. Williams. Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* MIT Press, Cambridge, MA, USA.

[4] D. West. Neural network credit scoring. *Computer & Operations Research, 2000, vol. 27*, Pergamon.

[5] C. Wu and X.M. Wang. A Neural Network approach for analyzing small business lending decisions. *Review of Quantitative Finance and Accounting, 2000, vol. 15*, Kluwer Academic Publishers.