Creditworthiness Decision-Making System based on Self-Organising Maps

József Bozsik*, Maksay Dorottya** and Ilonczai Zsolt***
Department of Software Technology and Methodology, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary
e-mail: bozsik@inf.elte.hu*, madraai@inf.elte.hu**, ilzraai@inf.elte.hu***

Abstract—Credit rating evaluates the creditworthiness of a company. The evaluation traditionally relies on algorithms and sophisticated programs requiring the analysis of a large amount of data. The aim of this article is to demonstrate a new method of financial classification, which uses artificial intelligence. This method is based on the analysis of data using a neural network to get accurate and reliable results. In addition to presenting the theoretical algorithm, this article details the phases of evaluation process and solutions to problems that emerged during program's development.

I. INTRODUCTION

Nowadays most companies suffer from the risk-averse behavior of lending institutions and from the current financial crisis. In this situation banks try to reduce their risk exposure by giving loans only to solvent companies, raising the question of how to determine a company’s solvency. Besides analyzing a company’s economic environment (which includes inflation, interest rates, budget, balance of payment, political situation, the industry’s prospects and the competitors), banks assessing solvency must take into account certain indicators (see Table I.), using forecasting methods that are usually based on discriminant analysis or statistical regression. Both of these methods require the analysis of a large amount of data, are difficult to use and require technical support. In contrast, the method presented here is based on artificial intelligence, especially artificial neural networks, which like the human brain are able to learn certain attributes, patterns and rules. This adaptive ability helps in categorizing data. Neural networks also have the ability to generalize, making this approach less sensitive to data loss. Neural networks have been used to classify companies with good results. Solvency has been shown to be a suitable property in this classification process [1], [2]. It is evident that a given company’s indicator set is linearly separable. Linear separability means that a single decision surface is enough to separate classes of patterns [3].

The neural network method is expected to be able, after a complex learning process, to classify the input patterns correctly with a high degree of certainty. In addition to implementing the method, numerous tests were needed in order to calculate the best parameter values for stable and reliable results. The application can be said to work correctly if the network determines the outcome of the so-called learning data with a high degree of accuracy, and if the error rate of new data is relatively low. It is important to avoid over-learning, which can cause database-like behavior. In database-like behavior, the appli-

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick liquidity ratio (0,+),</td>
<td>( \frac{\text{current assets} - \text{inventories}}{\text{current liabilities}} )</td>
</tr>
<tr>
<td>Liquidity ratio (0,+),</td>
<td>( \frac{\text{current assets}}{\text{current liabilities}} )</td>
</tr>
<tr>
<td>Proportion of Cash (%),</td>
<td>( \frac{\text{cash}}{\text{current assets}} )</td>
</tr>
<tr>
<td>Proportion of Cash flow and total debts (-,+),</td>
<td>( \frac{\text{cash flow}}{\text{total debt}} )</td>
</tr>
<tr>
<td>Proportion of current assets (%),</td>
<td>( \frac{\text{balance sheet total}}{\text{current assets}} \times 100 )</td>
</tr>
<tr>
<td>Capital Adequacy ratio (0,+),</td>
<td>( \frac{\text{fixed assets} + \text{inventories}}{\text{own equity}} \times 100 )</td>
</tr>
<tr>
<td>Turnover of assets (0,+),</td>
<td>( \frac{\text{balance sheet total}}{\text{net revenue}} \times 100 )</td>
</tr>
<tr>
<td>Turnover of Inventories (0,+),</td>
<td>( \frac{\text{inventories}}{\text{net revenue}} \times 100 )</td>
</tr>
<tr>
<td>Turnover of Trade Receivables (days) (0,+),</td>
<td>( \frac{\text{customers} \times 360}{\text{net revenue}} )</td>
</tr>
<tr>
<td>Indebtedness (%), (0,+),</td>
<td>( \frac{\text{balance sheet total}}{\text{balances}} \times 100 )</td>
</tr>
<tr>
<td>Proportion of Own Equity (%)</td>
<td>( \frac{\text{balance sheet total}}{\text{own equity}} \times 100 )</td>
</tr>
<tr>
<td>Solvency ratio (0,+),</td>
<td>( \frac{\text{balances} \times 100}{\text{own equity}} \times 100 )</td>
</tr>
<tr>
<td>Fixed assets covered by long term loans (%), (0,+),</td>
<td>( \frac{\text{long-term loans}}{\text{invested assets}} \times 100 )</td>
</tr>
<tr>
<td>Current assets covered by short term loans (%), (0,+),</td>
<td>( \frac{\text{short-term loans}}{\text{current assets}} \times 100 )</td>
</tr>
<tr>
<td>Net Profit Margin (%), (-,+),</td>
<td>( \frac{\text{net income}}{\text{net revenue}} \times 100 )</td>
</tr>
<tr>
<td>Return on Equity (%), (-,+),</td>
<td>( \frac{\text{net income}}{\text{own equity}} \times 100 )</td>
</tr>
<tr>
<td>Trade Receivables covered by Trade Payables (0,+),</td>
<td>( \frac{\text{trade receivables}}{\text{trade payables}} \times 100 )</td>
</tr>
</tbody>
</table>
cation produces a rate of one hundred percent when learning data, but a relatively low classification rate when analyzing new, unseen companies (so-called test data). The classification rate is an indicator of the network’s performance; it is the ratio of correctly classified companies to incorrectly classified companies. Related notions are error rate, first kind and second kind errors, which give information about incorrectly classified data.

After the learning process, statistical results become available which allow users to teach the Kohonen self-organizing map easily and precisely. The traceability of the network during the teaching process is essential in the choice of additional parameters because the effectiveness of the operation can be manipulated in a heuristic way by these values.

Although empirical determination of the optimal parameter values for the program’s operation is not a primary requirement, multiple test runs are required to obtain an optimal rate of classification. The application is subject to additional requirements, including supportive user interface and error-tolerant capability. Because input error can cause incorrect results, data must be verified at import and export. To accelerate the operation, data is accessed from a database rather than from files.

The credit rating process uses 17 client attributes (named financial indicators), which reflect the current financial state. (see Table I).

II. KOHONEN MAPS

The self-organizing map, commonly known as Kohonen network, is a computational method for the analysis and visualization of multi-dimensional data. It uses spacial imaging to model the complex data structures [4], and has the ability to recognize semantic relationships [5] among experimentally acquired information. Its difference from other neural networks lies in treating the data in a geometrical way. Usually the Kohonen network consists of two different layers:

1) input layer
2) output layer - processing neurons’ layer

The input layer is fully connected to the output layer. The elements of the output layer are located on a grid. This is the so-called quadratic topology, but hexagonal topology is also often applied, in which case a processing element has only three immediate neighbors instead of four. (see Fig. 1.) Hexagonal topology is not considered to be better or worse than the quadratic topology, but its implementation is more complicated - thus the preference for quadratic topology.

In the beginning of the process the inner weights of the network can be filled by random initialization or with learning data. If using learning data, the adaptation takes place faster but there is a chance for over-learning. A network is over-learned if the input data produces a hundred percent result, thus operating like a database, yet new data is classified with low accuracy.

A. Representation

A neuron can be represented by an m-dimensional vector which contains the financial indicators of a company. The neuron can be replaced with m simple neurons, each containing a single indicator. In this case a properly defined order has to be assured, because the result will vary depending on the indicator.

It is preferred that the input layer should consist of one single neuron to transmit the data to the processing elements’ layer. In the processing elements’ layer, neurons are placed at every snap-point of the grid. The neurons carry two type of information:

1) weight vector containing the financial index numbers
2) quality indicator showing the solvency of the company represented by the neuron

These two components are manipulated during the training process.

B. Initializing the network

There are two ways to initialize a Kohonen map. One possibility is to fill the neurons’ weights with small random numbers. The limits of the range within which random numbers are generated are set by the user. Another possibility is to fill in the weights from the training data. While in case of the first method the values are uniformly distributed (this is a consequence of the random number generation), the second method assures a lower iteration number of the training algorithm, because the weight vectors of the network will be geometrically closer to the input data.

C. Training the network

Kohonen network doesn’t need supervised training; because of its main characteristic of self-organization, it is able to learn by itself. In the following the abstract and general teaching algorithm is presented, adapted to the task and completed with new solutions.

During the learning process, three different factors must be determined in every step:

1) BMU: Best Matching Unit
2) Environmental coefficient
3) Learning coefficient
D. Calculation of the BMU

The training algorithm utilizes a competitive learning. Every neuron in the network calculates its euclidean distance from the input vector using the equation below (see Equation 1). The best matching unit will be the neuron at the least distance from the input weights. This means that the winning neuron is the most similar to the training example.

$$\text{EuclideanDistance}(v_1, v_2) = \sqrt{\sum_{i=1}^{m} (v_{1i} - v_{2i})^2}$$ \hspace{1cm} (1)

E. Adjusting the weights

Besides the best matching unit, other neurons’ weights are also adjusted towards the input data. The limits of the spread of information are determined by the neighborhood function, and the magnitude of the change depends on the learning coefficient, which is between 0 and 1. Both the neighborhood function and the learning coefficient are monotonically decreasing functions of the iteration number. At the beginning the neighborhood is broad, the learning coefficient is close to 1, and the teaching process takes place on the global scale. As the iteration number increases, self-organizing focuses on local modifications.

F. Training algorithm

1) Initialize the networks’ weights with random values or training data. Consider the sequence of the training data, one by one.
2) \(v_k\) is the \(k-th\) input vector.
3) Search for the closest point in the map form \(v_k\) using euclidean distance.
4) Traverse each node of the map \((W)\) and modify the weights according to the formulas below.

\[W_{i,j}(t + 1) = (W_{i,j} - v_k)\theta(t)\alpha(t),\] \hspace{1cm} (2)

\[\theta(t) = e^{\frac{-d}{2r}},\] \hspace{1cm} (3)

\[r = Re^{-\frac{t}{T}},\] \hspace{1cm} (4)

\[\alpha(t) = Ae^{-\frac{t}{T}},\] \hspace{1cm} (5)

where:
- \(t\) - actual iteration number
- \(T\) - total iteration number
- \(W_{i,j}(t)\) - weight vector at \([i, j]\) in the \(t-th\) iteration
- \(v_k\) - input vector
- \(\theta\) - neighborhood function
- \(\alpha\) - learning coefficient
- \(d\) - euclidean distance between \([i, j]\) place and the best matching unit
- \(r\) - actual neighborhood radius
- \(R\) - radius of the map
- \(A\) - initial value of the learning coefficient
5) Repeat the algorithm from step 2 until every learning data is traversed.

When the training process is finished, it has to be decided which part of the network will represent the solvent companies’ data and which part the insolvent companies. To fulfill these expectations, the map has to be traversed again, assigning a value between 0 and 1 to every node. This will be a flag value for every neuron. 0 will represent insolvent companies, while 1 will represent the solvent companies. The nearer a value is to 1, the likelier that the company represented by the neuron is solvent. By default, every indicator value is set to 0.5. During the process called clustering, every indicator value in the map is determined. These will be used to create a grey-scaled image that helps in visualizing the status of the self-organizing network. The process will be explained in detail later on.

G. Clustering algorithm

1) Initialize the network’s flag values to 0.5. Traverse each learning data.
2) Calculate the neighborhood function as in the training procedure using euclidean distance.
3) Traverse each node of the map \((W)\) and modify the flag values according to the formula below, if \(v_{res}\) is the current input vector’s quality indicator value (0 or 1).

\[W_{i,j}.flag(t + 1) = \frac{W_{i,j}.flag(t) + v_{res}(t)}{2},\] \hspace{1cm} (6)

\[\theta(t) = e^{\frac{-d}{2r}},\] \hspace{1cm} (7)

\[r = Re^{-\frac{t}{T}},\] \hspace{1cm} (8)

The quality indicators calculated with the algorithm above which are the output data of the Kohonen network are suitable for imaging to the set of colors. Thus the development of the self-organizing map can be visualized easily.

H. Learning coefficient and neighborhood function

The learning coefficient and the neighborhood function are both of gaussian type (see Fig. 2). This means that their effect decreases with time, which ensures that algorithm will converge to an optimally adapted map. At the end of the process the modification of the weights is only fine-tuning; the low influence of the training example is ensured by the above mentioned factors.

![Fig. 2. Gaussian function](image-url)
III. Realization

In essence, the algorithm represents a double imaging of the training data. The first round executes the adaptation of the weights to the input data, the second round maps the solvency of the companies to the appropriate point in the network. Regarding the first round, the training process does not need to know the quality of the input data. There the task is to adapt the network’s weights. The data quality is needed only in the second round, where the map is clustered - more precisely, the parts of the map are separated by giving quality indicator to each neuron. This imaging will create the map in its geometrical sense.

Despite the fact that the qualities of the training data are used only in the second round, the learning method is considered supervised. This means that every training example consists of an input vector and the desired output data. (that is, solvency). The statement that a Kohonen network does not need a supervised training process is not a contradiction. The adaptation of the map is self-organizing and unsupervised. The reason for a second round is that the adapted network is sufficient for visualization, but the classification task requires the ability to separate the map into parts.

After the training process, the flag values of the map become available for every neuron. Using these values between 0 and 1, a new input can be classified. (see Equation 9)

\[ \text{Result} = \text{BMU}.flag > \text{tolerance} \]  

(9)

To create an image of the network’s state, the flag values can be mapped to a set of colors. At first the flag value has to be multiplied to obtain a number between 0 and 255 (truncating division is used). This value will be applicable in the components of the color. If every point’s RGB color has the same components, the process leads to a grey-scaled image. Lighter parts will represent high solvency while darker parts will represent less solvent companies.

IV. Settings

Among the wide range of possibilities provided by the credit decision support system are manipulation of the learning and testing data and setting the main parameters of the self-organizing map.

A. Dependence on indicators

Regarding financial index numbers, the experience shows that the accuracy can be of a high level even if not all index numbers are taken into consideration. However, users of the system are advised to select a few index numbers for teaching from the first half of the queue, because the numbers are listed in the order of importance.

B. Parameters

The variables of the network can also be set. These refer to the size, training and visualization. The size of the map can be between 10 × 10 and 100 × 100. More precisely, the number of neurons can vary between 100 and 10,000. The number of training data has to be selected taking into account the size of the map, as well as number of iterations in the training algorithm.

The learning coefficient is a real number between 0 and 1. It determines an initial value which will decrease during the training. It is recalculated in every step using a gaussian function with the iteration number as a parameter.

The tolerance is a real number between 0 and 1, and determines a limit between solvent and insolvent companies. Visually, the tolerance represents a horizontal separator line on the map, showing solvent companies above the line, and insolvent companies below the line. It has a role in analyzing a new company after the learning process. The result will be determined by comparing the flag value of the best matching unit and the tolerance (see Equation 9). For the optimal parameters, see Table II.

V. Results

While analyzing the efficiency of the Credit Decision Support System using different parameter values, it was concluded that the classification rate is satisfactory. An optimal parameter combination can be clearly selected. Using these parameter values, the application is able to operate with a classification rate of over 70%.

While searching for optimal settings, it turned out that creating a network with a high number of neurons, and then running the training algorithm with insufficient data, did not lead to optimal results. In this case, the training process is very slow and parts of the network remain unexplored. The method is more effective when the size of the network is no more than ten times the amount of learning data.

If the network is taught with too much data or with a high iteration number, there is a risk for the network to lose its ability to generalize. This can also happen if the map is too small. If the map had 10,000 neurons, adaptation was too fast.

<table>
<thead>
<tr>
<th>Statistics results</th>
<th>For learning data</th>
<th>For test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification accuracy</td>
<td>97 %</td>
<td>73.5 %</td>
</tr>
<tr>
<td>First kind error</td>
<td>1 %</td>
<td>11 %</td>
</tr>
<tr>
<td>Second kind error</td>
<td>2 %</td>
<td>15.5 %</td>
</tr>
</tbody>
</table>

TABLE II

Optimal settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of neurons</td>
<td>50</td>
</tr>
<tr>
<td>Initializing method</td>
<td>learning data</td>
</tr>
<tr>
<td>Tolerance</td>
<td>0.4</td>
</tr>
<tr>
<td>Learning coefficient</td>
<td>1</td>
</tr>
<tr>
<td>Learning data</td>
<td>100</td>
</tr>
<tr>
<td>Iteration number</td>
<td>50</td>
</tr>
<tr>
<td>Considered financial index numbers</td>
<td>5</td>
</tr>
</tbody>
</table>

TABLE III

Results for optimal settings
Including an option for testing the parameter set more than once was a good decision because it facilitated the usage as well as the testing. The optimal network that is formed during the numerous iterations is available after the end of the process. Then analysis can proceed using this best adapted map. The method can be supplemented with heuristic elements to help achieve a higher classification rate. Consider Table II. These parameter values prove that the system is able to achieve high classification accuracy: even higher than 80%. The complexity of the problem lies in the fact that the number of possible cases is \( m^{10} \), roughly estimated, if there are \( m \) different kind of parameters. Only a small subset of these parameters leads to optimal accuracy.

VI. SUMMARY

The novelty of the credit rating system lies in using a neural network, which is able to learn certain patterns and attributes of the input data, in this case the solvency of a company. The number of the learning data is comparable with the size of the network. For this reason, a large amount of data is not required. Due to its promising results the Kohonen self-organizing maps are considered to be successfully applied in the problem of classifying companies by solvency.

The method’s efficiency can be improved on the basis of the analysis described above. The challenge of the development lies in empirically determining the parameters and the fact that the network is very sensitive to these values’ alteration. A further objective is to compare results with other hybrid methods, for example fuzzy support vector machine [6].

REFERENCES