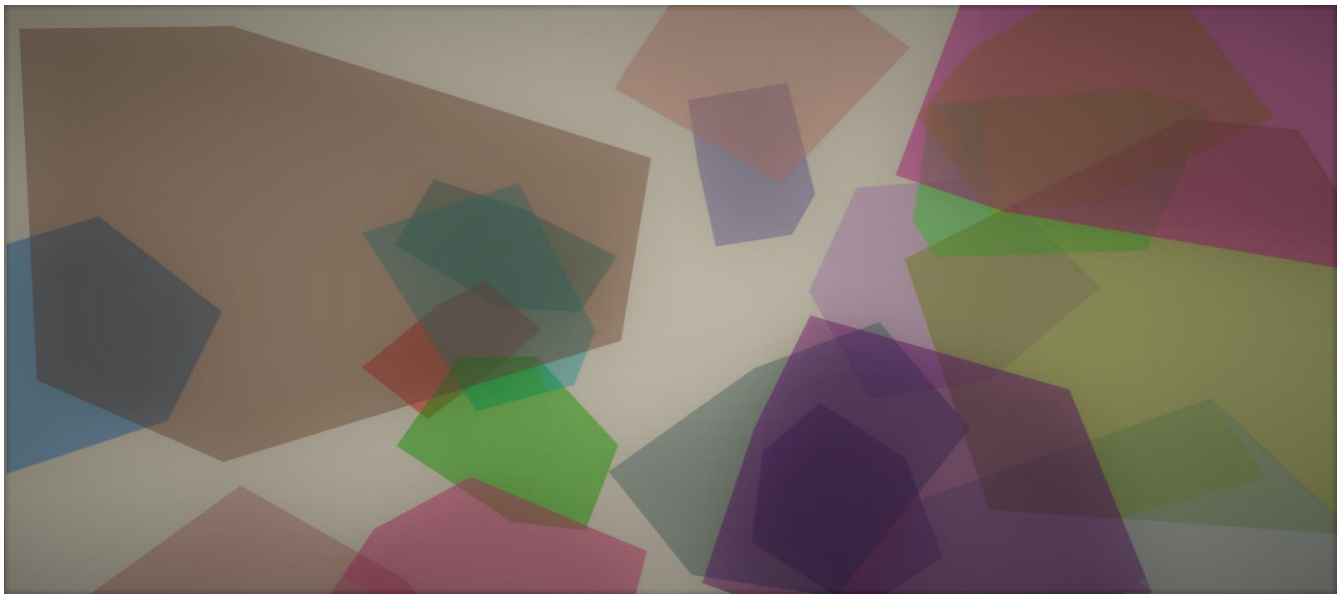


ESTIMATION OF THE PROBABILITY OF DEFAULT FROM INTERNAL DATA

Dr. Ventsislav Nikolov, Nikola Vasilev
October, 2018

Eurorisk Systems Ltd.
31, General Kiselov str.
9002 Varna, Bulgaria
Phone +359 52 612 367
Fax +359 52 612 371
info@eurorisksystems.com
www.eurorisksystems.com

Contents

Abstract: *In recent years, financial institutions have been in great need for an instrument for risk management. Several committees on banking supervision require for institutions to have reliable rating scales for probability of default (PD), which is the most important step is the transition towards an Internal Rating-Based (IRB) approach. This paper introduces a method for estimating the implied probability of default and classifying it into desired credit scales. The calculation of PD is based on Newton's method and the classification is performed via a competitively trained neural network.*

Keywords: *Credit rating, Internal rating-based approach, Newton's method, Classification, Self-organizing map.*

1. Introduction

Probability of default (PD) is a financial term that describes the possibility of bankruptcy within a given time horizon. In most cases, that horizon is set to one year. PD is widely used in a variety of credit analysis and risk management scenarios [1]. Results provided by PD are loss given default (LGD), expected loss and exposition associated with default. The loss value can be interpreted as a fund, where the borrower does not desire or is not able to pay the debt back in time. The probability of default can be measured by a variety of methods, including the discriminant function [2], logistic model, probability model, panel analysis, Cox model, decision trees and neural networks [4]. This paper introduces an approach that is based on a heuristic algorithm, estimating the implied PD for each contract by using a neural network with a self-organizing map (SOM), which clusters and classifies contracts in desired credit scales.

The main aim of this method is to comply with regulatory requirements specified in Regulation (EU) No 575/2013 of the European Parliament and Council, from 26 June 2013 [3]. In essence, the proposed module provides an automatic generation of credit scales and a calculation of PDs for credit contracts, based on real cooperative and government data. The module is designed to be easily transported into a variety of different software products.

The need for new methods of credit rating calculation emerges for the following reasons:

- Actions taken by credit rating agencies are oftentimes too slow, even though the market is dynamic;
- The evaluation process consumes a relatively long period of time. Daily obtained results can provide significant benefits;
- Institutions can create their own rating scales to classify borrowers. In this way, the credit rating would be more precise and more helpfull in the decision making process.

2. Architecture of the module

For the sake of an easier construction and future support, the execution of the system must be independent from the input data and separated from the module. Input data for algorithms are: numerical data for expected loss, credit contract details and rating scale. The conceptual design of the architecture is shown in Fig. 1. There are three basic modules:

- Portfolio evaluator: the output from this stage is represented in the form of a pair of data – contract ID and implied PD;
- Clustering and Optimization: uses data from the previous stage; it is based on artificial intelligence and contracts are classified according to desirable rating scales;
- Aggregation of result and generation of a rating scale, that includes two fields: credit rating and probability of default.

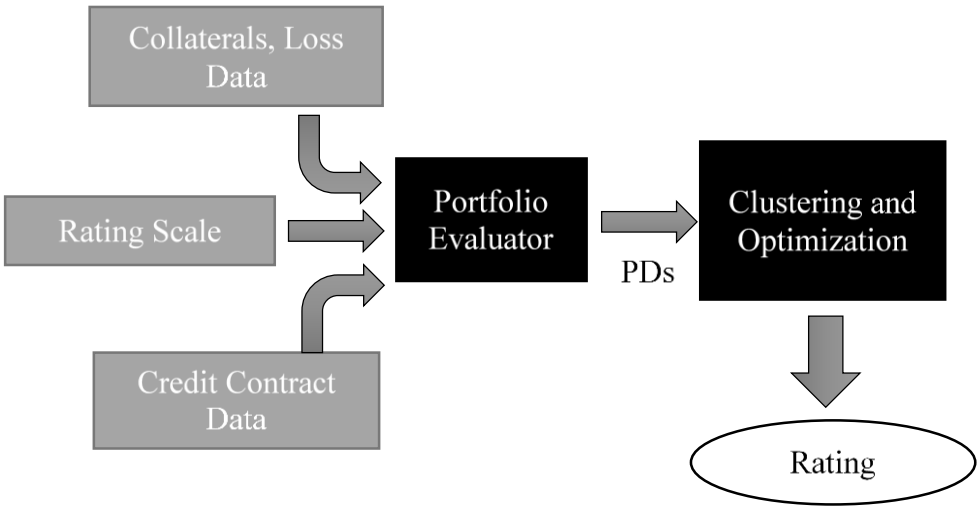


Fig. 1. Architecture of the module

The first stage in mathematical modules is to choose the rating scale. Different institutions work with different scales, containing mainly words with either lowercase or uppercase letters. Some agencies classify their borrowers according to groups, such as long and short term, cooperative and private, etc. In the next stage, a portfolio model is build and a PD for each contract is obtained. Afterwards, the data is analyzed and classified. The goal of the classification is to separate positions into different groups and calculate the average PD.

3. Newton-based optimizer for the PD calculation

The main idea of this algorithm is to perform a "what-if" analysis [7]. The end result of which is to obtain a target amount, followed by constrains and conditions. This approach aims at returning the global minimum of a given function, based on an adaptive search method. To accomplish this, adaptive steps are applied. With each iteration, this step decreases. The optimizer works with a general interface function and is described by a group of formulas. The final output represents the difference between target and estimated value. A pseudo-code is shown in Fig. 2.

The goal of this optimizer is to calculate the implied probability of default for a given position (Fig. 3). The resulting global minimum represents an absolute value and is gained through a difference between provisions and computed expected loss via a formula. The formula for expected loss is shown in (1):

$$EL = \sum_{t=0}^{k} e^{\frac{rT}{360}} LGD_i MPD_i \tag{1}$$

where:

r – rate;

T – time in days;

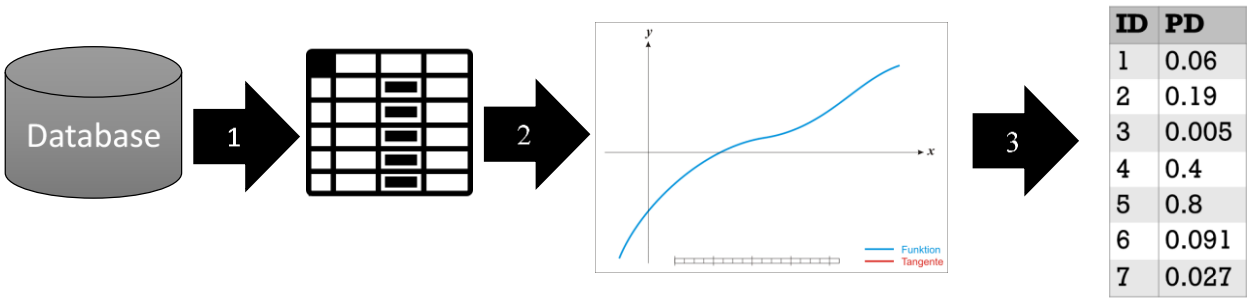
LGD_i – Loss given default for leg I;

MPD_i – Marginal PD, i.e. the difference between the current and the previously interpolated PD;

The pseudo-code of the algorithm is shown below.

```
searchMinimum () {
  for (num_search_iterations) {
    while (true) {
      double error = calculateError(currentValue);
      if (error is decreasing) {
        increase the step;
        currentValue = currentValue + step;
      } else {
            break;
      }
    }
    currentValue = currentValue - step;
    decrease the step;
    continue searching in the descending direction;
  }
  return currentValue;
}
```

Fig. 2. Pseudo-code of Newton-based optimizer



ID	PD
1	0.06
2	0.19
3	0.005
4	0.4
5	0.8
6	0.091
7	0.027

Fig. 3. Workflow of Newton-based optimizer

4. Clustering with self-organizing map

This solution uses a self-organizing map (SOM) [5] to obtain credit rating scales, where input data are presented in the form of contractors' PDs. The algorithm overview is shown in Figure 4. This type of network is organized using connections, where every input node is completely connected to every other node in the network. Each node represents a single credit rating. Input data required for this algorithm are expressed in the form of a pair of data – contract ID and implied PD. The algorithm also involves a rating scale (1), which is the key parameter to building a grid. Each node has specifically mapped positions (coordinates X and Y), as well as weights vectors. In the current solution, the grid looks like vector, because the size of Y is equal to one. The main reason for choosing this approach for clustering is because this network does not require predefined data.

After building the grid, the next step is to fill the classifier with input data (2). The idea of the SOM is to create zones that contain the same or similar credit contracts. Currently, for example, those are contracts with the same PD. At the beginning, the contracts are randomly assorted in the grid. In order for them to be more precise, the network must be trained (3). SOM evolves in epochs and by increasing the number of epochs, the solution will work more accurately. This type of approach has a saturation point. The goal of the training is to obtain grouped positions for each credit rating. After that, the learning rate is used, which is a variable that decreases with time. Another key variable is the radius of neighbourhood (ht), that also decreases with time. The node size can be defined as the range of each credit scale. The classification is focused on finding the smallest distance between the input contract and the distributed scale [6]. In other words, distance is the conditional factor for determining a contract's credit scale. In mathematics, this distance is known as the Euclidean distance. In fact, because of the unavailability of a decay factor, the formula looks as follows:

$$d = \sqrt{\sum_{i=0}^{k} (V_i - R_i)^2} \qquad (2)$$

where:

V – vector containing probabilities for each contract;

R – vector containing random weights;

K – length of vector.

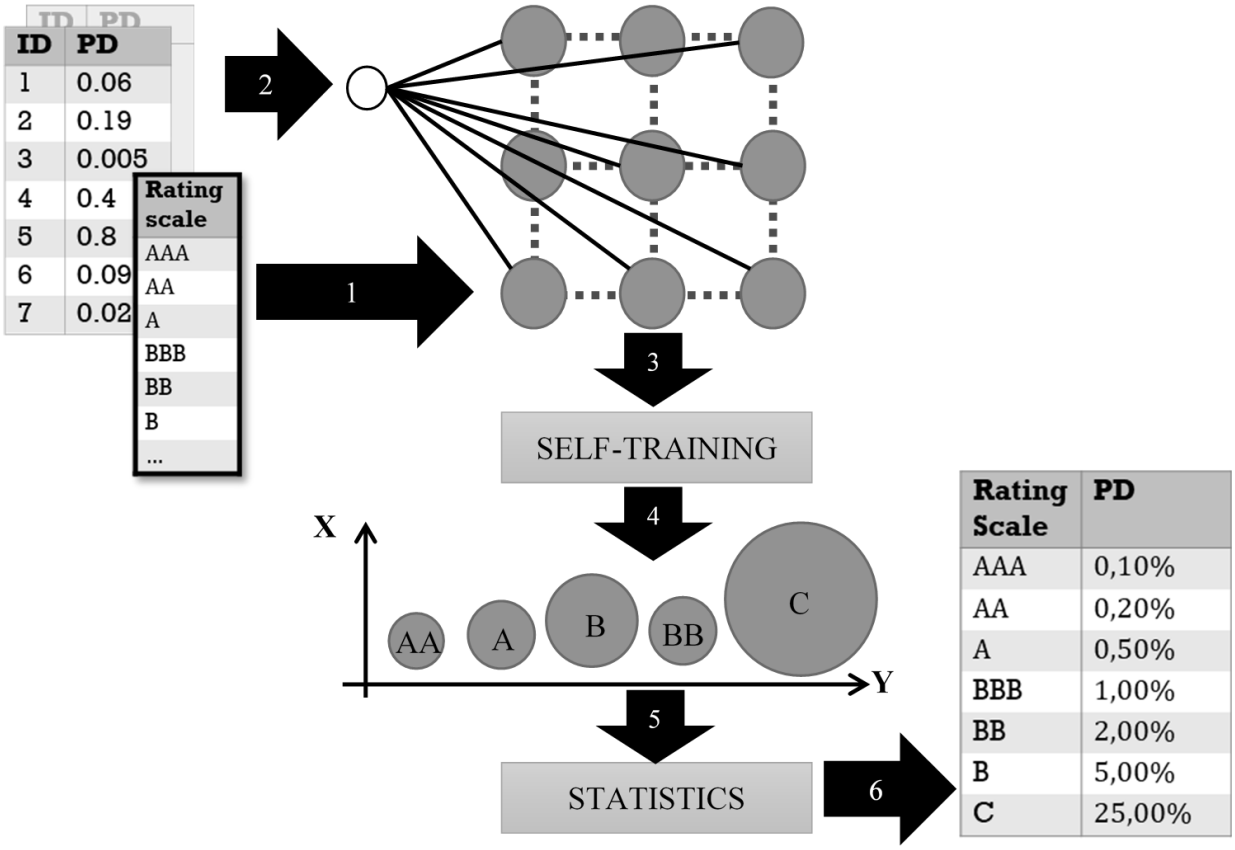


Fig. 4. Workflow of clustering with self-organizing map

The correction to other nodes is accomplished with the amount of influence:

$$\theta_t = e^{-\frac{d^2}{2h_t^2}} \tag{2}$$

After classifying input contracts, the clustering results are obtained, containing contract IDs. As demonstrated in Fig. 4, the size of the radius determines the size of each cluster. The basic idea is for similar PDs to be classified into one group.

5. Tests

5.1 Test of implied PD calculator

Tests are performed by exporting random positions into a spreadsheet software (e.g. Microsoft Excel). In this way, positions are evaluated in tables manually and the Microsoft Excel add-in program SOLVER is used for the calculation of implied PDs. Results generated from the algorithm must be equivalent to the ones in the Excel model.

5.2 Performance test of implied PD calculator

Fig. 5 demonstrates the configuration of a performance test for the calculation of implied PD. It is noticeable that there is a trend of linear time increase. The following test showed neither significant decreases nor increases in time.

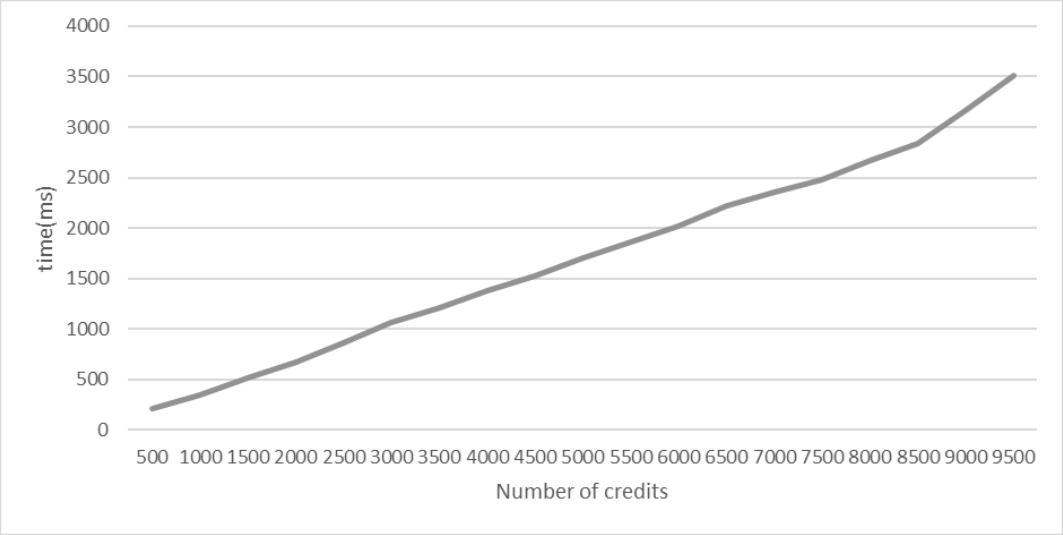


Fig. 5. Performance tests for the computation of implied PD

5.3 Performance test clustering

From the following analysis, it can be concluded that with the increasing number of position and epochs, time increases linearly. This is illustrated in Table.

Time (ms)		Epochs				
		250	**500**	**1000**	**2000**	**5000**
Position number	366	111	127	171	404	875
	693	103	159	318	774	1654
	1107	223	275	521	1061	2604
	2262	382	626	1146	2212	5444
	3417	567	926	1741	3634	8205

Table 1. Performance tests for clustering and building of credit scales

6. Conclusion and future work

The module prototype is developed in Java and is integrated in the third party software product. Targeted rating scales are used for building transition matrices. The generated scale is shown in column D in Fig. 6.

[%]	AAA	AA	A	BBB	BB	B	CCC	D
AAA	76,483617	6,628530	4,368130	4,189669	4,860872	1,718585	1,692698	0,057899
AA	6,628530	76,914201	3,020833	4,699269	3,129547	2,464819	2,630436	0,512364
A	4,368130	3,020833	70,906982	8,503522	7,146401	3,737581	1,042417	1,274134
BBB	4,189669	4,699269	8,503522	68,690014	6,961144	1,783850	2,624128	2,548405
BB	4,860872	3,129547	7,146401	6,961144	65,890968	5,297405	1,553352	5,160312
B	1,718585	2,464819	3,737581	1,783850	5,297405	63,842628	7,684537	13,470595
CCC	1,692698	2,630436	1,042417	2,624128	1,553352	7,684537	50,945947	31,826485

Fig. 6. Usage of generated credit scales

The computational part is implemented as a JAR library and can be used directly, as a module in the business layer logic of other software systems, or as a service operation. This module complies with the requirements stated in the EU regulations on credit institutions. In the development process, patterns for software design, software frameworks and libraries are used, enabling readability and easy support. Future developments could include expansions of the multi-threaded approach, as well as an optimization of mathematical calculations. Additionally, special schemes for private and cooperative contracts can be constructed.