JEL Classification: C45, C25, D81, G21
Keywords: retail banking, credit risk, logistic regression, learning vector quantization

Does Non-linearity Matter in Retail Credit Risk Modeling?

Vita JAGRIC
Davorin KRACUN
Timotej JAGRIC – *corresponding autho*r (timotej.jagric@uni-mb.si)
all authors: University of Maribor, Slovenia

Abstract
In this research we propose a new method for retail credit risk modeling. In order to capture possible non-linear relationships between credit risk and explanatory variables, we use a learning vector quantization (LVQ) neural network. The model was estimated on a dataset from Slovenian banking sector. The proposed model outperformed the benchmarking (LOGIT) models, which represent the standard approach in banks. The results also demonstrate that the LVQ model is better able to handle the properties of categorical variables.

1. Introduction

The credit approval process for loans to retail clients no longer relies on purely personal judgment decisions made by a loan officer, but incorporates models in order to enhance risk management policy and improve the quality of the bank's relationship with the customer. Financial institutions are looking for more effective ways to attract new creditworthy customers while at the same time control losses and lower the per-unit processing cost. Quantitative data about the borrower's credit quality and data warehouses have allowed the creation of tools for portfolio analysis as well as active asset management.

The current financial crisis has underlined the importance of transparency, objectivity, homogeneity, verifiability, and specificity of risk management policies undertaken by banks. Changes in banks were already encouraged by the Basel II accords with regard to relevant tools, policies, and organizational systems for managing and controlling credit risk. At the root of the larger credit risk management framework are credit risk models. Results based on credit risk models cannot be compensated by other component parts of the risk management system. Therefore, the first challenge in a bank's competitive efforts is to develop models with high predictive and/or classification powers.

For this task, most traditional techniques, such as linear discriminate analysis (LDA) and linear and logistic regression, have gained in popularity over the years thanks to their simplicity and satisfactory results. Despite this, the research challenge in credit risk modeling remains to improve the predictive accuracy of credit risk models. This would allow banks to achieve better final results in the credit-granting process. To capture the linkages of credit risk determinants, researchers have investigated newly developed techniques. The possible success of any suggested technique is determined by its ability to perceive the underlying properties of credit risk data. Consequently, linear techniques may not be the best solutions in the field of credit risk modeling. The possible presence of non-linearity may therefore be better investi-

gated by techniques such as neural networks (NNs). Neural networks have already proven themselves with good predictive and forecasting results in many other fields of financial modeling, such as stock markets (e.g. Barunik, 2008) and exchange rate modeling (e.g. Bildirici, 2010).

NNs have established themselves as a serious alternative to traditional statistical models not only because of their ability to capture non-linearity, but also due to complex data structures. They have turned out to be a successful strategy in the credit risk modeling of retail exposures (Desai et al., 1996; Lee et al., 2002; Malhotra and Malhotra, 2003; Baesens et al., 2003). Despite the better credit risk accuracy of NNs, critics generally emphasize the long training process in obtaining the optimal network topology, the difficulty of identifying the relative importance of potential input variables, and possible interpretive difficulties (Lee et al., 2002).

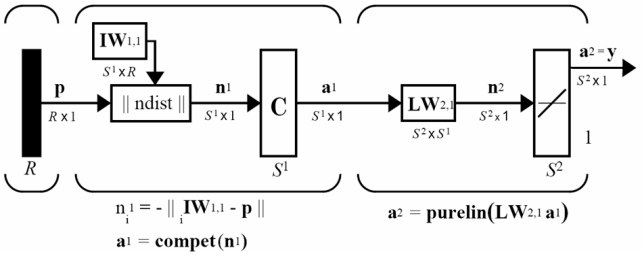
Many different types of neural networks have been applied to develop credit risk models. However, the most widely studied and used neural networks are feed-forward multilayer perceptions (MLPs) trained by back propagation (BP). Vellido et al. (1999) reported a large number of neural network applications in business. In credit risk applications, MLPs were used in Jensen (1992) to predict the payment history of credit applicants. The MLP model was compared with a commercial credit risk model. Desai et al. (1996) compared the multiplayer perceptron and modular neural network with linear discriminant analysis and logistic regression. It was concluded that neural networks are superior when the percentage of correctly classified bad loans is used as the measure of performance. In Lee (2005) the performance of credit risk models was studied using a two-stage hybrid modeling procedure with artificial neural networks and multivariate adaptive regression splines (MARS). The results reveal that the best model was based on the hybrid approach.

In this paper, we extend the current literature by proposing the use of LVQ neural networks in the field of credit risk modeling. In order to critically assess the performance of the proposed LVQ model for business use, we construct two groups of credit risk models for retail loans. The first group is based on standard logistic regression for benchmarking purposes and the second is the proposed alternative – learning vector quantization, which is an NN method. A comparison is made when choosing the best models from the two groups. Logistic regression is often used in credit risk models, but learning vector quantization has not been applied in this area of research.

Our models are developed on a real-life dataset provided by Slovenian banking sector. This makes the results of our study interesting not only for researchers, but also for the banking sector itself. We show that the new model outperforms the standard model based on logistic regression. We believe that this is a clear indication of non-linearity in the credit risk modeling problem. We also believe that higher classification accuracy is achieved in an NN model thanks to more efficient handling of categorical variables, which are typical of credit risk data.

This paper is organized as follows. Section 2 presents the methodological framework. Section 3 discusses the main properties of the data. The results for the credit risk models based on standard logistic regression and learning vector quantization are presented and discussed in section 4. Finally, section 5 presents the conclusions.

Figure 1 The LVQ Network



Notes:R – number of elements in input vector, S^1 – number of competitive neurons, S^2 – number of linear neurons, **p** – input vector, $\mathbf{IW}^{1,1}$ – input weight matrix, $\mathbf{LW}^{2,1}$ – layer two weight matrix, $\mathbf{n}^1$ – net input of a competitive layer, $\mathbf{n}^2$ – net input of linear layer, $\mathbf{a}^1$ – output of the competitive transfer function, $\mathbf{a}^2$ – output of linear transfer function

Source: Hagan et al. (1996).

2. Methodological Framework

Learning vector quantization (LVQ) has been assembled for a class of related algorithms which are widely used in the classification domain, particularly in non-linear separation problems or in potentially high-dimensional data. Successful applications of LVQ include medical imaging and data analysis, fault detection in technical systems, speech modeling and recognition, and the classification of satellite spectral data (Bojer et al., 2003; Kuncheva, 2004; Schleif et al., 2006; Villmann et al., 2003; Mäntysalo et al., 1994). LVQ procedures have several attractive features: they are easy to implement and intuitively clear, which contributes to the popularity of the approach. Biehl et al. (2007) argues that this makes them particularly interesting for researchers who need robust classification schemes without the black-box character of many neural methods.

In contrast to adaptive weights in feedforward neural networks or support vector machines, this procedure allows for immediate interpretation, since the weights are embedded in typical regions, while in feedforward NN and support vector machines (SVM) they are in a different space or at atypical borderline positions of the data. There are further advantages of LVQ, such as the natural way in which it can be applied to multi-class problems. Additionally, one can adjust the complexity of LVQ networks during the training process, according to specific needs.

The LVQ network architecture is shown in *Figure 1*. The LVQ network may be termed a hybrid network, since it uses both unsupervised and supervised learning. It is a two-layer network. The first layer, which is hidden, is also called the competitive layer due to the competitive transfer function that is used. The second one is called the linear layer, since it uses a linear transfer function. The number of neurons in the second layer is determined by the number of classes the network should learn to classify. The number of neurons in the first layer will always be at least as large as the number of neurons in the second layer, since each neuron in the first layer is assigned to a class, with several neurons often assigned to the same class.

In the competitive layer, each neuron's weight vector is a prototype vector for a different cluster, which allows it to classify a region in the input space. Thus, LVQ learning uses the calculation of distance directly, instead of calculating the inner product of the input vector and the weight vector. The advantage of this procedure is

that vectors do not need to be normalized. The net input of the hidden layer $\mathbf{n}^1$ of the LVQ is the Euclidian distance from the input weight matrix $\mathbf{IW}^{1,1}$ to the input vector $\mathbf{p}$:

$$\mathbf{n}^1 = -\left\|\mathbf{IW}^{1,1} - \mathbf{p}\right\| \tag{1}$$

Neurons compete among themselves to determine the winning neuron. The output of the hidden layer $\mathbf{a}^1$ will be 1 for the winning neuron, that is, the neuron for which the distance from its weight vector to the input vector is the smallest, and 0 for all other neurons.

In the second layer, the network combines the subclasses determined in the competitive layer to make up the final classes. This is done with the weight matrix for the second layer, $\mathbf{LW}^{2,1}$. This matrix has a single 1 in each column, indicating which class the appropriate subclass belongs to. It enables the LVQ network to overcome the limitations of standard competitive networks, which can only create decision regions that are convex. Diverse combinations of subclasses being put together in the same class enable LVQ networks to describe very complex, non-linear class boundaries in the input space.

The learning process of the LVQ network follows rules which combine unsupervised and supervised learning. The fundamental learning rule is Kohonen's LVQ1. Since it includes supervised learning, the network's input data must include target values, from which the network learns the proper network behavior. Before learning can take place, each neuron of the first layer is assigned to an output neuron, generating the weight vectors of the second layer. Normally, an equal number of hidden neurons is assigned to each neuron in the second layer. The learning algorithm will be applied, but the weight matrix in the linear layer will be left unchanged throughout. At each iteration of the learning process, an input vector $\mathbf{p}$ will be presented to the network. Hidden neurons compete according to their distance from each prototype vector. The winning neuron is i^*, whose first layer output is set to 1. All others are left at zero. The output vector of the first layer ($\mathbf{a}^1$) is multiplied by the weight matrix of the second layer $\mathbf{LW}^{2,1}$ to get the final output $\mathbf{a}^2$.

The Kohonen rule applies in two ways to the learning of the hidden layer an improvement over standard competitive learning. In the case where $\mathbf{p}$ is classified correctly, the ${}_{i^*}\mathbf{IW}^{1,1}$ weights of the winning hidden neuron are moved toward the input vector $\mathbf{p}$ in the q-th learning iteration, with α being the learning rate (2):

$${}_{i^*}\mathbf{IW}^{1,1}(q) = {}_{i^*}\mathbf{IW}^{1,1}(q-1) + \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{IW}^{1,1}(q-1)\right), \text{ if } a_{k^*}^2 = t_{k^*} = 1 \tag{2}$$

For α, Kohonen's (1997) suggestion is that it should initially be rather small, even smaller than 0.1. When the input vector $\mathbf{p}$ is classified incorrectly, then it is obvious that the wrong hidden neuron won the competition, therefore its weight vector must be shifted away from $\mathbf{p}$ as follows:

$${}_{i^*}\mathbf{IW}^{1,1}(q) = {}_{i^*}\mathbf{IW}^{1,1}(q-1) - \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{IW}^{1,1}(q-1)\right) \quad \text{if } a_{k^*}^2 = 1 \neq t_{k^*} = 0 \tag{3}$$

A hidden neuron will therefore move closer to vectors that fall into the class for which it represents a subclass and away from vectors that fall into subclasses of other classes. Since the LVQ1 learning algorithm suffers from some limitations, an improved version can be found in LVQ2.1 (Kohonen, 1997).

Learning here is similar to that in LVQ1, except now the two vectors of layer 1 that are closest to the input vector can be updated, provided that one belongs to the correct class and one belongs to a wrong class, and further provided that the input falls into a "window" near the midplane of the two vectors. This window is defined by:

$$min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) \rangle s \text{ , where } s = \frac{(1-w)}{(1+w)} \tag{4}$$

(where d_i and d_j are the Euclidean distances of $\mathbf{p}$ from ${}_{i*}\mathbf{IW}^{1,1}$ and ${}_{j*}\mathbf{IW}^{1,1}$. If the input is near the midplane, the two vectors are adjusted, provided that the input vector and ${}_{j*}\mathbf{IW}^{1,1}$ belong to the same class, and $\mathbf{p}$ and ${}_{i*}\mathbf{IW}^{1,1}$ do not belong to the same class. The adjustments made are:

$${}_{i*}\mathbf{IW}^{1,1}(q) = {}_{i*}\mathbf{IW}^{1,1}(q-1) - \alpha\left(\mathbf{p}(q) - {}_{i*}\mathbf{IW}^{1,1}(q-1)\right) \tag{5}$$

$${}_{j*}\mathbf{IW}^{1,1}(q) = {}_{j*}\mathbf{IW}^{1,1}(q-1) + \alpha\left(\mathbf{p}(q) - {}_{j*}\mathbf{IW}^{1,1}(q-1)\right) \tag{6}$$

In LVQ2.1, the relative distances of the codebook vectors from the class borders are optimized, whereas there is no guarantee for the codebook vectors being placed optimally to describe the forms of the class borders. Therefore, the LVQ2.1 should only be used in a differential fashion, using a small value for the learning rate and a relatively low number of training steps.

3. Description of the Dataset

The dataset used for the estimations in this paper come from Slovenian banking sector (see *Table 1*). The dataset contains different characteristics collected by the banks on 1,904 individual borrowers who were granted loans between 2006 and 2007. The outcome period for each month's observations ranges over the next 12 months. In the provided sample, 904 clients defaulted and 1,000 performed well. Each borrower in the sample had only one loan.

In performing the statistical experiments, a separate dataset for training and another separate dataset for testing had to be used. We partitioned our dataset into two subsamples (see *Table 1*): one for development (the development sample) and one for validation purposes (the validation sample). The dataset was randomly split so that the development sample contained 1,000 observations while the validation sample contained the rest of the available data (904 observations). The validation sample would later be used to test the discriminatory power of the model on a sample that was not used in the development stage of the model.

The definition of "default" for the evaluation of the borrower's performance with respect to the loan follows the Basel II standard: the borrower is in default if they are more than 90 days past due on any material credit obligation to the bank. On

Table 1 Properties of the Development and Validation Sample

Sample	Good	Bad	Total
Development sample	500	500	1000
Validation sample	500	404	904
Total	1000	904	1904

Notes: Good – borrowers who were granted loans and have not defaulted over the next 12 months.
Bad – defaulted borrowers.

Source: Data was provided by two banks in Slovenia, which provide loans to retail clientele. The data covers a period from 2006 to 2007.

the other hand, the Basel II rules do not indicate whether the definition of non-defaulted clients should be further enchased for modeling purposes. In our sample, "goods" are not just non-defaulted borrowers, but only those who, during the outcome period, were never more than 30 days overdue with any payment connected to a loan. As is very often done by banks, with this decision we aimed to relieve the model of the impact of "intermediates," i.e., borrowers who may share similarities with both groups of borrowers.

What follows below is an explanation of selected variables which are presented in *Table 2* along with their definitions. These variables characterize the borrower at the moment of application for the loan. They are typically used in credit risk modeling and are also recommended by the Basel II accord. Our dataset gathered socio-demographic variables, including the applicant's age, gender, education, marital status, number of dependents, housing situation, employment, job position, economic sector, and region of residence. The applicant's region is designated by the postal code of the region of the applicant's address. A second group of variables characterizes the applicant's financial data, the relationship between the applicant and the bank, and the features regarding the loan for which they are applying. These variables are: disposable income, other installments, debt-to-income ratio, relationship to the bank, amount of the loan, type of repayment, terms, installments, and interest rates.

Behavioral characteristics are very powerful indicators of the type of applicant (see Kocenda and Vojtek, 2009). However, the applicant needs to have a history with the bank in order to use these indicators. Equally relevant information might also be gathered by external data providers, if such a credit bureau is in place. Since we did not possess other behavioral variables, such as delinquency, they could not be used and therefore our model must be designated an application scorecard only. A new applicant has to be scored almost solely on the basis of their socio-demographic characteristics. A database where Slovenian banks provide data on their clientele and get information on new applicants' credit histories with other banks has been in existence since the beginning of 2008. However, for now it contains no variables like those included in our sample, i.e., socio-demographic ones.

Our data sample does not contain information on rejected applicants, since the banks did not collect this data. Therefore, potential selection bias may have arisen in our estimations. Banasik et al. (2003) compared the effects of selection bias on classification accuracy and found a minimal difference. Similar findings are presented in Hand and Henley (1993). We therefore decided not to make any further modifications to our dataset.

Table 2 Variables inthe Dataset

Variable	Description	Values	G	B	B+G	PD	WoE
X01 – applicants age	18 years to 23,82	0	66	106	172	0.62	-57.47
	From 23,82 to 29,64	1	98	108	206	0.52	-19.81
	From 29,64 to 35,46	2	150	119	269	0.44	13.06
	From 35,46 to 41,28	3	192	191	383	0.50	-9.57
	From 41,28 to 47,10	4	176	164	340	0.48	-3.03
	From 47,10 to 52,92	5	151	102	253	0.40	29.14
	From 52,92 to 58,74	6	83	72	155	0.46	4.12
	More then 58,74	7	84	42	126	0.33	59.22
X02 – applicants gender	Female	0	421	375	796	0.47	1.48
	Male	1	579	529	1108	0.48	-1.06
X03 – education	Elementary or vocational degree	0	363	357	720	0.50	-8.43
	High school degree	1	474	448	922	0.49	-4.45
	College degree, master or PhD	2	163	99	262	0.38	39.77
X04 – marital status	Married	0	545	422	967	0.44	15.49
	Other	1	455	482	937	0.51	-15.86
X05 – number of dependents	None	0	397	365	762	0.48	-1.69
	One	1	227	224	451	0.50	-8.76
	Two or more	2	376	315	691	0.46	7.61
X06 – housing	Living in own apartment	0	297	210	507	0.41	24.57
	Living in own house	1	291	205	496	0.41	24.94
	Living at parents, partner or rented	2	412	489	901	0.54	-27.23
X07 – employment	Employed	0	847	810	1657	0.49	-5.63
	Unemployed (retired, other)	1	153	94	247	0.38	38.62
X08 – job position	Manual, skilled worker or unemployed	0	310	326	636	0.51	-15.13
	Position requiring high school	1	385	387	772	0.50	-10.61
	Professional position	2	121	84	205	0.41	26.40
	Middle and top management	3	184	107	291	0.37	44.12
X09 – economical sector	Public sector	0	253	133	386	0.34	54.21
	Service and merchandising industry	1	183	228	411	0.55	-32.08
	Manufacturing sector	2	136	85	221	0.38	36.91
	Not employed (retired or other)	3	153	95	248	0.38	37.56
	Financial sector	4	275	363	638	0.57	-37.86
X10 – amount of loan	< 2.290 Euro	0	163	180	343	0.52	-20.01
	2.290 <= x < 4.580	1	176	216	392	0.55	-30.57
	5.580 <= x < 8.870	2	169	138	307	0.45	10.17
	8.870 <= x < 12.260	3	130	102	232	0.44	14.16
	12.260 <= x < 16.650	4	100	76	176	0.43	17.35
	16.650 <= x< 21.970	5	89	60	149	0.40	29.34
	21.970 <= x < 27.340	6	107	98	205	0.48	-1.31
	Over 27.340	7	66	34	100	0.34	56.24
X11 – type of repayment	Employer's withdrawal from the salary	0	759	441	1200	0.37	44.20
	Payment via direct debt at the account, postal order, others	1	241	463	704	0.66	-75.39

X12 – relationship to the bank	Account open at the bank	0	454	589	1043	0.56	-36.13
	No account at this bank	1	546	315	861	0.37	44.91
X13 – term	Up to 12 months	0	310	400	710	0.56	-35.58
	More then 12 months	1	690	504	1194	0.42	21.32
X14 – other installments	None	0	645	731	1376	0.53	-22.61
	Yes	1	355	173	528	0.33	61.79
X15 – disposable income	Up to 490 Euro	0	99	233	332	0.70	-95.68
	To 600 Euro	1	217	255	472	0.54	-26.23
	To 710 Euro	2	232	195	427	0.46	7.28
	To 820 Euro	3	162	97	259	0.37	41.20
	Over 820 Euro	4	290	124	414	0.30	74.87
X16 – region of residence	List of post codes A	0	410	472	882	0.54	-24.17
	List of post codes B	1	590	432	1022	0.42	21.08
X17 – interest rate	Fix interest rate	0	422	442	864	0.51	-14.72
	Variable interest rate	1	578	462	1040	0.44	12.31
X18 – installment	Up to 184 Euro	0	432	448	880	0.51	-13.73
	184<= x < 260 Euro	1	265	211	476	0.44	12.69
	over 260 Euro	2	303	245	548	0.45	11.15
X19 – debt-to--income ratio	Up to 18%	0	259	169	428	0.39	32.60
	From 18% up to 42%	1	569	548	1117	0.49	-6.33
	Over 42%	2	172	187	359	0.52	-18.45

Notes: Dependent Variable: Y, Sample: 1904, Obs. (with Dep. = 0) = 1000, Obs. (with Dep. = 1) = 904, G – good, B – bad.

Source: Data provided by two Slovenian banks.

For our analysis, we decided to categorize the numerical variables. Although it is possible to build a model using both numerical and categorical variables, the standard practice in credit risk modeling is to use categorized numerical variables. The number of categories and their width was determined by considering two rules: (i) all categories should have a similar number of observations; (ii) the calculated PD values should be different between classes upon calculated values for the "weights of evidence" (WoE) measure. These two steps were also performed for the categorized variables.

Neural network algorithms are generally not meant to operate directly on raw data. Most pattern recognition tasks are preceded by a pre-processing transformation that extracts invariant features from the raw data. Selecting a proper pre-processing transformation for a particular task usually requires careful consideration, and no general rules can be given. It is cautioned that if the LVQ method is used for benchmarking against other methods, proper pre-processing should always be used. Therefore, we scaled all variables between -1 and 1 by using the following transformation ($x^*_{max} = 1, x^*_{min} = -1$, where x_{max} is the maximum of the selected variable, and x_{min} is the minimum of the selected variable):

$$x^* = \left(x^*_{max} - x^*_{min}\right) * \left(x - x_{min}\right) / \left(x_{max} - x_{min}\right) + x^*_{min} \tag{7}$$

By scaling the data we increase the effectiveness of the neural network learning process. The results show that the pre-processing stage is valuable in building credit risk models of high effectiveness.

In addition to scaling data, it is useful to perform a reduction of variables. Variable selection, or feature selection, is the problem of choosing a small subset of features that is sufficient to describe the target concept. The objective of variable selection in the credit risk model is to obtain a model with low dimensionality. Most intelligent credit risk models developed have used the independent variables provided by banks without modifications (Lahsasna et al., 2010). Variable selection may affect the performance of the model, and using a formal method for choosing the most suitable customer variables for the model may improve the accuracy and reduce the complexity of the model by eliminating irrelevant variables. In Huang et al. (2007), input selection was applied as a technique to build a less complicated credit risk model with relatively few inputs.

4. Comparison of LOGIT and LVQ Models

The benchmarking model was the LOGIT model. The theoretical background has been described in the literature (see, for example, Gardner and Mills, 1989; Lawrence and Arshadi, 1995; Hand and Henley, 1997; or Charitou et al., 2004). Model development through the use of logistic regression started with the simplest model (a regression on a constant only). After each step, an additional variable is added and a decision is made on whether the new variable can be included based on the change in the value of the information criterion and the significance of the estimated coefficient. The coefficients are estimated using the maximum likelihood method. A statistical analysis was performed using the Eviews 6.0 software.

Since many alternative models were tested, we applied different statistical measures to select the best models. As some models do not include a constant term, we were not able to compare the models based on McFadden *R*-squared and the LR statistic. We therefore used two types of measures:

- measures based on the log likelihood value (Schwarz criterion, Hannan-Quinn criterion, and Akaike information criterion);
- measures based on prediction-expectation evaluation for a binary specification (fraction of a true positive, fraction of a true negative, and fraction of all correctly classified observations).

As it turned out, the variables seem to be correlated (see *Table 3* for a correlation analysis). In the presence of multicollinearity, the estimators have large variances and covariances, making precise estimations difficult, which could possibly lead to statistically insignificant coefficients. We therefore developed one LOGIT model with all variables (even if all of them are not statistically significant) and five alternative models that considered the following solutions for multicollinearity:

- dropping the variables that may be the reason for multicollinearity;
- applying principal component analysis in order to produce linearly independent (orthogonal) variables;
- combining forecasts of estimated LOGIT models.

Table 3 Correlation Analysis for Explanatory Variables (Development Sample)

	X01	X02	X03	X04	X05	X06	X07	X08	X09	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19
X01	1.00																		
X02	-0.15	1.00																	
X03	0.11	-0.24	1.00																
X04	-0.33			1.00															
X05		-0.10			1.00														
X06	-0.41	0.11	-0.13	0.26	-0.10	1.00													
X07	0.57	-0.06	0.05	-0.10	-0.25	-0.18	1.00												
X08	0.48	-0.17	0.54	-0.10	-0.14	-0.21	0.72	1.00											
X09		0.12	-0.16		-0.08		0.20	0.09	1.00										
X10			0.11	-0.09	0.10		-0.14		-0.25	1.00									
X11					-0.07	0.05		0.05	0.20	-0.40	1.00								
X12									-0.18	0.27	-0.41	1.00							
X13	-0.05						-0.06	-0.09	-0.26	0.66	-0.57	0.41	1.00						
X14	0.16	-0.06	0.19	-0.13		-0.12	0.10	0.18	-0.10		-0.14	0.15	0.11	1.00					
X15			0.33	-0.12	0.11	-0.09	-0.14	0.10	-0.19	0.51	-0.19	0.17	0.19	0.16	1.00				
X16				0.05					-0.05			0.08				1.00			
X17	-0.08						-0.13	-0.08	-0.14	0.36	-0.22	0.15	0.31	0.11	0.24		1.00		
X18			0.06	-0.05					-0.08	0.37	-0.07	0.06	0.12		0.32		0.08	1.00	
X19			-0.14			0.08	0.05	-0.07	0.05	0.05				-0.14	-0.24		-0.06	0.60	1.00

Notes: Sample: 1904, Obs. (with Dep. = 0) = 1000, Obs. (with Dep. = 1) = 904. Only significant (p<0.05) values for correlation coefficient are showed.
Source: Authors calculations (estimated with Eviwes v.6.0).

In some of the LOGIT models the intercept was dropped in order to avoid the dummy variable trap, since we converted categorical variables into dummy variables (this procedure was applied to all variables and we tested whether such a transformation could produce better results than the application of categorical variables).

The first strategy for solving the multicollinearity problem by dropping variables was followed in the LOGIT1, LOGIT3 and LOGIT4 and meant selecting variables in such a way that in the final model there would be variables that would ideally have little or no linear relationship among them. We added or removed variables according to the coefficient significance in several repeated steps. With the different strategies that were followed, we came up with many very different models. We will present three of them, all of which performed the best among the many temporal model solutions. In the first model the following variables have positive coefficients: housing, job level, economic sector, amount of the loan, and type of repayment. The variables with a negative coefficient are: relationship to the bank, other installments, disposable income, and region of residence. In this model, all but one variable was in the original categorical form, so it was only for the job level that dummies were used. In LOGIT3, another combination of variables is used: applicant age (two dummies only), employment, loan amount (one dummy), type of repayment, interest rate, installments, and debt-to-income ratio. In LOGIT4, all the selected variables were transformed into dummies. The following set of variables was selected: applicant age (three dummies), education, economic sector (two dummies), loan amount (six dummies), type of repayment, relationship to the bank, term, other installments, disposable income (three dummies), and region of residence.

In the LOGIT2 model, we included the first two factors from the principal component analysis in order to achieve linearly independent (orthogonal) variables but not to exclude variables with possible explanatory power. Finally, another alternative solution is presented in the LOGIT5 model. As explanatory variables, we included the forecast from the first and fourth model, and an intercept was included in this model. From the available variables, none of the best performing LOGIT models included the following: customer gender, marital status, and number of dependents.

The results for all the selected LOGIT models are presented in *Table 4*. In the first column, only those variables that were used in the selected models are listed. Some of the variables are presented in two different ways: as a normal categorical variable, or as a series of dummy variables. If a variable is presented as a series of dummy variables, then only the significant dummies are listed. The meaning of the dummies can be found in *Table 2* (second column). All results refer to the development sample.

In order to guarantee an objective comparison between the LOGIT models and the LVQ model, we also estimated a LOGIT model with all available variables (LOGIT6). The results are presented in *Table 5*. Due to the large impact of multicollinearity, many of the variables are statistically insignificant, even if they proved to have valuable information in previous LOGIT models.

The comparison of the classification performance of all six LOGIT models for the development sample is presented in *Table 6*, where we also include results for the LVQ model, which will be explained later on. For the measure of performance, we followed what is commonly used in the existing literature – the share of correctly

Table 4 Estimations of Logit Models (Development Sample)

	LOGIT1		LOGIT2		LOGIT3		LOGIT4		LOGIT5	
Variable	**Coef.**	**Prob.**	**Coef.**	**Prob.**	**Coef.**	**Prob.**	**Coef.**	**Prob.**	**Coef.**	**Prob.**
C									-2.543	0.000
X01 = 0					0.517	0.043	0.602	0.027		
X01 = 5					-0.650	0.002	-0.603	0.009		
X01 = 7							-0.685	0.030		
X03							0.215	0.069		
X06	0.222	0.005								
X07					-0.564	0.005				
X08 = 1	0.342	0.028								
X08 = 2	0.570	0.042								
X09	0.081	0.051								
X09 = 0							-0.628	0.002		
X09 = 2			-0.533	0.016			-0.612	0.007		
X10	0.220	0.000								
X10 = 0			-0.445	0.067			-0.446	0.056		
X10 = 2					-0.508	0.006				
X10 = 3			0.907	0.000			0.718	0.006		
X10 = 4			1.237	0.000			1.001	0.001		
X10 = 5			1.303	0.000			0.973	0.002		
X10 = 6			2.091	0.000			1.600	0.000		
X10 = 7			1.747	0.000			1.117	0.003		
X11	1.350	0.000	1.206	0.000	1.154	0.000	1.306	0.000		
X12	-0.411	0.006					-0.399	0.012		
X13							-0.466	0.043		
X14	-0.688	0.000					-0.648	0.000		
X15	-0.530	0.000	-0.355	0.000						
X15 = 0							0.900	0.000		
X15 = 1							0.727	0.000		
X15 = 4							-0.926	0.000		
X16	-0.518	0.000	-0.455	0.001			-0.487	0.001		
X17					-0.318	0.008				
X18					-0.351	0.001				
X19					0.360	0.003				
PC01			-0.350	0.000						
PC02			-0.217	0.000						
MODEL1_FORECAST									1.502	0.045
MODEL4_FORECAST									3.535	0.000
S.E. of regression	0.442		0.443		0.468		0.439		0.434	
Sum squared resid.	193.340		194.068		216.979		188.954		187.649	
Log likelihood	-572.840		-574.681		-624.800		-562.913		-588.178	
Akaike info criterion	1.166		1.173		1.266		1.166		1.122	
Schwarz criterion	1.215		1.232		1.305		1.264		1.137	
Hannan-Quinn criter.	1.184		1.196		1.281		1.203		1.128	
Convergence achieved after	3 iterations		3 iterations		3 iterations		4 iterations		3 iterations	

Notes: Dependent Variable: Y, Method: ML – Binary Logit (Quadratic hill climbing), Sample: 1000, Covariance matrix computed using second derivatives, Mean dependent var. = 0.5, S.D. dependent var. = 0.50025, Obs. (with Dep. = 0) = 500, Obs. (with Dep. = 1) = 500.

McFadden *R*-squared and LR statistic can only be computed for models which include constant term. Therefore, we do not report these two statistical measures in this table.

Source: Authors' calculations (models estimated with Eviwes v.6.0).

Table 5 Estimations of the Logit Model with All Variables (Development Sample)

Variable	LOGIT6			
	Coefficient	**Std. Error**	***z*-Statistic**	**Prob.**
C	0.090	0.437	0.207	0.836
X01	-0.044	0.051	-0.853	0.393
X02	0.006	0.157	0.039	0.969
X03	0.274	0.188	1.461	0.144
X04	0.042	0.166	0.253	0.800
X05	-0.073	0.092	-0.785	0.432
X06	0.213	0.095	2.245	0.025
X07	-0.219	0.476	-0.460	0.646
X08	-0.050	0.178	-0.281	0.779
X09	0.107	0.051	2.114	0.035
X10	0.290	0.057	5.112	0.000
X11	1.346	0.188	7.157	0.000
X12	-0.320	0.161	-1.986	0.047
X13	-0.309	0.244	-1.269	0.205
X14	-0.582	0.174	-3.350	0.001
X15	-0.514	0.083	-6.214	0.000
X16	-0.459	0.146	-3.145	0.002
X17	-0.027	0.163	-0.166	0.868
X18	-0.071	0.136	-0.523	0.601
X19	0.047	0.172	0.273	0.785
McFadden R-squared	0.177	Mean dependent var.		0.500
S.D. dependent var.	0.500	S.E. of regression		0.443
Akaike info criterion	1.182	Sum squared resid.		192.717
Schwarz criterion	1.280	Log likelihood		-570.798
Hannan-Quinn criter.	1.219	Restr. log likelihood		-693.147
LR statistic	244.699	Avg. log likelihood		-0.571
Prob(LR statistic)	0.000			

Notes: Dependent Variable: Y, Method: ML – Binary Logit (Quadratic hill climbing), Sample: 1000, Convergence achieved after 4 iterations, Covariance matrix computed using second derivatives, Obs. (with Dep. = 0) = 500, Obs. (with Dep. = 1) = 500.

Source: Authors calculations (models estimated with Eviwes v.6.0).

classified loans. We also had to consider the fact that the selected performance measure must be applicable to all models in the same way and must guarantee that the comparison of the models is not biased due to the selection of the performance measure.

If we first compare only the LOGIT models, we can see that LOGIT2 has the best classification performance for non-defaulted borrowers and the best overall classification performance. The LOGIT4 model has the best classification performance for defaulted borrowers. If we were considering measures based on log likelihood values, LOGIT5 would be selected. However, the results for the validation sample (see *Table 7*) show that the best performing LOGIT model is LOGIT1.

Table 6 Performance of Logit and LVQ Models for Development Sample

	GOOD		BAD		TP (in %)	TN (in %)	TP+TN (in %)
	TRUE	FALSE	TRUE	FALSE			
LOGIT1	362	138	355	145	72.40	71.00	71.70
LOGIT2	374	126	352	148	74.80	70.40	72.60
LOGIT3	332	168	334	166	66.40	66.80	66.60
LOGIT4	358	142	365	135	71.60	73.00	72.30
LOGIT5	360	140	360	140	72.00	72.00	72.00
LOGIT6	364	136	350	150	72.80	70.00	71.40
LVQ	393	107	402	98	78.60	80.40	79.50

Notes: Sample: 1000, Obs. (with Dep. = 0) = 500, Obs. (with Dep. = 1) = 500, TP – true positive, TN – true negative, TP+TN – true positive and true negative.

Source: Authors' calculations.

After the development of the LOGIT models, we started with the LVQ model. The LVQ model was implemented in Matlab (The MathWorks, Inc., Massachusetts, USA) running on a conventional personal computer. The Matlab files needed to run the LVQ model use a library of functions available in Matlab.

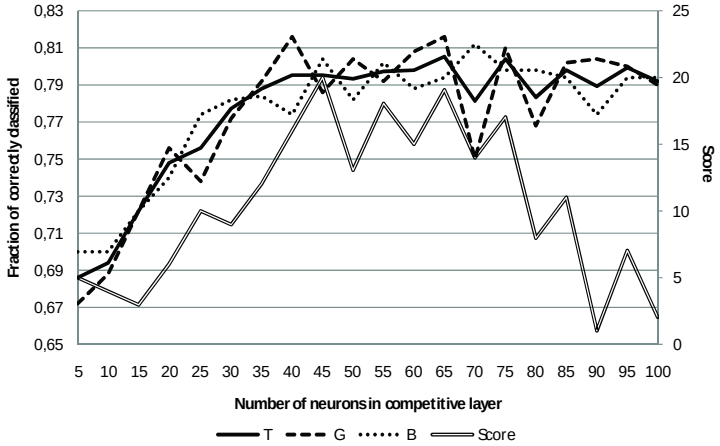
In many practical applications, even when the a *priori* probabilities for the samples in different classes are very different (as is the case in credit risk modeling), a very good strategy is to start with the same number of codebook vectors in each class. We therefore decided to have half of the neurons in the competitive layer reserved for defaulted borrowers in each tested architecture of the LVQ model.

The first-layer weights are initialized to the centers of the input ranges with the function midpoint in Matlab. The second-layer weights have 50% of the columns with a 1 in the first row (corresponding to class 1 – non-defaulted borrowers), while 50% of the columns have a 1 in the second row (corresponding to class 2 – defaulted borrowers).

As previously recommended (Kohonen, 1997), we started learning with the LVQ1 algorithm (the learning rate was set to 0.01), which has very fast convergence. Based on the experiments, we observed that asymptotic recognition accuracy was achieved after 1,000 learning steps. In an attempt to improve the recognition accuracy, we continued with the LVQ2.1, using a low initial value of the learning rate (0.01), which is then the same for all classes. We decided to take a value where $w = 0.25$, and then $s = 0.6$. This means that if the minimum of the two distance ratios is greater than 0.6, the two vectors are adjusted (see eq. 4). The LVQ2.1 algorithm was used in 200 learning steps. We discovered, however, that no major improvement could be achieved by the LVQ2.1 algorithm in any of the tested network architectures.

It often happens that neural network algorithms "over-learn", i.e., when the learning and test phases are alternated, the recognition accuracy is first improved until an optimum is reached. After that, when the learning is continued, the accuracy starts to decrease slowly. A possible explanation in the present case is that when the codebook vectors become very specifically tuned to the training data, the ability of the algorithm to generalize for new data is hindered. It is therefore necessary to stop the learning process after the "optimal" number of steps. Based on our experiments, we stopped learning with the LVQ1 algorithm after 1,000 steps. In this way, we followed the recommendation of Kohonen (1997) that the number of steps should not

Figure 2 Performance of LVQ Models for Different Sizes of the Competitive Layer (Development Sample)



Notes: X-axis indicates the number of neurons in the competitive layer of each tested network. All tested networks had 2 neurons in the output layer since we have two classes of borrowers.

The left y-axis shows the fraction of correctly classified (between 0 for no correctly classified samples and 1 for all correctly classified samples): T – the fraction of correctly classified (all), G – fraction of correctly classified (good), B – the fraction of correctly classified (bad).

The right y-axis shows score points (from 1 for worst LVQ model to 20 for best LVQ model): Score – the weighted result of scores for the size of the model, fraction of those correctly classified, and the difference in classification performance between good and bad.

Source: Authors' calculations (models estimated with Matlab v.7.0).

exceed 200 times the total number of codebook vectors (depending on the particular algorithm and learning rate and architecture). Such a stopping rule can only be found by experience, and it also depends on the input data.

The development of the LVQ model was performed iteratively. We started with a network which has five neurons in the competitive layer. After training, we measured the performance on the development sample and increased the size of the competitive layer by adding five neurons (we chose a step of five in order to reduce the computing time). These steps were repeated until we reached a size of 100 neurons in the competitive layer. We did not test larger networks due to the data size limitation.

In *Figure 2*, the performance of the LVQ models for the development sample is presented. In order to compare the results of the LVQ models with different numbers of neurons and to evaluate which one is the best, we take several criteria into account. These are the size of the model, the fraction of correctly classified applicants, and the difference in classification performance between good and bad. A higher number of neurons can, to some extent, improve the classification accuracy of the network, but this has two important disadvantages. The first is the problem of over-fitting and the second is that a longer training time is needed. From the weighted result, we may assume that the best performing LVQ model is the one with 45 neurons, which we also use for benchmarking purposes with the LOGIT models.

The performance of the best models, LVQ and LOGIT, is compared in *Tables 6* and *7*. In the literature, very different approaches to choosing the measure of

Table 7 Performance of Logit and LVQ Models for Validation Sample

	GOOD		BAD		TP (in %)	TN (in %)	TP+T (in %)
	TRUE	FALSE	TRUE	FALSE			
LOGIT1	335	165	272	132	67.00	67.33	67.15
LOGIT2	333	167	249	155	66.60	61.63	64.38
LOGIT3	288	212	249	155	57.60	61.63	59.40
LOGIT4	324	176	269	135	64.80	66.58	65.60
LOGIT5	328	172	266	138	65.60	65.84	65.71
LOGIT6	337	163	260	144	67.40	64.36	66.04
LVQ	376	124	295	109	75.20	73.02	74.23

Notes: Sample: 904, Obs. (with Dep. = 0) = 500, Obs. (with Dep. = 1) = 404, TP – true positive, TN – true negative, TP+TN – true positive and true negative.

Source: Authors' calculations.

performance are taken (see, e.g., Jensen et al., 1992, Desai et al., 1996, and Baesens et al., 2003). However, we consider the most objective measure to be the fraction of all correctly classified applicants (TP+TN), namely, true positives (TP) and true negatives (TN). On the other hand, another very important measure is the true negative (TN) due to the high cost to the bank of accepting bad borrowers. The performance was measured on the development (*Table 6*) and validation data (*Table 7*).

If we first compare the performance of the LVQ model for the development sample against the performance of the LOGIT models (*Table 6*), we can see that the LVQ model achieved the best results for all three classification performance measures. It is also interesting to note that the LVQ model is slightly better at distinguishing defaulted borrowers (80.40% correctly classified) than non-defaulted borrowers (78.60% correctly classified). This is similar to the results in *Table 7*. It is evident that for all performance measures, the LVQ model outperformed the LOGIT models on the validation data. These results emphasize the advantages of the LVQ method when used for credit evaluation purposes. Even though the training process requires more time for the LVQ model than for LOGIT models, this argument cannot be relevant for a bank when assessing risk management decisions.

As has already been demonstrated (*Table 3*), the variables seem to be highly correlated, which made the process of estimating a well working LOGIT model extremely difficult. This is not, however, the case for the LVQ model, where all variables were included. Also, the categorical form of the variables is not well suited for use in LOGIT models. Finally, possible non-linear relations can only be captured in the LVQ model.

5. Concluding Remarks

This study aims at capturing possible non-linearities in credit risk modeling for retail exposures. It stems from the results of the existing literature, which indicate that standard, linear techniques may be outperformed by non-linear models in credit risk modeling. To explore the proposed hypothesis, a method from the neural network family was applied, namely, learning vector quantization (LVQ). The LOGIT model was chosen as the benchmarking model, since it is the most popular in the banking industry and also very often referred to as a benchmarking model in the existing literature. To verify the feasibility of this proposed approach, the credit

risk modeling task was conducted on a real-life dataset from Slovenian banking sector.

When training an LVQ network, the size of the network must be set in advance. However, there is no clear answer on how to make a decision on this. This indicates that the success of the modeling procedure may finally result in either poorer or stronger model performance depending on the initial conditions determined by the size of the network. We therefore estimated with more than a single model and chose the best among all of them. Similarly, multiple models were developed with the logistic regression as well. The differences between the LOGIT models came out of the different strategies taken to battle the problem of multicollinearity.

Neural networks are often criticized for their long training process. Since the time required for the training process increases with the size of the network, we considered size as a determinant when choosing the best model. Additionally, size must be taken into account in order to avoid the problem of "over-fitting." Finally, we chose the best LVQ model according to the weighted result of scores combining four determinants: the size of the model, the fraction of those correctly classified, and the difference in classification performance between good and bad. The best LVQ model turned out to have 45 neurons in the competitive layer, while the size of the linear layer was determined by the number of classes, which is the same for all models, i.e., two.

The results of our study are based on the validation data and demonstrate that the LVQ model has the highest average correct classification rate in comparison with the benchmark models, which is the main result of this study. These findings are in line with our hypothesis that the LVQ model is better able to capture non-linear relationships among the variables and can handle the properties of categorical variables better than linear techniques such as logistic regression.

In the credit risk modeling problem, many classification techniques yield performances that are quite competitive with each other. A possible explanation for this phenomenon is that it is usually very difficult to achieve good separation between good and bad customers. Credit risk datasets are typically very noisy. Two credit applicants with the same characteristics can easily belong to different classes, one good and the other bad. However, in the credit risk modeling problem, small differences in classification accuracy may already be large enough to have commercial implications representing vast profits. In our study, the improvement in classification accuracy of 8.52 percentage points, when measured by the fraction of all correctly classified applicants, is a vast one. It leads to less costs associated with default risk and less opportunity costs for banks.

REFERENCES

Baesens B, Setiono R, Mues C, Vanthienen J (2003): Using Neural Network Rule Extraction and Decision Tables for Credit Risk Evaluation. *Computer Journal of Management Science*, 49(3): 312–329.

Banasik J, Crook J, Thomas L (2003): Sample Selection Bias in Credit Scoring Models. *Journal of the Operational Research Society*, 54(8):822–832.

Barunik J. (2008): How Does Neural Networks Enhance the Predictability of Central European Stock Returns? *Finance a úvěr - Czech Journal of Economics and Finance*, 58(7-8):359–376.

Biehl M, Ghosh A, Hammer B (2007): Dynamics and Generalization Ability of LVQ Algorithms. *Journal of Machine Learning Research*, 8:323–360.

Bildirici M, Alp EA, Ersin ÖÖ (2010): TAR-cointegration neural network model: An empirical analysis of exchange rates and stock returns. *Expert Systems with Applications*, 37(1):2–11.

Bojer T, Hammer B, Koers C (2003): Monitoring technical systems with prototype based clustering. In: Verleysen M (Ed.): *European Symposium on Artificial Neural Networks, d-side*: 433–439 (Evere, Belgium).

Charitou A, Neophytou E, Charalambous C (2004): Predicting Corporate Failure: Empirical Evidence for the UK. *European Accounting Review*, 13(3):465–497.

Desai V, Crook J, Overstreet G (1996): A Comparison of Neural Networks and Linear Scoring Models in the Credit Union Environment. *European Computer Journal of Operational Research*, 95(6):24–37.

Gardner MJ, Mills DL (1989): Evaluating the Likelihood of Default on Delinquency Loans. *Financial Management*, 18(4):55–63.

Hagan MT, Demuth HB, Beale M (1996): *Neural network design*. PWS publishing company, Boston, MA.

Hand DJ, Henley WE (1993): Can Reject Inference Ever Work? *IMA Journal of Mathematics Applied in Business and Industry*, 5(4):45–55.

Hand DJ, Henley WE (1997): Statistical Classification Methods in Consumer Credit Scoring. *Journal of the Royal Statistical Society A*, 160(3):523–541.

Huang L, Chen C, Wang J (2007): Credit Scoring with a Data Mining Approach Based on Support Vector Machines. *Computer Journal of Expert Systems with Applications*, 33(4):847–856.

Jensen L (1992): Using Neural Networks for Credit Scoring. *Computer Journal of Managerial Finance*, 18(15):26–29.

Kocenda E, Vojtek M (2009): Deafult Predictors and Credit Scoring Models for Retail Banking. *CESIFO Working Paper*, no. 2862:1–53.

Kohonen T (1997): *Self-organizing Maps*. 2nd Edition. Springer Verlag, Berlin.

Kuncheva LI (2004): Classifier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (Eds): Multiple Classifier Systems: 5th International Workshop, MCS2004. Springer, Berlin. *Lecture Notes in Computer Science*, no. 3077:1–15.

Lahsasna A, Ainon RN, Wah TY (2010): Credit Scoring Models Using Soft Computing Methods: A Survey. The *International Arab Journal of Information Technology*, 7(2):115–123.

Lawrence E, Arshadi N (1995): A Multinomial Logit Analysis of Problem Loan Resolution Choices in Banking. *Journal of Money, Credit and Banking*, 27(1):202–216.

Lee T-S, Chen IF (2005): A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications*, 28(4):743–752.

Lee T-S, Chiu CC, Lu CJ, Chen IF (2002): Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications*, 23(3):245–254.

Liu JL, Sun X (2009): Application of Learning Vector Quantization network in fault diagnosis of power transformer. *Mechatronics and Automation*, ICMA 2009:4435–4439.

Malhotra R, Malhotra K (2003): Evaluating Consumer Loans Using Neural Networks. *Omega Computer Journal*, 31(2):83–96.

Mäntysalo J, Torkkola K, Kohonen T (1994): Mapping content dependent acoustic information into context independent form by LVQ. *Speech Communication*, 14(2):119–130.

Schleif FM, Villmann T, Hammer B (2006): Local metric adaptation for soft nearest prototype classification to classify proteomic data. In: Bloch I, Petrosino A, Tettamanzi AGB (Eds.): International Workshop on Fuzzy Logic and Applications, Springer, Berlin. *Lecture Notes in Computer Science*, no. 3849:290–296.

Vellido A, Lisboa G, Vaughan J (1999): Neural Networks in Business: A Survey of Applications. *Proceedings of Expert Systems with Applications*, 17(1):51–70.

Villmann T, Merenyi E, Hammer B (2003): Neural maps in remote sensing image analysis. *Neural Networks*, 16(3-4):389–403.