Istanbul University Journal of the School of Business
İstanbul Üniversitesi İşletme Fakültesi Dergisi

Vol/Cilt: 44, No/Sayı:2, November/Kasım 2015, 3-12
ISSN: 1303-1732 – http://dergipark.ulakbim.gov.tr/iuisletme

Artificial Neural Networks with Gradient Learning Algorithm for Credit Scoring

Derya Soydaner[1], Ozan Kocadağlı[2]✦

[1] *(Department of Statistics, Faculty of Arts and Sciences, Mimar Sinan Fine Arts University, İstanbul, Turkey)*
[2] *(Department of Statistics, Faculty of Arts and Sciences, Mimar Sinan Fine Arts University, İstanbul, Turkey)*

Keywords:

Artificial Neural Networks

Credit Scoring

Information Criteria

Gradient Based Optimization Algorithms

ABSTRACT

Recently, credit scoring problems have come into prominence depending on growing the number of applicants. As known from literature, the traditional techniques are not sufficient to model this kind of problems accurately. For this reason, the researchers are still struggling to develop the novel techniques and improve the current ones to achieve better solutions. In this paper, credit scoring problem is handled by artificial neural networks (ANNs) because they provide flexible modeling procedure and superior performances in the nonlinear environments. However, the researchers mostly overlook some important requirements such as model complexity, overfitting and selection of optimization algorithm during training of ANNs. This paper presents an efficient procedure that allows estimating more robust credit scoring models by means of the information criteria and the early stopping approach based on the cross-validation technique. In the application section, ANNs are trained by various gradient based algorithms over German credit scoring data, and then their classification performances are compared with each other and logistic regression. According to results, the performance of ANNs is better than logistic regression.

1. Introduction

In recent years, the credit scoring is an important research area in financial and banking industry. The applications for loans have been increasing day by day, so the decision makers suffer from evaluating them without the aid of credit scoring models. These models can be built using the historical information from thousands of actual customers by means of statistical and operations research methods. Generally, credit scoring problem mainly splits into two categories. The first category is “application scoring” that classifies the credit applicants into the risk groups, and decides whether any credit application should be worthy to approve it or not. In this context, “application scoring” can be expressed roughly as a classification problem. The second category is “behavioral scoring” that deals with the existing customers of bank and their payment history (Thomas, 2000). Behavioral scoring models help the decision makers to determine the critical strategies about the credit limits of customers, the payment difficulties and bankruptcy, etc. These models classify the existing customers into groups and also predict future purchasing behavior or credit status of customers (Hsieh, 2004).

Depending on the structure of credit scoring problem, various statistical methods can be performed such as discriminant and logistic regression analysis, linear and nonlinear programming, artificial neural networks (ANNs), support vector machines (SVMs) and expert systems, etc. Among these, despite discriminant and

✦ Corresponding author Tel: +90 212 246 0011/5511
E-mail: derya.soydaner@msgsu.edu.tr (D. Soydaner)
ozankocadagli@gmail.com (O. Kocadağlı)

logistic regression analysis are mostly used approaches, they are criticized because of their inherent shortcomings (Abdou et al., 2008; Šušteršič et al., 2009; Wang et al., 2011). For instance, discriminant analysis is not realistic in terms of enforcing some assumptions on real-world data. Although logistic regression might perform well in many applications, its efficiency might disappear in the excessive non-linear environments and high dimensional parameter spaces. Recently, the researchers have applied ANNs, SVMs and expert systems to the credit scoring problems because these techniques do not suffer from compelling assumptions unlike the traditional approaches. In this study, we will restrict ourselves only implementations of ANNs in the context of credit scoring problems. More detailed information about SVMs and expert systems can be found in (Schebesch and Stecking, 2005; Ben-David and Frank, 2009; Setiono et. al., 2012; Niklis et. al., 2014; Bazmara and Donighi, 2014).

Essentially, ANNs are artificial intelligent techniques, and they can be applied easily to various problems such as regression, time-series, classification and pattern recognition problems in the areas of statistics, finance, medical and engineering, etc (Faraway and Chatfield, (1998); Blanco et. al., 2013; Kocadagli and Asikgil, 2014; Niklis et al., 2014). They provide natural way to model the related problems, and give superior performances than the classical approaches (Ong et al., 2005). One of the advantages of ANNs is ability of modeling the non-linear systems in a flexible way without using restrictive assumptions.

Despite of some superior advantages, the researchers suffer from some complicated problems related to ANNs such as controlling model complexity, efficient parameter estimation, training time, gradient information and data separation when they are trained by gradient based optimization algorithms (Kocadagli and Basikgil, 2014). Especially, controlling complexity is closely related to over/lower-fitting to training data. During training ANNs, the researchers mostly tend to minimize the mean squared error by gradient based algorithms. However, this tend causes two types errors, namely approximation and estimation errors which should be reduced simultaneously (Freitas, 2000). Also, some gradient based algorithms require tuning some specific parameters by trial and errors as well as derivative information.

In the excessive parameter cases, Quasi-Newton known as Broyden, Fletcher, Goldfarb, and Shanno (BFGS), Levenberg-Marquardt (L-M) and Scaled Conjugate Gradient (SCG) optimization algorithms provide more superior performances than Gradient Descent (GD) and GD with momentum (GDM). Especially, BFGS utilizes approximations of Hessian matrix; hence it doesn't require an excessive memory allocation. L-M algorithm uses a non-negative damping parameter when its larger values make the algorithm closer to Gauss-Newton Method whereas its smaller values tend it closer the GD. By using an efficient damping parameter in the L-M, it is possible to make fast searching in the excessive parameter cases. SCG doesn't require user dependent parameters and decreases the time consuming arisen from line search differently from its pioneers. In context of implementations to ANNs, more detailed information about gradient based algorithms can be found in (Moller, 1993; Golden, 1996; Bishop, 2010; Kocadagli and Asikgil, 2014).

2. Motivation and Overview

In context of implementation of ANNs to credit scoring problems, there are many studies in the literature in which this issue is handled in the different perspectives. In these studies, the researchers have compared traditional ANNs with the classical statistical methods. For instances, Desai et al. (1996) compared the performances of discriminant and logistic regression analysis with ANNs. Malhotra and Malhotra (2003) compared the multiple discriminant analysis with ANNs. In another study, Abdou et al. (2008) focused to estimate the correct classification rates on the Egyptian banking credit data using the results of ANNs, discriminant, probit, logistic and regression analysis. Blanco et al. (2013) studied credit scoring models based on the multilayer perceptron approach (MLP) and benchmarks their performance against other models which employ the traditional linear discriminant analysis, quadratic discriminant analysis and logistic regression techniques. West (2000) investigated the credit scoring accuracy of five neural network models and compared the results with linear discriminant analysis, logistic regression, k nearest neighbor, kernel density estimation and decision trees. Lee and Chen (2005) proposed a two-stage hybrid scoring model and compared the results with discriminant and logistic regression analysis. From the analysis results of these studies, it can be seen that ANNs provide better performance than classical approaches in terms of correct classification in the credit scoring problems.

In addition to traditional training techniques of ANNs for credit scoring problems, it is possible to hybridize ANNs with evolutionary algorithms and expert systems, so this collaboration can achieve much more successful performances than classical ones for credit scoring problems. Recently, there have been many successful implementations of hybrid ANNs in terms of determining the optimum topology and making accurate classification for the related problems (Arifovic and Gençay, 2001; Blanco et al., 2001; Chalkiadakis et al., 2001; Seiffert, 2001; Lee et al., 2002; Oreski et al., 2012; Hamadani et al., 2013).

Despite of the superiority of hybrid ANNs to traditional approaches, some inherent problems and model

complexity are still overlooked in the most studies. The aim of the study is to propose an efficient approach which allows the decision makers to handle the application scoring problems much more accurately, and overcomes the mentioned problems easily. For this reason, the feed-forward ANNs are trained by various gradient based algorithms; thus their performances are examined against the high dimensional parameter cases. Moreover, to control the model complexity in the training process; Akaike Information Criterion (AIC), Corrected AIC (AIC_c) and Bayesian Information Criterion (BIC) are used as well as early stopping approach based on the cross-validation (Silva et al., 2008).

This paper is constructed as following. Section 3 introduces to the structure of feed-forward ANN and some important elements related to estimating the classification models. Also, the traditional training procedure is introduced in this section. Section 4 includes the credit scoring implementations of the logistic regression and ANNs which are trained by gradient based algorithms. In this section, to examine the performances of various approaches, German credit scoring data is used because this benchmark data allows the researchers to compare their analysis results with the previous studies in the literature. Finally, the analysis results are discussed and interpreted in detail in the Section 5.

3. Constructing the Topology of ANNs

In this study, the network structure consists of three layers as seen in Figure 1. The first layer includes the system inputs called the features of applicants, the second one named as the hidden layer composes of certain number of neurons, and the last one has output vector which consists of the binary digits {0, 1}. These binary digits represent that whether any credit application is worthy or not to approve it. In order to use information criteria, these binary digits should be normalized as follow (Mirkin, 1996):

$$x_{\text{normalized}} = \frac{b_i - p_c}{\sqrt{p(1-p)}} \quad b_i \in [0,1] \qquad (1)$$

where;

p_c: The frequency of any class on data

That is, binary digits are normalized according to their class frequency. In the context of classification problem, the mathematical representation of ANN structure can be given as follow:

$$y = f(w^I, w^{II}, x) = \frac{1}{1+e^{-[b^{II}+w^{II}A(w^I x+b^I)]}} \quad y \in [0,1] \qquad (2)$$

where;

w^I : Matrix that includes all the weight values among inputs and neurons in the hidden layer.

w^{II}: Vector that includes the weight values among the neurons in the hidden layer and output.

x: The input vector

b^I: Vector that includes all the bias values for the tangent hyperbolic activation functions in the hidden layer.

b^{II}: Bias value for logistic activation function in the output layer

The matrix w^I can be expressed as follow:

$$w^I = [w_1 \;\; w_2 \; \ldots \; w_j \; \ldots \; w_n] \qquad (3)$$

where w_j is a vector that is defined as $w_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]$ (j=1,2,…n). That is, w_j includes all the weights between the neuron j^{th} and all the inputs

$A(w^I x + b^I)$: $R^n \rightarrow R^n$ represents a vector function that consists of *n* activation functions as follow:

$$A(w^I x + b^I) = G(A_1(w_1 x + b_1), A_2(w_2 x + b_2), \ldots, A_n(w_n x + b_n)) \qquad (4)$$

where;

n: The number of neurons in the hidden layer

w_j: The weight vector defined between the neuron *j* and all of the inputs

A_j (j=1,2,... ,n) is defined as j^{th} tangent hyperbolic function in the hidden layer:

$$A_j = \frac{e^{NET_j} + e^{-NET_j}}{e^{NET_j} - e^{-NET_j}}; \quad NET_j = w_j x + b_j \;\; j = 1, 2,\ldots,n \qquad (5)$$

In order to make an accurate classification over the related dataset, all the weight and bias values must be estimated efficiently by minimizing an error function as known risk function as well. To do so, different kinds of error functions can be used such as Mean Absolute Deviations (*L1* norm), Mean Squared Errors (MSE) (*L2* norm), Mean Absolute Percentage Errors, Classification Errors and MSE with Penalty function also known as regularization, etc. Much more detailed information can be found in (Golden, 1996; Nocedal and Wright, 2006; Bishop, 2010). In this study, as the error function, MSE is used as well as considering Classification Errors over training, validation and test data in terms of improving the classification performance.

In the context of training ANNs, another important issue is to determine the number of neurons because this issue is directly related to model complexity (Kocadagli and Asikgil, 2014). In this study, to determine the efficient number of neurons providing the best performance of network; Akaike (AIC), Corrected Akaike (AIC_c) and Bayesian (BIC) criteria are used (Faraway and Chatfield, 1998; Bozdogan, 2000; McQuarrie and Tsai, 2007; Kocadagli and Asikgil, 2014). Doing so, determining the

efficient number of neurons is handled much more accurately instead of doing this by trial and errors.

In the literature, mostly AIC and BIC are applied to control the model complexity. Although AIC is asymptotically unbiased and efficacious for large-sample, it leads to overfitting for the small-sample. In this case, it is well known that AIC_c copes with the small-sample overfitting tendencies, so it outperforms than AIC (McQuarrie and Tsai, 2007). Even so, AIC is asymptotically equivalent to AIC_c in large-sample and it achieves almost same performance. Furthermore, BIC penalizes the extra parameters more effectively than AIC, and it provides simple models rather than complex ones. In terms of penalization ability, it can be said that AIC_c is rather closer to BIC (Faraway and Chatfield, 1998).

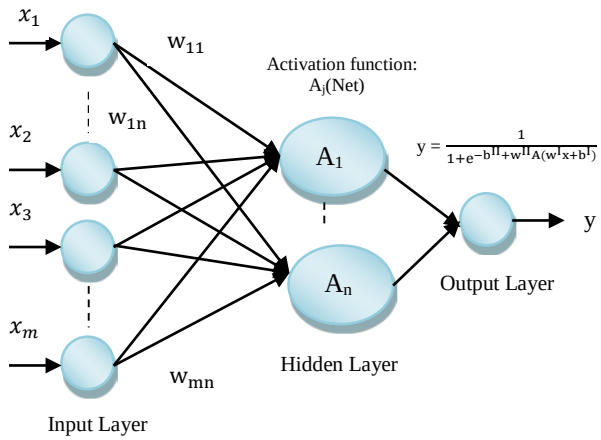

Figure 1: The structure of a feed-forward ANN

3.1. Training Procedure with Gradient Based Algorithms

As known, Back-Propagation (BP) method is a common training procedure that is based on minimizing MSE by GD optimization algorithm (Golden, 1996; Bishop, 2010). In analysis, to examine the performance of other gradient based optimization algorithms on the feed-forward ANNs in the context of classification issue; GDM, SCG, L-M and BFGS are preferred as well as GD. Before the training process, the application data is normalized to get rid of adverse effect of different scales of input variables, and it is partitioned into three subsets as training, validation and test. Although three subsets are mostly preferred in the cross-validation literature, it is desirable to use only training and test datasets as well (Abdou et al., 2008; Šušteršič et al., 2009). While ANNs are trained over training data, validation data is used only to control the overfitting. When training procedure comes up to the certain error limit over the validation data or maximum iteration number, it is terminated automatically. Test data is not introduced to ANNs while training procedure is being processed, because it is used only to compare the classification performances providing over the estimated models. Moreover, the efficient number of neurons is determined by information criteria as well as comparing the estimated models.

After the training process is complicated, the statistical indicators and accuracy ratios are calculated over training, validation and test data. These ratios indicate the percentages of that whether credit applicants are assigned to true class or not. The training framework is given roughly in Figure 2.

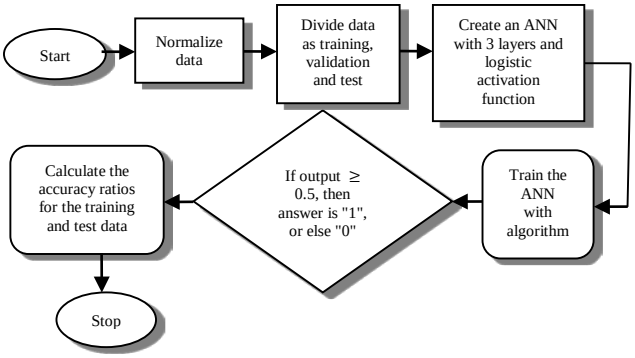

Figure 2: Scheme of Training Procedure

4. Application

In this section, credit evaluation was made by using the real-world German credit dataset. This data includes one dependent and twenty independent variables regarding to 1000 loan applicants. Dependent variable represents whether the credit application is credit-worthy, so it is denoted as binary digits {0, 1}. Independent variable set consists of totally twenty categorical and numerical variables with different scales. For this reason, these variables were normalized before training process. All the variables and their descriptions are given in Table 1. Also, to compare different approaches, and to control overfitting, the credit scoring data was portioned into training, validation and test data. To examine the performances of different approaches in the context of the credit scoring; the logistic regression, ANNs with gradient based algorithms were considered. The structure of feed-forward ANNs were constructed by using three layers, and the neuron numbers in their hidden layers were determined by considering AIC, AIC_c and BIC. While the hidden layers compose of tangent hyperbolic activation functions, output layer includes the logistic activation function. All the analysis related to different approaches is given different sections below, and then comparing of their performances is presented in the section of Results and Discussion. The software of algorithms used in training processes was written in Matlab 7.12.

Table 1: German Credit Dataset

DEPENDENT VARIABLE	Type of measurement
Credit	Categorical
CATEGORICAL VARIABLES	
Status of existing checking account	Categorical
Credit history	Categorical
Purpose	Categorical
Savings account / bonds	Categorical
Present employment since	Categorical
Personal status and sex	Categorical
Other debtors / guarantors	Categorical
Property	Categorical
Other installment plans	Categorical
Housing	Categorical
Job	Categorical
Telephone	Categorical
Foreign worker	Categorical
CONTINUOUS VARIABLE	
Duration in month	Numerical
Credit amount	Numerical
Installment rate in percentage of disposable income	Numerical
Present residence since	Numerical
Age in years	Numerical
Number of people being liable to provide maintenance for	Numerical
Number of existing credits at this bank	Numerical

Source: UCI, Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

4.1. Logistic Regression

In the logistic regression analysis, to estimate much more efficient models, variable selection procedures were developed. By using the Wald technique, which is one of the forward stepwise techniques, the number of independent variables was reduced to 10 from 20. Thus, the logistic regression estimations were made over only ten significant independent variables instead of all of them. In order to examine the performances of the estimated models, the analysis data was partitioned into two subsets as training and test data with respect to the different ratios. All the estimated models and their performance indicator are summarized in Table 2 (This table is given in the Appendix). As mentioned before, accuracy ratios indicate the percentages of that whether credit applicants are assigned to true class or not.

As seen from Table 2, the performances of the estimated models vary depending on the partition ratios. According to the accuracy ratio of test data, it can be said that the first configuration with bold font is better than the others. However, the second bold configuration is better than the others according to the accuracy ratios of both training and all datasets. Also, this configuration is superior to the first bold configuration with respect to information criteria.

4.2. Training ANNs by Gradient Based Algorithms

Before the training the feed-forward ANNs with gradient based optimization algorithms, German credit data was partitioned into training, validation and test data sets with different ratios. During the training process, the early stopping approach was used in order to control the overfitting to training data as well as information criteria. In analysis, the feed-forward ANNs were trained by GD, GDM, SCG, L-M and BFGS, and then their performance were examined in detail. Essentially, these algorithms have different features each other's. For instances, while GD requires the learning rate parameter; GDM also needs an extra parameter known as the momentum constant in addition to the learning rate. L-M uses a non-negative damping parameter when its larger values make the algorithm closer to Gauss-Newton Method whereas its larger values tend it closer the GD. On the contrary, BFGS and SCG are able to work without any specific parameter. Further information about these algorithms and their specific parameter settings can be found in (Moller, 1993; Arifovic and Gençay, 2001; Ong et. al., 2005).

4.2.1.Performance of Gradient Descent

GD algorithm needs a learning rate parameter to work. During training ANNs, to improve the classification performance, different learning rates were treated. As mentioned before, to control overfitting, early stopping approach was used as well as information criteria. In analysis, the efficient number of neurons was determined by means of information criteria. The training results are summarized in Table 3 with respect to different partition ratios (This table is given in the Appendix).

From Table 3, it can be seen that information criteria penalize the complex models because it is possible to make accurate classification using small number of neurons too. Hence, the maximum three neurons in the hidden layer are sufficient to obtain the efficient accuracy rates over training and test datasets. Comparing to the performances of models, the second configuration with bold font gives superior performance over test data. However, the first bold configuration has better accuracy rates over training and all datasets as well as better values of information criteria. Here, information criteria help to determine the efficient neuron number as well as comparing the estimated models.

4.2.2.Performance of Gradient Descent with Momentum

GDM requires a momentum term in addition to the learning rate differently from the classic GD. To examine the effect of combination of learning rate and momentum constant on the classification performance, ANNs were trained using by different combinations of these parameters. Similarly to training with GD, to control overfitting and improve the accuracy performances, the early stopping and information criteria were used. The best configurations are summarized in Table 4 (This table is given in the Appendix).

As seen from Table 4, the best configurations were estimated by means of different combinations of learning rate and momentum parameters with respect to different cross-validation ratios. According to results, the first three configurations with bold font give the best performance over test data while the last bold configuration has better performance over training data. Here, information criteria help to determine the efficient neuron number as well as comparing the estimated models.

4.2.3.Performance of Levenberg-Marquardt Algorithm

In this application, the classical ANNs were trained with the L-M algorithm. This algorithm needs an initial damping parameter μ. To examine the effect of damping parameter, training was made using different values of it as well as running the algorithm at the different iteration numbers. The best configurations are summarized in Table 5 (This table is given in the Appendix). As seen from Table 5, the best configurations were estimated by means of initial damper parameters with respect to different cross-validation ratios. According to results, the first configuration gives the best performance over test data while the last bold configuration has better performance over training data as well as the values of information criteria. Here, information criteria help to determine the efficient neuron number as well as comparing the estimated models.

4.2.4.Performance of Scaled Conjugate Gradient

In this section, training was made by SCG. As known, SCG doesn't require any initial parameter to work. The best configurations obtained from this algorithm are summarized in Table 6 with respect to different cross-validation ratios (This table is given in the Appendix). As seen from Table 6, the first configuration with bold font gives the best performance over test data while the second bold configuration has better performance over training data. Here, information criteria help to determine the efficient neuron number as well as comparing the estimated models. Here, information criteria help to determine the efficient neuron number as well as comparing the estimated models.

4.2.5.Performance of BFGS

In the last application, ANNs were trained with the BFGS algorithm. As known, there is no need to use any initial parameter in BFGS. The best configurations obtained from this algorithm are summarized in Table 7 (This table is given in the Appendix). As seen from Table 7, the first configuration with bold font gives the best performance over test data while the second bold configuration has better performance over training data. Comparing to the models, the first bold configuration is better than the second one according to information criteria.

5. Results and Discussion

From analysis results, it can be seen that training of ANNs by the traditional optimization algorithms gives better solution than logistic regression. However, this training process needs to determine the suitable cross-validation ratios, early stopping criteria and controlling over/lower-fitting in addition to tuning the specific parameters of some gradient based algorithms. To control overfitting problem during training by the traditional algorithms, cross-validation and early stopping criteria were used simultaneously; thus the efficiency of algorithms were improved. Some remarkable advantages and disadvantages of gradient based algorithms are summarized as following.

As seen from Table 3, GD requires to a suitable learning rate to accomplish the efficient search in the high dimensional parameter case. If this rate is too small, it runs for a long time unnecessarily; hence overfitting might be inevitable. In such a case, although MSE calculated over training data decreases, MSE of test data increases inversely. Otherwise, if training process is terminated too early, then the lower-fitting is inevitable too. From results, it can be concluded that the iteration number of GD varies depending on the learning rate parameter.

As seen from Table 4, GDM achieves more superior performance than GD by means of suitable couples of learning rate and momentum constant. However, GDM requires much more training time as well as making trial and errors for the best couple of parameters.

As seen from Table 5, L-M needs lower iterations, so it works very quickly and gives better results than the other gradient based algorithms in the high dimensional parameter cases. In analysis, L-M was treated by a couple of initial damper values, and then this parameter was fixed as 0.01 by means of trial and errors.

As seen from Table 6 and 7; BFGS and SCG do not require any parameter tuning, so this feature provides an important advantage to the researchers in the high dimensional cases. However, to estimate the best models, BFGS and SCG should be worked a couple of times, or they should be initialized by the efficient initial points.

In order to compare the performances of all the methods, their best configurations are summarized in Table 8 (This table is given in the Appendix). According to these results, L-M gives better accuracy ratios with respect to training, test and whole datasets. Besides, as expected, ANNs are superior the logistic regression.

6. Conclusions

According to the analysis results performed over the German credit scoring data, the proposed credit scoring approach provides the substantial advantages in terms of

estimating process and determining the best configuration. First, the early stopping approach terminates the training process automatically when the classification errors over validation data begin to grow. Second, the model selection criteria helps the analysts to make a decision about which model is the best. Moreover, using different gradient based algorithms provides a broad perspective about the performances of these algorithms in the context of credit scoring problems.

For future studies, we are planning to hybridize ANNs with evolutionary algorithms instead of training gradient based algorithms. In this context, the studies related to accelerating and modifying of hybrid approach will be continued, and then a user-friendly interface will be designed by means of Matlab GUI.

References

Abdou, H., Pointon, J., and El-Masry, A. (2008). Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications,* 35, 1275-1292.

Arifovic, J. and Gençay, R. (2001). Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A,* 289, 574-594.

Bazmara, A. and Donighi, S.S. (2014). Bank customer credit scoring by using fuzzy expert system. *I.J. Intelligent Systems and Applications,* 11, 29-35.

Ben-David, A. and Frank, E. (2009). Accuracy of machine learning models versus "hand crafted" expert systems - A credit scoring case study. *Expert Systems with Applications,* 36, 5264-5271.

Bishop, C. (2010). Neural networks for pattern recognition, Oxford University Press.

Blanco, A., Delgado, M. and Pegalajar, M.C. (2001). A real-coded genetic algorithm for training recurrent neural networks. *Neural Networks,* 14, 93-105.

Blanco, A., Pino-Mejias, R., Lara, J. and Rayo, S. **(**2013). Credit scoring models for the microfinance industry using neural networks: Evidence from Peru. *Expert Systems with Applications,* 40, 356-364.

Bozdogan, H. (2000). Akaike's information criterion and recent developments in information complexity. *Journal of Mathematical Psychology,* 44(1), 62-91.

Chalkiadakis, I., Rovithakis, G. and Zervakis, M. (2001). A structural genetic algorithm to optimize high order neural network architecture, ESANN'2001 proceedings-European Symposium on Artificial Neural Networks Bruges (Belgium), 185-192.

Desai, V.S., Crook, J.N. and Overstreet, G.A.J. (1996). A comparison of neural networks and linear scoring models in the credit union environment. *European Journal of Operational Research,* 95, 24-37.

Faraway, J. and Chatfield, C. (1998). Time series forecasting with neural networks: A comparative study using the airline data. *Journal of the Royal Statistical Society: Series C (Applied Statistics),* 47(2), 231-250.

Freitas, J.F.G. (2000). Bayesian methods for neural networks, PhD. Thesis, Trinity College University of Cambridge and Cambridge University Engineering Department, UK.

Golden, R.M. **(**1996). Mathematical methods for neural network analysis and design, The MIT Press, England.

Hamadani, A.Z., Shalbafzadeh, A., Rezvan, T. and Moghadam, A. **(**2013). An integrated genetic-based model of naive bayes networks for credit scoring. *International Journal of Artificial Intelligence & Applications (IJAIA),* 4(1).

Hsieh, N.C. (2004). An integrated data mining and behavioral scoring model for analyzing bank customers. *Expert Systems with Applications,*27(4), 623-633.

Kocadağlı, O. and Aşıkgil, B. (2014). Nonlinear time series forecasting with Bayesian neural networks. *Expert Systems with Applications,* 41, 6596-6610.

Lee, T.S., Chiu, C.C., Lu, C.J. and Chen, I.F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications,* 23, 245-254.

Lee, T.S. and Chen, I.F. **(**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.

Malhotra, R. and Malhotra, D.K. (2003). Evaluating consumer loans using neural networks. *The International Journal of Management Science,* 31, 83-96.

Matlab 7.12, http://www.mathworks.com/help/

McQuarrie, A. D. R. and Tsai, C.L. **(**2007). Regression and time series model selection. *World Scientific Publishing Co.*

Mirkin, B. (1996). Mathematical classification and clustering. *Kluwer Academic Publishers,* 74-76.

Moller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks,* 6(4), 525-533.

Niklis, D., Doumpos, M. and Zopounidis, C. (2014). Combining market and accounting-based models for credit scoring using a classification scheme based on support vector machines. *Applied Mathematics and Computation,* 234, 69-81.

Nocedal J. and Wright S. J. (2006). Numerical Optimization, 2nd Edition, Springer.

Ong, C., Huang, J. and Tzeng, G. (2005). Building credit scoring models using genetic programming. *Expert Systems with Applications,* 29(1), 41-47.

Oreski, S., Oreski, D. and Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert Systems with Applications,* 39, 12605-12617.

Schebesch, K.B. and Stecking, R. (2005). Support vector machines for classifying and describing credit applicants: Detecting typical and critical regions. *Journal of the Operational Research Society,* 56, 1082-1088.

Seiffert, U. (2001). Multiple layer perceptron training using genetic algorithms, *ESANN'2001 proceedings-European Symposium on Artificial Neural Networks Bruges (Belgium)*,159-164.

Setiono, R., Baesens, B. and Martens, D. (2012). Rule extraction from neural networks and support vector machines for credit scoring, *Data Mining: Foundations and Intelligent Paradigms Intelligent Systems Reference Library*, Springer, Book Chapter, 25, 299-320.

Silva, L. M., Marques de Sá, J. and Alexandre, L. A. (2008). Data classification with multilayer perceptrons using a generalized error function. *Neural Networks*, (21) 1302 – 1310.

Šušteršič, M., Mramor, D., and Zupan, J. (2009). Consumer credit scoring models with limited data. *Expert Systems with Applications*, 36, 4736-4744.

Thomas, L.C. (2000). A survey of credit and behavioural scoring: Forecasting financial risk of lending to customers. *International Journal of Forecasting*, 16(2), 149-172.

http://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit +Data). UCI, Machine Learning Repository.

Wang, G., Hao, J., Ma, J. and Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications*, 38, 223-230.

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, 27, 1131-1152.

APPENDIX

Table 2: Performances of Logistic Regression Models

LOGISTIC REGRESSION MODELS								
						Accuracy Ratios (%)		
Partition Ratios	**Training MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Test**	**All**
0.60 – 0.40	0.1620	-1070.246	-1069.714	-1010.880	0.1632	76.8	76.0	76.4
0.70 - 0.30	0.1589	-1265.541	-1265.087	1204.479	0.1690	77.6	74.3	76.6
0.75 – 0.25	**0.1617**	**-1344.567**	**-1344.144**	**-1282.747**	**0.1598**	**76.7**	**77.2**	**76.8**
0.80 – 0.20	0.1597	-1445.763	-1445.367	-1383.232	0.1679	77.0	74.0	76.4
0.85 – 0.15	**0.1583**	**-1544.930**	**-1544.557**	**-1481.732**	**0.1751**	**78.0**	**75.3**	**77.5**
0.90 – 0.10	0.1608	-1623.070	-1622.718	-1559.244	0.1634	77.1	75.0	76.8

Table 3: Performance of Gradient Descent

GRADIENT DESCENT												
									Accuracy Ratios (%)			
Partition Ratios	**N**	**lr**	**Iter.**	**Train MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Val**	**Test**	**All**
0.80 - 0.05 - 0.15	2	0.5	756	0.1551	-1401.18	-1395.44	-1145.37	0.149	79.0	79.**0**	74.0	78.5
0.80 - 0.05 - 0.15	**3**	**0.1**	**1550**	**0.1540**	**-1362.57**	**-1349.73**	**-981.70**	**0.145**	**80.1**	**81.0**	**76**.0	**79.8**
0.80 - 0.10 - 0.10	4	0.05	4679	0.1428	-1379.10	-1356.00	-873.17	0.143	80.6	82.0	74.0	80.1
0.75 - 0.15 - 0.10	2	0.01	5928	0.1572	-1230.90	-1224.42	-980.21	0.168	78.3	77.6	78.3	78.2
0.75 - 0.10 - 0.15	3	0.1	1353	0.1494	-1310.50	-1296.72	-933.95	0.167	78.4	74.7	77.0	77.7
0.70 - 0.20 - 0.10	3	0.01	10000	0.1592	-1152.44	-1137.57	-780.52	0.144	77.0	76.5	81.0	77.3
0.70 - 0.20 - 0.10	**3**	**0.05**	**4724**	**0.1544**	**-1173.63**	**-1158.75**	**-801.70**	**0.134**	**78**.0	**78**.0	**81**.0	**78.3**
0.70 - 0.20 - 0.10	3	0.1	1007	0.1533	-1178.85	-1163.98	-806.93	0.145	78.1	73.5	79.9	77.0

N: Number of neurons, **lr:** Learning rate, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data

Table 4: Performance of Gradient Descent with Momentum

GRADIENT DESCENT WITH MOMENTUM											
								Accuracy Ratios (%)			
Partition Ratios	**N**	**Iter.**	**Training MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Training**	**Val**	**Test**	**All**
Learning Rate = 0.1 ; Momentum Term = 0.1											
0.70 - 0.20 - 0.10	2	1932	0.1544	-1217.71	-1211.09	-967.91	0.152	78.1	76.0	79.0	77.8
0.70 - 0.20 - 0.10	3	743	0.1539	-1176.18	-1161.31	-804.26	0.153	78.4	77.0	78.0	78.1
Learning Rate = 0.1 ; Momentum Term = 0.5											
0.70 - 0.20 - 0.10	**2**	**1850**	**0.1571**	**-1205.78**	**-1199.16**	**-955.98**	**0.132**	**77.4**	**74.5**	**82**.0	**77.4**
0.70 - 0.20 - 0.10	3	2390	0.1561	-1166.02	-1151.15	-794.10	0.136	77.4	76.5	82.0	77.7
0.80 - 0.10 - 0.10	3	299	0.1480	-1394.66	-1381.83	-1013.79	0.148	79.1	79.0	77.0	78.8
Learning Rate = 0.01 ; Momentum Term = 0.5											
0.70 - 0.20 - 0.10	**2**	**1850**	**0.1571**	**-1205.78**	**-1199.16**	**-955.98**	**0.132**	**78.6**	**73.5**	**82**.0	**77.9**
0.75 - 0.15 - 0.15	2	2628	0.1476	-1276.09	-1269.61	-1025.40	0.177	78.3	72.3	79.0	77.3
Learning Rate = 0.5 ; Momentum Term = 0.5											
0.70 - 0.20 - 0.10	2	462	0.1566	-1207.87	-1201.25	-958.08	0.145	77.7	71.0	80.0	76.6
0.70 - 0.20 - 0.10	**3**	**461**	**0.1536**	**-1177.45**	**-1162.58**	**-805.52**	**0.145**	**79.1**	**76**.0	**82**.0	**78.8**
0.75 - 0.10 - 0.15	**3**	**760**	**0.1511**	**-1283.54**	**-1269.76**	**-906.99**	**0.145**	**80.9**	**81**.0	**77**.0	**80.5**

N: Number of neurons, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data

Table 5: Performance of Levenberg-Marquardt

THE LEVENBERG-MARQUARDT ALGORITHM												
									Accuracy Ratios (%)			
Partition Ratios	**N**	**Initial μ**	**Iter.**	**Train MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Val**	**Test**	**All**
0.60 - 0.30 - 0.10	2	0.01	12	0.1543	-1031.30	-1023.48	-788.44	0.1811	77.2	76	78.0	77.0
0.70 - 0.15 - 0.15	2	0.01	15	0.1639	-1175.77	-1169.15	-925.97	0.1586	77.4	75.3	79.3	77.4
0.70 - 0.20 - 0.10	2	0.01	10	0.1551	-1214.77	-1208.15	-964.97	0.1382	77.0	74.5	81.0	77.4
0.70 - 0.20 - 0.10	**3**	**0.01**	**9**	**0.1773**	**-1200.12**	**-1195.64**	**-994.73**	**0.1715**	**78.6**	**76**.0	**82**.0	**78.4**
0.75 - 0.15 - 0.10	2	0.01	19	0.1519	-1323.61	-1317.46	-1070.71	0.1650	76.3	73.3	78.0	76.0
0.80 - 0.10 - 0.10	**2**	**0.01**	**14**	**0.1359**	**-1506.51**	**-1500.77**	**-1250.70**	**0.1330**	**80.6**	**78**.0	**78**.0	**80.1**

N: Number of neurons, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data

Table 6: Performance of Scaled Conjugate Gradient

SCALED CONJUGATE GRADIENT ALGORITHM											
								Accuracy Ratios (%)			
Partition Ratios	**N**	**Iter**	**Train MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Val**	**Test**	**All**
0.70 - 0.15 - 0.15	**2**	**32**	**0.1483**	**-1246.12**	**-1239.50**	**-996.32**	**0.1401**	**80.6**	**74.7**	**81.3**	**79.8**
0.70 - 0.15 - 0.15	3	25	0.1608	-1145.50	-1130.62	-773.5	0.1504	80.0	74.1	81.0	79.2
0.70 - 0.20 - 0.10	**2**	**27**	**0.1575**	**-1204.04**	**-1197.41**	**-954.24**	**0.1367**	**78**.0	**76.5**	**82**.0	**78.1**
0.70 - 0.20 - 0.10	3	26	0.1600	-1148.79	-1133.92	-776.87	0.1354	77.0	75.3	80.7	77.3
0.75 - 0.15 - 0.10	2	26	0.1527	-1319.64	-1313.49	-1066.74	0.1711	75.9	76.7	79.0	76.0
0.80 - 0.05 - 0.15	2	37	0.1573	-1429.85	-1428.03	-1287.73	0.1728	77.4	77.0	79.0	78.3

N: Number of neurons, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data

Table 7: Performance of BFGS

BFGS ALGORITHM											
								Accuracy Ratios (%)			
Partition Ratios	**N**	**Iter**	**Train MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Val**	**Test**	**All**
0.60 - 0.30 - 0.10	2	22	0.1757	-909.36	-891.69	-547.77	0.162	78.0	75.7	78.0	77.4
0.70 - 0.20 - 0.10	**2**	**21**	**0.1647**	**-1172.42**	**-1165.80**	**-922.63**	**0.140**	**77.6**	**76.5**	**82**.0	**77.9**
0.70 - 0.20 - 0.10	**3**	**28**	**0.1583**	**-1156.16**	**-1141.29**	**-784.24**	**0.153**	**78.3**	**78**	**81**.0	**78.5**
0.75 - 0.15 - 0.10	3	14	0.1594	-1243.03	-1229.25	-866.48	0.163	76.7	75.3	77.0	76.5
0.80 - 0.05 - 0.15	2	20	0.1704	-1325.78	-1320.04	-1069.97	0.135	75.6	72.0	81.3	75.6
0.80 - 0.05 - 0.15	3	35	0.1585	-1339.83	-1326.99	-958.96	0.148	77.5	74.0	78.0	77.0
0.80 - 0.10 - 0.10	3	12	0.1638	-1313.09	-1300.25	-932.22	0.155	78.0	78.0	77.0	76.3

N: Number of neurons, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data

Table 8: Comparing to the Performances of all the methods

								Accuracy Ratios			
Method	**Partition Ratios**	**N**	**Train MSE**	**AIC**	**AIC_c**	**BIC**	**Test MSE**	**Train**	**Val**	**Test**	**All**
GD	0.70 - 0.20 - 0.10	3	0.154	-1173.63	-1158.75	-801.70	0.134	78.0	78.0	81.0	78.3
GDM	0.70 - 0.20 - 0.10	2	0.157	-1205.78	-1199.16	-955.98	0.132	77.4	74.5	82.0	77.4
L - M	0.70 - 0.20 - 0.10	3	0.177	-1200.12	-1195.64	-994.73	0.171	78.6	**76.0**	**82.0**	**78.4**
SCG	0.70 - 0.20 - 0.10	2	0.157	-1204.04	-1197.41	-954.24	0.136	78.0	76.5	82.0	78.1
BFGS	0.70 - 0.20 - 0.10	2	0.164	-1172.42	-1165.80	-922.63	0.140	77.6	76.5	82.0	77.9
Logistic	0.85 – 0.15	-	0.158	-	-	-	0.175	78.0	-	75.3	77.5

N: Number of neurons, **Iter:** Iterations, **MSE:** Mean Squared Error, **Val:** Validation data