Robert Verner (Slovakia), Ladislav Rosocha (Slovakia)

Yield spreads prediction using genetic neural network

Abstract

In this paper, the authors aim at prediction of demanded yield spreads on primary bond market using biologically inspired algorithms. The researchers combine genetic algorithms and multilayered feedforward neural network trained by Levenberg-Marquardt algorithm in order to present a genetic artificial neural network. Consequently it is estimated demanded yield spread of bonds based on parameters of individual offerings. The results indicate that compared to conventional types of artificial neural networks, genetic network reached the lowest mean squared error and highest determination coefficient on the investigated sample of 23 844 initial bond offerings and outperfomed other networks, primarily on out-of-sample data.

Keywords: artificial neural networks, bonds, genetic algorithms, initial public offerings.
JEL Classification: C45, C21, G12.

Introduction

The idea of resolving complicated finance and business problems using an artificial intelligence approach has been an interesting task for academic researchers and biologically inspired computing methods have proven themselves to be an important tool across a wide range of functional areas affecting most businesses. These methods are simple computational instruments for exploring the data and developing models that help to identify crucial patterns, characteristics or structures. Standard econometric methods might have several limitations regarding the complexity of public offering problems. Conventional models require various assumptions of the data and variables. But public issues include many variables with unknown or ill-defined relationships.

Artificial neural networks can be considered as computational structures that mimic the biological nervous system. Their ability to learn and generalize enable them to produce reasonable outcomes for inputs not seen during the training (learning). In most cases the neural network is presented with examples and its free parameters are modified to minimize the error between the actual and desired output. The learning is repeated for many examples in the dataset until it reaches a steady state with no further significant improvements. Since they are very adaptive, neural networks can operate in nonstationary environment. Using parallel structure they are able to resolve nonlinear, stochastic and ill-defined tasks.

Since artificial neural networks have been successfully applied to solve nonlinear and challenging problems, they have been actively used for applications such as bankruptcy prediction, predicting costs, forecast revenue, credit scoring and more (Lee and Chen, 2005; Hayashi et al., 2010; Moosmayer et al., 2013; Tang and Chi, 2005; West, 2000). Other more specific discipline-based reviews have appeared in auditing (Koskivaara, 2004), manufacturing (Sick, 2002), management (Boussabaine, 1996), and resource management (Maier and Dandy, 2000). These neural network systems are typically only a few basis points more precise than their alternatives, but because of the amounts of money involved, these methods are very profitable. Jain and Nag (1995) developed a neural network model for pricing initial public offerings. The neural network model significantly improved accuracy of prediction and reduced underpricing costs. Robertson et al. (1998) proposed neural networks models in order to estimate the first-day return of an initial public offering. They divided the data set into technology and non-technology offerings and constructed a regression model and two neural network models. Their results indicated that neural network models performed better on both technology and non-technology groups and overwhelmed linear regression model at predicting the first-day return of a public offering.

The most applied network learning method in practice is the error backpropagation based on gradient descent. However, it has several shortcomings, major of which is the significant risk of being stuck in a local valley of the cost function. In case of cost function with many local minima, its performance depends on the beginning point given by the vector of initial synaptic weights. But somewhere in the connection weight space there may be alternative vector of initial weights that results into considerably lower error (either better local minimum or possibly global minimum). The aim of this work is to present neural network learning and initial weights selection algorithm which might overcome the related local minimum and initial weights selection problems without requiring any intervention from the researcher.

Ing. Robert Verner, Ph.D., MBA, Department of Economics, Faculty of Business Economy with seat in Košice, University of Economics in Bratislava, Slovakia.
Ladislav Rosocha, Ph.D. Student, MUDr., Department of Quantitative Methods, Faculty of Business Economy with seat in Košice, University of Economics in Bratislava, Slovakia.

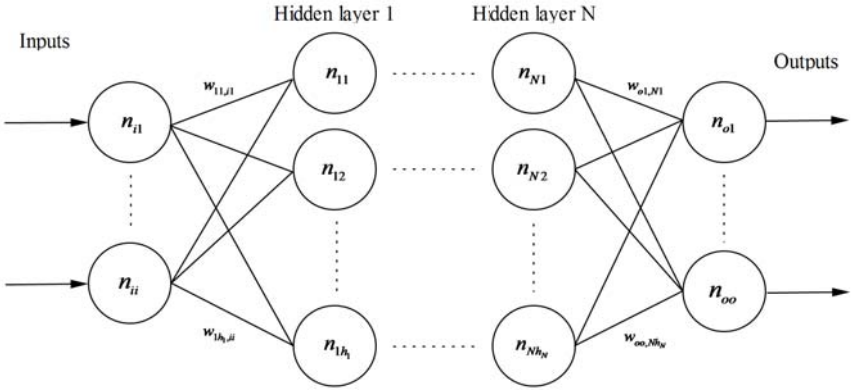
The presented hybrid learning algorithm should consistently examine the space of initial synaptic weights and possibly overcome the chance of getting stuck in the local minimum of network error function (Zgodavova, 2015).

1. Methodology

In order to avoid above mentioned premature convergence, we propose implementation of genetic algorithms into the learning process of neural network and incorporate the element of stochasticity which should enable the learning algorithm to escape from the local minimum of error function and converge to a better (potentially global) solution. We might define the topology of multilayer feedforward network as $i - h_1 - h_2 - \ldots - h_N - o_i$ with i as the number of input neurons, $h_1 - h_2 - \ldots - h_N$ representing hidden neurons in N hidden layers and o as the output nodes, weights of the network may be captured in a form of vector:

$$w^T = (w_{11,i1}, w_{12,i1}, \ldots, w_{1h_1,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}),$$

where $w_{11,i1}$ is the weight between first input neuron and the first neuron in hidden layer h_1, $w_{12,i1}$ is the weight between first input neuron and the second neuron in h_1, etc. Finally, w_{oo,Nh_N} represents the connection between the last output neuron and last neuron from final hidden layer h_N and as shown on Figure 1.



Source: Processed by authors.

Fig. 1. Connection weights

The logic behind application of genetic algorithms in the initial weights selection is the following:

1. At the beginning of learning process, genetic algorithm generates the first random population of initial weight vectors w_s^u – i.e. the generation of neural networks (individuals) with pre-defined architecture. $s = 1$ to S denotes the total number of individuals in one generation, while $u = 1$ to U is the total number of generations. At this step $s = 1$ to S and $u = 1$.
2. Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) evaluates the error function of each neural network (fitness of each individual in population) using the second-order information about the error surface. The final outcome of the network is its performance in terms of mean squared error on the never seen testing test (out-of-sample data).
3. Based on the achieved MSE (fitness), genetic algorithm performs breeding (selection, crossover, mutation, replacement) of current population of initial weight vectors in order to create new generation of initial weight vectors w_s^{t+1} (i.e. w_s^2 in this case).
4. Continue with step 2, until the termination criteria are met.

According to genetic algorithms methodology, at the first step of the learning process presented algorithm generates initial population of vectors w. If we set the total number of individuals in the generation as $S = 100$ and the total number of generations as $U = 1000$, then at the first step we would have generation of chromosomes (genetic representations of individuals):

$$w_1^{1T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$
$$w_2^{1T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$
$$w_3^{1T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$
$$w_4^{1T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$
$$\vdots$$
$$w_{100}^{1\ T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right),$$

while at the last step we would obtain final generation of chromosomes:

$$w_1^{1000T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$
$$w_2^{1000T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N}\right)$$

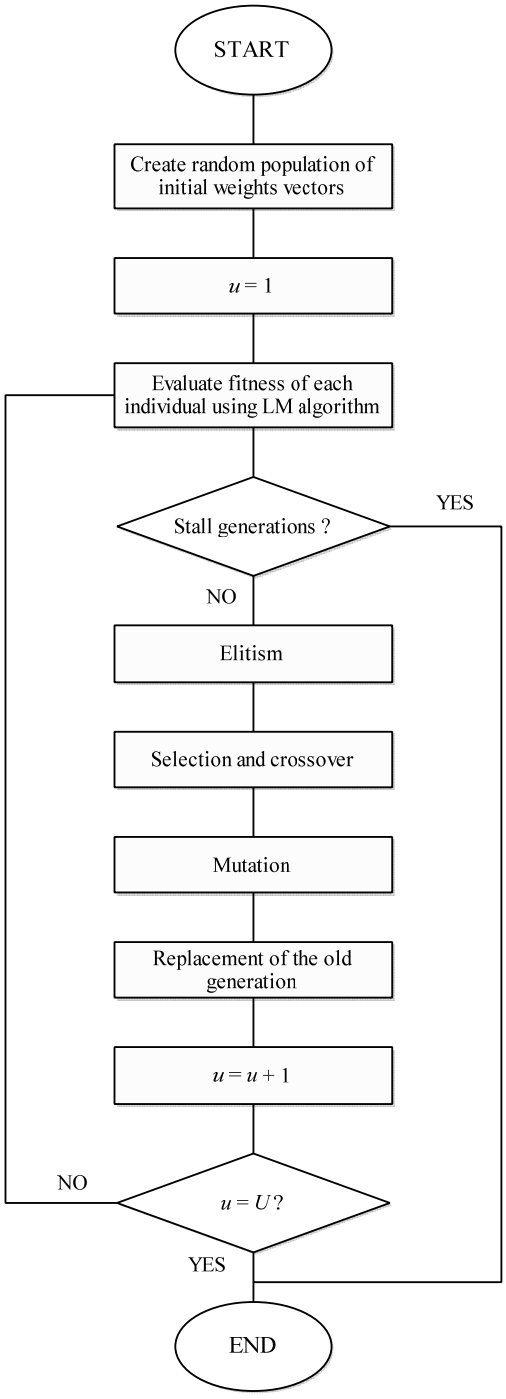
$$w_3^{1000T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N} \right)$$

$$w_4^{1000T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N} \right)$$

$$\vdots$$

$$w_{100}^{1000T} = \left(w_{11,i1}, w_{12,i1}, \ldots, w_{1h,ii}, \ldots, w_{o1,N1}, \ldots, w_{oo,Nh_N} \right).$$

Genetic algorithm performs roulette wheel selection of individual initial weights vectors and realizes single line crossover, mutation and replacement so that we can obtain next generation with superior genetic information. The algorithm runs until the predefined maximal number of generations is reached.



Source: Processed by authors.

Fig. 2. Genetic neural network

Even though the selection of stopping criteria is solely up to researcher, excessively strict termination conditions inhibit the convergence ability of the algorithm. Despite the computational costs, in order to achieve the diversibility and profoundness of the search, number of individuals in every generation had been established as $S = 200$. It is worth to notice that the process of tuning the functional parameters of most stochastic algorithms is usually the matter of trials and errors. However, if small change in input

parameter results into large modification of algorithm performance and output, such method cannot be considered as very robust and its general practicability is questionable. Figure 2 depicts the flowchart of proposed genetic neural network.

2. Data

Our data sample consisted of 23 844 EUR and USD denominated straight bond offerings with fixed coupon issued between January 2003 and April 2015 from the BondRadar information service. Perpetual and floating rate obligations have been excluded from the sample. Regarding the sample, we focused on following independent variables which characterized every debt offering:

- volume (in USD mil. equivalent);
- maturity (in years);
- rating by Moody's;
- rating by Standard & Poor's;
- rating by Fitch;
- subordinated status (yes/no);
- collateral (yes/no);
- prestige of issue leader (in total size of led issues);
- prestige of bookrunner 1 to 4 (in total size of led issues).

Independent variable was the spread over middle value of interest rate swaps (in case of EUR issues) or over US Treasury yields (in case of USD issues) with corresponding maturity in basis points. Yields of US Treasuries are approximately equal to USD interest rate swaps. Since the rating grades of credit rating agencies are in form of symbols, we evaluated the symbols on equidistant basis from 1 (default) to 21 (prime grade). Issue without credit rating from particular rating agency was assigned with 0. Tables 1 and 2 present summary statistics of independent variables. We might see that the size of an average deal was 953.23 mil. USD eq. and the largest issue was enormous 15 bln. USD eq. Average maturity of the examined bond sample was slightly below 9 years and maximal maturity 100.32 years for one obligation maturing in 2114.

In order to check the normal distribution of explanatory variables we applied Shapiro-Wilk (Shapiro and Wilk, 1965) and Jarque-Bera (Jarque and Bera, 1980) tests. Null hypothesis of both above mentioned tests is the normal distribution of data. Based on p-values (zero or very close to zero) we can reject the null hypothesis of normal distribution for all independent variables (Tkáč et al., 2012).

Table 1. Summary statistics of independent variables 1

	Volume	Maturity	Moodys	SP	Fitch	Subord.
Mean	953.23	8.9875	14.019	13.674	8.4833	0.027973
Median	700	7.0384	15	16	0	0
Minimum	10	1.9973	0	0	0	0
Maximum	15000	100.32	21	21	21	1
Standard deviation	924.06	7.4932	6.1955	6.4311	8.99007	0.1649
Variation coefficient	0.9694	0.83373	0.44194	0.47033	1.0598	5.8949
Skewness	3.3684	3.2372	-0.80427	-0.88986	0.23455	5.7251
Ex. Kurtosis	20.325	17.13	-0.16694	-0.13651	-1.7524	30.777
Shapiro-Wilk	0.691913	0.65056	0.895309	0.181956	0.746506	0.151567
p-value	4.74E-108	2.21E-111	1.28E-81	9.19E-86	5.38E-103	2.73E-136
Jarque-Bera	455499	333170	2598.28	3165.35	3269.53	1.07E+06
p-value	0	0	0	0	0	0

Source: Processed by authors.

Table 2. Summary statistics of independent variables 2

	Covered	Lead	Book1	Book2	Book23	Book2
Mean	0.086437	1.11E+06	1.01E+06	7.53E+05	4.11E+05	1.86E+05
Median	0	1.14E+06	1.14E+06	7.05E+05	19327	0
Minimum	0	0	0	0	0	0
Maximum	1	1.88E+06	1.88E+06	1.88E+06	1.88E+06	1.88E+06
Standard deviation	0.28101	4.95E+05	6.35E+05	6.61E+05	5.78E+05	4.24E+05
Variation coefficient	3.2511	0.44602	0.6252	0.8769	1.4073	2.2713
Skewness	2.9434	-0.71138	-0.33345	0.30836	1.24	2.4483
Ex. Kurtosis	6.6638	-0.2142	-1.205	-1.3066	0.27329	5.2193
Shapiro-Wilk	0.314425	0.904278	0.903363	0.88161	0.734626	0.508991
p-value	4.50E-130	1.35E-79	8.25E-80	1.93E-84	3.59E-104	1.25E-120
Jarque-Bera	78546.8	2056.7	1884.54	2073.94	6184.59	50885.4
p-value	0	0	0	0	0	0

Source: Processed by authors.

Out of 23 844 deals only 667 (2.80%) were subordinated and 2 061 (8.64%) were backed by some collateral (covered). In case of credit rating variables we cannot take average values as fundamental, because the absence of rating is valued by zero which decreases the total mean value. While median credit rating grade from Moody's was 15 (A3) and from Standard & Poor's was 16 (A2), median for Fitch was 0 (no rating).

3. Results

In order to examine the abilities of standard artificial neural network structures, we created architecture consisting of one input layer with 12 input neurons (for independent variables) and two hidden layers with 20 hidden neurons in every layer. Output layer had single neuron with desired yield spread as outcome. Activation function had been set as hyperbolic tangent for neurons in input and hidden layers and linear function for output neuron. Since hyperbolic tangent limits the values to $< -1.1 >$ this activation for output neuron would not be able to correctly estimate actual yield spreads. Another possibility was to scale the data sample to $< -1.1 >$ and apply hyperbolic tangent activation for all neurons. We have tried this approach with no significant difference in obtained results, therefore we used unscaled data and linear activation function for output neuron.

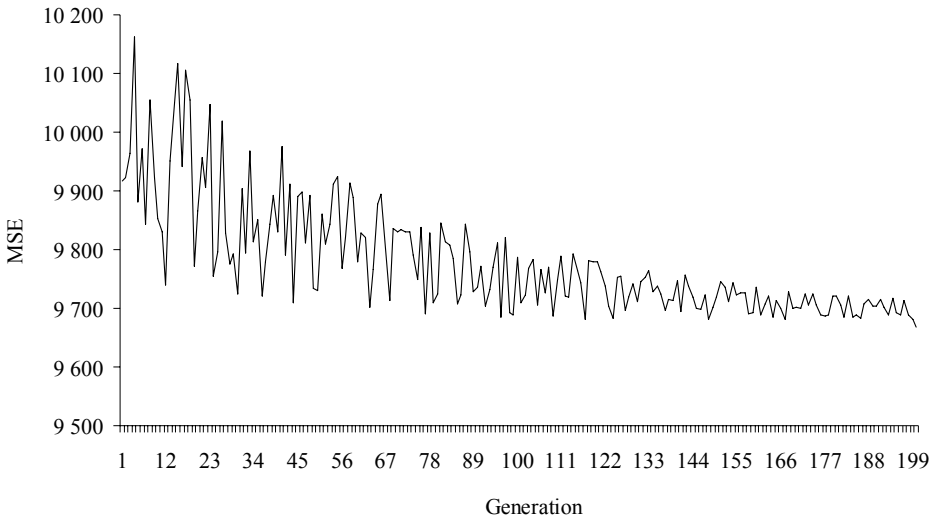
We selected six conventional neural network learning algorithms and evaluated their mean squared error and determination coefficient on the sample of presented bond offerings. The most simple was the standard Gradient descent (GD), followed by Gradient descent with momentum (GDM), Scaled conjugate gradient (SCG) proposed by Møller (1993), Levenberg-Marquardt (LM) algorithm and Conjugate gradient with Powell/Beale restarts (CGPB). Powell/Beale restart algorithm improves the search direction using specific restart strategy (Powell, 1977; Bale, 1972). Quasi-Newton methods were represented by Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). For all neural networks was the data sample divided as follows: training set comprised of 70% (16 690 observations), validation set consisted of 15% (3577 observations) and testing set contained 15% (3577 observations). This distribution should have assured not only the proper learning process, but also should have preserved the generalization ability of networks and the prevented overfitting the training data. Out-of-sample testing group is usually set from 10% to 20% of the total observations, i.e. 15% was reasonable compromise.

Genetic network was created in Java programming language, specifically for the purpose of investigating initial bond offering yields. Its characteristics had been determined to be consistent with standard methods built in Matlab, i.e. architecture with one input layer containing with 12 input neurons and two hidden layers containing 20 hidden neurons in every layer. Output layer had single neuron with desired yield spread as outcome and hyperbolic tangent activation function for neurons in input and hidden layers, while linear activation function for output neuron. After roulette wheel selection of individual initial weights vectors algorithm realized single line crossover, mutation and replacement and produced the next generation until one of the two termination criteria (maximum generations U = 200 or stall generations = 10) was met. One generation comprised of 50 individual neural networks. The data sample was divided in a same way as in case of other explored neural networks, i.e. training set contained 70%, validation set 15% and testing set 15% of observations.

The algorithm stopped after 200 generations which took more than 28 hours. A total of 10 000 networks were created and their fitness was evaluated using Levenberg-Marquardt algorithm. Final fitness was measured on testing set (out-of-sample data). The outcome of the program was in form of actual and fitted values as well as residuals of final best individual (neural network). Moreover, the program stored the best individual in every generation. Figure 3 presents the learning process of Genetic neural network in terms of the fitness function of the best individual in the generation obtained on testing set. It might be stated that despite oscillations, means squared error of Genetic network gradually decreased and converged to MSE values around 9 700.

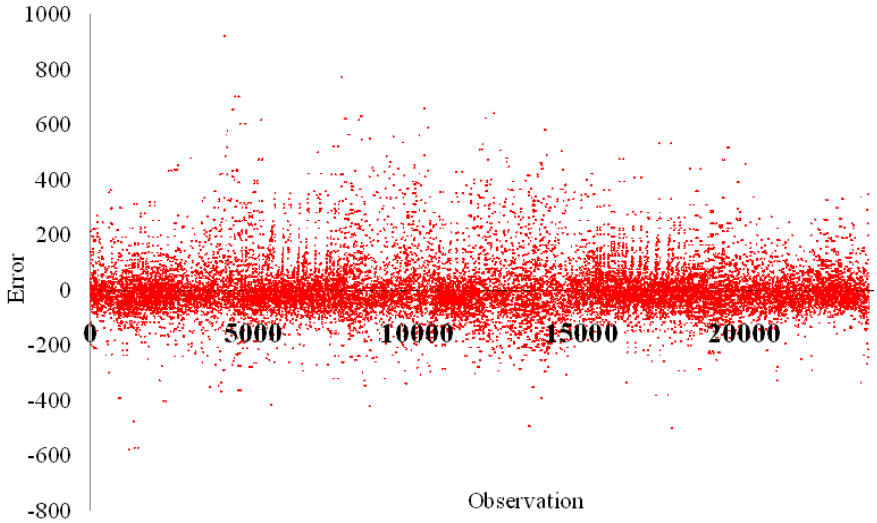
Figure 4 depicts the residuals of Genetic network, while Figure 5 compares actual bond offering yields to its fitted values.

Table 3 introduces the performance of all explored techniques. We might see that the weakest performance in terms of means squared error was unambiguously achieved by Gradient descent and Gradient descent with momentum. Other learning algorithms reached comparable results, however, Broyden-Fletcher-Goldfarb-Shanno algorithm had approximately double MSE than remaining three algorithms. Good outcomes by means of mean squared error on all three data sets were obtained by Levenberg-Marquardt algorithm. Focusing on the mean squared error, the lowest value of MSE was obtained by Genetic network (GNET) on training set and comparing to other applied techniques, GNET reached the smallest MSE on all three sets of data.



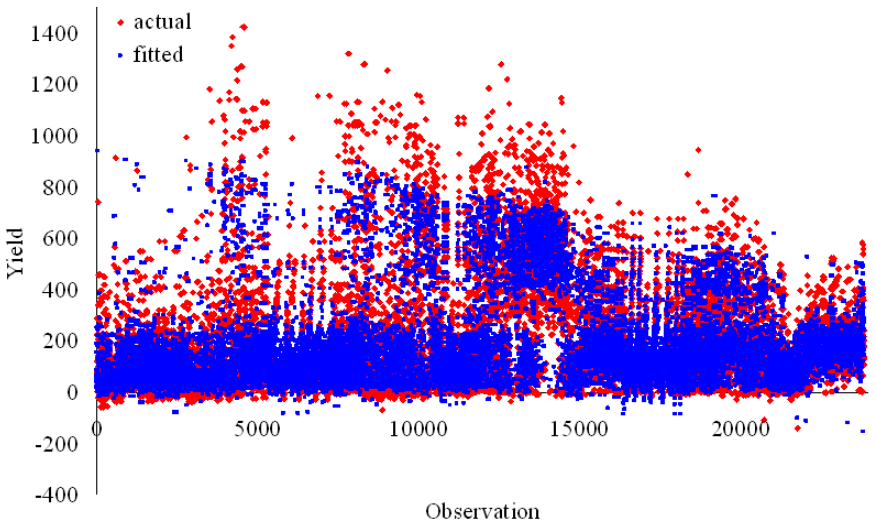
Source: Processed by authors.

Fig. 3. Progress of best individuals



Source: Processed by authors.

Fig. 4. Genetic network residuals by observation



Source: Processed by authors.

Fig. 5. Actual vs. Genetic network fitted values by observation

Table 3. Overview of performance of all techniques

	GD	GDM	CGPB	BFGS	SCG	LM	GNET
Training MSE	4543600	2004600	11140	21713	11528	10071	9487
Validation MSE	4580600	2023900	11415	23444	11760	9679	9598
Testing MSE	4602200	1972200	12257	21273	11522	9923	9681
	GD	GDM	CGPB	BFGS	SCG	LM	GNET
Training R-squared	47.32%	37.05%	85.55%	68.91%	85.08%	87.17%	88.79%
Validation R-squared	47.03%	37.44%	84.88%	69.17%	85.27%	87.51%	88.15%
Testing R-squared	47.48%	37.62%	84.65%	68.77%	84.42%	87.07%	88.36%

Source: Processed by authors.

Similar results were obtained for determination coefficient, where the highest value of GNET was produced on training set and confronted to other approaches, genetic network had the best outcomes on all data sets. Regarding overall performance, it might be concluded that more sophisticated artificial neural networks significantly overran linear methods as well as simply neural networks based on gradient descent backpropagation. Moreover, on explored sample of bond offering yields achieved presented Genetic network the best overall performance not only in terms of mean squared error, but also in form of highest determination coefficient on training, testing and validation sets. However, it must be noted that Levenberg-Marquardt algorithm required higher computational time than gradient based methods. Table 4 describes the processing time of all explored neural network learning algorithms.

Table 4. Computational time of neural network learning algorithms

	GD	GDM	CGPB	BFGS	SCG	LM
Time (seconds)	3	4	8	15	8	11

Source: Processed by authors.

Fastest methods were the Gradient descent and Gradient descent with momentum. Both conjugate gradient based techniques took 8 seconds, Levenberg-Marquardt took 11 seconds and Broyden-Fletcher-Goldfarb-Shanno algorithm worked 15 seconds. Despite the fact that the difference of three seconds between best performing Levenberg-Marquardt and both conjugate gradient methods can be ignored in case of single neural network, it has significant impact on design of proposed Genetic network which generates thousand of networks as individuals. Providing 1200 individuals, the three second difference results into one hour longer computational time. Nevertheless, the bond issuing process is not a real-time matter and preparations often take more than a month. Thus the computational time is not the primary parameter of algorithm efficiency and we might focus on the precision of yield prediction using Levenberg-Marquardt method as Genetic network learning algorithm.

Conclusion

Presented Genetic network implements the methodology of genetic algorithms into the initial weight selection process of conventional artificial networks. It creates and breeds generations of individual neural networks trained by Levenberg-Marquardt algorithm based on their fitness function in terms of mean squared error. Such network should be able to escape from the first found local minimum of error function and potentially converge to its global minimum. Even though that the Genetic network reached the best determination coefficient and lowest mean squared error among explored techniques on all three groups of data (training, validation, testing set), it required significantly higher computational time. This fact does not overly matter in case of bond yields offerings, but duration of optimization process in magnitude of hours complicates its real-time applications. One possibility is to amend additional termination criterion in form of stall generations which stops the algorithm if there is no significant improvement in the objective function for a sequence of consecutive generations. This might save the computational time and produce satisfactory solution in more reasonable period. However, it is crucial that on examined sample of bond offerings Genetic network outperformed other analyzed methods and obtained best results.

In our following research we would like to decrease the necessary computational time of learning process thus permit the real-time utilization of proposed hybrid methodology. Perspective classes of algorithms, which have already been implemented into neural network framework, are artificial immune systems and swarm intelligence techniques such as fish swarm algorithm, cuckoo search algorithm and firefly algorithm. Alternatively, Invasive weed colony optimization or Shuffled frog leaping algorithm might be tested.

References

1. Beale, E.M.L. (1972). A derivative of conjugate gradients. In Lootsma, F.A. (Ed.) *Numerical methods for nonlinear optimization*, Academic Press.
2. Boussabaine, A.H. (1996). The use of artifical neural networks in construction management: a review, *Construction Management and Economics*, 14 (5), pp. 427-436.
3. Broyden, C.G. (1970). The convergence of a class of double-rank minimization algorithms, *Journal of the Institute of Mathematics and Its Applications*, 6 (1), pp. 76-90.
4. Fletcher, R. (1970). A new approach to variable metric algorithms, *The Computer Journal*, 19 (3), pp. 317-322.
5. Goldfarb, D. (1970). A family of variable-metric methods derived by variational means, *Mathematics of Computation*, 24 (109), pp. 23-26.
6. Hayashi, Y. et al. (2010). Understanding consumer heterogeneity: A business intelligence application of neural networks, *Knowledge-Based Systems*, 23 (8), pp. 856-863.
7. Jain, B.A., Nag, B.N. Artificial neural network models for pricing initial public offerings, *Decision Sciences*, 26 (3), pp. 283-302.
8. Jarque, C.M., Bera, A.K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals, *Economics letters*, 6 (3), pp. 255-259.
9. Koskivaara, E. (2000). Artificial neural network models for predicting patterns in auditing monthly balances, *Journal of the Operational Research Society*, 51, pp. 1060-1069.
10. Lee, T.S., 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), pp. 743-752.
11. Levenberg, K. (1944). A method for the solution of certain problems in least squares, *Quarterly of applied mathematics*, 2 (2), pp. 164-168.
12. Maier, H.R., Dandy, G.C. Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications, *Environmental Modelling and Software*, 15 (1), pp. 101-124.
13. Marquardt, D.W. (1963). An algorithm for least-squares estimation of nonlinear parameters, *Journal of the Society for Industrial & Applied Mathematics*, 11 (2), pp. 431-441.
14. Møller, M.F. (1993). A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks*, 6 (4), pp. 525-533.
15. Moosmayer, D.C. et al. (2013). A neural network approach to predicting price negotiation outcomes in business-to-business contexts, *Expert Systems with Applications*, 40 (8), pp. 3028-3035.
16. Powell, M.J.D. (1977). Restart procedures for the conjugate gradient method, *Mathematical programming*, 12 (1), pp. 241-254.
17. Robertson, S.J. et al. (1998). Neural network models for initial public offerings, *Neurocomputing*, 18 (1), pp. 165-182.
18. Shaphiro, S.S., Wilk, M.B. (1965). An analysis of variance test for normality, *Biometrika*, 52, pp. 591-611.
19. Shanno, D.F. (1970). Conditioning of quasi-Newton methods for function minimization, *Mathematics of Computation*, 24 (111), pp. 647-656.
20. Sick, B. (2002). On-line and indirect tool wear monitoring in turning with artificial neural networks: A review of more than a decade of research, *Mechanical Systems and Signal Processing*, 16 (4), pp. 487-546.
21. Tang, T.C., Chi, L.C. (2005). Neural networks analysis in business failure prediction of Chinese importers: A between-countries approach, *Expert Systems with Applications*, 29 (2), pp. 244-255.
22. Tkáč, M., Czillingová, J., & Petruška, I. (2012). Financial and economic analysis of steel industry by multivariate analysis, *Ekonomický časopis*, 4, pp. 388-405.
23. West, D. (2000). Neural network credit scoring models, *Computers & Operations Research*, 27 (11-12), pp. 1131-1152.
24. Zgodavova, K. (2015). Self-Assessment. In Dahlgaard-Park, S.M. (Ed.). *The SAGE Encyclopedia of Quality and the Service Economy*. SAGE Publications, pp. 664-668.