Adv Syst Sci Appl 2018; **02**; 53-58
Published online at http://ijassa.ipu.ru/index.php/ijassa/article/view/582

Use of neural network models in market risk management

Mariya Radosteva[1], Vladimir Soloviev[1*], Vera Ivanyuk[1,2], Anatoliy Tsvirkun[2]

[1)] Financial University under the Government of the Russian Federation, Moscow, Russia

E-mail: vsoloviev@fa.ru

[2)] Institute of Control Sciences RAS, Moscow, Russia

E-mail: tsvirkun@ipu.ru

Abstract*:* This topic is of high relevance due to the fact that many market risk assessment mathematical models currently available contain many limitations for their effective use. However, these limitations are often not feasible, which leads to a decrease in forecast accuracy. To avoid this, more accurate models are necessary. Neural network-based models can show a more accurate result due to their basic property – nonlinearity. The goal of this paper is to build a model that can enable us to assess a market risk for a company.

Keywords: risk, forecasting, neural networks

1. INTRODUCTION

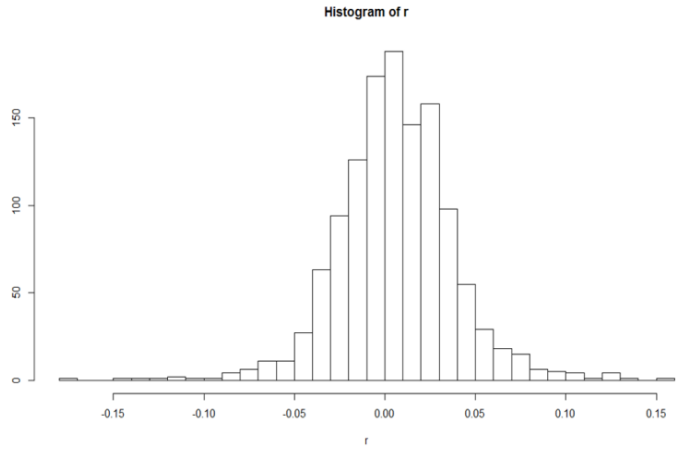

Fig. 1.

The primary goal of this paper is to determine a lower bound of the yield to be forecast by the neural network model with a certain level of significance. Current actual yields will be fed to the neural network output, and some factors will be fed to the neural network input.

Starbucks stock prices for the period from 2011 to 2016 were taken as the factors fed to the neural network input.

[*] Corresponding author: vsoloviev@fa.ru

Fig. 2

However, there is no point in using this data as is, so the values of stock yield were fed to the neural network input. It was decided to take weekly yields with a lag of one day as the yield basic indices. In other words, the yield values were calculated as follows:

$$r_t = \frac{P_t - P_{t-5}}{P_{t-5}},$$

where the *t* is a time interval assumed as a day (the *t* assumes the values of 6,7,8,... from the beginning of the counting period taking into account that the initial value is indexed by the one);

The P_t is the closing price on the *t* day without taking into account dividends (this choice is due to the fact that we cannot forecast dividends upon stocks and we do not need this because we assess the market risk of a company);

The r_t is the earned weekly yield which has been used for modeling.

Fig. 3

To improve the model quality volatility was also used as input and output data, so the next step was to calculate daily yields and weekly volatilities (standard deviations) on their basis with a lag of one day similarly with the yields. The volatilities finding procedure can be described as follows:

$$r_k = \frac{P_k - P_{k-1}}{P_{k-1}}$$

$$\sigma = \sqrt{\sum_{k=1}^{4} (r_k - \bar{r_k})^2}$$

Another input parameter was trade volume change values.

$$volume_t = \frac{Volume_t - Volume_{t-5}}{Volume_{t-5}},$$

where $Volume_t$ is an absolute trade volume value on the *t* day; $volume_t$ is a trade volume relative change on the *t* day.

2. DISTRIBUTION OF THE NEURAL NETWORK ERRORS WHEN FORECASTING THE YIELD

The model is based on the architecture of the artificial neural network of a multilayer perceptron or a direct propagation network with error back propagation learning method.

The model is aimed at forecasting yield lower bound for the time being, which can be crossed by the actual yields graph not more than α in 100% of cases. To solve this problem, it was decided that the neural network should learn based on historical data. It was also assumed that the current yield and volatility depended on lagged yield and lagged volatility values.

An empirical distribution function was constructed and a quantile of 0.95 level was taken. The error distribution density is shown in Fig. 4.

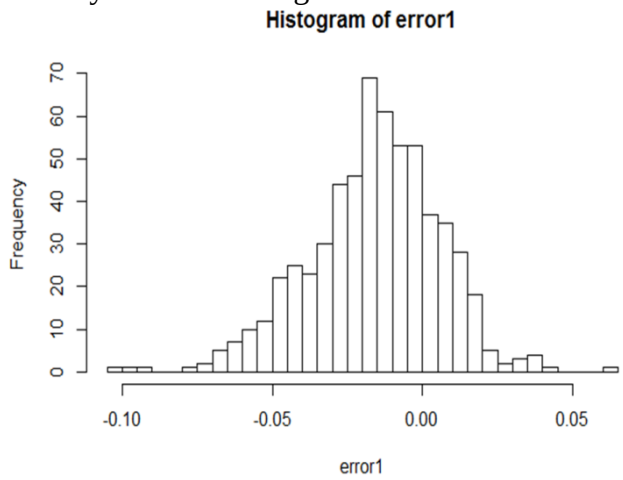

Fig. 4

The resulting value was calculated as a correction level to the initially specified VaR curve. That is the final formula for the VaR based on the neural network is as follows:

$$VaR_{neur} = r_{neur} - \sigma_{neur} - correction$$

This curve was constructed in two ways. The first way of constructing is based on the fact that the neural network learns once and then a forecast is made using it. That is essentially a model with static weights in that the weights inside the network are calculated once and no longer change. This approach to the VaR curve constructing gives an advantage in terms of program running time, since the learning occurs once. For the sake of convenience this model will be hereinafter referred to as a neural network model with static weights.

The second way is based on the assumption that the market structure is changeable (which is actually the case) and to make a forecast more accurate the neural network learns each

time when calculating the VaR. That is, to forecast the yield for the time being the neural network learns using N size sample, which includes the previous N values of this index. This is a model with dynamic weights. At each calculation of the VaR value the neural network learns again. Of course, such method of the VaR constructing is inferior to the first one, but it should be more accurate. For the sake of convenience this model will be hereinafter referred to as a neural network model with dynamic weights.

Kupic test consists in checking the following statistical hypothesis:

$$H_0: \alpha_0 = \alpha \text{ against alternative } H_{alt}: \alpha_0 \neq \alpha,$$

where the $\alpha = \frac{K}{N}$, K is the number of the VaR line breaks, the N is the quantity of forecast data, and the $1 - \alpha_0$ is a specified level of significance.

Checking is performed using statistics

$$S_{Cup} = -2\ln((1-\alpha)^{N-K}\alpha^K) + -2\ln((1-\alpha_0)^{N-K}{\alpha_0}^K),$$

which has the $\chi^2(1)$ distribution if the null hypothesis is true.

Another quality index that helps to choice between the models that passed the Kupic test is the loss function value, which measures the average value of the VaR level excess by actual losses. The smaller the loss function value, the more adequately risk is assessed by the considered model. The loss functions most commonly used are the Lopez and Blanco-Ihle's ones:

$$L_{Lo} = \frac{1}{K}\sum_{t=1}^{T}(y_t - VaR_t)^2 I(y_t < VaR_t),$$

$$L_{BI} = \frac{1}{K}\left(\frac{y_t - VaR_t}{VaR_t}\right) I(y_t < VaR_t)$$

The Lopez's loss function is different in that it gives a greater weight to significant deviations. This is justified from a substantive point of view, since single large excesses are usually more dangerous than a few small ones.

When constructing the VaR curve, the programming language R with the neuralnet library was used, which allowed to construct direct propagation neural networks with the error back propagation methods in different versions. In particular, the RPROP learning method was used.

During the research, experiments to change the number of layers in the multilayer perceptron, as well as the number of neurons in each layer in order to improve forecast accuracy were carried out. The learning sample-based forecast error formula built into the neural network was used as a guide. Such error formula was calculated as follows:

$$E = \frac{1}{2}\left(y_{fact} - \tilde{y}\right)^2$$

and was output by the program itself. Since the neural network showed different results at each learning session, it was decided to take the arithmetic mean of this error in order to generalize this parameter for the neural network with such layers and quantity.

$$E_{\text{cp}} = \frac{1}{n}\sum_{i=1}^{n} E_i,$$

where the *n* was taken equal to the order of 30.

The result of the research is the **Error! Reference source not found.**.

Table 1.

Number of neurons in layers	**Yield**		**Volatility**
	With a relative change in trade volume	Without a relative change in trade volume	Without a relative change in trade volume
(6, 3)	0.14399	0.163759	0.00813

(8, 3)	0.135139	0.157891	0.00813
(11, 6)	0.12661	0.14632	0.00813
(12, 3)	0.12789	0.139695	0.00813
(12, 6)	0.13096	0.132457	0.00813
(22, 6)	0.1176	0.14441	0.00801
(22, 11)	0.10761	0.15441	0.00771
(24, 3)	0.107	0.14177	0.00713
(24, 6)	0.11826	0.17130	0.00712
(22, 11, 3)	0.12405	0.1789	0.01519
(22, 11, 6)	0.10574	0.17111	0.02907
(24, 12, 6)	0.11581	0.1753	0.01813

From the implementation point of view the back propagation method converges for a long time, therefore its various modifications are often used. In particular, during the neural network learning the RPROP method has been used which is as follows.

The algorithm is based on the partial derivative sign. New weights are updated according to:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

$$\begin{cases} \Delta w_{ij}^{(t)} = -sign\left(\frac{\partial E}{\partial w_{ij}}^{(t)}\right)\Delta_{ij}^{(t)}, \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} \geq 0 \\ \Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)} \text{ и } \left(\frac{\partial E}{\partial w_{ij}}^{(t)}\right) = 0, \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \end{cases}.$$

where the $sign()$ is a function that assumes the value of +1 or -1 depending on which sign has a value in parentheses;

$$\Delta_{ij}^{(t)} = \begin{cases} \min\left(\eta^{+}\Delta_{ij}^{(t-1)}, \Delta_{\max}\right), & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \max\left(\eta^{-}\Delta_{ij}^{(t-1)}, \Delta_{\min}\right), & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \frac{\partial E}{\partial w_{ij}}^{(t)} = 0 \end{cases}$$

where the zero values $\Delta_{ij}^{(0)}$ and $0 < \eta^{-} < 1 < \eta^{+}$ are chosen arbitrarily. Such an algorithm converges faster than a usual back propagation method.

The result of this study is the VaR curve constructed on the basis of the neural network model.

One of the ways to implement the neural network model for the VaR level forecasting is shown in Figure 5.

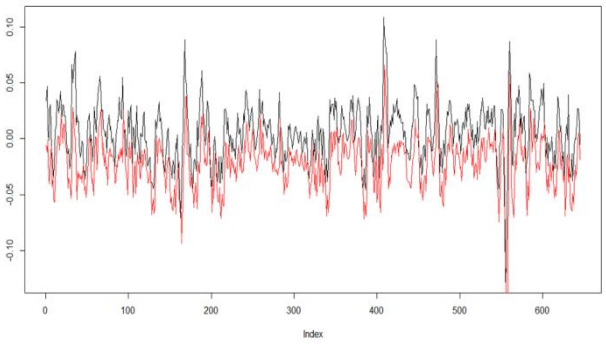

Fig. 5

The VaR curve constructed on the basis of the neural network model with static weights and a significance level of 0.95 (the black line is the yield curve, the red line is the VaR level).

In this case, the actual percentage of break is at the level of 4.65%. The Kupic test has shown a p-value= 0.6809912418, which is a good result.

The Lopez and Blanco-Ihle's tests for such a situation assumed the following values:

$$L_{Lo} = 0.0001344051456,$$

$$L_{BI} = 0.1513434227,$$

which generally evidences a good quality of the model, since the break depth is relatively small. In particular, the Lopez's index indicates that existing breaks are light, which is a good sign.

The result of the paper is the market risk assessment neural network model, which has shown good results according to the tests performed.

REFERENCES

[1] Ben, K. & Van der Smagt, P. (1996). An introduction to neural networks. University of Amsterdam.

[2] Haykin, S. (1999) Neural Networks: A Comprehensive Foundation, Prentice Hall

[3] Ivanyuk, V. & Tsvirkun, A. (2013). Intelligent system for financial time series prediction and identification of periods of speculative growth on the financial market. IFAC Proceedings Volumes, 46(9), 1128-1133.

[4] Ivanyuk, V. & Pashchenko, F. (2015) Methods and models for the forecasting and management of time series. // ITISE 2015, Granada, 283-292

[5] Simonov, B., Ivanyuk, V., & Simonova, I. (2016). Existence of Best Approximation Elements in the spaces L. Journal of Mathematical Sciences, 217(5), 1-21.

[6] Koroteev, M., Terelyanskii,, P. & Ivanyuk V. (2016) Approximation of Series of Expert Preferences by Dynamical Fuzzy Numbers. Journal of Mathematical Sciences, 216(5), 692-695.

[7] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.

