TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
ISSN 2029-4913 print/ISSN 2029-4921 online

2014 Volume 20(1): 116–132
doi:10.3846/20294913.2014.889051

CONFIGURING ARTIFICIAL NEURAL NETWORKS FOR STOCK MARKET PREDICTIONS

Gheorghe RUXANDA, Laura Maria BADEA

Bucharest University of Economic Studies, 15–17 Calea Dorobanti, District 1, Bucharest, Romania

Received 28 May 2013; accepted 24 November 2013

Abstract. Making accurate predictions for stock market values with advanced non-linear methods creates opportunities for business practitioners, especially nowadays, with highly volatile stock market evolutions. Well suited for approaching non-linear problems, Artificial Neural Networks provide a number of features which make possible reasonably accurate forecasts. But, like the old Latin saying "Primus inter pares", not all Artificial Neural Networks perform the same, end results depending very much on the network architecture and, more specifically, on the chosen training algorithm. This paper provides suggestions on how to configure Artificial Neural Networks for performing stock market predictions, with an application on the Romanian BET index. Final results are confirmed by testing the trained networks on the Croatian Stock Market data. End remarks entitle Broyden-Fletcher-Goldfarb-Shanno training algorithm as a good choice in terms of model convergence and generalization capacity.

Keywords: prediction, Artificial Neural Networks, nonlinear programming, gradient descent, BFGS, numerical differentiation, stock exchange market.

Reference to this paper should be made as follows: Ruxanda, G.; Badea, L. M. 2014. Configuring Artificial Neural Networks for stock market predictions, *Technological and Economic Development of Economy* 20(1): 116–132.

JEL Classification: C45, C51, C53, C63.

Introduction

Driven by the prospect of high profits and benefits that can be extracted from speculative activities, over the past decades, predicting stock market prices has become an attainable goal for business practitioners due to powerful estimation tools and advanced computing resources. Available theories such as efficient market hypothesis (EMH) and random walk

Corresponding author Laura Maria Badea
E-mail: *laura.maria.badea@gmail.com*

support the idea that it is virtually impossible to forecast stock prices. According to EMH, stock values reflect all available information, any new knowledge quickly being absorbed by the market in an efficient manner. On the other hand, the random walk theory assumes that past values do not impact the current values as no trend exists, all variations being the result of a random process. Nevertheless, with the proper approach and by using advanced forecasting models, the market response can be speculated. For many years, the classical Auto-Regressive Integrated Moving Average (ARIMA) technique has been used to make stock exchange predictions. However, the stock market evolution is difficult to predict using linear approaches. Recent advancements in computational area make non-linear models a viable option for time series estimations, and Artificial Neural Networks (ANNs) are such mathematical representations, very popular these days in many fields, including stock market prediction.

In this study differently configured ANNs are built and compared in terms of forecasting errors when making predictions on Bucharest Stock Market Index, BET. The paper is organized as follows: Section 1 gives an overview on the related work regarding the use of ANNs in stock market predictions. Section 2 presents relevant information about Bucharest Stock Exchange, establishing a context. Also, information about employed data and sampling technique are briefly discussed in this part of the paper. Section 3 details the model building steps and presents the mathematical backgrounds of some optimization algorithms used for network training: gradient descent and Broyden-Fletcher-Goldfarb-Shanno method (henceforth also BFGS). These are further used to make predictions on BET index. This section also presents a powerful algorithm for numerical differentiation, which is a method used to evaluate first order and second order derivatives. Section 4 provides the results of the models built with ANNs and those obtained when testing the networks on the Croatian market data. At the end of the paper the main conclusions are presented regarding the use of Artificial Neural Networks in stock market forecasting applications, and proposals for future developments.

1. ANNs and stock market forecasting – literature review

After the reticence period from the late 60's and from the 70's, when Minsky, Papert (1969) criticised Neural Networks, especially the perceptron model, many improvements and developments have been delivered by researchers to sustain the use of ANNs. As such, nowadays, ANNs have gained success in many areas, from mathematics and informatics to medicine and economics (Iordache, Spircu 2011). Given their flexibility in handling non-linear data, and considering superior results offered when compared with traditional estimation techniques, over the past years, Artificial Neural Networks have been intensely used in forecasting applications. Exchange rate prediction and stock price forecasting are common areas where ANNs have proven the ability to reach good results. Egeli *et al.* (2003) used ANNs to predict Istanbul Stock Exchange market index and they observed that these methods attain better results compared with Moving Average approach. Coupelon (2007) also showed that Artificial Neural Networks provide good solutions when predicting stock movements. Faria *et al.* (2009) compared the forecasting power of ANNs with that of the adaptive exponential smoothing method using the principal index of the Brazilian stock market. Isfan *et al.* (2010) compared

ANNs with traditional estimation techniques using forecast results obtained on Portuguese stock market, proving also that Neural Networks are very efficient when dealing with the non-linear character of financial data. Also, Georgescu (2010) used one-value-ahead Neural Networks forecasting methods to make stock exchange predictions on the Romanian market.

More recently, Vahedi (2012) used ANNs to predict stock price in Tehran Stock Exchange using investment income, stock sales income, earnings per share and net assets. His results support the ability of ANNs to perform well in stock exchange forecasting applications by using quasi-Newton training algorithms. Using the observed values between 2003 and 2006 of the Nigerian Stock Exchange (NSE) market index, Idowu *et al*. (2012) showed that ANNs can generate good predictions, however, without disregarding the configuration process which needs to be performed very carefully and which represents an essential factor in generating meaningful results with these modelling techniques. Khan, Gour (2013) compared the forecasting power of different technical methods with ANNs and in the end reached to the conclusion that back-propagation Neural Networks generate better outcomes.

2. Datasets and sampling methods

The Romanian Stock Market had a tradition of over 70 years before restarting its' activity in November 1995. Starting from 1996, when the mass management/employee buy-out (MEBO) process took place, the number of transactions performed by Bucharest Stock Exchange Market has significantly increased, marking the beginning of a viable and promising trading mechanism. In 1997, the Bucharest Stock Exchange introduced its' first synthetic index, BET, aimed to give a general image of the stock exchange performance. BET is a weighted index of the free-float capitalization of the top ten most liquid companies listed at the Bucharest Stock Exchange. The liquidity ratio is calculated semi-annually and this methodology allows BET index to represent a support asset for financial derivatives and for structured products.

The evolution of the Romanian stock exchange followed an increasing trend after the implementation of BET index, gaining the attention of many investors. In 2004, Bucharest Stock Exchange capitalization reached the level of almost 12 billion USD, which back then represented about 17% of the Romanian GDP. In 2006, the average value of daily transactions surpassed the 10 million EUR threshold, this being also highlighted by the ascending trend of the Bucharest Stock Market index, BET. Nevertheless, the peak was reached in July 2007, when BET index hit a value that was ten times higher compared with September 1997, when the index was first introduced. Thus, mid of 2007 brought a maximum point for BET index, marking also the beginning of the Romanian Stock Exchange decrease. In 2008, the Romanian Stock Market severely felt the shocks induced by the economic crisis, moving towards a new inflexion point reached at the beginning of 2009, representing a new minimum this time. Since then, the general evolution of BET index outlined a rising trend.

Using the data observed between 1st of January 2005 and 31st of March 2013, the aim of this study was to forecast BET index value using lagged prices and also macroeconomic indicators which might prove relevance in explaining the evolution of this indicator. Previous studies (Zoicas, Făt 2005; Ungureanu *et al.* 2011) have already emphasized some bonds between certain Bucharest Stock Market indexes and macroeconomic indicators like: EUR/RON

exchange rate (where RON denotes the Romanian New Leu), unemployment rate, inflation rate and different forms of interbank average interest rates (ROBID and ROBOR by maturity bands[1]). Nevertheless, considering the low frequency of the information provided by the inflation rate and unemployment rate, only the other two indicators (EUR/RON and interest rates), which offer daily observations, were further considered in this study.

Taken from the perspective of an investment, ROBID represents an alternative to stock exchange and foreign currency capital placing. Thus, this leaves only EUR/RON and ROBID macroeconomic indicators for the analysis. In order to prevent the model from considering irrelevant information which unnecessarily overloads the training process of ANNs, a simple regression model between BET index and each of the remaining macroeconomic indicators was computed. Table 1 shows that ROBID_12M provided the highest R-square (determination coefficient) for BET compared with the other ROBID ratios, namely 0.101884. However, this is still much lower compared with EUR/RON indicator which generated an R-squared of 0.556330. Thus, only EUR/RON exchange rate was further kept for predicting BET index values.

Table 1. R-squared results for simple regressions of BET index against ROBID and EUR/RON

	ROBID_ON	ROBID_TN	ROBID_1W	ROBID_1M
R-squared	0.028535	0.033433	0.050462	0.056978
	ROBID_3M	**ROBID_6M**	**ROBID_9M**	**ROBID_12M**
R-squared	0.077268	0.092286	0.101399	0.101884
	EUR/RON			
R-squared	0.556330			

Figure 1a provides the evolution of the closing BET index over the selected timeframe compared with EUR/RON exchange rate available on the official website of the National Bank of Romania. Missing values resulted from non-transactional days, such as legal holidays, were replaced by values from previous available days. The negative correlation between the two time series is visible with the naked eye, grounding the process of further searching for connexions between these two ratios. Up until mid of 2007, the blooming Romanian economy was reflected by a rising trend observed in the evolution of BET index and by significant appreciations of the national currency as related to EUR. However, beginning with 2008, the developments of these two ratios have taken the opposite trends, severe depreciations being observed especially starting with October 2008, the point when the worldwide economic crisis has installed.

When building a model with Artificial Neural Networks, data partitioning is a very important step. The initial data was split into three datasets, as follows:

– Training set – 80% of the initial dataset, which was used for model development (1st of January 2005 – 5th of August 2011);

[1] ROBID is the interbank average interest rate for deposits, and ROBOR is the interbank average interest rate for loans granted. Each one is available for 8 maturities: overnight (ROBID_ON), tomorrow-next (ROBID_TN), one week (ROBID_1W), one month (ROBID_1M), three months (ROBID_3M), six months (ROBID_6M), nine months (ROBID_9M), and twelve months (ROBID_12M).

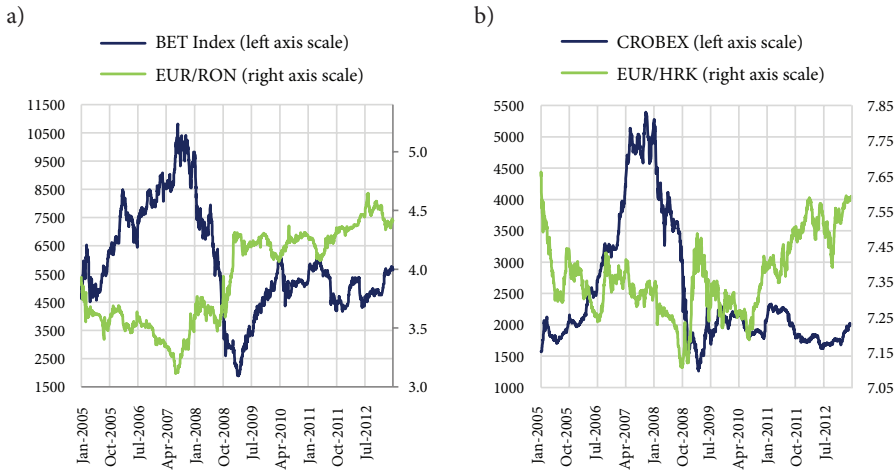

Fig. 1. BET Index vs. EUR/RON (a); CROBEX Index vs. EUR/HRK (b)

- Validation set – 10% of the initial dataset, used for model assessment (8th of August 2011 – 31st of May 2012);
- Test set – the remaining 10% of the initial dataset, which offers a completely out-of-time reassessment of the model (1st of June 2012 – 31st of March 2013).

The partitioning rule is based on the chronological dimension, meaning that the oldest 80% of the values have fallen into the training set, the following 10% of these were included in the validation set, and the most recent 10% were part of the test set.

For additional out-of-sample testing of the final results obtained on the Romanian market data, the models will be also checked on the Croatian Stock Exchange official index, CROBEX. Croatia is a country situated between South-Eastern Europe and Central Europe that has made remarkable progress over the past years, enhancing its adherence to EU. CROBEX was introduced in 1997 and it measures the performance of Zagreb Stock Exchange (ZSE) by including the 25 most liquid companies listed at ZSE. Figure 1b shows that within the time period 1st of January 2005 – 31st March 2013 there is a high resemblance between BET index and CROBEX index evolutions. Also, the relationship between EUR/HRK exchange rate and Zagreb Stock Exchange indicator highlights a similar evolution to the one observed on the Romanian market between BET stock index and EUR/RON exchange rate, suggesting that this testing data was properly chosen. Nevertheless, the testing will be performed only on the data corresponding to the period included in the test set of the Romanian data set, meaning on the timeframe 1st of June 2012 – 31st of March 2013.

3. Configuring ANNs for stock exchange predictions

Artificial Neural Networks are modelling techniques which have successfully been used in previous stock exchange forecasting applications. However, as with any other Neural Network model, their performance depends on a number of elements such as: the network type, the

training method and other configuration components that will be further approached. Often, some of these elements are selected based on a process of trial-and-error comparison aimed to identify the model with the lowest error, usually on the test set. However, the end decision should always be taken, based on a trade-off between training costs and benefits emerged from using a certain network.

The basic principle of ANNs stands in generating a signal or an outcome based on a weighted sum of inputs which is afterwards passed through an activation function as below:

$$y = f(\sum wx + b), \tag{1}$$

where: x is the vector of input variables; w is the vector of weights; b is the bias; $f(.)$ is the activation function, and y is the output vector.

One of the most used types of ANNs within stock exchange applications is the feed-forward multilayer perceptron (MLP). This is organized in three categories of layers (input layer, hidden layers and output layer) and the information flow is performed in a feed-forward manner. In this work, differently configured multilayer feed-forward Neural Networks are developed to make predictions on Bucharest Stock Exchange BET index. These configurations are further detailed in the upcoming sections.

3.1. Input and output variables

This study seeks to predict BET index evolution using lagged values, but also the signals induced by the evolution of EUR/RON exchange rate. Egeli *et al.* (2003) used the price of the Istanbul Stock Exchange value from one day before and the previous day TL/USD exchange rate, along with other variables to predict the evolution of the stock exchange index. Considering the non-stationary character of the stock exchange data series (Georgescu 2011), the first difference of the log time series were performed on BET index and on EUR/RON exchange rate. Usually, for financial predictions the best outputs are reached when short forecasting periods are considered. Therefore, the time horizon to be predicted was set to one day ahead.

In this paper, input variables were established based on a stepwise forward regression. Thus, for EUR/RON exchange rate two steps backwards were tested and for BET index five previous steps were evaluated in respect with their *p-values*. Stepwise forward regression is performed by initially estimating a linear regression of the dependent variable against each independent variable. After selecting the independent variable with the lowest *p-value*, all possible two-variable regressions in which one of the variables is the one resulted as significant after the initial estimation are computed. If more of the two-variable regressions are significant in respect with the *p-values*, then the model which generates the lowest *p-values* is selected. Next, both of the added variables are checked against the *backwards p-value* criterion, and variables with *p-value* higher than the selected criterion are removed from the model. After that, the next variable is added after choosing the three-variable regression with the lowest *p-values*. After each new variable entry, they are again all tested against the backwards criterion and removed from the model if they don't meet the *p-value backwards* criterion. The process stops when the lowest *p-value* of the variables not yet included in the regression exceeds the forward stopping criterion.

Table 2 provides the results of the stepwise forward regression using the selected stopping criteria: *p-value forward* greater than 0.1 and *p-value backwards* exceeding 0.1. The stepwise regression was performed using the training and validation datasets. Results highlight that the following indicators should be used as input variables for predicting BET index at time point *t*:

- d_ln_BET_(-1) – the modification of BET index from one day before;
- d_ln_BET_(-3) – the modification of BET index from three days before;
- d_ln_EUR_RON_(-2) – the modification of EUR/RON exchange rate from two days before.

Table 2. Input variables for BET index

Dependent Variable: D_LN_BET_T
Method: Stepwise Regression
Included observations: 1924 after adjustments
Number of always included regressors: 3
Selection method: Stepwise forwards
Stopping criterion: p-value forwards/backwards = 0.1/0.1

Variable	Coefficient	Std. Error	t-Statistic	Prob.
D_LN_BET_T_1	0.077819	0.022683	3.430684	0.0006
D_LN_BET_T_3	–0.038682	0.022700	–1.704051	0.0885
D_LN_EUR_RON_T_2	–0.274951	0.094916	–2.896787	0.0038

3.2. Hidden layers and hidden nodes

Even though there is no rule of thumb when setting the number of hidden layers and hidden nodes, care must be taken when selecting these elements. A high number of hidden nodes might generate an over-fitted model, while a network with a small number of hidden units is at risk of performing poor on new observations. Usually, these elements are selected after performing a series of experiments in which different values are tested and final forecasting errors are compared. Thenmozhi (2006) outlines that most of the studies on stock exchange prediction using ANNs include up to 12 hidden nodes. However, past research has only given some hints on how to set these values, the end decision still depending on the analysed problem and, more precisely, on the available data. For the current experiment one hidden layer was selected, and the number of hidden nodes ranged between a minimum of 2 and a maximum of 6, twice the number of input variables (Jha 2009).

3.3. Activation functions

Activation functions are applied to the weighted sum of inputs of a node in order to generate a certain outcome. Sibi *et al.* (2013) provide examples of activation functions that can be used when training Neural Networks with the back-propagation method. As the non-linear character of ANNs is given by the form of the activation functions, the most common types especially within the hidden layers are those taking a non-linear form. Among these, sigmoid ("*S*" shape) functions are often preferred for their continuous character, which makes possible differentiation, an important feature when training with back-propagation, but also

for their bounded range ([0,1] or [–1,1]), which makes them easily interpreted. Some of the most common types of sigmoid functions are:

– Logistic function (Verhulst 1845):

$$f(x)=\frac{1}{1+e^{-x}}; \tag{2}$$

– Hyperbolic tangent function (tanh) (Abbé Sauri 1774):

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}; \tag{3}$$

– Elliott function (Elliott 1993):

$$f(x)=\frac{1}{1+|x|}. \tag{4}$$

Based on previous studies and indications (Kaastra, Boyd 1995; Bishop 1995), this paper analyses, in turns, the use of logistic and hyperbolic tangent functions in the hidden layer. In the output layer, linear activation function was selected for all networks built.

3.4. Error function

Every training cycle of an ANN generates a certain cost, measured by using an error function which analyses the differences between the network outputs and the target (desired) outputs. During each cycle, the error corresponding to all training observations is reassessed, further generating new adjustments in the network weights with the purpose of minimizing the selected error function. Often used when training MLP networks are the sum of squared errors, taking the following form:

$$SOS_{dataset}=E(w)=\frac{1}{2}\sum_{d=1}^{m}\left(t^{d}-o^{d}\right)^{2}, \tag{5}$$

where: $SOS_{dataset}$ is the error calculated on the analysed dataset; m is the number of observations within the dataset; t^{d} is the target value for observation d; and o^{d} is the output of the network for observation d.

3.5. Training algorithms

The way in which the network weights are adapted for meeting the desired purpose defines the training algorithm and is essentially an optimization problem. When learning with ANNs, the optimization problem becomes the minimization of the error function *E(w)*. For networks having more than one layer of weights, there may be many local minima points for which the gradient of the weights space satisfies the condition $\frac{\partial E}{\partial w}=0$. Therefore, in search of that global minimum point for which the error function has the lowest value, several adjustments are performed using the formula below:

$$w_{t+1}=w_{t}+\Delta w_{t+1}, \tag{6}$$

where t is the number of the training cycle (epoch).

Local optimization can be divided into the following three classes: non-derivative methods, first derivative (gradient) methods, and second derivative methods. Financial applications mostly use first derivative method gradient descent algorithm to make adjustments in the weights during the training cycles. Nevertheless, the rating of this algorithm is many times shaded by the local minima problem and by a slow convergence process. Second derivative optimization methods use the Hessian matrix to determine the search direction. Examples of second derivative methods are discrete Newton, quasi-Newton, and Levenberg-Marquardt. Newton's methods assume that the objective function can be locally approximated as a quadratic around the optimum, and uses the first and second derivatives to find the stationary point. In quasi-Newton methods the Hessian matrix of second derivatives of the function to be minimized does not need to be computed at any stage. The Hessian is updated by analysing successive gradient vectors instead.

Past studies (Ruxanda, Smeureanu 2012; Antucheviciene *et al.* 2012; Dadelo *et al.* 2012) indicate that decision making is mostly about finding preferable solution, within an acceptable decision time and with a bearable error level. This is where optimization algorithms play an important role as they offer a solution for the trade-off between decision process time and results. The optimization algorithms presented below will be further analysed in respect with their errors when performing stock market predictions.

3.5.1. Gradient descent

First introduced by Rumelhart *et al.* (1986), back-propagation algorithm using gradient descent technique is a first derivative method which uses gradient information calculated from the optimization function to determine the search direction on the response surface (Ruxanda 2010). Given a three layer MLP (one input layer, one hidden layer and one output layer), the training process within the back-propagation algorithm is described below.

The network weights are initially set to small random values. Afterwards, the input model is applied and propagated through the network generating outputs:

$$h_j = f(net_j) = f(\sum_k w_{jk} x_k), \tag{7}$$

where: h_j is the output of the hidden unit j; net_j is the input of the hidden node j; w_{jk} is the weight given to input k for hidden node j; x_k is the input node k; and $f(.)$ is the activation function for the hidden layer.

These outputs are further used as entries for the output layer. Weighted and summed up, they are passed through an activation function in order to produce the final output:

$$o_i = g(net_i) = g(\sum_j w_{ij} h_j) = g(\sum_j w_{ij} f(\sum_k w_{jk} x_k)), \tag{8}$$

where: o_i is the response of the output unit i; net_i is the input of the output node i; w_{ij} is the weight given to the hidden node j for the output node i; and $g(.)$ is the activation function from the output layer.

Considering the form of the error function provided in Equation (5), for p output nodes and m input-output pairs, the error becomes:

$$SOS_{training} = E(w) = \frac{1}{2}\sum_{d=1}^{m}\sum_{i=1}^{p}\left(t_i^d - g(\sum_j w_{ij} f(\sum_k w_{jk} x_k^d))\right)^2 . \quad (9)$$

Then, the errors are passed back through the network using the gradient method by calculating the contribution of each hidden node and deriving the adjustments needed to generate a better output. The gradients for the hidden to output layer, and for the input to hidden layer are presented in Equations (12) and (15) respectively:

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta\sum_{d=1}^{m}\left(t_i^d - o_i^d\right)\cdot g'(net_i^d)\cdot(h_j^d); \quad (10)$$

$$\delta_i^d = g'(net_i^d)(t_i^d - o_i^d); \quad (11)$$

$$\Delta w_{ij} = \eta\sum_{d=1}^{m}\delta_i^d h_j^d; \quad (12)$$

$$\Delta w_{jk} = -\eta\frac{\partial E}{\partial w_{jk}} = -\eta\sum_{d=1}^{m}\frac{\partial E}{\partial h_j^d}\cdot\frac{\partial h_j^d}{\partial w_{jk}} = \eta\sum_{d=1}^{m}\sum_{i=1}^{p}\delta_i^d\cdot w_{ij} f'(net_j^d)\cdot x_k^d; \quad (13)$$

$$\delta_j^d = f'(net_j^d)\sum_{i=1}^{p} w_{ij}\delta_i^d; \quad (14)$$

$$\Delta w_{jk} = \eta\sum_{d=1}^{m}\delta_j^d x_k^d, \quad (15)$$

where η is the learning rate.

The new weights can be adjusted using also the momentum rate, which considers the modifications performed in previous cycles:

$$\Delta w_{t+1} = -\eta\frac{\partial E(w_t)}{\partial w_t} + \alpha\Delta w_t, \quad (16)$$

where: α is the momentum rate; Δw_{t+1} is the weight modification for cycle $t+1$; and Δw_t is the modification in weights from the previous cycle.

The learning rate controls the size of the step from each iteration, and the momentum rate speeds up the convergence process in flat regions, or reduces the jumps in regions with high fluctuations by considering a fraction of the previous weight change. Although very popular in practice, a downside of the gradient descent algorithm is that the learning process is slow and thus, the convergence is highly dependent on the values chosen for the learning and momentum rates.

3.5.2. Broyden-Fletcher-Goldfarb-Shanno

Introduced in 1970 (independently by Broyden (1970); Fletcher (1970); Goldfarb (1970); Shanno (1970)), Broyden-Fletcher-Goldfarb-Shanno is a quasi-Newton optimization method

which provides good convergence. Although second derivative algorithms usually require more computational resources, BFGS algorithm uses only an approximation and not the fully explicit calculation of the Hessian inverse matrix, based on estimations obtained only from first order information.

In case of BFGS algorithm the necessary condition for optimality is the minimization of the error function $E(w)$. The weights adjustments when using BFGS training algorithm are performed in an iterative manner, as follows:

$$\Delta w_{t+1} = w_{t+1} - w_t = -\eta H_t^{-1} \frac{\partial E(w_t)}{\partial w_t}, \tag{17}$$

where: t indicates the training cycle; and H_t^{-1} is an approximation of the Hessian inverse matrix $[\partial^2 E(w_t)]^{-1}$ at time point t.

Quasi-Newton methods require that the approximation of matrix H_{t+1}^{-1} satisfies the condition $H_{t+1}^{-1}\gamma_t = \delta_t$. The approximation of the Hessian inverse matrix used by BFGS algorithm is provided in the equation below:

$$H_{t+1}^{-1} = H_t^{-1} - \frac{\delta_t \gamma_t^T H_t^{-1} + H_t^{-1}\gamma_t \delta_t^T}{\delta_t^T \gamma_t} + \left(1 + \frac{\gamma_t^T H_t^{-1} \gamma_t}{\delta_t^T \gamma_t}\right) \cdot \frac{\delta_t \delta_t^T}{\delta_t^T \gamma_t}, \tag{18}$$

where: $\delta_t = w_{t+1} - w_t$ and $\gamma_t = \frac{\partial E(w_{t+1})}{\partial w_{t+1}} - \frac{\partial E(w_t)}{\partial w_t}$.

The initial value of H_0^{-1} is the identity matrix. The adjusting process is performed until a stopping criterion is met such as verifying the performance of the training process on an additional validation data set which prevents the over-fitting phenomenon from affecting the model's performance on new data.

Although in the specialized literature many other training algorithms and derivations from these are proposed (Cocianu, State 2013), in this paper, gradient descent method and BFGS training algorithm were used to make predictions on BET index values and were compared in terms of estimation errors.

3.6. Stopping conditions

With non-linear optimization algorithms it is important to choose certain stopping rules. Bishop (1995) presents five types of stopping criteria which refer to: performing a number of cycles, a certain time has elapsed, the error function has decreased below a certain value, the relative change in error is below a threshold, or the error calculated on an independent dataset (validation set) has started to increase, meaning that there is a risk of over-fitting the model. In this paper, the training process was set to stop when one of the following events is first reached: 500 cycles, or a variation of the average error for 20 consecutive epochs below 0.0000001. The error function is the sum of squared errors between the network outputs and the target values, computed as in Equation (5). The evolution of the error function on the training set was also compared with the one from the validation dataset in order to make sure that additional decreases in the training error (training SOS) don't bring increases in the validation set error (validation SOS).

3.7. An algorithm for numerical differentiation

The most important aspect related to learning algorithms which use gradient and Hessian matrix information, is based on the numerical evaluation of the first order and second order derivatives. The success of ANNs training process is strictly related to the method used to perform the numerical evaluation of the derivatives. An efficient numerical differentiation algorithm was proposed by Professor Gheorghe RUXANDA in the context of developing a language for analysis and prediction – EMI. The algorithm is based on determining an optimal variation of the argument for which the differentiation of a real function is performed, and allows the estimation of first order and second order derivatives (simple and mixed) with a high precision rate. Below is presented the description of the algorithm written in pseudo-code:

```
algorithm deriv;
external
    function f(x), x;
const
    cmin=$MinMachineNumber,cmax=$MaxMachineNumber,cprec=$MachinePrecision;
    climvs=cmin 10^7, climrs=cmin 10^16, cunit=1.0, cvd=1.005;
    precmax=1.5*10^(-IntegerPart(cprec)), precwrk=0.75*10^(-IntegerPart(cprec-10.5));
begin
    rs=cmin, rd=cunit, xabs=abs(x);
    if xabs > climrs then rs=precmax*xabs endif;
    if xabs > cunit then rd=xabs/precwrk endif;
    val=f(x), vs= abs(val);
    if vs < climvs then vs=cunit endif;
    vs=precwrk*vs, vd=cvd*vs, limps=rs, limpd=rd;
    sign=1.0, iter=1, cont=1;
    while (cvd*limps <= limpd) && (iter <= maxiter) && (cont ==1)
        p=sqrt(limps*limpd), delt=f(x+sign*p) -val;
        if abs(delt) <= vs then
                limps=p;
        else
                if abs(delt) >= vd then limpd=p, cont=0 endif;
        endif;
        if (delt == 0.0) && (iter > 2) then cont=0, break endif;
        iter++;
        if (iter > maxiter) && (cont == 1) && (sign == 1) then
                sign=–1, iter=0, limps=rs, limpd=rd ;
        endif;
        if cont == 0 then
                if sign == 1 then
                        vder=(f(x+p) – f(x-p))/(2.0*p);
                else
                        vder=(val-f(x-p) /p, p=sign*p;
```

```
            endif;
            vder2=(f(x+2*p) - 2*f(x) + f(x-2*p))/(4.0*p*p);
        endif;
    endwhile;
    return cont, vder, vder2;
end.
```

The above pseudo-code of the algorithm is applicable for the calculation of first order and second order derivatives of a single-variable function. Nevertheless, the algorithm can easily be adapted for the evaluation of multi-variable functions. Tested on several classes of functions, the proposed algorithm has revealed an average precision of 1.0e-9 for first order derivatives, and an average precision of 1.0e-5 for second order derivatives. The obtained average number of iterations needed to determine the optimal variation of the argument used for differential evaluation equals 12.

4. Results

The model building consisted of generating 100 different networks from combining the following elements:

- The number of hidden nodes, which varied from 2 to 6;
- The types of activation functions from the hidden layer: logistic sigmoid and hyperbolic tangent sigmoid;
- The training algorithms: gradient descent (GD) and BFGS.

For each distinct configured building-block, five networks were trained for results consistency in which the initial weights were different (initial weights were picked using a normal distribution of mean 0 and variance 0.1). For the gradient descent method, the learning rate and the momentum parameters were set to 0.1 each. For each type of training algorithm analysed in this paper, the best five networks in terms of sum of squared error on the test sample (test SOS) were retained. The test sample, acting as a totally independent dataset, gives indications on the model predictive power on new information. Table 3 provides the errors reached by the best five networks for each learning algorithm. Values indicate that BFGS provides more accurate predictions on all three datasets.

Table 3. Neural Networks' results on Romanian data

Network name	Training SOS	Validation SOS	Test SOS	Training algorithm and No. of cycles	Hidden layer activation function
MLP 3-4-1 BFGS 7 T	3.225E-01	2.345E-02	8.120E-03	BFGS 7	Tanh
MLP 3-5-1 BFGS 5 T	3.226E-01	2.342E-02	8.130E-03	BFGS 5	Tanh
MLP 3-3-1 BFGS 28 L	3.191E-01	2.319E-02	8.140E-03	BFGS 28	Logistic
MLP 3-2-1 BFGS 5 L	3.225E-01	2.345E-02	8.140E-03	BFGS 5	Logistic
MLP 3-3-1 BFGS 7 T	3.224E-01	2.348E-02	8.150E-03	BFGS 7	Tanh
MLP 3-2-1 GD 3 T	3.932E-01	3.142E-02	1.908E-02	GD 3	Tanh

Continued Table 3

Network name	Training SOS	Validation SOS	Test SOS	Training algorithm and No. of cycles	Hidden layer activation function
MLP 3-3-1 GD 3 T	3.933E-01	3.144E-02	1.910E-02	GD 3	Tanh
MLP 3-2-1 GD 4 T	3.938E-01	3.150E-02	1.916E-02	GD 4	Tanh
MLP 3-5-1 GD 7 T	3.938E-01	3.149E-02	1.917E-02	GD 7	Tanh
MLP 3-2-1 GD 5 T	3.946E-01	3.159E-02	1.926E-02	GD 5	Tanh
In case of all networks, linear activation function was used in the output layer					

The lowest error is achieved by the model *MLP 3-4-1 BFGS 7 T* which uses BFGS training algorithm, four hidden nodes and hyperbolic tangent function in the hidden layer. This network is obtained after performing seven epochs of weights adjustments and generates a test SOS that is by 57% lower compared with the best network using gradient descent algorithm.

Considering the overall ten best networks available in Table 3 and the activation functions used in the hidden layer, hyperbolic tangent sigmoid function performs better on the Romanian Stock Market data. Regarding another ranged element, the number of hidden nodes, results showed that networks using the maximum selected number of six hidden nodes are nowhere in the best five most performing models in this experiment. Therefore, this proves that there is no need of including too many hidden nodes in the network, as this will result only into increased training time and complexity, and not into improved outcomes.

Evaluating the results obtained from applying these ten best networks on the Croatian data (Table 4), we observe that the activation function that reached the lowest error is logistic sigmoid this time. Nevertheless, the best network using BFGS training algorithm, *MLP 3-3-1 BFGS 28 L*, has reached an error that is by 69% lower compared with the one generated when applying the best network which uses the gradient descent method, *MLP 3-2-1 GD 3 T*. This gives us the reason to state that BFGS learning algorithm is a better option when modelling volatile data such as stock market values.

Table 4. Results on Croatian data

Network name	SOS for Croatian data
MLP 3-3-1 BFGS 28 L	4.572E-03
MLP 3-5-1 BFGS 5 T	4.573E-03
MLP 3-2-1 BFGS 5 L	4.580E-03
MLP 3-3-1 BFGS 7 T	4.587E-03
MLP 3-4-1 BFGS 7 T	4.605E-03
MLP 3-2-1 GD 3 T	1.473E-02
MLP 3-3-1 GD 3 T	1.476E-02
MLP 3-2-1 GD 4 T	1.481E-02
MLP 3-2-1 GD 5 T	1.482E-02
MLP 3-5-1 GD 7 T	1.490E-02

Conclusions

Although Artificial Neural Networks are flexible non-parametric methods that perform well on non-linear data series, their predictive power is conditioned by a set of elements which define the network configuration features. When building a predictive model for stock market prices with ANNs, it is important that the modeller selects the proper values for items like: the number of input and output variables, the number of hidden layers and hidden nodes, the activation functions in the hidden and output layers, the initial weights, the training algorithm, and the stopping criteria. Similar to Coupelon (2007) remarks, we can state that in most of the cases there are no values for these elements to be considered as best choices when forecasting stock exchange prices, the selection process being based on performing several experiments and choosing the network that offers the best results in terms of a performance metric. However, in respect with the training algorithm, this study has given evidence that BFGS outperforms the classical gradient descent method, providing lower errors even in the context of highly volatile data such as the one revealed by the stock exchange market.

Idowu *et al*. (2012) pointed out that although ANNs do not allow perfect estimations on volatile data such as the stock exchange market, they certainly provide closer results to the real ones compared with other techniques. In the present study, estimation results have indeed revealed small errors on the test datasets. Thus, Artificial Neural Networks can be used in an efficient manner to forecast stock market prices based on past observations. Therefore, we can affirm that EMH theory stating that stock prices cannot be predicted based on past values, can be rejected.

Further steps and research directions regarding the evaluation of ANNs in stock market predictions should consider the followings:

- Analyse how results differ when performing random partitioning for selecting the cases for training and validation datasets;
- Include more predictors to estimate stock exchange market, such as international stock exchange market index or qualitative factors;
- Using other training algorithms such as Levenberg-Marquardt (Zayani *et al*. 2008) which gives an optimized approach to local minima problem.

References

Abbé Sauri, M. 1774. *Cours complet de mathématiques*. A Paris, Aux dépens de Ruault. 656 p.

Antucheviciene, J.; Zavadskas, E. K.; Zakarevicius, A. 2012. Ranking redevelopment decisions of derelict buildings and analysis of ranking results, *Economic Computation and Economic Cybernetics Studies and Research* 46(2): 37–62.

Bishop, C. 1995. *Neural Networks for pattern recognition*. Oxford: Clarendon Press. 482 p.

Broyden, C. 1970. The convergence of a class of double-rank minimization algorithms, *Journal of Institute Mathematical Applications* 6(1): 76–90. http://dx.doi.org/10.1093/imamat/6.1.76

Cocianu, C.; State, L. 2013. Kernel-based methods for learning non-linear SVM, *Economic Computation and Economic Cybernetics Studies and Research* 47(1): 41–60.

Coupelon, O. 2007. *Neural network modeling for stock movement prediction: a state of the art*. Blaise Pascal University. 5 p.

Dadelo, S.; Turskis, Z.; Zavadskas, E. K.; Dadeliene, R. 2012. Multiple criteria assessment of elite security personal on the basis of ARAS and expert methods, *Economic Computation and Economic Cybernetics Studies and Research* 46(4): 65–88.

Egeli, B.; Ozturan, M.; Badur, B. 2003. Stock market prediction using Artificial Neural Networks, in *Proceedings of the 3rd Hawaii International Conference on Business*, 26–28 July, 2012, Honolulu, Hawaii, USA. 8 p.

Elliott, D. L. 1993. *A better activation function for Artificial Neural Networks*. Institute for Systems Research, University of Maryland.

Faria, E. L.; Albuquerque, M. P.; Gonzalez, J. L.; Cavalcante, J. T. P.; Albuquerque Marcio, P. 2009. Predicting the Brazilian stock market through Neural Networks and adaptive exponential smoothing methods, *Expert Systems with Applications* 36(10): 12506–12509. http://dx.doi.org/10.1016/j.eswa.2009.04.032

Fletcher, R. 1970. A new approach to variable metric algorithms, *Computer Journal* 13(3): 317–322. http://dx.doi.org/10.1093/comjnl/13.3.317

Georgescu, V. 2011. An econometric insight into predicting Bucharest stock exchange mean, return and volatility – return processes, *Economic Computation and Economic Cybernetics Studies and Research* 45(3): 25–42.

Georgescu, V. 2010. Robustly forecasting the Bucharest stock exchange BET index through a novel computational intelligence approach, *Economic Computation and Economic Cybernetics Studies and Research* 44(3): 23–42.

Goldfarb, D. 1970. A family of variable-metric methods derived by variational means, *Mathematical Computations* 24: 23–26. http://dx.doi.org/10.1090/S0025-5718-1970-0258249-6

Idowu, P. A.; Osakwe, C.; Kayode, A. A.; Adagunodo, E. R. 2012. Prediction of stock market in Nigeria using artificial neural network, *International Journal of Intelligent Systems and Applications* 4(11): 68–74. http://dx.doi.org/10.5815/ijisa.2012.11.08

Iordache, A. M.; Spircu, L. 2011. Using Neural Networks in ratings, *Economic Computation and Economic Cybernetics Studies and Research* 45(3): 101–112.

Isfan, M.; Menezes, R.; Mendes, D. A. 2010. Forecasting the Portuguese stock market time series by using Artificial Neural Networks, *Journal of Physics: Conference Series* 221(1): 13 p.

Jha, G. K. 2009. *Artificial Neural Networks*. Indian Agricultural Research Institute, PUSA, New Delhi. 8 p.

Kaastra, I.; Boyd, M. S. 1995. Forecasting futures trading-volume using Neural Networks, *The Journal of Futures Markets* 15(8): 953–970. http://dx.doi.org/10.1002/fut.3990150806

Khan, A. U.; Gour, B. 2013. Stock Market trends prediction using neural network based hybrid model, *International Journal of Computer Science Engineering and Information Technology Research* 3(1): 11–18.

Minsky, M.; Papert, S. 1969. *Perceptrons*. Cambridge: MIT Press. 81 p.

Rumelhart, D.; Hinton, G.; Williams, R. 1986. Learning representations by backpropagation errors, *Nature* 323: 533–536. http://dx.doi.org/10.1038/323533a0

Ruxanda, G.; Smeureanu, I. 2012. Unsupervised learning with expected maximization algorithm, *Economic Computation and Economic Cybernetics Studies and Research* 46(1): 28 p.

Ruxanda, G. 2010. Learning perceptron neural network with backpropagation algorithm, *Economic Computation and Economic Cybernetics Studies and Research* 44(4): 37–54.

Shanno, D. 1970. Conditioning of quasi-Newton methods for function minimization, *Mathematical Computations* 24: 647–656. http://dx.doi.org/10.1090/S0025-5718-1970-0274030-6

Sibi, P.; Allwyn Jones, S.; Siddarth, P. 2013. Analysis of different activation functions using Backpropagation Neural Networks, *Journal of Theoretical and Applied Information Technology* 17(3): 1264–1268.

Thenmozhi, M. 2006. Forecasting stock index returns using Neural Networks, *Delhi Business Review* 7(2): 59–69.

Ungureanu, E.; Burcea, F.-C.; Pîrvu, D. 2011. The analysis of interest rate and exchange rate influence's on stock market. Medium run evidence from Romania, *Annals Economic Science Series* 17: 163–170.

Vahedi, A. 2012. The predicting stock price using artificial neural network, *Journal of Basic and Applied Scientific Research* 2(3): 2325–2328.

Verhulst, P. F. 1845. Recherches mathematiques sur la loi d'accroissement de la population, *Nouveau Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles* 18: 41 p.

Zayani, R.; Bouallegue, R.; Roviras, D. 2008. Levenberg-Marquardt learning neural network for adaptive predistortion for time-varying HPA with memory, in *OFDM Systems: 16th European Signal Processing Conference (EUSIPCO 2008)*, 25–29 August, 2008, Lausanne, Switzerland.

Zoicas, I. A.; Făt, M. C. 2005. The analysis of the relation between the evolution of the Bet Index and the main macroeconomic variables in Romania (1997–2008), *Annals of the University of Oradea: Economic Science* 3(1): 632–637.

Gheorghe RUXANDA. PhD in Economic Cybernetics, Editor-in-chief of ISI Thompson Reuters Journal "Economic Computation and Economic Cybernetics Studies and Research" and Director of Doctoral School of Economic Cybernetics and Statistics. Is a Full Professor and PhD Adviser within the Department of Economic Informatics and Cybernetics, The Bucharest Academy of Economic Studies. He graduated from the Faculty of Economic Cybernetics, Statistics and Informatics, Academy of Economic Studies, Bucharest (1975) where he also earned his Doctor's Degree (1994). Had numerous research visits in USA, England and France. He is a Full Professor of Multidimensional Data Analysis (Doctoral School), Data Mining and Multidimensional Data Analysis (Master Studies), Modeling and Neural Calculation (Master Studies), Econometrics and Data Analysis (Undergraduate Studies). Scientific research activity: over 35 years of scientific research in both theory and practice of quantitative economy and in coordinating research projects; 50 scientific papers presented at national and international scientific sessions and symposia; 65 scientific research projects with national and international financing; 79 scientific papers published in prestigious national and international journals in the field of economic cybernetics, econometrics, multidimensional data analysis, microeconomics, scientific informatics, out of which eleven papers being published in ISI – Thompson Reuters journals; 18 manuals and university courses in the field of econometrics, multidimensional data analysis, microeconomics, scientific informatics; 31 studies of national public interest developed within the scientific research projects. Fields of scientific competence: evaluation, measurement, quantification, analysis and prediction in the economic field; econometrics and statistical-mathematical modelling in the economic–financial field; multidimensional statistics and multidimensional data analysis; pattern recognition, learning machines and Neural Networks; risk analysis and uncertainty in economics; development of software instruments for economic-mathematical modelling.

Laura Maria BADEA is a PhD candidate in Economic Cybernetics at the Bucharest Academy of Economic Studies, has an MA in Corporate Finance (2010) and graduated the Faculty of Finance, Insurance, Banking and Stock Exchange from Bucharest University of Economic Studies (2008). Scientific research activity: 2 published articles in ISI Thompson Reuters Journals. Fields of scientific interest: machine learning and other modelling techniques used for classification matters in economic and financial domains, with a focus on Artificial Neural Networks.