European Research Studies,
Volume XV, Issue (2), 2012

Time Series Prediction with Neural Networks for the Athens Stock Exchange Indicator

M. Hanias[1], P. Curtis[2], E. Thalassinos[3]

Abstract:

The main aim of this study is to predict the daily stock exchange price index of the Athens Stock Exchange (ASE) using back propagation neural networks. We construct the neural network based on the minimum embedding dimension of the corresponding strange attractor. Multistep prediction for nine days ahead is achieved with this particular network indicating the increased possibility of this technique for immediate forecasts for very time-short data sets, mostly daily and weekly.

Key Words: *Time Series Forecasting, Neural Networks, Perceptions, Neuron*

JEL Classification*: C02, C22, C69*

[1] *TEI Chalkidas, Athens,Greece.*
[2] *TEI Chalkidas, Athens, Greece, email: pcurtis@teihal.gr*
[3] *Professor, Department of Maritime Studies, University of Piraeus,Chair Jean Monnet, email: thalassi@unipi.gr*

1. Introduction

The common characteristic of all stock exchange indicators have the term of uncertainty, which is related with their short and long-term future state. This feature is undesirable for the investor but it is also unavoidable whenever the stock exchange indicator is selected as an investment tool. Among the best possible alterations that the researcher desires to make is to reduce this uncertainty. Stock exchange prediction or forecasting is one of the interesting issues in this process as it is also referred in other works (Helstrom T. & Holmstrom K., 1998, Tsibouris G. & Zeidenberg M., 1996, White H., 1993).

Time series forecasting, or time series prediction, takes an existing series of data $x_{t-n},\ldots,x_{t-2},x_{t-1},x_t$ and forecasts the $x_{t+1},x_{t+2},\cdots$ data values. The goal is to observe or model the existing data series to enable future unknown data values to be forecasted accurately. Examples of data series include financial data series (stocks, indices, rates, etc.), physically observed data series (sunspots, weather, etc.), and mathematical data series (Fibonacci sequence, integrals of differential equations, etc.).

A neural network (Tsibouris G. & Zeidenberg M., 1996) is a computational model that is loosely based on the neuron cell structure of the biological nervous system. Given a training set of data, the neural network can learn the data with a learning algorithm; in this research the most common algorithm, back propagation, is used. Through back propagation, the neural network forms a mapping between inputs and desired outputs from the training set by altering weighted connections within the network. The origin of neural networks dates back to the 1940s.

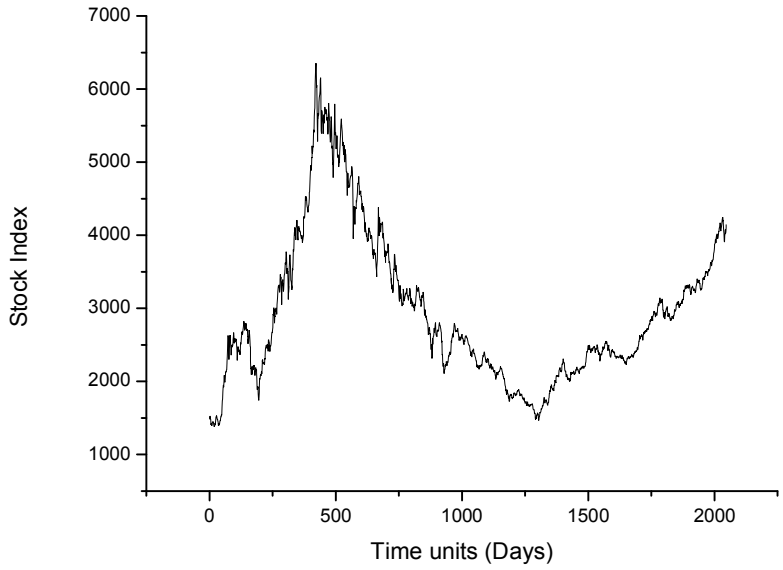
McCulloch and Pitts (1943) and Hebb (1949) researched networks of simple computing devices that could model neurological activity and learning within these networks, respectively. Later, the work of Rosenblatt (1962) focused on computational ability in perceptrons, or single-layer feed-forward networks. Proofs showing that perceptrons, trained with the Perceptron Rule on linearly separable pattern class data, could correctly separate the classes generated excitement among researchers and practitioners.

This excitement waned with the discouraging analysis of perceptrons presented by Minsky and Papert (1969). The analysis pointed out that perceptrons could not learn the class of linearly inseparable functions. It also stated that the limitations could be resolved if networks contained more than one layer, however no effective training algorithm for multi-layer networks was available. Rumelhart, Hinton, and Williams (1986) revived interest in neural networks by introducing the generalized delta rule for learning by back propagation, which is today the most commonly used training algorithm for multi-layer networks.

2. Athens Stock Exchange Indicator Time Series

The Athens Stock Exchange Indicator is presented as a signal x=x(t) as it shown in Figure 1. It covers data from the period 1998 to 2005. The sampling rate is Δt=1 day.

Figure 1. Time Series Data for the Athens Stock Exchange Indicator



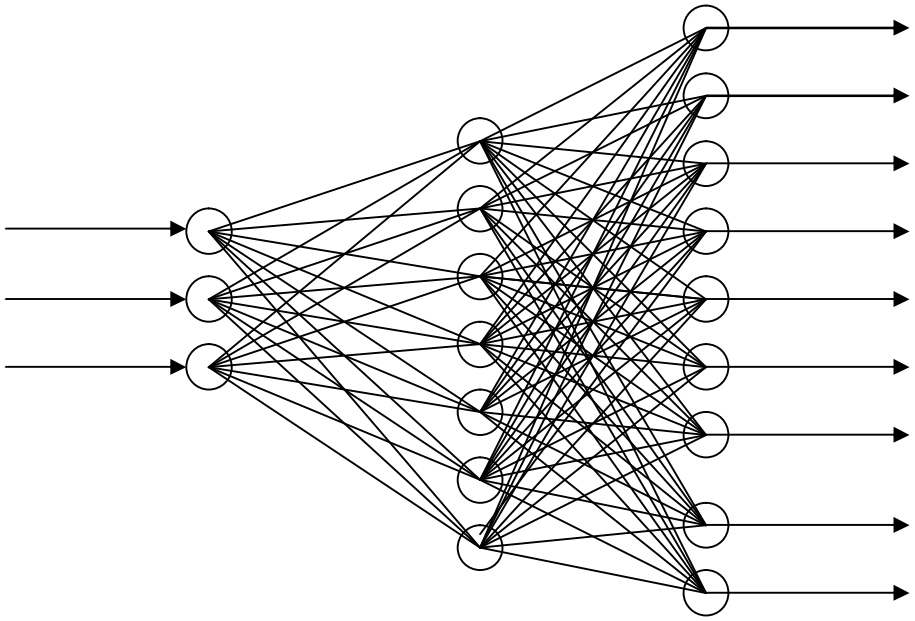
3. Neural Network Constructions

In order to predict the time series for the Athens Stock Exchange Indicator we construct a back propagation network (Wan A.E., 1990, Widrow B., and Lehr M., 1990, Rumelhart D., McClelland J. et al., 1986) that consists of 1 input layer 1 middle or hidden layer and 1 output layer. The input layer has 3 processing elements (PE) neurons; the middle layer has 2x3+1=7 neurons according to Kolmogorov theorem (Kolmogorov N. A., 1950, Blackwell D., 1947).

We choose the input layer to have 3 inputs because of minimum embedding dimension 3, and the attractor is embedded at a three –dimensional phase space (Hanias P.M., Curtis G.P., and Thalassinos E.J., 2006). Beginning with the first value of the time series, the first set of inputs data is x1, x2, x3 and our outputs are x4, x5, x6, x7, x8, x9, x10, x11, x12. Similarly, the second set of inputs is x2, x3, x4 , and the outputs we are trying to predict are x5, x6, x7, x8, x9, x10, x11, x12, x13. So we construct the network having an input layer of 3 neurons. The output layer has 9 neurons. In other words with every 3 input values of stock market we can

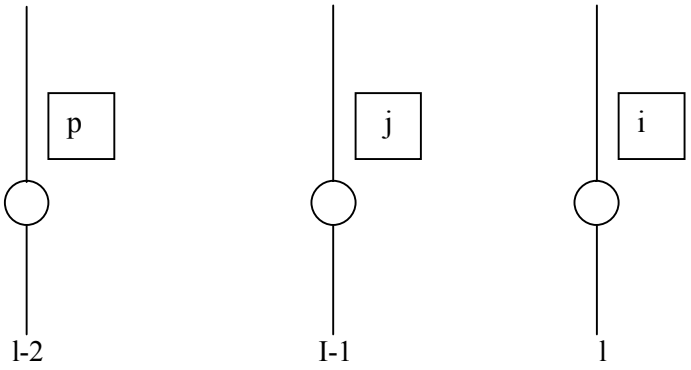
predict 9 steps ahead. The network architecture is shown in Figure 2. There is a full interconnection between each layer.

Figure 2. Network Architecture with one Input Layer one Hidden Layer and one Output Layer



As it is shown in Figure 2 this neural network is composed of a layered arrangement of artificial neurons in which each neuron of a given layer feeds the neurons of the next layer. Singles neurons extracted from the corresponding layers are represented in Figure 3.

Figure 3. Singles Neurons Extracted from the Input, Hidden and Output Layer



In Figure 3 there is a neuron p at input layer l-2, a neuron j at middle layer l-1, a neuron i at output layer i. If m is the number of neurons in layer l, in our case m=9, n is the number of neurons in layer l-1, in our case n=7, and N is the number of examples, in our case N=1024, while t is an index that points to the corresponding case. We denote yli(t) as the output of the ith neuron of the layer l and y(l-1)j(t) as the output of the jth neuron of the middle layer l-1 while Ti(t) is the desired case.

The inputs y(l-1)j(t) to the neuron i are multiplied by variable coefficients wlji called weights which represent the synaptic connectivity between neuron j in the previous layer and neuron i in layer l.

At output layer l the input at neuron i is

$$I_{li}(t) = \sum_{j=1}^{m} w_{lji} y_{(l-1)j}(t) \qquad (1)$$

while the output of neuron i is

$$y_{li}(t) = \tanh(I_{li}(t)) \qquad (2)$$

where tanh is the hyperbolic tangent function.

Simplistically we can say that the output of a neuron is a tanh function of the weighted sum of its inputs.

The error δlj(t) for neuron i of output layer l, for the case t is

$$\delta_{li}(t) = T_i(t) - y_{li}(t) \qquad (3)$$

where yli(t) is the predicted output while Ti(t) is the desired output.

Using a back propagation network we propagate backwards the errors for each output layer neuroses to the middle layer, using the same interconnection and weights as the middle layer use to transmit their outputs to the output layer. Then we can compute a gain error for each neuron in the middle layer, based on its portion of the blame for the output layer's error.

The gain error OE for neuron i of output layer l is as it is pointed out in Wan A.E., (1994):

$$OEli(t)=\delta li(t)yli(t)[1-yli(t)] \qquad (4) \text{ and}$$

$$\frac{\partial E(t)}{\partial w_{lji}} = -OE_{li}(t) y_{(l-1)}(t) \qquad (5) \text{ and finally}$$

$$w_{lji}^{new} = w_{lji}^{old} - \beta \frac{\partial E(t)}{\partial w_{lji}} \quad (6) \text{ or}$$

$$w_{lji}^{new} = w_{lji}^{old} + \beta OE_{li}(t) y_{(l-1)}(t) \quad (7)$$

is a weight change algorithm called generalized delta rule (Danilo P., Mandic J., and Chambers A., 2001).

The notation w_{lji}^{old} represents the weight vector before and w_{lji}^{new} after the weight has been adjusted. This rule is called the Widrow –Hoff training rule, or the Least Mean Squared Rule. What this law attempts to do is to ensure that the aggregate statistical Least Mean Square Error is achieved in the network. The constant β is the learning constant.

However, we used a more robust learning rule as the Generalized Delta Rule with momentum term α. The new statement of the Delta Rule is:

$$W_{lji}^{new}(t) = w_{lji}^{old} + \beta OE_{li}(t) y_{(l-1)}(t) + \alpha(w_{lji}^{new}(t-1) - w_{lji}^{old}(t-1)) \quad (8)$$

The momentum term α is simply a constant α, times the weight vector of this neurode from the previous presentation of the input pattern. So, if the last weight change was in a particular direction the momentum term tends to make the next weight change, more or less, into the same direction.

Applying the same procedure to middle layer we have the following:

At middle layer l-1 the input at neuron j is:

$$I_{(l-1)j} = \sum_{p=1}^{n} w_{(l-1)pj} y_{(l-2)p}(t) \quad (9)$$

while the output from neuron j is

$$y_{(l-1)j}(t) = \tanh(I_{(l-i)j}(t)) \quad (10)$$

the error δ(l-1)j(t) for neuron j of middle layer l-1, for case t is

$$\delta_{(l-1)j} = \sum_{j=1}^{m} w_{lji} OE_{li}(t) \quad (11)$$

The gain error ME for neuron j of middle layer l-1 is:

$$ME(l\text{-}1)j(t)=\delta(l\text{-}1)j(t)y(l\text{-}1)j(t)[1\text{-}y(l\text{-}1)j(t)] \quad (12)$$

$$\frac{\partial E(t)}{\partial w_{(l-1)pj}} = -ME_{(l-1)j}(t)y_{(l-2)p}(t) \quad (13)$$

and the Delta rule is

$$w^{new}_{(l-1)pj} = w^{old}_{(l-1)pj} - \beta \frac{\partial E(t)}{\partial w_{(l-1)pj}} \quad (14) \text{ or}$$

$$w^{new}_{(l-1)pj} = w^{old}_{(l-1)pj} + \beta ME_{(l-1)j}(t)y_{(l-2)p}(t) \quad (15)$$

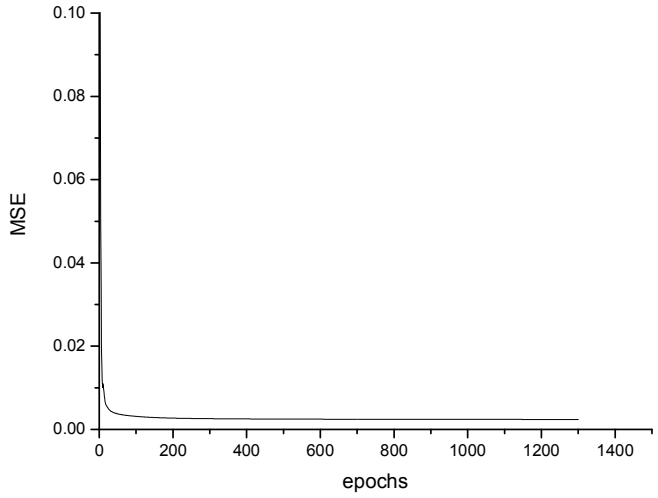
Therefore the generalized Delta Rule is:

$$w^{new}_{(l-1)pj}(t) = w^{old}_{(l-1)pj}(t) + \beta ME_{(l-1)j}(t)y_{(l-2)p}(t) + \alpha(w^{new}_{(l-1)pj}(t-1) - w^{old}_{(l-1)pj}(t)) \quad (16)$$

4. Time Series Prediction

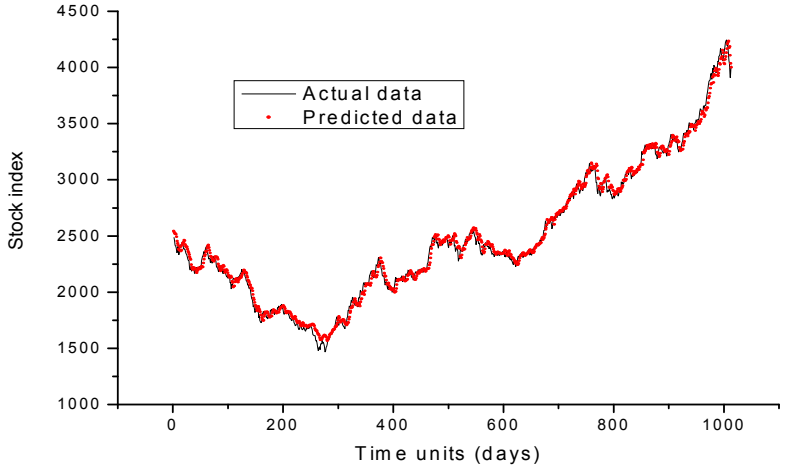
In this section we construct the neural network of Figure 2. The input layer has 3 neurons the hidden layer has 7 neurons and the output layer has 9 neurons. The time series has 2047 points. We train the network with a training set of 1024 exemplars, the learning rate β=1 and the momentum α=0.7. The network was training so the Mean Square Error (MSE) to be less than 0.0025. After 1300 epochs the minimum training was, MSE=0.0024. Figure 4 presents the dependence of MSE on epochs. As it is shown in Figure 4 there is a rapid convergence.

Figure 4. MSE Versus Epochs at the Training Data Set



Then we can test the network on the last 991 data values that the network has never seen before. For each 3 inputs we can predict the next 9 values as it is shown in Figure 5.

Figure 5. Actual and Predicted Time Series for 9 Time Steps Ahead



At it is shown in Figure 5 there is a very good fit between the actual and the predicted data.

5. Conclusions

In this paper we have shown that a back propagation neural network can predict in a very well manner the Athens Stock Exchange Indicator. Using three days as inputs we predicted nine days ahead in the future. The actual and the predicted data are very close with a minimum Mean Square Error (MSE) = 0.0024 which is an acceptable error for this kind of data.

References

1. Blackwell D., (1947), "Conditional Expectation and Unbiased Sequential Estimation", *Ann. Math. Stat., 18 pp. 105–110.*
2. Danilo P., Mandic, J., and Chambers A., (2001), Recurrent Neural Networks for Prediction, John Wiley & Sons Ltd, London.
3. Hanias P.M., Curtis G.P., Thalassinos E.J., (2006), "Non-Linear Dynamics and Chaos: The Case of the Price Indicator at the Athens Stock Exchange", *European Research Studies Vol. VII issue (3-4) and Eastern Economic Association Annual Conference, February 2007, New York.*
4. Helstrom T., Holmstrom K., (1998), Predicting the Stock Market, published as Opuscula ISRN HEV-BIB-OP-26-SE.
5. Kolmogorov N.A., (1950), "Unbiased Estimates", *Izv. Akad. Nauk SSSR Ser. Mat.,14: 4 pp. 303–326 (In Russian).*
6. Maddala G.S., (1992), Introduction to Econometrics, New York, Toronto: Macmillan Publishing Company.
7. Pesaran H.M., Timmermann A., (1994), "Forecasting Stock Returns: An Examination of Stock Market Trading in the Presence of Transaction Costs", *Journal of Forecasting, Vol. 13, pp 335-367.*
8. Rumelhart D., McClelland J, et al, (1986), Parallel Distributed Processing, Cambridge, MA MIT Press.
9. Thalassinos E., Thalassinos J., (2004), "European Stock Markets: Integration Analysis", *Eastern Economic Association Conference, February 2007, New York.*
10. Thalassinos E., Thalassinos P., (2005), "Stock Market Analysis", *European Research Studies Journal, Vol., IX issue (1-2), and International Conference in Finance", Istanbul, 2005.*
11. Tsibouris G., Zeidenberg M., (1996), "Testing the Efficient Market Hypothesis with Gradient Descent Algorithms", in Reffenes A.P., Neural Networks in the Capital Markets, England: John Wiley & Sons Ltd, pp 127-136.
12. Wan A.E., (1990), "Temporal Backpropagation: An Efficient Algorithm for Finite Impulse Response Neural Networks", *Proceedings of the 1990 Connectionist Models, Summer School, San Mateo, CA.*
13. Wan A. E., (1994), "Time Series Prediction by Using a Connectionist Network with Internal Delay Lines Network with Internal Delay Lines" in Time Series Prediction: Forecasting the Future and Understanding the Past, Eds. by
14. White H., (1993), "Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns", in Trippi R.R., and Turban E., Neural Networks in Finance and Investment, Chicago, Cambridge England: Probus Publishing Company, pp 315-329.
15. Widrow B., Lehr M., (1990), "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", *IEEE Proceedings, Vol. 78, No. 9, pp. 1415–1442.*