FORECASTING OF THE GROSS DOMESTIC PRODUCT IN TURKEY VIA NEURAL NETWORKS USING MACROECONOMIC FACTORS

Elmas Anli AK*
Ilyas AKHİSAR**

Abstract

This paper examines whether macroeconomic factors such as deposit interest, domestic credit volume, Istanbul Stock Exchange (ISE-100) index, capacity utilization rate, net international reserve, manufacturer price index, USD rate, M1, M2, M2Y, export, import, domestic debt stock and foreign debt stock might be used in forecasting the gross domestic product (GDP) in Turkey. These 14 macroeconomic factors belonging to the 168 monthly periods between the years 1991 and 2004 have been used to train a back-propagation artificial neural network. A total of 64 networks with different configurations have been tried and the one converging to the least total error has been selected. Monthly data from the year 2005 has been used to test the one-year-ahead forecasting capability of the final artificial neural network.
Key Words: Artificial neural network, back-propagation algorithm, selection of the topology, macroeconomic factors, forecasting capability.

Özet

Bu çalışmada, mevduat faizi, kredi hacmi, İstanbul Menkul Kıymetler Borsası (IMKB-100) endeksi, kapsite kullanım oranı, net uluslararası rezervler, üretici fiyat endeksi, US dolar oranı, M1, M2 M2Y ihracat, ithalat, iç borç stoğu ve dış borç stoğu gibi makro ekonomik faktörler kullanılarak Tükiye' nin gayri safi yurtiçi hasılasının tahmin edilebilmesini araştırıldı. 1991-2004 yılları arasındaki 168 aylık makro ekonomik veri, geri beslemeli yapay sinir ağı ile eğitildi. Toplam 64 farklı yapı

---

* *Araş.Gör,.İstanbul Teknik Üniversitesi Uçak ve Uzay Bilimleri Fakültesi.*
** *Yrd. Doç. Dr., M.Ü. Bankacılık ve Sigortacılık Yüksek Okulu,*

uygulanarak en düşük toplam hataya yakınsayanı seçildi. 2005 yılı aylık datası yapay sinir ağının tahmin kapasitesini test etmek için kullanıldı.
Anahtar Kelimeler: Yapay sinir ağı, Geri beslemeli algoritma, Topoloji seçimi, Makro ekonomik faktörler, Tahmin kapasitesi

Introduction

Neural networks can be used to estimate relations from sample data. Unlike statistical techniques, however, neural systems do not require any mathematical model describing the functional dependency of outputs on the inputs.

In this study, it is desired to forecast the GDP in Turkey one-month-ahead, using a feed-forward back-propagation neural network. The data used in this study covers the period 1991-2005. Monthly data from 1991-2004 has been used in training while the data from they year 2005 has been used to test the results. It is seen that such a one-month-ahead forecasting of the GDP is possible and the results are satisfactory.

This work is organized as follows. Section I gives back-ground information and a literature survey on the subject and data. Section II describes neural networks. Section III describes the factors chosen as input variables, how the network architecture used in this study was chosen and the training procedure. Section IV gives results and comments.

1.1 Literature Survey

Due to the problems and costs that arise in relation to economic crises, forecasting crises plays a crucial role in decreasing costs. Various economic factors are used to monitor crises beforehand. Recently, in addition to econometric forecasting approaches, neural networks have been demonstrated to provide promising results in financial forecasting and trading.

Perez-Rodrigueza and Torrab examined the out of sample forecast performance of smooth transition auto-regression (STAR) models and the results indicate that ANNs provide better forecasts than the linear AR model and the STAR models for some forecast horizons and forecasting methods. (Kim, 2006) has proposed genetic algorithm approach to instance selection in artificial neural networks for financial data mining. (Chiang et al., 2006) have provided empirical evidence on the relationship between country-fund discounts and time varying risk factors by analyzing monthly data of 39 closed-end country funds. (Kiymaz, 2000) has empirically analyzed the initial and after-market returns for the Turkish initial public offerings (IPOs) to provide an emerging market case of international evidence on performances of IPOs. (Muradoglu and Metin, 1996) have tested the semi-strong form of the efficient market hypothesis in Turkey by using the recently developed techniques in time series econometrics, namely unit roots and cointegration. (Hooker, 2004) has investigated the predictive power of several candidate macroeconomic factors for emerging market equity returns using the Bayesian model selection approach. (Yim and Mitchell, 2005) have used hybrid neural networks to predict country risk rating. (Tkacz, 2001) has improved the accuracy of financial and monetary forecasts of Canadian output growth by using leading indicator neural network model and found that neural networks yield statistically lower forecast errors for the year-over-year growth rate of real GDP(Gross Domestic Product) relative to linear and univariate models.

2. Neural Networks as a Multivariable Optimization Tool

Heuristic algorithms are self learning or adaptive processes that seek a solution among all possible solutions but do not guarantee that the optimal solution will be found. This makes heuristic algorithms approximate, rather than exact. Since such algorithms are self-learning, they get better with experience. The two most popular heuristic algorithms are neural networks and genetic algorithms. (Anli, 2005) has described neural networks in detail. Elaborate explanations related to neural networks, back-propagation algorithm and advantages can be found in the mentioned reference. Neural networks are adopted as a problem-solving tool in many branches of engineering. NNs classify input patterns and adapt to pattern populations. They have numerous applications because they replace expensive programming efforts with simple self-training hardware. Their main advantage lies in the fact that they can learn new things without having to be explicitly programmed.

2.1 Back-Propagation Algorithm

In neural network architectural structures, nodes are organized into groups called layers. Input layers receive inputs, output layers produce outputs and internal (or hidden) layers provide the interconnections between input and output. The activity of an artificial neuron consists of two distinct steps;
Step-1: Evaluation of the input signal:

$$input_i^l = \sum_j w_{ij} output_j^{l-1}$$

Step-2: Transformation of the net input signal :

$output_i^l = TF(input_i^l)$ where TF denotes a transfer function applied in the ith node in layer l, j denotes the jth node in layer (l-1) connected to the ith node in layer l via the weight w_{ij}. One of the most popular non-linear transfer function (TF) is called sigmoidal function, defined as

$$output_i^l = [2/(1+\exp(-2(input_i^l))] - 1$$

In a feed-forward NN with the back-propagation learning algorithm, the nodes are organized into layers. They are connected via links and each link is associated with a weight. It is that weight that determines the nature and strength of one node's influence on another. All nodes in one layer have the same input signal simultaneously. Back-propagation neural networks, developed by Rumelhart and others, are currently the most widely used algorithm for learning connections,. Back-propagation is a systematic method for training multilayer neural networks. Training is usually done by iterative updating of weights; the training paradigm finds a set of weight values that minimizes the error across the set of facts. During training, differences between actual outputs and the outputs predicted by the model are propagated back through the architectural structure of the network. Back-propagation involves the computation of errors in each layer of the neural network and propagation of errors backwards through the network in order to update weights and biases. This strategy is based on comparing the output of the model and the actual answer that should be produced. In simple words, it is based on error minimization.

When a network fulfils the convergence criteria, it means that the training is complete. If the network is validated, it can be used to predict outputs based on new input values that were not in the training set. It should be taken into consideration that for back-propagation to yield fine results, the topology of the NN, meaning the number and size of the hidden layers should be carefully chosen. Unfortunately, there is no mathematical rule to determine the optimal values of these parameters, thus they must be found by trial-and-error.

2.2 Total Error

Feeding one data pattern forward in the network, propagating the errors backwards and updating weights and biases until all training data are included in the computations is called one epoch. These steps can be repeated as many times as possible until the total error of the network falls below an acceptable level. Total error is the way to monitor how well the NN performs. A sum squared error or an absolute error may be used for this purpose. Since the aim of the NN, during training, is to learn to produce the output value in the training set, which in this case is the GDP, once it has been provided with the input values in the training set, the error can simply be the difference between the output of the NN and the target values that it is supposed to produce. This error is calculated across all training patterns by using the updated weights and biases at the end of each epoch. Thus, the mathematical measure of the performance of the NN, in the case of calculating an absolute error, is

$$\text{total error} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left|\text{output(j)} - \text{target(j)}\right| \tag{1}$$

where output(j) is the output of the jth neuron in the output layer

target (j) is the target value of the jth neuron in the data pattern

n is the number of data patterns in the training set, which, in this case, is 168, pertaining to the number of months in 14 years of training data and m is the number of output neurons, which in this case is 1, since the GDP is to be forecast.

Average error is this error per data pattern in the training set. It is monitored continuously during the training of the neural network as a measure of how well the network is learning.

A quick examination of the algorithm reveals that the back-propagation network trains itself to recognize features in the input patterns and slowly changes the network connection weights and biases to minimize the error across all training data patterns. Back-propagation might require extremely long training times but it must be kept in mind that training a NN for a particular application will only be done once. After the correct weights and biases are obtained, only the feed-forward part of the NN will be used in order to obtain the right output values for an input pattern. NN will simply produce the right outputs for an input data pattern.

It should be noted here that in the case of forecasting, the data patterns need to be presented to the network chronologically. Input data belonging to time t and output data belonging to time t+1 form one data pattern, i.e., the idea is that the macroeconomic factors of time t affect the GDP of time t+1. For this reason, the data needs to be presented to the network in order. If the problem had not been time dependent, then the order of the data patterns presented to the network would not matter. In the case of robot kinematics, for example, as studied by Anli, a neural network was trained to solve for the position and orientation of a parallel manipulator when the lengths of the six prismatic legs were given. Apparently, the input data consisted of leg lengths and the output data consisted of the position and orientation data of the mechanism. One data pattern was formed by taking both the input data and the output data at the same time interval. Thus, since the same leg lengths would result in the same position and orientation, regardless of time, data patterns could be presented to the network in any order. In the case of forecasting, on the other hand, what is to be done is to determine the GDP one year ahead of time. For this reason, the input data (14 macroeconomic factors) belonging to time t and the output data (GDP) belonging to time t+1 form one data pattern and the order of those data patterns is of great importance. For example, the first data pattern presented to the network during training has the macroeconomic data of January 1991 as inputs and the GDP of February 1991 as the output.

2.3 Selection of the topology of the network

The topology of a neural network is related to the number of neurons in each of its layers. Since there is no mathematical procedure for choosing the best architecture for a given problem, the procedure is based primarily on experimentation. The number of neurons in the input and output layers, in this case, are 14 and 1, respectively, since the aim is to forecast the GDP in the presence of 14 macroeconomic factors. This means 14 data will enter the network to produce 1 output, which clarifies why there have to be 14 neurons in the input layer and 1 in the output layer. As for the hidden layers, several experiments were carried out to figure out the configuration resulting in the least average error. Average error, or the error per data pattern, is the way to tell how well the network is adapting to the data patterns present in the training set.

A neural network training program was written in FORTRAN because running the neural network packages of commercially available programs, such as MATLAB, was found to require ~10 times longer processing time, due to the additional functions and sub-routines running in the background. In addition, programming in FORTRAN gives the opportunity to be in control of every aspect of the program, such as how the learning rate is going to decay, etc. Programming in FORTRAN gave more flexibility and control over the algorithm. Many trials were carried out in order to find the best architecture. In order to do so, the neural network program was run with many combinations of the hidden layer neuron numbers up to 100000 epochs and the behavior of the average error in each trial was recorded. The best 30 were selected and re-run, this time up to 500000 epochs. The main criteria in selection were the following.
Is the average error steadily declining or is it fluctuating?
Is the average error declining too slowly?
Is the average error continuing to fall or does it rise after a while?

The best 7 architectures were selected, taking into account the criteria listed above, and re-run until the average error fell below an acceptable limit, which in this case was on the order of 10-3. Table 1 gives some examples of the results of these runs. The first two columns give the number of neurons in the first and second hidden layers, respectively, and columns 3 – 6 give the average error per data pattern after 50000, 100000, 200000 and 500000 epochs, respectively. An inspection of table 1 reveals why some neural network architectures were only run up to 100000 epochs while others were run up to 500000 epochs. The criteria for deciding whether a certain architecture is disadvantageous or not is listed in the above paragraph. For instance, the (1, 2) architecture was found to be disadvantageous because the average error was still on the order of 10-1 after 100000 epochs, (4, 1) architecture was found to be disadvantageous because the average error was stuck at a certain value and did not keep on decreasing, and the (5, 3) architecture was found to be disadvantageous because the average error started increasing steadily afterwards.

Table 1. Sample training data.

no	1st hid. lay.	2nd hid. lay.	50000 epochs	100000 epochs	200000 epochs	500000 epochs
1	1	1	7.82*10-1	8.34*10-1		
2	2	1	3.08*10-1	3.28*10-1		
3	2	2	1.26*10-1	1.11*10-1	6.65*10-2	6.10*10-2
4	2	8	3.62*10-1	3.73*10-1		
5	3	1	1.64*10-1	1.66*10-1		
6	3	8	1.90	1.90		
7	4	1	3.02*10-2	2.26*10-2		
8	4	4	3.27*10-2	3.26*10-2		
9	4	7	2.10*10-2	1.42*10-2	1.41*10-2	1.39*10-2
10	5	2	2.01*10-1	2.56*10-2		
11	5	3	8.3*10-2	2.00*10-2		
12	5	6	2.56*10-2	1.24*10-2	1.24*10-2	1.24*10-2
13	6	2	2.70*10-2	3.65*10-2		
14	6	4	1.08*10-2	1.08*10-2		
15	6	5	7.94*10-3	8.14*10-3	8.37*10-3	8.70*10-3
16	7	4	1.68*10-2	1.01*10-2	9.92*10-3	9.54*10-3
17	7	6	4.14*10-2	9.38*10-3	5.99*10-3	5.81*10-3
18	7	7	3.80*10-2	1.68*10-2	1.48*10-2	6.18*10-3
19	8	2	9.75*10-3	9.77*10-3	9.85*10-5	1.01*10-2
20	8	3	2.06*10-2	1.14*10-2	5.04*10-3	4.07*10-3
21	8	8	7.03*10-3	7.00*10-3	6.91*10-3	6.48*10-3

It should be emphasized that it is not only the values of the average error at certain epochs that was monitored, it is the progress or the inclining/declining characteristics of the average error that was observed in choosing the best architectures. The architecture with 8 neurons in the first hidden layer and 3 in the second hidden layer was finally selected as the best architecture for this problem because the average error per data pattern was found to drop to 3.69*10-3 at 339000th epoch. Even though there was a rise in the average error between the 339000th and 500000th epochs, 3.69*10-3 was the smallest average error experienced in this problem, and it is a satisfactory result.

3. Forecasting the GDP by considering national macroeconomic factors using neural networks

The type of the NN used in this application is the feed-forward network with two hidden layers and is trained using back-propagation algorithm. The reason why it is useful to use neural networks in solving the GDP forecasting problem is that it is very simple to train a neural network for this type of problem of obtaining outputs for a given set of inputs. Once a neural network is trained for a particular problem and the right connection weights and biases are obtained and saved, it takes the network milliseconds to produce the output value for a given input set of data patterns. In this case, this input data pattern consists of the 14 the internal macroeconomic data in a certain month (t) and the output data pattern is the GDP of the next month (t+1) in Turkey.

The NN application given in this work has 14 neurons in the input layer and 1 neuron in the output layer, due to the nature of the problem considered. Since the problem solves for the GDP in the presence of 14 values, it means that the NN needs to produce 1 output in the presence of 14 inputs. Thus, the 14 neurons (factors) that the NN architecture has in the input layer.

Each of the neurons in the input layer represents one of the macroeconomic data. The neuron in the output layer represents the GDP. The feed-forward network trained for this study has the following characteristics: Learning rate slowly decayed from 0.2 to 0.0001 with a rate of 0.001 to give better convergence, momentum factor was chosen to be 0.02 and the training stopped once the average error per data pattern reached 0.00369. Data from years 1991 – 2005 were used in training and testing the network. A network should be tested with data patterns that are not in the original training set to see whether the network has reached right connection weights and biases to produce accurate results when it is provided with data that it has not been presented before, thus the network was trained with data from the years 1991 – 2004 (168 months) and tested with the data from the year 2005. It has been observed by trial and error that the best NN configuration is the one having 8 neurons in the first hidden layer and 3 in the second hidden layer. The activation function f(net) used in the hidden layers is the tangent-sigmoid function, the mathematical equivalent of the tangent-hyperbolic function, defined as

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

It has also been found out that gradually decreasing the learning rate to is helpful for convergence. The measure of performance of the NN is an error function defined as

$$\text{error} = \sum_{i=1}^{n} |\text{output}(i) - \text{target}(i)| \qquad (3)$$

where output(i) is the value produced by the ith output neuron
target is the desired value that should actually be produced by the ith output neuron and
n is the number of output neurons. It should be noted here that in this case n=1. |output– target| is the absolute value of the difference between the output value and the target value. The errors from all data patterns are added through one epoch, and then divided by the number of data patterns to obtain the average error. This error function is the way to monitor the performance of the NN, thus the training of the NN stops only when the average error per data pattern drops below an acceptable value. Figure 1 gives the GDP in the year 2005 and the values forecasted by the neural network.

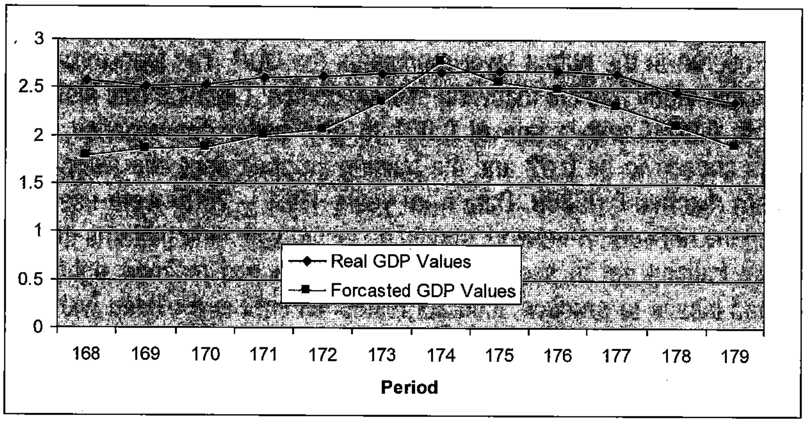

Figure 1. GDP and the forecasted GDP in 2005.

4. Results and Comments

A back-propagation neural network was trained to forecast the GDP in Turkey one year ahead of time, using 14 macroeconomic factors as the inputs. Training of the NN by using 168 data patterns (data from the years 1991 – 2004) was performed until the average error per data pattern dropped to 3.69*10-3. This is considered very accurate in many applications. Results of forecasting the GDP in the year 2005 are given in figure 1. Figure 1 shows and compares the GDP of the 12 data patterns (data from the year 2005) that were not in the training set and the GDP values that the neural network forecasts.

A neural network has been trained to forecast the GDP in Turkey one year ahead of time in the presence of macroeconomic data, in other words, macroeconomic data of the month t have been used to forecast the GDP of the month t+1

Consequently, once the network is trained, the method yields accurate results fast enough to find the GDP index so that it is applicable for real-time situations. Using more data patterns or changing the input factors could give more accurate results.

References

Elmas Anli, "Positional Kinematics of the 6-3 Stewart Platform Mechanism Using Heuristic
Algorithms", MSc Thesis, 2005, Istanbul Technical University, Insititute of Science and Technology, Interdisciplinary Programs, Aeronautical and Astronautical Engineering.

Halil Kiymaz, "The initial and aftermarket performance of IPOs in an emerging market: evidence from Istanbul stock exchange", Journal of Multinational Financial Management, 10 (2000) 213–227.

Greg Tkacz, "Neural network forecasting of Canadian GDP growth", International Journal of Forecasting, 17 (2001) 57–69.

Jorge V. Perez-Rodrigueza, Salvador Torrab, Juliain Andrada-Felixa, “STAR and ANN models: forecasting performance on the Spanish bIbex-35Q stock index”, Journal of Empirical Finance, 12 (2005) 490–509.

Juliana Yim, Heather Mitchell, “Comparison of country risk models: hybrid neural networks,
logitmodels, discriminant analysis and cluster techniques”, Expert Systems with Applications, 28 (2005) 137–148.

Kyoung-Jae Kim, “Artificial neural networks with evolutionary instance selection for financial
forecasting”, Expert Systems with Applications, 30 (2006) 519–526.

Mark A. Hooker, “Macroeconomic factors and emerging market equity returns: a Bayesian model selection approach”, Emerging Markets Review, 5 (2004) 379–387.

Thomas C. Chiang, Doseong Kim, Euiseong Lee, “Country-fund discounts and risk: Evidence
from stock market volatility and macroeconomic volatility”, Journal of Economics and Business. 58 (2006) 303-322.

Yaz Gulnur Muradoglu and Kivilcim Metin, “Efficiency of the Turkish Stock Exchange with respect to monetary variables: A cointegration analysis”, European Journal of Operational Research, Volume 90, Issue 3 (1996) 566-576.