Inflation Forecasting in Pakistan using Artificial Neural Networks

Adnan Haider and Muhammad Nadeem Hanif

State Bank of Pakistan, Karachi, Pakistan

13. July 2007

INFLATION FORECASTING IN PAKISTAN USING ARTIFICIAL NEURAL NETWORKS

Adnan Haider[†]
Muhammad Nadeem Hanif[*]

Abstract: An artificial neural network (hence after, ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. In previous two decades, ANN applications in economics and finance; for such tasks as pattern reorganization, and time series forecasting, have dramatically increased. Many central banks use forecasting models based on ANN methodology for predicting various macroeconomic indicators, like inflation, GDP Growth and currency in circulation etc. In this paper, we have attempted to forecast monthly YoY inflation for Pakistan by using ANN for FY08 on the basis of monthly data of July 1993 to June 2007. We also compare the forecast performance of the ANN model with conventional univariate time series forecasting models such as AR(1) and ARIMA based models and observed that RMSE of ANN based forecasts is much less than the RMSE of forecasts based on AR(1) and ARIMA models. At least by this criterion forecast based on ANN are more precise.

JEL Classification: C51, C52, C53, E31, E37
Key Words: artificial neural network, forecasting, inflation

[*] The authors are, respectively, Research Analyst (adnan.haider@sbp.org.pk) and Senior Economist (nadeem.hanif@sbp.org.pk) in Economic Modeling Division, Research Department, State Bank of Pakistan, I.I. Chundrigar Road, Karachi, Pakistan.

[†]***Corresponding Author Note:*** The authors are grateful to an anonymous referee for providing his helpful comments. They also thankful to Safdar Ullah Khan for his insightful discussions on the earlier draft of this paper. Finally, Views expressed here are those of the authors and not necessarily of the State Bank of Pakistan. Any errors or omissions in this paper are the responsibility of the authors.

I. INTRODUCTION

In modern age of forecasting, there has been a great interest in studying the artificial neural network (ANN) forecasting in economics, financial, business and engineering applications including GDP growth, stock returns, currency in circulation, electricity demand, construction demand and exchange rates (see, Fernandez-Rodriguez, Gonzalez-Martel and Sosvilla-Rivero [2000] and Redenes and White [1998]). Many central banks, for example: CZECH National Bank, (Marek Hlavacek, Michael Konak and Josef Cada [2005]), Bank of Canada, (Greg Tkacz and Sarah Hu [1999]), Bank of Jamaica, (Serju [2002]), are currently using their forecasting models based on ANN methodology for predicting various macroeconomic indicators. Using forecasts based on this methodology one may make correct classification and decisions such as stock selection, bond rating, credit assignment, property evaluation, and many others (see Chen, Racin and Swanson [2001], Swanson and White [1997] and Stock and Watson [1998]).

Inflation forecast is used as guide in the formulation of the monetary policy by the monetary authorities in the world. Monetary policy decisions are based on inflation forecast extracted from the information from different models and other information suggested by relevant economic indicators of the economy. The main purpose of this paper is to forecast monthly YoY inflation for Pakistan by using ANN methodology for FY08 on the basis of available monthly data since July 1993. We also compare the forecast performance of the ANN model with that of univariate AR(1) and ARIMA based models. It is observed that forecasts based on ANN are more precise than then those based upon AR(1) and ARIMA models.

The rest of the paper is organized as follows: Neural network methodology is presented in section 2. Section 3 provides data and model specifications. Empirical findings are discussed in section 4 and the summary of our findings is presented in the last section.

II. BACKGROUND AND METHODOLOGY

Neural network theory grew out of Artificial Intelligence research, or the research in designing machines with cognitive ability. An artificial neural network is an information processing paradigm that is inspired by the way

biological nervous systems, such as the brain, process information.[1] The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements, called neurons, working in unison to solve specific problems. ANN learns by experience like people. An ANN is configured for a specific application, such as pattern recognition and time series forecasting, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. The basic building block of a brain and the neural network is the neuron. The human neuron[2] is shown in figure 1.

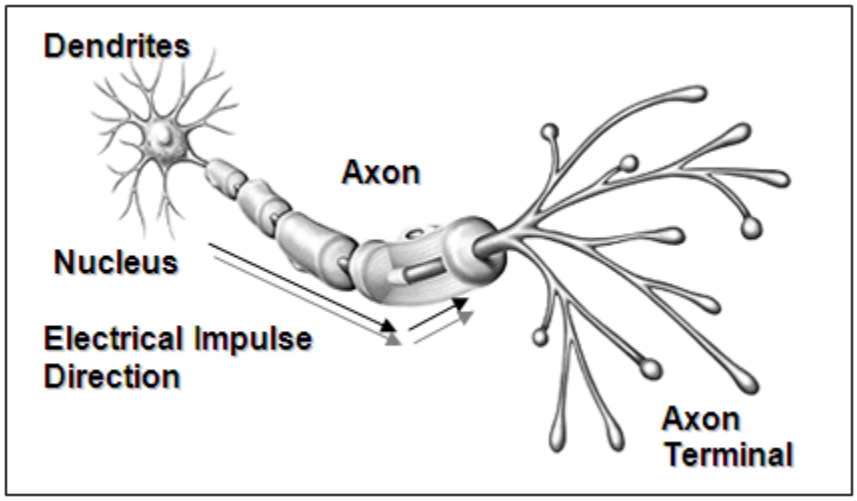

Figure 1. Biological Model of Human Neuron

As described by Beal and Jackson [1990], all inputs to the cell body of the neuron arrive along *dendrites*. Dendrites can also act as outputs interconnecting inter-neurons. Mathematically, the dendrite's function can be approximated as a summation. *Axons,* on the other hand, are found only on output cells. It has an

[1]A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: Knowledge is acquired by the network through a learning process and interneuron connection strengths known as synaptic weights are used to store the knowledge, see for instance, Hykin [1994].

[2] Adapted from Beale and Jackson [1990] and JOONE Documentation.

electrical potential. If excited, past a threshold, it will transmit an electrical signal. Axons terminate at *synapses* that connect it to the dendrite of another neuron. The neuron sends out spikes of electrical activity through a long axon, which splits into thousands of branches, see, figure 2. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The human brain contains approximately 10 billion interconnected neurons creating its massively parallel computational capability.

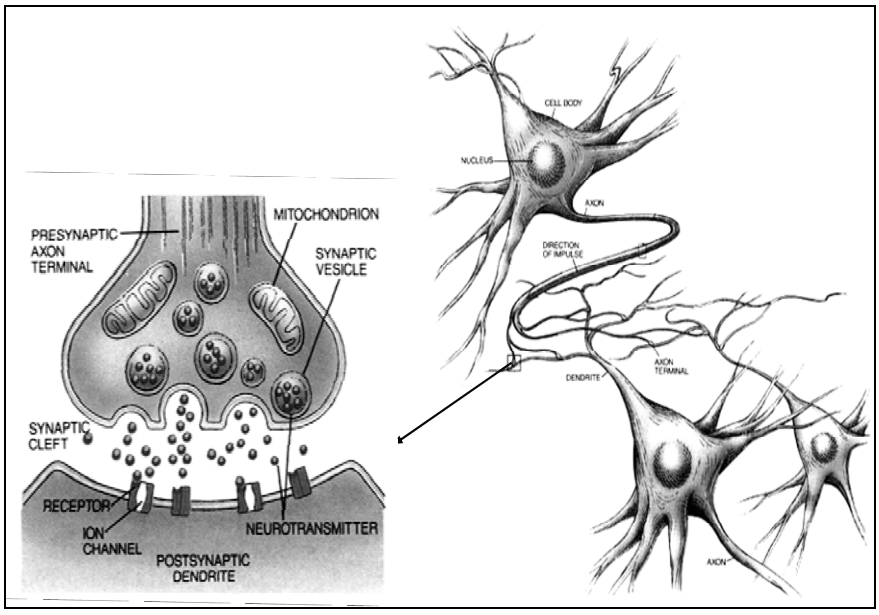

Figure 2. Neural Signal Transmission

ARTIFICIAL NEURON

The artificial neuron was developed in an effort to model the human neuron. The artificial neuron depicted in figure 3[3]. Inputs enter the neuron and are multiplied by their respective weights.

For analytical purposes, a neuron may be broken down into three parts:

- input connections
- summing and activation functions
- output connections

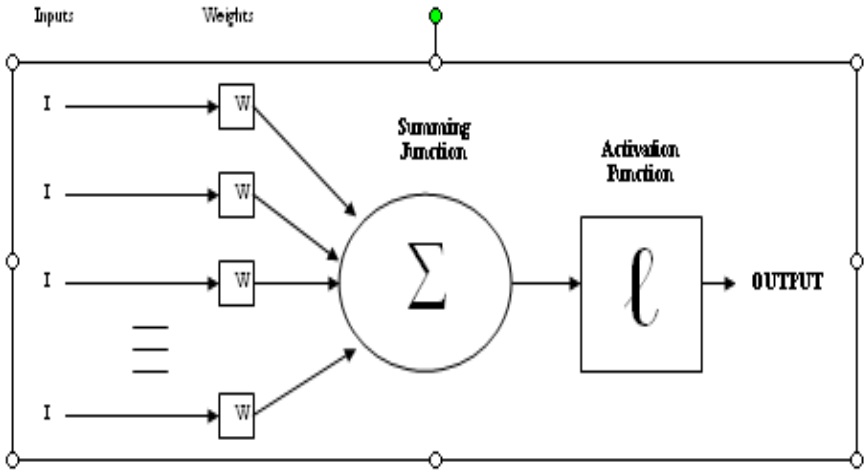

Figure 3. Artificial Neuron Model

INPUT CONNECTIONS

In artificial neural network, a neuron is connected to other neurons and depends on them to receive the information that it processes. There is no limit to the amount of connections a neuron may receive information from. The information that a neuron receives from others is regulated through the use of weights. When a neuron receives information from other neurons, each piece of information is multiplied by a weight with a value between -1 and +1, which allows the neuron

[3] Adapted from Kartalopoulos [1996] and Haykin [1994].

to judge how important the information it receives from its input neurons is. These weights are integral to the way a network works and is trained: specifically, training a network means modifying all the weights regulating information flow to ensure output follows the given criteria, e.g., minimization of RMSE or MAE.

SUMMING AND ACTIVATION FUNCTIONS

The second portion of a neuron is the summing and activation functions. The information sent to the neuron and multiplied by corresponding weights is added together and used as a parameter within an activation function.[4] Numerous activation functions exist in ANN literature, but we will discuss below the one which we used and that is hyperbolic tangent function: a continuous function with a domain of $(-\infty,\infty)$ and a range of (-1, 1):

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

By providing a function with a limitless domain and a range of (-1, 1), it is perfect for predicting whether or not inflation will rise ($\tanh(x) = 1$) or fall ($\tanh(x) = -1$).

OUTPUT CONNECTIONS

Finally, once the activation function returns a corresponding value for the summed inputs, these values are sent to the neurons that treat the current neuron as an input. The process repeats again, with the current neuron's output being summed with others, and more activation functions accepting the sum of these inputs. The only time this may be ignored is if the current neuron is an output neuron. In this case, the summed inputs and normalized sum is sent as an output and not processed again.

---

[4]In a biological context, a neuron becomes activated when it detects electrical signals from the neurons it is connected, see, Beale, R., Jackson, T., [1990]. If these signals are sufficient, the neuron will become "activated" - it will send electrical signals to the neurons connected to it.

NEURAL NETWORK ARCHITECTURE

While each neuron is, in and of itself, a computational unit, neurons may be combined into layers to create complex but efficient groups that can learn to distinguish between patterns within a set of given inputs. Indeed, by combining multiple layers of such groups, it is theoretically possible to learn any pattern. There are many combinations of neurons that allow one to create different types of neural networks, but the simplest type is a single-layer feedforward network. In this case, a network is composed of three parts: a layer of input nodes, a layer of hidden neurons, and a layer of output nodes, as is shown in the figure 4.

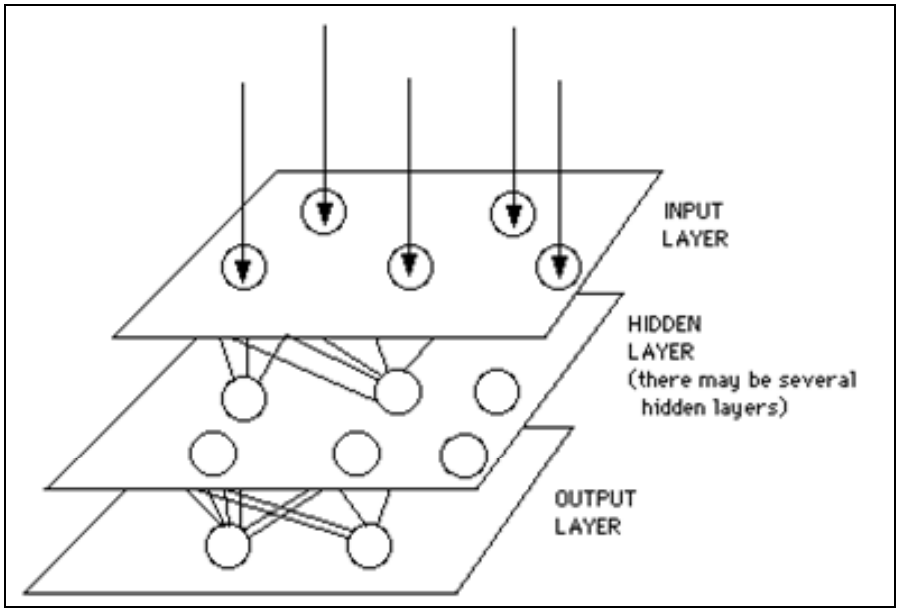

Figure 4. Layers of Artificial Neural Networks

A multilayer feedforward network is similar to a single-layer one. The main difference is that instead of having a hidden layer pass its calculated values to an output layer, it passes them on to another hidden layer. Both types of networks are typically implemented by fully connecting each layer's neurons with the preceding layer's neurons. Thus, if *Layer A* has *k* neurons and sends its information to *Layer B*, with *n* neurons, each neuron in *Layer A* has *n*

connections for its calculated output, while each neuron in *Layer B* has *k* input connections.

Interestingly, such a network can be represented mathematically in a simple manner. Supposing there are *k* neurons in *Layer A*, let *a* represent a vector, where a_i is the i^{th} neuron's activation function output. Let *b* represent the input values to neurons in *Layer B*, with b_j be the j^{th} neuron. Let W be a *n* by *k* matrix where w_{ji} represents the weight affecting the connection from a_i to b_j. Keeping this in mind, we can see that for a single-layer feedforward network, we can mathematically represent the flow of information by,

$$Wa = b$$

and the learning thus becomes a modification of each w_{ji} in W. A similar mathematical analogy applies to multilayer feedforward networks, but in this case, there is a W for every layer and '*b'* is used as the value for '*a'* when moving to subsequent layers. The most popular type of learning within a single-layer feedforward network is the Delta Rule, while multilayer feedforward networks implement the Backpropagation algorithm, which is a generalization of the Delta Rule, (see, Beale, R., Jackson, T., [1990]).

The Delta Rule may be summarized with the following equation:

$$\Delta w_{ij} = - \in \delta_j x_i$$

In this case, Δw_{ij} represents the change of the weight connecting the i^{th} neuron with the j^{th} output neuron, xi is the output value of the i^{th} neuron, ε is the learning rate, and δj is the error term in the output layer, defined as:

$$\delta_k = -(t_k - o_k)$$

where t_k is the expected output, while o_k is the actual output. While this rule works well when there is only one hidden layer In case of multiple layers we use generalized delta rule described below.

$$\delta_j = (\sum_k \delta_k w_{jk})(h_j(1-h_j))$$

In this case, one uses the same equation for Δw_{ij} but uses the term above instead, with k representing the neurons receiving information from the current neuron being modified. $\delta_k w_{jk}$ is the error term of the k^{th} neuron in the receiving layer, with w_{jk} being the connecting weight. The activation functions of all the neurons in a network implementing backpropagation must be differentiable, because:

$$h_j = \sigma'(z_j)$$

with z_j being the net input for the neuron.

Finally, if biases are present, they are treated like regular neurons, but with their output (x) values equal to 1:

$$\Delta B_j = - \in \delta_j$$

When implemented, this algorithm has two phases. The first deals with having the network evaluate the inputs with its current weights. Once this is done, and all the neuron and output values are recorded, phase two begins. The algorithm begins this phase by applying the original Delta Rule to the output neurons, modifying weights as necessary. Then the generalized Delta Rule is implemented, with the previous δ values sent to hidden neurons, and their weights changing as required.

III. DATA AND MODEL SPECIFICATIONS

DATA SPECIFICATION

The main objective of this study is to forecast monthly YoY inflation for Pakistan for FY08 using feed-forward artificial neural network model with 12 hidden layers. We used data on monthly basis since July-1993. Figure 5 represents graphically the data we have used.

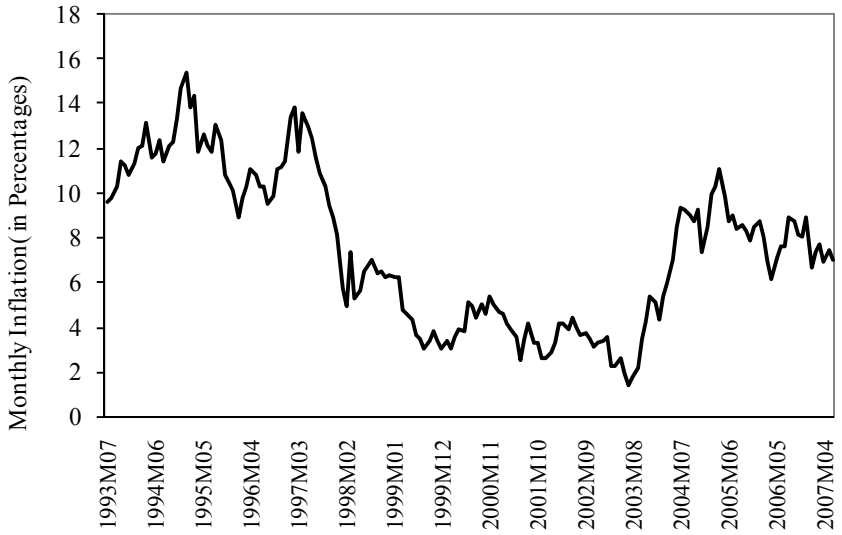

Figure 5. Monthly Inflation rate (YoY)

MODEL SPECIFICATION

We estimate a very simple neural network for inflation based on *'feedforward with backpropagation'* architecture.

$$\hat{\pi}_{t+j} = \sum_{k=1}^{n} \Theta_k \tanh(w_k x_{t-k} + b_k) \qquad (1)$$

Where: $\hat{\pi}_{t+j}$ is the neural network inflation forecast *j* months ahead, x_{t-1} is a vector of lagged inflation variables [$\hat{\pi}_{t-1}$, $\hat{\pi}_{t-2}$], tanh which is the hyperbolic tangent function used as transformation process. Θ 's are layer weights, w_i are input weights and b's are biases.

While implementing our model, it required following steps:

- Selection of input variables
- Preprocessing the input data
- Specifying a neural network

- Training the network
- Forecast Accuracy

SELECTION OF INPUT VARIABLE

This step identifies the variables that contribute the most to forecasting the target variable. If we omit important variables, then its effect on the performance of the neural network can be significant. For our simplicity, we only consider univariate forecast neural network model. Given the limitation of the data, the simple neural network architecture given by (1) was chosen with very minimal search over alternative network architectures.

PREPROCESSING THE INPUT DATA

Neural networks need properly transformed data to be able to process them and generate sound forecasts.[5] For our simplicity, we assume there is no statistical bias in our data.

SPECIFYING A NEURAL NETWORK

A typical *feedforward with backpropagation* network should have at least three-layers[6]. Appropriate selection of number of layers is an art. This selection criteria needs experimentation. The countless combinations of layers and neurons that we can make and the time it takes to train a network after each selection is an arduous exercise.[7] In this paper, the number of layers specified is 12. We could also use many layers but that would make the training time prohibitive. The resulting improvement in forecast accuracy may not be worth the extra time.

---

[5]Transformation, normalization and data smoothing are three common ways of preprocessing data. Transformation and normalization makes the statistical distribution of each input and output data roughly uniform. The values are scaled to match the range that the input neurons use. Data normalization methods, which include simple linear scaling and statistical measures of central tendency and variance, remove outliers and spread out the distribution of the data. Data smoothing filters out noise in the data.

[6] Input, output and at least one hidden layer is compulsory.

[7]It is found that, a single or two-layer network would be rather inadequate in capturing the complex dynamics of market variables. For example, Barnett, Medio and Serletis [2003]

Furthermore, a backpropagation network should have at most fifteen layers.[8] In the network specification stage we can adjust a number of default parameters or values that influence the behavior of the training process. These deal with the learning, forgetting and error tolerance rates of the network, the maximum number of runs, stop value for terminating training and randomizing weights with some specified dispersion[9].

TRAINING THE NETWORK AND FORECASTING

we train the ANN using the Levenberg-Marquardt algorithm, a standard training algorithm from the literature. The algorithm is terminated according to the early stopping procedure.[10] The validation set used in the early stopping procedure is chosen in a somewhat unusual manner. Finally, the training function produce forecast results on the basis of RMSE minimization criteria.

FORECAST EVALUATION

To evaluate the forecast, we calculate root mean of squared errors (RMSE) as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(Y_{it} - Y_{it})^2}{n}}$$

The training algorithm is run on the training set until the RMSE starts to increase on the validation set.

---

[8] See, Serju [2002].
[9] For neural network default values specification, see MATLAB 7.0 guidelines. [www.mathworks.com]
[10] This procedure is related to the innovative approach used by Nakamura, Emi, [2005]

IV. EMPIRICAL RESULTS

Inflation forecasting based on model described in section 3.2 requires the following steps. First of all we normalize data by using the following algorithm.

$$normalized\ \ value(\pi_t) = 2 * \left(\frac{(current\ \ value(\pi_t) - \max value(\pi_t))}{(\min value(\pi_t) - \max value(\pi_t))} \right)$$

This process change the original scaling of data within the range [-1 , +1]. Results are shown here in figure 6.

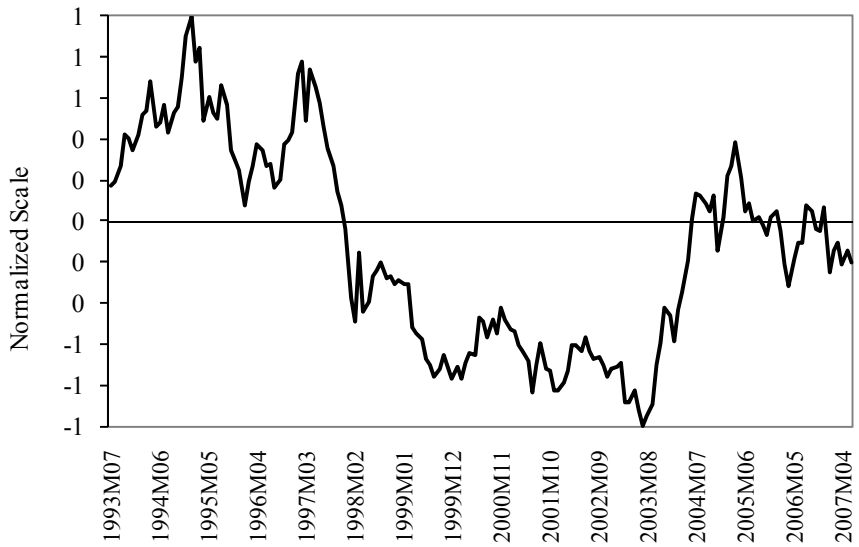

Figure 6. Inflation rate in normalized Scale [-1, +1]

Our neural network was trained using the MATLAB. Before training, this model requires some default values, which are given in table 1. After setting the basic properties, model required normalized data as input and transfers this to training function by using tan hyperbolic function. Then 'trainlm' function is used to train the data.

Forecast efficiency is captured by minimizing the root mean square errors (RMSE). In our experiment, we try to find forecast monthly YoY inflation

for Pakistan for FY08 on the basis of monthly data since July 1993. The out-of-sample forecast is presented in figure 7. It shows an upward continuous trend in inflation rate for FY08. In the next subsection we compare the forecast performance of ANN with that of AR(1) model.

Table 1: Default Parameters Values for ANN
hidden_layers = 12
max lag = 12
training set = 80
forecast period = 12
learning rate = 0.25
learning increment = 1.05
learning decrement = 0.07
training parameter epochs = 1000
target RMSE = 0.00005;
Table Key:
MALTAB Neural Network toolkit uses these parameter values.

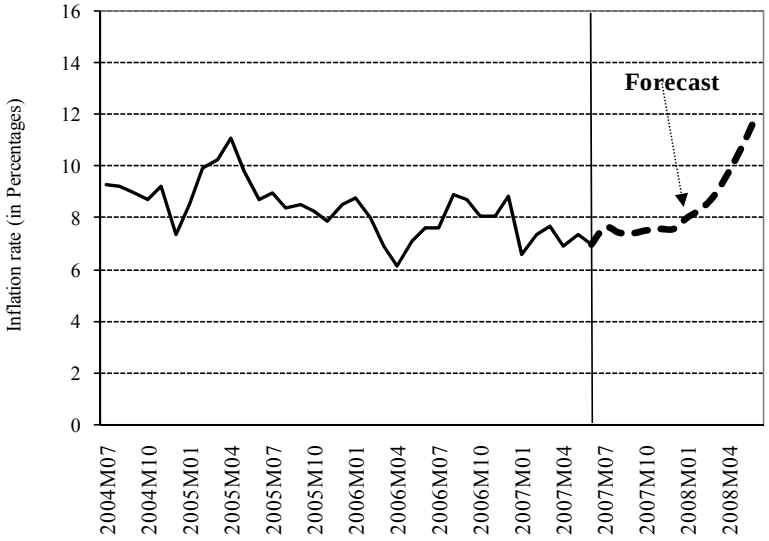

Figure 7. Monthly YoY Inflation Forecast for FY08

COMPARING FORECAST PERFORMANCE OF ANN AND AR(1) AND ARIMA BASED MODELS

In order to compare the out-of-sample forecast performance of ANN with AR(1) and ARIMA based models we find the out-of-sample forecast for July 06 to June 07 from both of these model based on data for July 1993 to June 2006. Results based on ANN methodology as well as both AR(1) and ARIMA methodologies are presented in table 2. Forecasting performance is evaluated on the basis of RMSE criteria. We observed that RMSE of ANN based forecasts is less than the RMSE of forecasts based on AR(1) and ARIMA models. At least by this criterion forecast based on ANN are more precise. This forecast comparison result based on Pakistan data is also consistent with earlier findings; Nakamura (2006) for US case and Choudhary and Haider (2008) for the case of twenty eight OECD countries.

Table 2: Performance Comparison based on RMSE

Months	Actual Inflation	Forecast by ANN	Forecast by AR(1)	Forecast by ARIMA(1,3,7,11;1;1, 2)*
Jul-06	7.63	7.68	7.62	7.92
Aug-06	8.93	7.92	7.60	8.34
Sep-06	8.73	8.43	7.59	8.86
Oct-06	8.11	8.45	7.58	9.34
Nov-06	8.06	8.33	7.56	9.54
Dec-06	8.88	8.26	7.55	9.79
Jan-07	6.64	7.98	7.53	10.12
Feb-07	7.39	7.70	7.52	11.23
Mar-07	7.67	7.68	7.51	11.23
Apr-07	6.92	7.59	7.49	10.76
May-07	7.41	7.44	7.48	10.37
Jun-07	7.00	7.29	7.47	10.14
Average:	7.78	7.90	7.54	9.80
RMSE:		0.59	0.75	2.25

Table Key:

*/ usual ARIMA (AR terms; Order of integration; MA terms) convention is used

V. CONCLUSION

This paper applied a simple univariate artificial neural network model to forecast monthly YoY inflation for Pakistan by using ANN methodology for FY08 on the basis of monthly data for July 1993 to June 2007. The main purpose of this exercise is to find consistent out-of-sample forecast based on RMSE minimization criteria, in which error volatility is minimized after training network with 12 hidden layers. The leaning rate of our model is 0.25. Model simulation used feedforward with backpropagation methodology which requires an activation function which used generalized delta rule. Our forecast results indicate that inflation projection for the end of next FY08 is on high on average as compared with FY07. In last, we compared out-of-sample forecast performance with forecast based on AR(1) and ARIMA based models and found that RMSE of ANN based forecasts is much less than the RMSE of forecasts based on AR(1) and ARIMA based models. At least by this criterion forecast based on ANN are more precise.

REFERENCES

Barnett, William A., Alfredo Medio, and Apostolos Serletis, (2003), *Nonlinear and Complex Dynamics in Economics*, Memo

Beale, R., Jackson, T., (1990), *Neural Computing: An introduction*, Adam Hilger, Bristol England.

Brock, William A. and Cars H. Hommes, (1997), A Rational Route to Randomness, *Econometrica*, , 65 (5), pp 1059–1095.

Chen, Xiaohong, J. Racine, and N. Swanson, (2001), Semiparametric ARX Neural Network Models with an Application to Forecasting Inflation," *IEEE Transactions on Neural Networks*, 12, pp 674–683.

Choudhary, M. A. and A. Haider (2008), *Neural Network models for Inflation Forecasting: An Appraisal*, Discussion Paper no 0808, Department of Economics, University of Surrey, UK.

Fernandez-Rodriguez, Fernando, Christian Gonzalez-Martel, and Simon Sosvilla-Rivero, (2000), On the profitability of technical trading rules based on artificial neural networks:: Evidence from the Madrid stock market, *Economics Letters*, 69 (1), pp 89–94.

Gonzalez Steven, (2000), Neural Networks for Macroeconomic Forecasting: A Complementary Approach to Linear Regression Models, *Finance Canada Working Paper* 2000-07.

Greg Tkacz, Sarah Hu, (1999), *Forecasting GDP growth using Artificial Neural Networks,* Working Paper 99-3, Bank of Canada.

Hornik, K., M. Stinchcombe, and H. White, (1989), Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, pp 359–366.

Haykin, Simon, (1994), *Neural Networks: A comprehensive foundation*, *Macmillian College Publishing Company, New York.*

Kartalopoulos, Stamations V., (1996), Understanding Neural Networks and Fuzzy Logic: Basic concepts and applications, *IEEE Press*, New York.

Lebaron, B. and A.S. Weigend, (1998), A bootstrap evaluation of the effect of data splitting on financial time series, *IEEE Transactions on Neural Networks*, 9 (1), pp 213–220.

Marek Hlavacek, Michael Konak and Josef Cada, (2005), *The application of structured feedforward neural networks to the modeling of daily series of currency in circulation*. Working paper series 11, Czech National Bank.

McCracken, M. W. and K. D. West, (2001), *Inference about Predictive Ability, in Michael Clements and David Hendry, eds., A Companion to Economic Forecasting*, Oxford, U.K. Blackwell Publishers.

Moshiri, Saeed and Norman Cameron, (2000), Econometrics Versus ANN Models in Forecasting Inflation, *Journal of Forecasting*, February Issue, 19.

Nakamura, Emi, (2006), Inflation forecasting using a neural network, Economics Letter, 86 (3), pp 373-378.

Refenes, A. P. and H. White, (1998), Neural Networks and Financial Economics, *International Journal of Forecasting*, 6 (17).

Stock, James H. and Mark W. Watson, (1998), *A Comparison of Linear and Nonlinear Univariate Models for Forecasting Macroeconomic Time Series*, NBER Working Paper 6607, 7

Swanson, Norman R. and Halbert White, (1997), A Model Selection Approach to Real-Time Macroeconomic Forecasting Using Linear Models and Artificial Neural Networks, *The Review of Economics and Statistics*, November Issue, 79 (4), pp 540–550.