Volume 2, No. 2, Mar-Apr 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Application of a Modified Generalized Regression Neural Networks Algorithm in Economics and Finance

Eleftherios Giovanis
Department of Economics
Royal Holloway University of London
Egham, United Kingdom
Eleftherios.Giovanis.2010@live.rhul.ac.uk

Abstract: In this paper we propose an alternative and modified Generalized Regression Neural Networks Autoregressive model (GRNN-AR) in S&P 500 and FTSE 100 index returns, as also in Gross domestic product growth rate of Italy, USA and UK. We compare the forecasts with Generalized Autoregressive conditional Heteroskedasticity (GARCH) and Autoregressive Integrated Moving Average (ARIMA) models. The results indicate that GRNN outperform significant the conventional econometric models and can be an efficient alternative tool for forecasting. The MATLAB algorithm we propose is provided in appendix for further applications, suggestions, modifications and improvements.

Keywords: Autoregressive Moving Average, Forecasting, GARCH, Generalized Regression Neural Networks, MATLAB, Stock Returns

I. INTRODUCTION

Empirical analysis in macroeconomics as well as in financial economics is largely based on times series. The existence of unexpected shocks or innovations to the economy plus measurement errors, strongly suggest that economic variables are stochastic. The last two decades new approaches are applied in economics and finance. Most of them support the artificial intelligence procedures Aryal and Yao-Wu [1] applied a MLP network with 3 hidden layers to forecast the Chinese construction industry and they compare the forecasting performance of the MLP networks with that of ARIMA and they found that the RMSE of the MLP estimation is 49 percent lower than the ARIMA counterpart. Swanson and White [2]-[3] applied neural networks to forecast nine seasonally adjusted US macroeconomic time series and they found generally neural networks outperform the linear models. Keles *et al*. [4] developed Adaptive Neuro-Fuzzy Inference System for the prediction of domestic debt presenting very good results

II. METHODOLOGY

A. Autoregressive (AR) model

We consider a series $y_1, y_2, \ldots, y_n$. An autoregressive model of order p denoted AR(p), states that y_t is the linear function of the previous p values of the series plus an error term:

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \varepsilon_t \qquad (1)$$

, where $\varphi_1, \varphi_2 \ldots, \varphi_p$ are weights that we have to define or determine, and ε_t denotes the residuals which are normally distributed with zero mean and variance σ^2 [5]. Conditioned on the full set of information available up to time i and on forecasts of the exogenous variables, the one-period-ahead forecast of y_t would be

$$\hat{y}_{t+1|t} = \hat{\phi}_{0\,t+1|t} + \hat{\phi}_1 y_t + \hat{\phi}_2 y_{t-1} + ...+ \hat{\phi}_p y_{t-p+1} + \hat{\varepsilon}_{t+1|t} \qquad (2)$$

B. Moving Average (MA)

We consider the q order moving average MA(q) specification [5]:

$$R_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - .... - \theta_q \varepsilon_{t-q} \qquad (3)$$

, where the $\theta_1, ..., \theta_q$ are the parameters of the model, μ is the constant and $\varepsilon_1, ..., \varepsilon_q$ are again the white noise error terms. The forecasts are given by Eq. (4)

$$\hat{R}_{t+j} = \hat{\mu} - \hat{\theta}_1 \varepsilon_t - \hat{\theta}_2 \varepsilon_{t-1} - .... - \hat{\theta}_q \varepsilon_{t-q+1} \qquad (4)$$

C. Autoregressive Moving Average (ARMA) models

From the previous two sections we combine Autoregressive (AR) Moving Average (MA) Models and the Autoregressive Moving Average (ARMA) which encompasses (1) and (3) is defined as:

$$R_t = \mu + \phi_1 R_{t-1} + \phi_2 R_{t-2} + ... + \phi_p R_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - .... - \theta_q \varepsilon_{t-p} \qquad (5)$$

ARMA*(p, q)* process has p autoregressive, lagged dependent-variable, terms and q lagged moving-average terms. The series R_t is said to be integrated of order one, denoted *I (1)*, because taking a first difference produces a stationary process. A nonstationary series is integrated of order d, denoted *I (d)*, if it becomes stationary after being first differenced d times autoregressive integrated moving-average model, or ARIMA *(p, d, q)* and will be [5]:

$$\Delta^{\mathrm{d}} R_t = \mu + \phi_1 \Delta^{\mathrm{d}} R_{t-1} + \phi_2 \Delta^{\mathrm{d}} R_{t-2} + ... + \phi_p \Delta^{\mathrm{d}} R_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - .... - \theta_{\mathrm{q}} \varepsilon_{t-q} \quad (6)$$

The forecasts for ARMA *(p, q)* model is given by Eq. (7)

$$\hat{R}_{t+j} = \hat{\mu} + \hat{\phi}_1 R_t + \hat{\phi}_2 R_{t-1} + ... + \hat{\phi}_p R_{t-p+1} + \hat{\theta}_1 \varepsilon_t + \hat{\theta}_2 \varepsilon_{t-1} + ... + \hat{\theta}_{\mathrm{q}} \varepsilon_{t-q+1} \quad (7)$$

In all case we choose the lag order based on Akaike criterion.

D. Generalized Autoregressive conditional Heteroskedasticity (GARCH)

We estimate with symmetric GARCH (1,1) model [6]. The general GARCH (p,q) model is

$$\sigma_t^{\,2} = a_0 + \sum_{i=1}^{q} a_i u^2_{t-i} + \sum_{j=1}^{p} \beta_j \sigma^2_{t-j} \quad (8)$$

We do not bother to examine other GARCH models as the asymmetric, because the forecasts are not significant.

E. Generalized regression Neural Networks

The GRNN [7] is defined as:

$$E[y \mid x] = \frac{\int_{-\infty}^{\infty} y g(x, y) dy}{\int_{-\infty}^{\infty} g(x, y) dy} \quad (9)$$

,where E[y | x] is the expected value of y given x and g(x,y) is the Parzen probability density estimator . If the value of g(x,y) is unknown , then it can be estimated from a sample of observations of x and y. The predicted output obtained by GRNN is:

$$\hat{y}(x) = \frac{\sum_{i}^{n} y_i \exp(-\frac{\| x - x_i \|^2}{2\sigma_{\mathrm{i}}^{\,2}})}{\sum_{i}^{n} \exp(-\frac{\| x - x_i \|^2}{2\sigma_{\mathrm{i}}^{\,2}})} \quad (10)$$

Usually the GRNN consists of four layers. The first layer, which are the input data, the synaptic and the activation functions are linear. In the second layer, the pattern layer, the synaptic function is the radial and the activation function is the negative exponential. The third layer, the summation layer, has as the first layer, linear synaptic and activation functions. The output layer has a synaptic function a division and linear activation function. More specifically input layer receives the input vector *X* and distributes the data to the pattern layer. Each neuron in the pattern layer generates an output θ, which is:

$$\theta_{\mathrm{i}} = \exp(-\frac{\| x - x_i \|^2}{2\sigma_{\mathrm{i}}^{\,2}}) \quad (11)$$

In this layer the numerator and denominator neuron compute the weighted and simple sums based on the values of *w* and θ , which is $w_{ij}\theta_j$, the numerator is

$$S_j = \sum_i w_{ij} \theta_j \quad (12)$$

and denominator is

$$S_d = \sum_i \theta_j \,. \quad (13)$$

In the output layer output y is computed as

$$Y_j = S_j / S_d \quad (14)$$

We examine an GRNN Autoregressive models, which is nothing else by taking as inputs the dependent or output variable with lags.

III. DATA

For the gross domestic product we examine the period 1991-2009, where the period 1991-2006 is obtained as the in-sample or training period and 2007-2009 is taken for testing or for the out-of-sample forecasts. In the case of stock returns we obtain the year 2009 and the last 20 trading days of December are taken as testing period. It should be noticed that even if we take much longer samples for GARCH and ARMA processes the forecasting performance in the out-of-sample period is not changed.

IV. EMPIRICAL RESULTS

A. Stock Indices

The training in GRNN differs from Feed-Forward Neural Networks (FFNN), where in the last models the training involves the learning and momentum rates with delta rule for the computation of the optimum weights in input-hidden and hidden-output layers. In the case of GRNN the training is based on the sigma value in Eq. (10)-(11). Furthermore we developed a MATLAB routine in appendix, where also the GRNN training is based on weight initialization. In table I the lag order for the Autoregressive and Moving average models are reported. In tables II and III we present the correct percentage sign as well as the Root Mean squared Error (RMSE) and Mean Absolute Error (MAE). The estimations for MA and ARMA in the case of FTSE 100 are not reported because we found that the MA process is zero. Furthermore, we have an ARMA process and not ARIMA as the stock returns are always stationary. We confirmed b applying Augmented Dickey-Fuller-ADF [8]-[9].

Table I. AR, MA and ARMA processes for stock indices

Indices	*AR*	*MA*	*ARMA*
S&P 500	5	3	5,3
FTSE 100	2	0	2,0

We take for example the S&P 500. In the initial phase we estimate with the programming routine 1 in appendix and spread=1 and weight=1 with *a* and *b* equal with -0.05 and 0.05 respectively. We present the following in-sample forecasts in Fig. 1. The sigma value is found equal with 0.1556. Then we

change the weights and specifically a=-0.05 and b=0.05 to a=-0.03 and b=0.03 and we present the actual versus the forecasting values in Fig. 2.

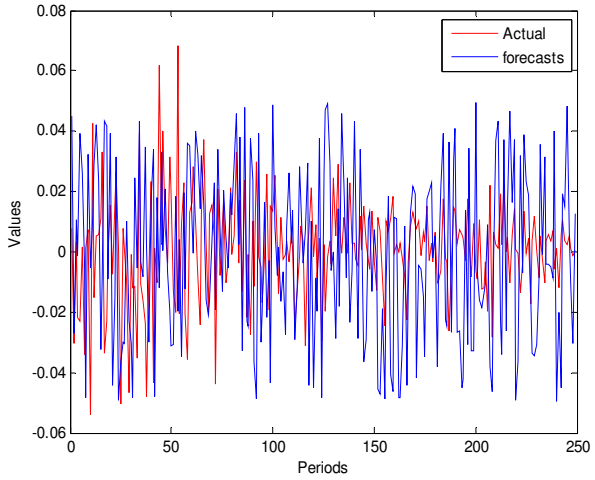

Figure 1. In-sample forecasts for S&P 500 with GRNN-AR(1), a =-0.05, b=0.05

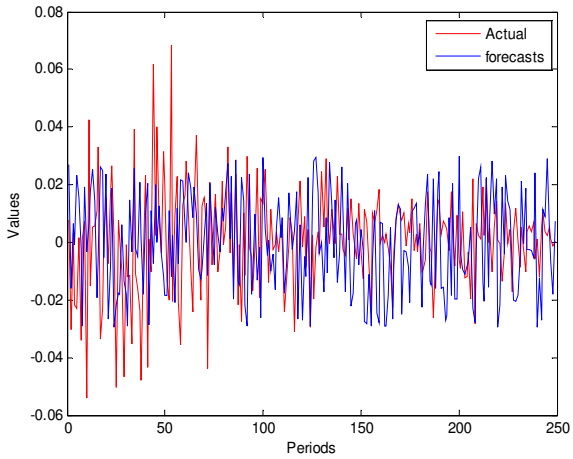

Figure 2. In-sample forecasts for S&P 500 with GRNN-AR(1), a =-0.03, b=0.03

We estimate GRNN-AR (1) which means that we have an autoregressive model of one lag. Finally, for S&P we set up sigma value at 0.4 and for the weights initialization we have set up *a* and *b* at -0.03 and 0.03 respectively. Once again we conclude that RMSE and MAE have no role in relation to correct percentage sign. To be specific RMSE and MAE values of GRNN, in the case of S&P 500, are significant higher to conventional econometric models, but the forecasting performance based on the correct sign is significant higher. On the other hand GRNN presents the lowest RMSE and MAE values in the case of FTSE 100 but again has the highest correct percentage sign. This can be easily explained by the fact that the movements of GRNN in S&P 500 are closer to actual values, but there are cases where there are extreme deviations. For example GRNN predicts the correct sign in eighth or nineteenth observation in Fig. 3. But there is a great deviation. For example the actual value in eighth observation is 0.0069 and the forecasting is 0.0269. On the other hand GARCH predicts the wrong sign, but its forecasting value is -0.00017, which is closer to 0.0069.

Table II. Forecasting Performance for S&P 500 and FTSE 100

Indices	*AR*	*MA*	*ARMA*
S&P 500			
Correct Percentage Sign	40.00	55.00	55.00
RMSE	0.0063	0.0061	0.0065
MAE	0.0053	0.0052	0.0047
FTSE 100			
Correct Percentage Sign	45.00		
RMSE	0.0088		
MAE	0.0072		

Table III. Forecasting Performance for S&P 500 and FTSE 100

Indices	*GARCH*	*GRNN-AR*
S&P 500		
Correct Percentage Sign	55.00	75.00
RMSE	0.0062	0.0122
MAE	0.0052	0.0095
FTSE 100		
Correct Percentage Sign	55.00	75.00
RMSE	0.0088	0.0074
MAE	0.0072	0.0055

We observe also the same situation regarding linear procedures. To be specific the RMSE and MAE values of AR are lower than the respective values of ARIMA in the case of S&P 500, but the correct percentage sign is significant lower. This indicates that the studies supporting some models in stock returns and exchange rates forecasting is not necessary, because the correct sign plays the major role. The financial traders are not interested in RMSE and MAE, as also they are not interesting at all about the information criteria, Log-Likelihood, autocorrelation, heteroskedasticity, residuals tests and many others, but they are interesting about the signal. The simplicity of neural networks and artificial intelligence procedures in finance is that we do not bother for econometric misspecification and residuals tests. Furthermore, a practitioner or a financial trader tests the models and chooses this one based on its forecasting performance and not on various residuals and other tests, because it is a waste of time and there is no time on this field. Also we observe that GARCH process is not superior to ARIMA, concerning the S&P 500. This indicates that even GARCH solves about autocorrelation, heteroskedasticity and ARCH effects, does not present high forecasting performance. But even if we try to forecast the volatility with GARCH process the results will be again the same.

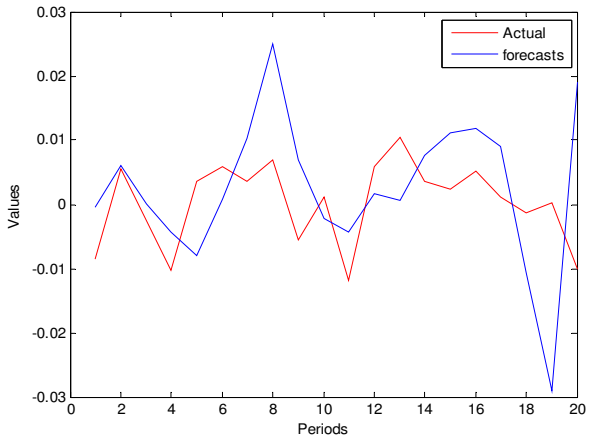

Figure 3. Out-of-sample forecasts for S&P 500 with GRNN-AR, a =-0.03, b=0.03 and sigma=0.4

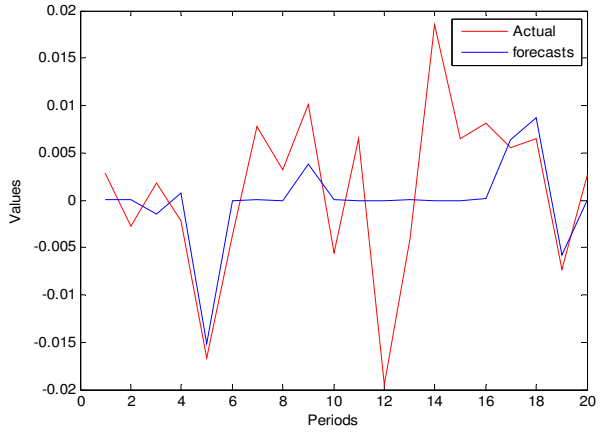

Figure 4. Out-of-sample forecasts for FTSE-100 with GRNN-AR, a =-0.01, b=0.01 and sigma=0.05

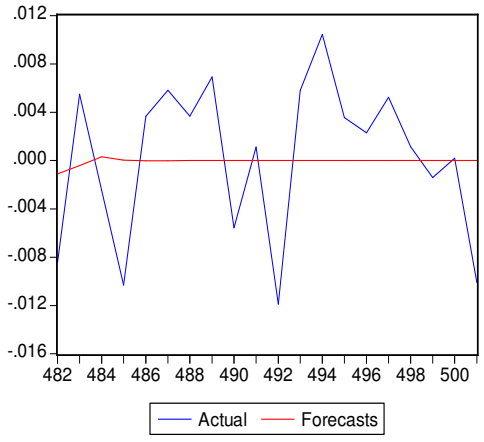

Figure 5. Out-of-sample forecasts for S&P 500 with GARCH process

In Fig. 5 we present the out-of-sample forecasts for S&P 500 with GARCH during period 2008-2009. The forecasts for the in-sample period are similar. We do not bother of presenting the forecasts of the other models, because the situation is almost the same, indicating that the forecasting performance of conventional econometric models is extremely poor. Even if we get a longer sample to satisfy the statistical properties, the forecasts are not changed at all. It should be noticed that we estimated an autoregressive AR(1) for GRNN as one input is enough to get satisfying results.

B. Gross Domestic Product

In the next example we examine the one-step ahead forecasts for the gross domestic product of Italy, UK and USA. The Autoregressive-AR process is found to be 1, 3 and 3 respectively for Italy, UK and USA, while the respective values for Moving Average-MA process are 3, 5 and 5 respectively.

In tables IV and V the in-sample and out-of-sample one-step ahead forecasts are reported. For GRNN we estimated the same autoregressive process with the respective lag values we mentioned previously for each country. The results among the linear models are mixed. To be specific AR presents the lowest RMSE and MAE values in Italy and UK, while ARIMA has the highest forecasting performance in gross domestic product growth rate of USA. In the case of Italy *a* and *b* have been set up at – 2 and 2 respectively and the sigma value found equal with 2.4321. In the case of UK we set up a=-4 and b=4 and the sigma found equal with 2.0387. Finally, for USA we set up at -1 and 1 for *a* and *b* respectively and the sigma found equal with 2.1203. We observe that in all cases GRNN outperforms the linear models. Furthermore, if we think that we have not set up at the optimum values for sigma and weights, then RMSE and MAE values can be reduced further. In a few words, if we set up different values for sigma and weights than we can get even higher forecasting performance. This is the simplicity and flexibility of neural networks and the algorithm that changing the settings the forecasts can be significant improved.

Table IV. In-Sample Forecasting performance of AR and GRNN-AR model

	In sample period 1990-2006			
	AR		**MA**	
	MAE	RMSE	MAE	RMSE
Italy	2.3555	2.9334	2.8313	3.7331
UK	2.2230	2.8528	2.2825	2.8361
USA	2.0581	2.5179	2.1797	2.5440

Table V. Out-of-Sample Forecasting performance of AR and GRNN-AR model

	Out-of sample period 2007-2009			
	ARMA		**GRNN -AR**	
	MAE	RMSE	MAE	RMSE
Italy	2.8679	3.4071	1.8485	2.1193
UK	2.4255	2.9177	1.8995	2.6360
USA	1.9961	2.2236	1.5694	1.9416

Finally, we examine again the gross domestic product, but this time we apply a four-step ahead period forecast for GDP of UK and USA. The estimating or training period is 1991-2008 and the year 2009 is left as the testing period or for out-of-sample forecasts. In Fig. 6 and 7 we present the actual values versus the best linear models. More specifically, in the case of UK we take the AR and we take ARIMA for USA.

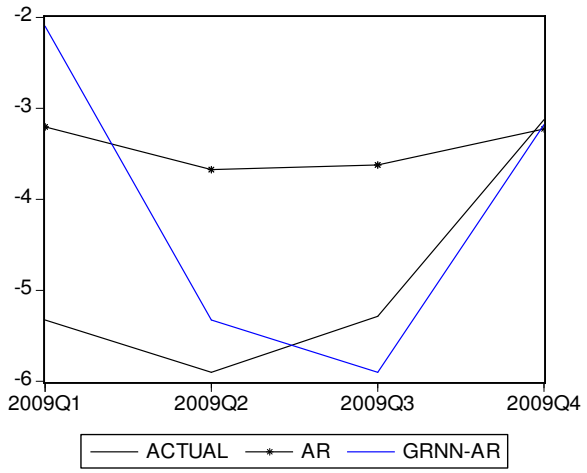

Figure 6. Out-of-sample forecasts for GDP of UK

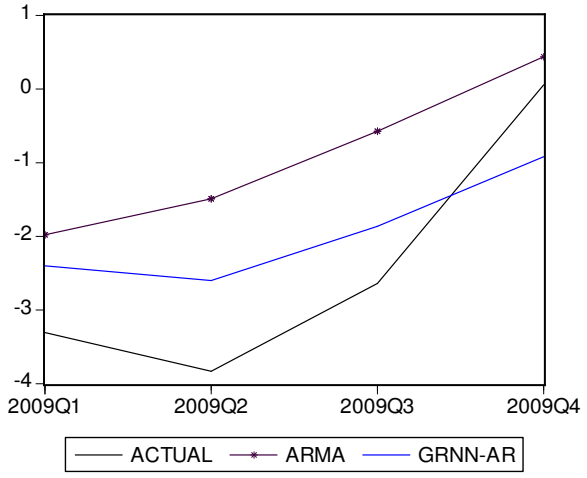

Figure 7. Out-of-sample forecasts for GDP of USA

From Fig. 6 and 7, we observe that the forecasting performance of GRNN is very high, especially in UK. On the other hand AR is a straight line. Also if we think that we might have not found the optimum values for sigma or weights then GRNN can be significant improved.

The problem with GRNN, as with any other kind of neural network model, is that the process is not straightforward. To be specific in the case of GRNN we have to set up the appropriate sigma value. For example with even larger sigma the predicted curve will get flatter more smooth as well. In some cases this can be desirable. But the value of σ depends on the time-series we examine each time. For example in stock returns we need a significant lower sigma value than we need in GDP. Furthermore, even the times-series concern the same field, for example the stock returns, the sigma value is varied among the different stock returns we examine each time. The last crisis is a proof that conventional econometric modeling, in economic institutions and financial industry, has tragically failed and there is a great need of adopting new approaches in economic academic departments and institutions, as also the introduction of new courses in the economic university departments

V. CONCLUSIONS

We examined and presented a simple Generalized Regression Neural Network Autoregressive (GRNN-AR) Model and we compared its forecasting performance with the respective of AR, MA, ARMA and GARCH models. Our findings support GRNN approach because of its flexibility to be adjusted in the actual values by changing the sigma value as also the values for the weights initialization. Furthermore, we propose additional GRNN models as GRNN Moving Average, or Autoregressive Moving Average GRNN. Also we propose genetic algorithms optimization in order to compute the optimum sigma value

VI. REFERENCES

[1] R. D. Aryal R.D. and W. Yao-Wu, W. "Neural Network Forecasting of the Production Level of Chinese Construction Industry," Journal of comparative international management, vol. 29, pp. 319-33, 2003

[2] N.R. Swanson and H. White, "A model selection approach to real time macroeconomic forecasting using linear models and artificial neural networks", Review of Economics and Statistics, vol. 79, pp. 540-50, 1997

[3] N.R. Swanson and H. White, "Forecasting economic time series using adaptive versus non-adaptive and linear versus nonlinear econometric models", International Journal of Forecasting, vol. 13, pp. 439-61, 1997

[4] A. Keles, M. Kolcak and A. "The adaptive neuro-fuzzy model for forecasting the domestic debt", Knowledge-Based Systems, vol. 21, no. 8, pp. 951-957, 2008

[5] W. H. Greene, Econometric Analysis. 6th ed.., Prentice Hall, New Jersey, 2008

[6] Bollerslev, T. "Generalized Autoregressive Conditional Heteroskedasticity", Journal of Econometrics, vol. 31, pp. 307-327, 1986.

[7] D. F. Specht "A Generalized Regression Neural Network", IEEE Transactions on Neural Networks, vol. 2, pp. 568-576, 1991

[8] D. A. Dickey and W. A. Fuller "Distribution of the Estimators for Autoregressive Time Series with a Unit Root", Journal of the American Statistical Association, vol. 74, pp. 427–431, 1979

[9] D. Kwiatkowski, P .C. B. Phillips, P. Schmidt and Y. Shin "Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root", Journal of Econometrics, vol. 54, pp. 159-178, 1992

Appendix

MATLAB script file

```
clear all;
load file.mat
nforecast=12
spread=1          % set up the sigma value. 1 for computing 2 for setting up manually
weights=1;        % Set the weight initialization. 1 and 2 for random and 3 for random
                  %and  Nguyen - Widrow initialization
default_sigma=2;  %The default spread-sigma value
u=1;              % Set the AR process for autoregresive models
for jj=nforecast:-1:1
y=data(1:end-jj,:)
d=length(y)
```

```
t=length(y)
clear x
for p=1:u
x(:,p)=lagmatrix(y,p)
end
i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
  error('Series cannot contain NaN).')
end
if any(sum(isnan(x)) == size(x,1))
  error('A realization of "x" is completely missing (all NaN"s).')
end
first_Row  =  max(sum(isnan(x))) + 1;
x          =  x(first_Row:end , :);
y=y(first_Row:end,:)
[nh,nj]=size(y);
[nk,ni]=size(x);
t=length(y)
center_matrix=x - ones(nk,1)*mean(x);
% Compute sigma
if spread==1;
maxmin= [max(center_matrix),min(center_matrix)]
distance=abs(2*max(maxmin))
sigma=distance/(sqrt(2*ni))
elseif spread==2;
  sigma =default_sigma
end
% Set up the weight matrix
if weights==1;
   rand('state',0)
a=-1
b=1
W=a + (b-a) *rand(nk,ni);

elseif weights==2;
gamma = 0.7*nj^(1/ni);
a=-0.01
b=0.01
%rand('state',sum(100*clock))            % Resets it to a different
state each time.
rand('state',0)                    % Resets the generator to its initial
state.
W=a + (b-a) *rand(nk,ni);
W = gamma*W/sqrt(sum(sum(W.^2)));
end
%-------------------------------------------%
%------------------ Training----------------%

    % compute summation neuron output
s=0
    for ii=1:ni
      for kkk=1:nk
       norm_Input = x(kkk,ii) - center_matrix(ii,:);
       norm_Res= (sqrt(norm_Input*norm_Input'))
       exp_Par = (-norm_Res / sigma^2)
       exp_Res(ii) = -exp(exp_Par)
       s=-s+exp_Res(ii)
     end
   end
     sum_Neuron = -exp_Res* W';
     yf = sum_Neuron'/s;
%-----------------------------------------------------%
%--------------------Testing-----------------------%
      p_test=y(end,:)
  for ii=1:ni
       norm_Input_test = p_test - center_matrix(end,:);
       norm_Res_test=
(sqrt(norm_Input_test*norm_Input_test'))
       exp_Par_test = (-norm_Res_test / sigma^2)
       exp_Res_test(ii) = -exp(exp_Par_test)
       s_test(ii)=-s+exp_Res_test(ii)
     end
     sum_Neuron_test = exp_Res_test* W(end-u+1:end,:);
     yhat(jj,:) = sum_Neuron_test/s_test;
end
test_y=data(end-nforecast+1:end,:)
for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end
figure, plot(y,'r-'); hold on;plot(yf,'b-');
xlabel('Periods')
ylabel('Values')
h1 = legend('Actual','forecasts',1);
%title('In_sample forecasts')
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
h1 = legend('Actual','forecasts',1);
%title('Out_of_sample forecasts')
```