A Neural Network Approach to Time Series Forecasting

Iffat A. Gheyas, Leslie S. Smith

***Abstract*—We propose a simple approach for forecasting univariate time series. The proposed algorithm is an ensemble learning technique that combines the advice from several Generalized Regression Neural Networks. We compare our algorithm with the most used algorithms on real and synthetic datasets. The proposed algorithm appears as more powerful than existing ones.**

***Index Terms*— Time series forecasting, Box-Jenkins methodology, Multilayer Perceptrons, Generalized Regression Neural Networks.**

I. Introduction

A univariate time series is a sequence of observations of the same random variable at different times, normally at uniform intervals. The goal of univariate time series data mining is to predict future values of a given variable by looking at its behaviour in the past. Share prices, profits, imports, exports, interest rates, popularity ratings of politicians, amount of pollutants in the environment and number of SARS cases over time are some of examples of time series. Lagged variables, autocorrelation and nonstationarity are the major characteristics that distinguish time series data from spatial data. The difficulties posed by these special features make forecasting time series notoriously difficult.

In time series forecasting, the magnitude of the forecasting error increases over time, since the uncertainty increases with the horizon of the forecast. When forecasting time series, interval estimates are more informative than simple point estimates. Without a doubt, the ARIMA (Autoregressive Integrated Moving Average) modelling methodology (popularized by Box and Jenkins (1976)) and the GRACH (Generalized Autoregressive Conditional Heteroskedasticity) modelling methodology (proposed by Bollerslev (1986)) are the most popular methodologies for forecasting time series and future volatility, respectively [1, 2]. Neural Networks (NNs) are now the biggest challengers to conventional time series forecasting methods [3-20]. A variety of NNs are available. However, multilayer perceptrons (MLP) with backproapagation learning are the most employed NNs in time series studies.

We present a novel approach, using a Generalized Regression Neural Networks (GRNN) ensemble to the forecasting of time series and future volatility. The remainder of this paper is organized as follows: the new algorithm is described in section 2 (with an overview of GRNN in section 2.1 and details of the proposed algorithm in section 2.2), research methodology in section 3, results and discussions in section 4, followed by summary and conclusions in section 5.

Manuscript received February 15, 2009.

I. A. Gheyas and L.S. Smith are both with the Department of Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, Scotland, UK (corresponding author is IAG, phone: +44 (0) 1786 46 7430; fax: +44 (0) 1786 467434; e-mail: iag@cs.stir.ac.uk).

II. Developing a new algorithm

We present an improved algorithm, based on GRNN, for the time series forecasting. GRNN is a neural network proposed by Donald F. Specht in 1991 [3]. This algorithm has a number of advantages over competing algorithms. GRNN is non-parametric. It makes no assumptions concerning the form of the underlying distributions. A major problem with the ARIMA and GARCH methodology and the MLP algorithm is that they are global approximators, assuming that one relationship fits for all locations in an area. Unlike these algorithms, the GRNN is a local approximator. In these algorithms local models are turned into heterogeneous forecasting models adequate to local approximation. GRNN is simpler than other existing algorithms. It has only one parameter (smoothing factor σ, where $0 < \sigma \leq 1$) that needs to be specified, but our research suggests that the performance is not very sensitive to the parameter σ. However, we face a dilemma when applying the GRNN to the time series forecasting task. If we provide only the most recent past value, the GRNN generates the smallest forecasting error but does not accurately forecast the correct direction of change. On the other hand, if we provide multiple past observations, the GRNN can forecast the direction of change correctly, but the forecasting error appears to proportionally increase with an increasing number of input values. In order to overcome this problem, we propose a derivative of the GRNN, which we call GRNN ensemble. Using the MLP, the ARIMA & the GARCH methodologies, trial-and-error methods are applied in order to secure the best fitting model. The advantage of the proposed algorithm is that it is very simple to implement, neither the trial-and-error process nor prior knowledge about the parameters is required.

A. Generalized Regression Neural Networks

In GRNN (Specht, 1991) each observation in the training set forms its own cluster [3]. When a new input pattern x is presented to the GRNN for the prediction of the output value, each training pattern y_i assigns a membership value h_i to x based on the Euclidean distance $d = d(x, y_i)$ as in equation 1.

$$h_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d^2}{2\sigma^2}\right) \qquad (1)$$

Where σ= smoothing function parameter (we specify a default value: $\sigma = 0.5$).

Finally, GRNN calculates the output value *z* of the pattern *x* as in equation (2).

$$z = \frac{\sum_i (h_i \times \text{output of } y_i)}{\sum_i h_i} \qquad (2)$$

If the output variable is binary, then GRNN calculates the probability of event of interest. If the output variable is continuous, then it estimates the value of the variable.

B. Proposed Algorithm

We propose a novel GRNN-based ensemble algorithm for time series forecasting. Two GRNN ensembles, *A* and *B*, are built. A GRNN ensemble is a collection of GRNNs that work together. To make a forecast, every member of the ensemble forecasts the output value. The overall output of the ensemble is the weighted average of the outputs produced by the member GRNNs. The GRNN ensemble *A* forecasts the expected future value, and the GRNN ensemble *B* forecasts the expected future volatility of the time series. The trained GRNN ensemble *A* and the trained GRNN ensemble *B* are used to make successive one step ahead forecasts. This is done by rolling the sample forward one time step, using the forecast as an input, and making another one-step-ahead forecast and so on.

Pseudo code of the proposed algorithm

// Construct the GRNN ensemble *A* for forecasting conditional mean

- **Stationarize the series:** Transform a non-stationary time series into a stationary one by using a logarithmic or square root transformation and differencing.
- **Normalize** the values of stationary time series in the range of 0 to 1.
- **Selection of input variables:** Measure the periods of waveforms (that represents the time lags between two succeeding peaks or troughs) found in the whole observation period. Calculate the weighted average (N_A) of all of the periods, where recent waveform period carry more weight than those in the past. Select the N_A most recent past values for the current value of the series.
- **Create the GRNN ensemble *A*** with N_A separate GRNNs. Each GRNN is connected with a single input node. The input variable of each network is different.
- **Train each member GRNN** on the past values of the stationary time series data.
- **Estimate weights of each member GRNN:** Present training patterns to each GRNN separately for forecasting purposes and calculate weights for each GRNN:

$$W_j^A = 1 - \frac{\sum_{i=1}^{N} \left| \frac{(Z_i - \hat{Z}_{ij})}{Z_m} \right|}{N}, \quad j \in 1, \cdots, T_A \quad m \le N \quad (3).$$

where, W_j^A = weight of the *j-th* member GRNN of the ensemble *A*, Z_i =Actual output of the *i*-th pattern, $\hat{Z}_{ij}$ =output of the *i*-th pattern predicted by the *j*-th member GRNN, Z_m = maximum actual output in the training set, and *N* = number of observations.

- **Final output of the GRNN-ensemble *A*:** The final output Y_A of GRNN ensemble *A* is the weighted average of the outputs of all its member GRNNs.
- **Back-transform** the forecasted conditional means into the original domain.

// Construct the GRNN ensemble *B* for forecasting conditional variance

- **Create the training dataset for the ensemble *B*:** Present training patterns to the GRNN ensemble *A* for prediction purposes and find the residual series a_t by subtracting the predicted value $\hat{Z}_t$ from the actual value Z_t : $a_t = Z_t - \hat{Z}_t \qquad (4)$.

Normalize the squared residuals to lie from 0 to 1.

- **Identify predictors for the ensemble *B*:** Count the number of waves in the squared residual series and their associated periods. Calculate the weighted average of periods (N_B). Select the N_B most recent past squared residuals for the ensemble *B*.
- **Construct the GRNN ensemble *B*** with N_B separate GRNNs. Each GRNN consists of a single input node. The input of each member GRNN is a different lagged squared residual.
- **Estimate weight of each GRNN:** Present training patterns of the square residual series to each GRNN of the ensemble *B* for forecasting purposes and estimate weights for each member GRNN as in equation (5):

$$W_j^B = 1 - \frac{\left(\sum \left| \frac{(a_i - \hat{a}_{ji})}{a_m} \right| \right)}{k} \qquad (5)$$

where W_j^B = weight of the member GRNN of the ensemble *B*, a_i = actual square residual, $\hat{a}_{ji}$ = square residual predicted by the *j*-th GRNN of the ensemble *B*, a_m = maximum actual residual in the training set, and *k* = number of data points.

The final output (predicted square residual) of the ensemble *B* is the weighted average of predictions of the member GRNNs.

- **Calculate conditional variance** at time lag *t* (where *t*=1, 2, 3….):

$$\text{Conditional variance at time lag t} = \text{predicted squared residual} \times \text{time lag t} \qquad (6)$$

- **Compute 95% confidence intervals (CI)** associated with the conditional mean (predicted by GRNN Ensemble *A*) as in equation (7), assuming that the time series variable has a normal distribution:

$$95\%\ \text{CI associated with the expected value at lag t} = \text{conditional mean} \pm 1.96\sqrt{(\text{conditional variance})} \qquad (7)$$

III. Research Methodology

An observed time series can be decomposed into four components: (i) Mean value of the data: We arbitrarily choose the mean value, (ii) Long term trend: We used linear, exponential, logarithmic, or polynomial functions to estimate the trend at time point *t*, (iii) Cyclical change (Seasonality/Periodicity): We estimate the periodicity s_t at time lag *t* using a sinusoidal model, and (iv) Noise. The principle of generating synthetic time series datasets is to first estimate the values of these four components using mathematical models and then combine the values of components into one based on the additive, multiplicative, or additive-multiplicative decomposition model. Figures 1-2 show samples of synthetic data.

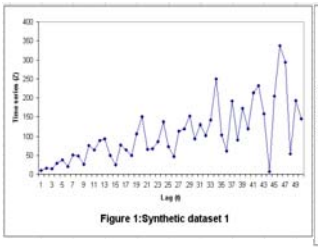
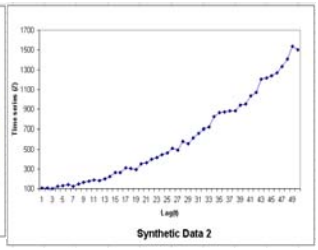


Figure 1:Synthetic dataset 1



Synthetic Data 2

IV. Results and Discussions

We compare our proposed algorithm (GRNN-Ensemble) with existing algorithms (ARIMA-GARCH methodology, MLP, GS (GRNN with a single predictor) and GM (GRNN with multiple predictors)) on thirty synthetic datasets and ten real-world datasets. The real-world datasets (obtained from http://statistik.mathematic.uniwuerzburg.de/timeseries) are the Airline Data, the Bankruptcy Data, the Electricity Data, the Hong Kong Data, the Nile Data, the Share Data, the Star Data, the Car Data, the Sunspot Data, and the Temperatures Data. The algorithms are applied to make 10-step-ahead out-of-sample forecasts. These algorithms were ranked in terms of their accuracy in the interval estimation, and interval length. We assign rank 1 to the best algorithm, the rank 2 to the next best algorithm and so on. Table 1 summarizes the results. Obviously, the higher the accuracy the better. The lower the average interval length, the better the performance of the algorithms and the lower the standard deviation, the more consistent and reliable the algorithm. Appendix Tables A1-A4 give an overview of the statistical test results.

Key Findings:

- GRNN-ensemble is statistically significantly superior to the other four algorithms both at very short horizons (one step-ahead) and at longer horizons (five and ten step-ahead).

- The GRNN with multiple predictors perform significantly worse compared with the other algorithms at all three forecast horizons. It is difficult to say for sure what causes this algorithm to generate a bad performance. One possible cause would be that the GRNN does not assign weights to the input variables. Lagged input variables are highly correlated and they might make each other redundant to a great extent. These algorithms match patterns based on the Euclidean distance between input patterns and stored reference pattern. The distance increases many fold above the actual distance as the number of input variables increases. We also tried with the Mahalanobis distance providing correlation weighted distance but the performance did not improve. Our initial suspect is the high degree of redundancy among predictors. Multicollinearity among the predictors may lead to systematic overestimation of Euclidean (and Mahalanobis) length. This issue deserves further investigation. However, our empirical results demonstrate that our new algorithm (GRNN-Ensemble) does not suffer from this shortcoming anymore.

V. Summary and Conclusion

We propose a simpler and more efficient algorithm (GRNN ensemble) for forecasting univariate time series. We compare GRNN ensemble with existing algorithms (ARIMA & GARCH, MLP, GRNN with a single predictor and GRNN with multiple predictors) on forty datasets. The one-step process is iterated to obtain predictions ten-steps-ahead. The results obtained from the experiments show that the GRNN ensemble is superior to existing algorithms.

References

[1] G.E.P. Box, and G.M. Jenkins, *Time series analysis: forecasting and control*, SAN Francisco: Holden-Day, 1976.

[2] T. Bollerslev. (1986). Generalized autoregressive conditional heterskedasticity. *Journal of Econometrics*. 31, 307-327.

[3] D.F.A. Specht. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*. 2, 568-576.

[4] R. Aliev, B. Fazlollahi, and B. Guirimov. (2008). Linguistic time series forecasting using fuzzy recurrent neural network. *Soft Computing.* 12(2), 183-190.

[5] R. Aliev, B. Fazlollahi, R. Aliev, and B. Guirimov, (2006). Fuzzy time series prediction method based on fuzzy recurrent neural network. *Lecture Notes in Computer Science*. 4233, 860-869.

[6] J. Hwang, and E. Little. (1996). Real time recurrent neural networks for time series prediction and confidence estimation. *IEEE International Conference on Neural Networks*. 4, 1889-1894.

[7] M. Huken, and P. Stragge. (2003). Recurrent neural networks for time series classification. *Neurocomputing*. 50, 223-235.

[8] V. Lopez, R. Huerta, and J.R. Dorronsoro, (1993). Recurrent and feedforward polynomial modelling of coupled time series. *Neural Computation.* 5(5), 795-811.

[9] P. Coulibaly, and F. Anctil. (2000). Neural network-based long term hydropower forecasting system. *Computer-Aided Civil and Infrastructure Engineering*. 15, 355-364.

[10] R. Drossu, and Z. Obradovic, (1996). Rapid design of neural networks for time series prediction. *IEEE Computational Science & Engineering.* 3(2), 78-89.

[11] G.P. Zhang, and M. Qi, (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research.* 160, 501-514.

[12] A.J. Conway. (1998). Time series, neural networks and the future of the Sun. *New Astronomy Reviews.* 42, 343-394.

[13] X. Yan, Z. Wang, S. Yu, and Y. Li., "Time series forecasting with RBF neural network". In: *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics* Guangzhou., August 2005.

[14] E.S. Cheng, S. Chen, and B. Mulgrew. (1996). Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *IEEE Transactions on Neural Networks.* 7, 190-194.

[15] G. Lin, and L. Chen. (2005). Time series forecasting by combining the radial basis function network and the self-organizing map. *Hydrological Processes*. 19, 1925-1937.

[16] H.K. Cigzoglu. (2005). Application of generalized regression neural networks to intermittent flow forecasting and estimation. *Journal of Hydrologic Engineering*. 10(4), 336-341.

[17] T. Yildirim, and H.K. Cigizoglu, "Comparison of generalized regression neural network and MLP performances on hydrologic data forecasting". In: *Proceedings of the 9th International Conference on Neural Information Processing*, vol.5, 2002.

[18] M.N. Islam, L.K.K. Phoon, and C.Y. Liaw, 2000. "Forecasting of river flow data with a general regression neural network". In: *Proceedings of International symposium on integrated water resource management*, 2001(272), 285-290.

[19] H.K. Cigizoglu, and M. Alp, 2005. Generalized regression neural network in modelling river sediment yield. Advances in Engineering Software, 37(2), 63-68.

[20] A. Chen, and M.T. Leung, 2004. Regression neural network for error correction in foreign exchange forecasting and trading. Computers & Operations Research, 31, 1049-1068.

[21] S. Singel, N.J. JR. Castellan, Nonparametric *statistics: for the behavioral sciences*, New York: McGraw-Hill, Second Edition.

Table 1: Summary Results

		One-step-ahead forecast			Five-Step-ahead forecast			Ten-step-ahead forecast		
	Accuracy (%)	All data	Synthetic data	Real data	All data	Synthetic data	Real data	All data	Synthetic data	Real data
GRNN ensemble (GE)	Rank	1			1			1		
	Mean	98	98	98	97	97	97	94	94	93
	STD	2	1	2	2	2	2	4	4	4
	Max	100	100	100	100	100	99	100	100	96
	Min	95	95	95	94	93	93	84	84	86
	CI width	10	10	11	23	22	28	43	43	45
GRNN with a single predictor (GS)	Rank	2			4			4		
	Mean	91	91	90	88	88	74	80	81	77
	STD	4	4	3	6	7	9	8	7	10
	Max	98	98	96	98	98	94	97	97	92
	Min	83	83	87	81	71	58	64	64	64
	CI width	13	13	14	31	35	22	49	51	46
GRNN with multiple predictors (GM)	Rank	5			5			5		
	Mean	82	82	84	76	74	83	72	71	74
	STD	7	7	6	10	9	10	12	13	9
	Max	95	95	95	94	94	94	94	94	91
	Min	70	70	71	61	58	58	37	37	63
	CI width	20	20	17	44	44	45	62	59	71
ARIMA & GARCH (A&G)	Rank	3			3			3		
	Mean	91	91	90	89	90	85	82	82	80
	STD	5	5	4	7	7	5	9	8	10
	Max	100	100	94	98	98	91	100	100	92
	Min	82	83	82	75	75	75	57	62	57
	CI width	17	16	20	28	27	25	45	45	42
MLP	Rank	4			2			2		
	Mean	90	90	90	90	90	92	84	85	83
	STD	5	4	5	5	5	3	10	9	13
	Max	99	99	97	100	100	95	100	100	99
	Min	79	81	79	80	80	80	63	67	63
	CI width	17	19	13	30	31	25	49	46	56

Appendix

Table A1: Friedman two-way analysis of variance by ranks

Hypothesis	Test Statistic	Test Result
Ho: There is no difference in rank totals of the 5 algorithms in the case of one-step-ahead-forecasting Ha: A difference exists in rank totals of the 5 algorithms in one-step-ahead forecasting.	*N*=40 Chi-square=107.302 df=4 Asymp. Sig.=0.000	Reject the null hypothesis and conclude that there is a difference in the performance of one-step-ahead-forecast across algorithms with *p<0.005*
Ho: There is no difference in rank totals of the 5 algorithms in the case of five-step-ahead-forecasting Ha: A difference exists in rank totals of the 5 algorithms in five-step-ahead forecasting.	*N*=40 Chi-square=96.840 df=4 Asymp. Sig.=0.000	Reject the null hypothesis and conclude that there is a difference in the performance of five-step-ahead forecasting across algorithms with *p<0.005*
Ho: There is no difference in rank totals of the 5 algorithms in the case of ten-step-ahead-forecasting Ha: A difference exists in rank totals of the 5 algorithms in ten-step-ahead forecasting.	*N*= 30 Chi-square=76.100 df = 4 Asymp. Sig = 0.000	Reject the null hypothesis and conclude that there is a difference in the performance of ten-step-ahead forecasting across algorithms with *p<0.005*

Table A2: Results of Multiple Comparison Tests (in one-step-ahead forecasting)

	GRNN- Ensemble	GS	GM	ARIMA & GARCH	MLP
GRNN-Ensemble	-	Yes	Yes	Yes	Yes
GS	Yes	-	Yes	No	No
GM	Yes	Yes	-	Yes	Yes
ARIMA & GARCH	Yes	No	Yes	-	No
MLP	Yes	No	Yes	Yes	

** 'Yes' indicates a significant difference in rank totals between two algorithms at the 5% level of significance, while 'No' indicates no significant difference.*

Table A3: Results of Multiple Comparison Tests (in five-step-ahead forecasting)

	GRNN-Ensemble	GS	GM	ARIMA & GARCH	MLP
GRNN- Ensemble	-	Yes	Yes	Yes	Yes
GS	Yes	-	Yes	Yes	No
GM	Yes	Yes	-	Yes	Yes
ARIMA & GARCH	Yes	Yes	Yes	-	No
MLP	Yes	No	Yes	No	

** 'Yes' indicates a significant difference in rank totals between two algorithms at the 5% level of significance, while 'No' indicates no significant difference.*

Table A4: Results of Multiple Comparison Tests (in ten-step-ahead forecasting)

	GRNN ensemble	GS	GM	ARIMA & GARCH	MLP
GRNN-Ensemble	-	Yes	Yes	Yes	Yes
GS	Yes	-	No	No	No
GM	Yes	No	-	Yes	Yes
ARIMA & GARCH	Yes	No	Yes	-	No
MLP	Yes	No	Yes	No	

- *'Yes' indicates a significant difference in rank totals between two algorithms at the 5% level of significance, while 'No' indicates no significant difference.*