Using Recurrent Neural Networks for Time Series Forecasting

Sandy D. Balkin
Pennsylvania State University
Department of Management Science and Information Systems [*]
University Park, Pennsylvania 16802
e-mail: sxb31@psu.edu

Abstract. In the past few years, artificial neural networks (ANNs) have been investigated as a tool for time series analysis and forecasting. The most popular architecture is the multilayer perceptron, a feedforward network often trained by back-propagation. The forecasting performance of ANNs relative to traditional methods is still open to question although many experimenters seem optimistic.

One problem with the multilayer perceptron is that, in its simplest form, it is similar to a pure autoregressive type model, so it lacks the ability to account for any moving average structure that may exist. By making a network recurrent, it is possible to include such structure.

We present several examples showing how an ANN can be used to represent an ARMA scheme and compare the forecasting abilities of feedforward and recurrent neural networks with traditional methods.

[*]Working Paper Series number 97-11. This paper was presented at the 1997 International Symposium on Forecasting, Barbados.

1. Introduction

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way the brain processes information. 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 (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. For an introduction to ANNs see, for example, Hinton (1988), Lippmann (1987), or Cheng and Titterington (1994).

Recently, ANNs have been investigated as a tool for time series forecasting. Examples include Tang, et al. (1991), Hill, et al. (1996), and Faraway and Chatfield (1995). The most popular class is the multilayer perceptron, a feedforward network trained by backpropagation. This class of network consists of an input layer, a number of hidden layers and an output layer. Figure 1 is an example of an ANN of this type with 3 neurons in the input layer, a single hidden layer with 2 neurons, and 1 neuron in the output layer. The first step in using an ANN concerns the choice of structure, known as the architecture of the network. Neural Networks are known to be universal function approximators (Hornik, Stinchcombe, & White, 1989), however, for an ANN to perform well, the inputs and number of nodes in the hidden layer must be carefully chosen. Since they learn by example, it is vital that the inputs characterize the important relationships and correlations in the time series being modeled.

For a feedforward neural network with one hidden layer, the prediction equation, given by Faraway and Chatfield (1995), for computing forecasts $\hat{x}_t$ using selected past observations, $x_{t-j_1}, \ldots, x_{t-j_k}$, at lags $(j_1, \ldots, j_k)$ and h nodes in the hidden layer will be referred to as $NN[j_1, \ldots, j_k; h]$. Thus, the network in Figure 1 is $NN[1, 12, 13; 2]$. The functional form

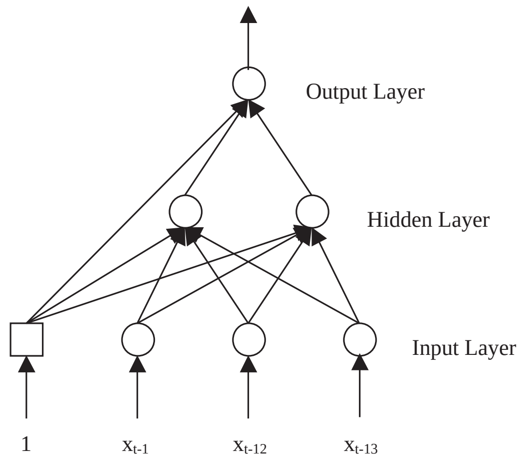

Figure 1: *A feedforward neural network*

may be written as

$$\hat{x}_t = \phi_o(w_{co} + \sum_h w_{ho}\phi_h(w_{ch} + \sum_i w_{ih}x_{t-j_i}))$$

where $\{w_{ch}\}$ denote the weights for the connections between the constant input and the hidden neurons and w_{co} denotes the weight of the direct connection between the constant input and the output. The weights $\{w_{ih}\}$ and $\{w_{ho}\}$ denote the weights for the other connections between the inputs and the hidden neurons and between the neurons and the output respectively. The two functions ϕ_h and ϕ_o denote the activation functions that define the mappings from inputs to hidden nodes and from hidden nodes to output(s) respectively. The logistic function, $y = \phi(x) = \frac{1}{1+e^{-x}}$ is widely used in the ANN literature. Faraway and Chatfield (1995) recommend, for forecasting applications, that ϕ_h be logistic and ϕ_0 linear. Forecasts are generated iteratively by performing successive one-step ahead forecasts using previous forecasts as estimates of observables. This approach represents a neural network implementation of a *non-linear autoregressive* (*NLAR*) model of order k as $\widehat{x}_t = \widetilde{f}(x_{t-j_1}, x_{t-j_2}, \ldots, x_{t-j_k}, \varepsilon_{t)}$ where $\{\varepsilon_t\}$ is a sequence of IID random variables and

$\widetilde{f} : \Re^{k+1} \rightarrow \Re$. It should be noted that the literature on ANNs rarely speaks in terms of random variables, or of any assumptions about them. However, almost all applications use the least squares criterion for fitting so that, by implication, the errors are assumed to be at least uncorrelated with equal variances. Further, the use of ANNs for forecasting requires constancy of model structure for future observations, implying temporal invariance of $\widetilde{f}$.

This *NLAR* network, in its simplest form, will have zero nodes in the hidden layer. The resulting prediction equation for computing a forecast for x_t becomes

$$\hat{x}_t = \phi_o(w_{co} + \sum_i w_{io} x_{t-j_i})$$

where ϕ_o is a linear function, w_{co} denotes the weight of the direct connection between the constant input and the output and $\{w_{io}\}$ are the weights of the connections between the input neurons and output neuron. This equation corresponds to the traditional autoregressive *(AR)* forecasting function

$$\hat{x}_t = c + \phi_1 x_{t-j_1} + \phi_2 x_{t-j_2} + \cdots + \phi_p x_{t-j_k}.$$

were k is the number of input nodes in the corresponding neural network at lags $(j_1, \ldots, j_k)$. So, in its simplest form, the neural network architecture reduces to the standard *AR* type model. The benefit of implementing a neural network is that it can account for nonlinear in addition to linear relationships between the inputs and output when a hidden layer is included.

A problem now arises when comparing neural networks and traditional time series methods in that the latter can account for moving average structure in addition to autoregressive structure. Although the neural network model with a non-zero number of nodes in the hidden layer can account for non-linear structure, it cannot model a moving average as in an *ARMA* scheme. Since an *MA* model can be expressed as an infinite order *AR* model, some moving average structure may be accounted for by using a neural network with a large number of inputs. However, trying to model an *MA* series in such a way is hampered by both short series and long computational time. So, to be able to make a fair comparison between neural network and *ARMA* forecasts, the neural network should be able to reduce to an *ARMA* form. Thus, it is necessary to consider another class which does account for moving average structure in a series.

This paper presents the recurrent neural network for modeling and forecasting time series. Section 2 of this paper reviews the Box-Jenkins method and then introduces recurrent neural networks and shows how they can be reduced to represent *ARMA* type models in Section 3. Section 4 shows empirical results comparing *ARMA*, feedforward and recurrent neural network forecasting abilities on some benchmark series. Section 5 presents the conclusions and suggests some areas for future work.

2. Box-Jenkins Method

The Box-Jenkins method is a well known paradigm used to identify the moving average, autoregressive and seasonal components of a stationary time series. The following is an explanation of model types and how they are integrated into a single linear model framework.

The first type of linear model is called the *moving average.* A moving average process of order q is characterized by

$$x_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q}$$

which is a weighted average of random shocks spanning q periods. Each of the random shocks is assumed to be independent and identically distributed. A moving average model is used when there is a linear dependence on past performance. It is interesting to note that the system has a q-period memory meaning that a random shock persists for only q periods.

A second type of linear model is the *autoregressive* scheme. An autoregressive process of order p is characterized by

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \varepsilon_t$$

which is a weighted average of the past performance of a time series. Such a model is used when the change in the series at any point in time is linearly correlated with previous changes.

Combining the two models above results in the *mixed autoregressive-moving average* model characterized by

$$x_t = c + \phi_1 x_{t-1} + \ldots + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q}.$$

This type of scheme is used when both moving average and autoregressive tendencies are present.

This can be extended into a seasonal $ARMA$(P,Q) model when defined as

$$x_t - \Phi_1 x_{t-s} - \ldots - \Phi_P y_{t-P_s} = \delta_t - \Theta_1 \varepsilon_{t-s} - \ldots - \Theta_Q \varepsilon_{t-Q_s} + \varepsilon$$

In general, when allowing for the series to be transformed and differenced, the Box-Jenkins method provides guidelines to follow when choosing the parameters to identify a model of the form:

$$\phi(B)\Phi(B^s)\nabla^d \nabla_s^D (x_t - c) = \theta(B)\Theta(B^S)\varepsilon_t$$

The experimenter identifies several possible models and then chooses which is best based upon a set of diagnostics. Forecasts are then based on the selected model. A more detailed explanation of the concepts just presented can be found in Box, et al. (1994).

The linear models can be extended to non-linear models. In Section 1, we saw that $\widehat{x}_t$ follows a nonlinear autoregressive model if there exists a function $\widetilde{f} : \Re^{p+1} \rightarrow \Re$ such that $\widehat{x}_t = \widetilde{f}(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \varepsilon_{t)}$, $t \in \mathbf{Z}$, where $\{\varepsilon_t\}$ is a sequence of IID random variables. Similarly, we may have a *non-linear moving average* (NLMA) model where $\widehat{x}_t = \widetilde{g}(\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q})$. Combining the two non-linear modeling schemes results in the *non-linear autoregressive moving average* (NLARMA) model $\widehat{x}_t = \widetilde{h}(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, \varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q})$.

3. Recurrent Neural Networks

Recurrent Neural Networks are useful in situations when there is a temporal (time dependent) relationship in data. Classes of this architecture type were introduced in Jordan (1986) and Elman (1990). They are constructed by taking a feedforward network architecture and adding feedback connections to previous layers. Such networks are trained by the standard backpropagation algorithm except that patterns must always be presented in time sequential order (see the seminal paper by Rumelhart, Hinton and Williams (1986) for a description of the backpropagation algorithm). The one difference in structure is that there are some extra nodes next to the input layer that are connected to the hidden layer just like the other input nodes. These extra nodes hold the contents or representation of the contents of one of the layers as it existed when the previous pattern was trained. In this way the network sees previous knowledge it had about previous inputs. This extra set of nodes are called the network's *context units* and are sometimes referred to as a network's *long term memory* for reasons explained later.

The primary types of recurrent networks are:

- *Input Layer Fed Back Into Input Layer:* The context units remember the new input data and use it when the next pattern is processed.
- *Hidden Layer Fed Back Into Input Layer:* The context units remember the hidden layer, which contains features detected in the raw data of previous patterns. It is also known as an *Elman Recurrent Network.*
- *Output Layer Fed Back Into Input Layer*: The context units remember outputs previous network outputs. It is also known as a *Jordan Recurrent Network.*

Figure 2 is an example of a Jordan recurrent neural network with the network output fed back into the context unit and Figure 3 is an example of an Elman recurrent network with the hidden layer fed back into the context layer. The adjustment of the weights in each

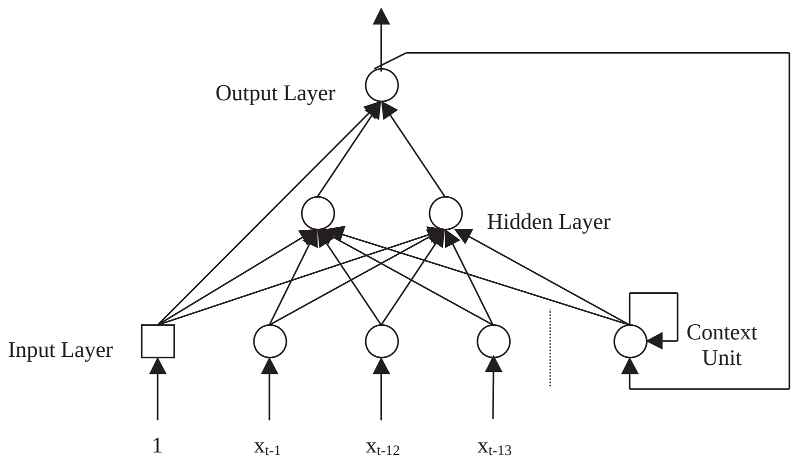

Figure 2: *A Jordan recurrent neural network*

successive iteration of the learning algorithm is a function of both the input vector as well as a compressed representation of all preceding inputs. If there is no temporal structure in the data, using a recurrent network will not work as well as a feedforward network because the long term memory nodes will be considered random noise to the net. If a recurrent network does not work well, there may not be time dependencies in the data.

It has been shown that a feedforward neural network has the ability to represent an AR type model. To use such a network to account for moving average structure, the investigator can use the ability to represent an MA scheme as an infinite order AR type model, however, trying to model a series in this fashion will result in a computationally infeasible number of nodes. However, adding recurrence to a network can accomplish the same end with far fewer processing nodes.

The forecasting function for a recurrent neural networks may be represented as

$$\hat{x}_t = \phi_o(w_{co} + \sum_h w_{ho}\phi_h(w_{ch} + \sum_i w_{ih}x_{t-j_i} + \sum_j w_{lh}h_{j,t-1}))$$

where all coefficients are as previously defined with addition of w_{lh} which denotes the weights

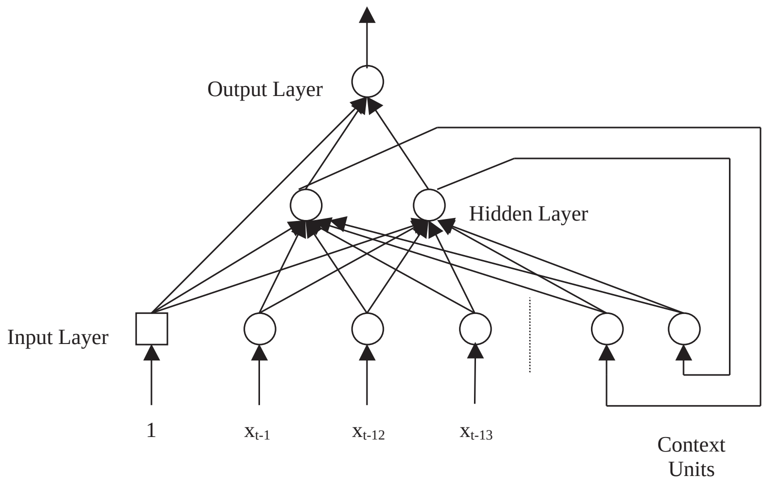

Figure 3: *An Elman recurrent neural network*

for connections between context and hidden neurons and $h_{j,t-1}$ which denotes the value of the network output from the previous time unit from a Jordan recurrent network or the value of hidden node j from the previous time unit from an Elman recurrent network.

Modified or extended versions of these architectures define $h_{j,t-1}$ as a convex function of the output from the recurrent nodes and value of the context units from the previous time unit. For example, the value of the context unit for a Jordan recurrent neural network is given by

$$h_{j,t-1} = \lambda_1 \widehat{x}_{t-1} + \lambda_2 h_{j,t-2}$$

where $\lambda_1, \lambda_2 \in [0,1]$ and $\lambda_1 + \lambda_2 = 1$. In this architecture, the context unit contains a compressed version of the long term behavior of the network along with a portion of the short term information. The lambda values, called decay parameters, are adjusted according to the importance the investigator puts on either the long or short term behavior information. In this case, the context unit serves as a simple lowpass filter whose output is a weighted value of some of its more recent past inputs. In the case of the Jordan context unit, the output is obtained by summing the past values multiplied by a scalar which decays exponentially over time. These context units are called long term memory units, because they remember past events. In general, λ_2 is set to $1 - \frac{1}{k}$; $k > 0$, where k is the number of time samples to remember.

Once again, consider the simplest neural network of this type. This network, shown in Figure 4, will have 1 input neural, 0 neurons in the hidden layer, 1 neuron in the output layer and 1 context unit. In Bulsari and Saxen (1993), it is shown that its forecasting function is

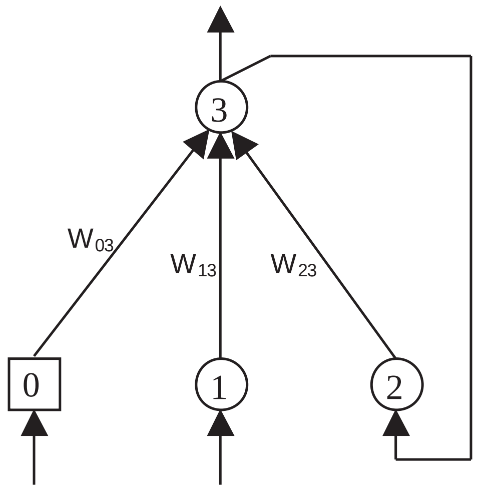

Figure 4: *A simple recurrent neural network*

given by

$$\hat{x}_t(1) = w_{03} + w_{13}x_t + w_{23}h_t$$

which can be transformed as:

$$\begin{aligned}\hat{x}_t(1) &= w_{03} + (w_{13} + w_{23})x_t - w_{23}(x_t - h_t) \\ &= w_{03} + (w_{13} + w_{23})x_t - w_{23}(x_t - \hat{x}_t) \\ &= w_{03} + (w_{13} + w_{23})x_t - w_{23}\varepsilon_t\end{aligned}$$

If we let $c = w_{03}$, $\phi = w_{13} + w_{23}$ and $\theta = w_{23}$, then

$$\hat{x}_t(1) = c + \phi x_t - \theta \varepsilon_t$$

which is the forecasting equation for an $ARMA(1,1)$ scheme. Further AR terms can be added by including the desired lags as inputs. For example, Figure 5 is a neural network representation of an $ARMA(2,1)$ scheme. It is important to mention that, with this type of architecture, namely one direct connection from the output neuron to the context unit, we are restricted to a single order moving average structure. It is also important to note

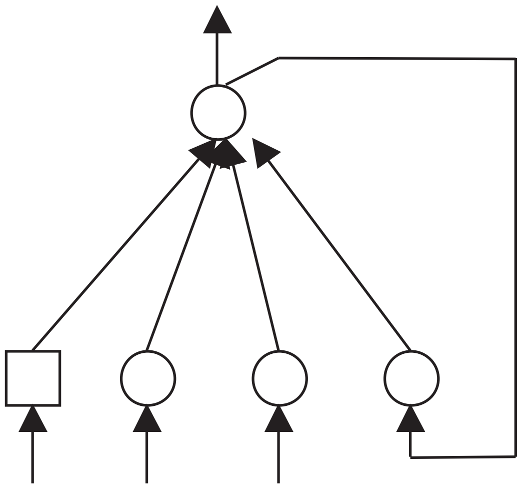

Figure 5: *A neural network representation of an ARMA(2,1) scheme*

that although what has been presented are neural network representations of $ARMA$-type schemes, the neural network coefficients are not constrained by stationarity conditions as they are for the conventional linear model.

4. Examples

For the first example, consider the Box-Jenkins airline series. Figure 6 is a plot of the series. The Box-Jenkins analysis of this monthly series entails taking one seasonal and one non-

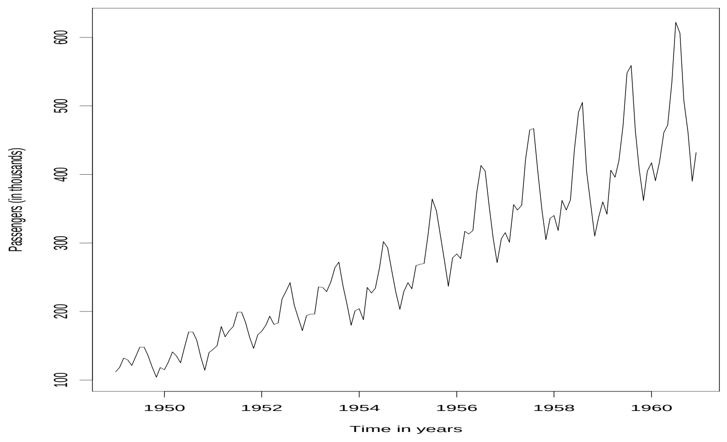

Figure 6: *Box-Jenkins Series G : International Airline Passengers : Monthly Totals for January 1949-December 1960*

seasonal difference and moving average of the logarithm of the original series. This model is known as the *airline model* and is written $(0,1,1) \times (0,1,1)_{12}$. The following is a summary table of the forecasts generated by the various methods being compared. The model is fit on the first 132 data points and used to forecast the remaining 12. The neural network methods were performed on the original series and incorporated the same lags (1, 12 and 13) as the airline model using a Jordan neural network with decay parameters set as $\lambda_1 = 0.6$ and $\lambda_2 = 0.4$ for the *RNN*. The table summarizes the accuracy of the forecasts shown in Figure 7.

Modeling Type	*Box-Jenkins*	*Feedforward ANN*	*Recurrent ANN*
Model	$(0,1,1) \times (0,1,1)_{12}$	*lags=1, 12, 13*	*lags=1, 12, 13*
# Hidden Neurons	NA	2	2
# Parameters	2	11	13
SS_{RES}	1.079	0.9994	1.1751
$\widehat{\sigma}$	0.095	0.0962	0.1053
AIC	-555.7	-546.786	-523.5157
BIC	-546.1	-505.216	-487.28
FMSE	315.3	310.5	306.3
FMAD	12.5	14.1	12.9

Although the within sample forecasts are not as good as the other methods, the recurrent network achieves the best mean out-of-sample forecasts. From the plot, it is interesting to note that each method does better than the rest for some forecasted values, but based on the FMSE, the recurrent network provides marginally better results. The FMAD tells us that the Box-Jenkins and recurrent network methods are comparable while the feedforward network performed significantly worse.

For another example, consider the Box-Jenkins Series A, chemical process concentration readings at 2 hour intervals, shown in Figure 8. Box et al. (1994) analyze this series and choose an $ARMA(1,1)$ as a parsimonious model. For comparison, a feedforward neural network with one input at lag 1, two nodes in the hidden layer and one output node is created. For the recurrent network, a connection is added from the output node to a context unit with decay parameters $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$. The following table is a summary of the forecasts shown in Figure 9. The networks are trained on the first 191 observations and all three methods are used to forecast the final 6 values.

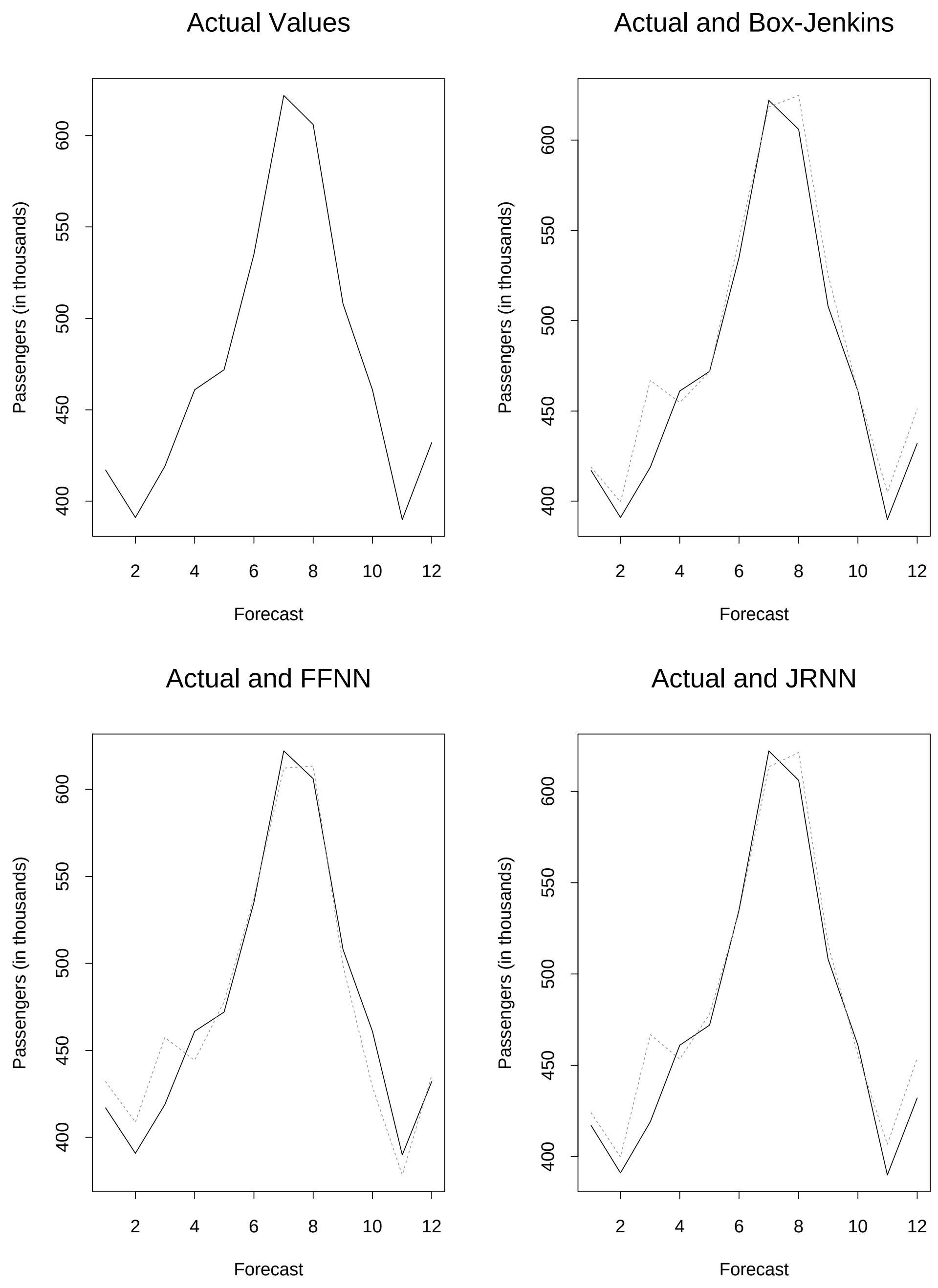

Figure 7: *Actual and forecasted values of last 12 observations from Airline series*

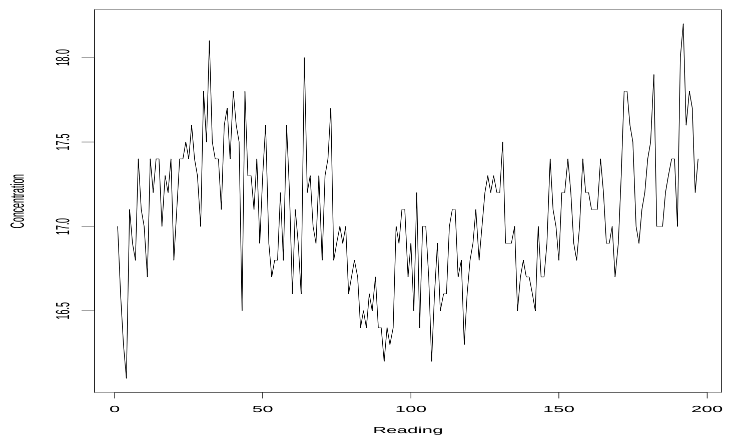

Figure 8: *Box-Jenkins Series A : Chemical Process Concentration Readings : Every 2 Hours*

Modeling Type	*Box-Jenkins*	*Feedforward ANN*	*Recurrent ANN*
Model	$(1, 0, 1)$	*lags=1*	*lags=1*
# Hidden Neurons	NA	2	2
# Parameters	2	7	9
SS_{RES}	19.188	21.56	19.208
$\widehat{\sigma}$	0.314	0.337749	0.320494
AIC	-451.47	-418.626	-437.266
BIC	-442.914	-395.341	-407.443
FMSE	0.157935	0.264902	0.1576944
FMAD	0.3256333	0.463793	0.3288155

Once again, the Box-Jenkins and recurrent neural network methods perform comparably while the feedforward network performs significantly worse. Moreover, perhaps the key result in both examples is that the recurrent ANN is able to match the performance of the *ARIMA* model, whereas the feedforward ANN does not.

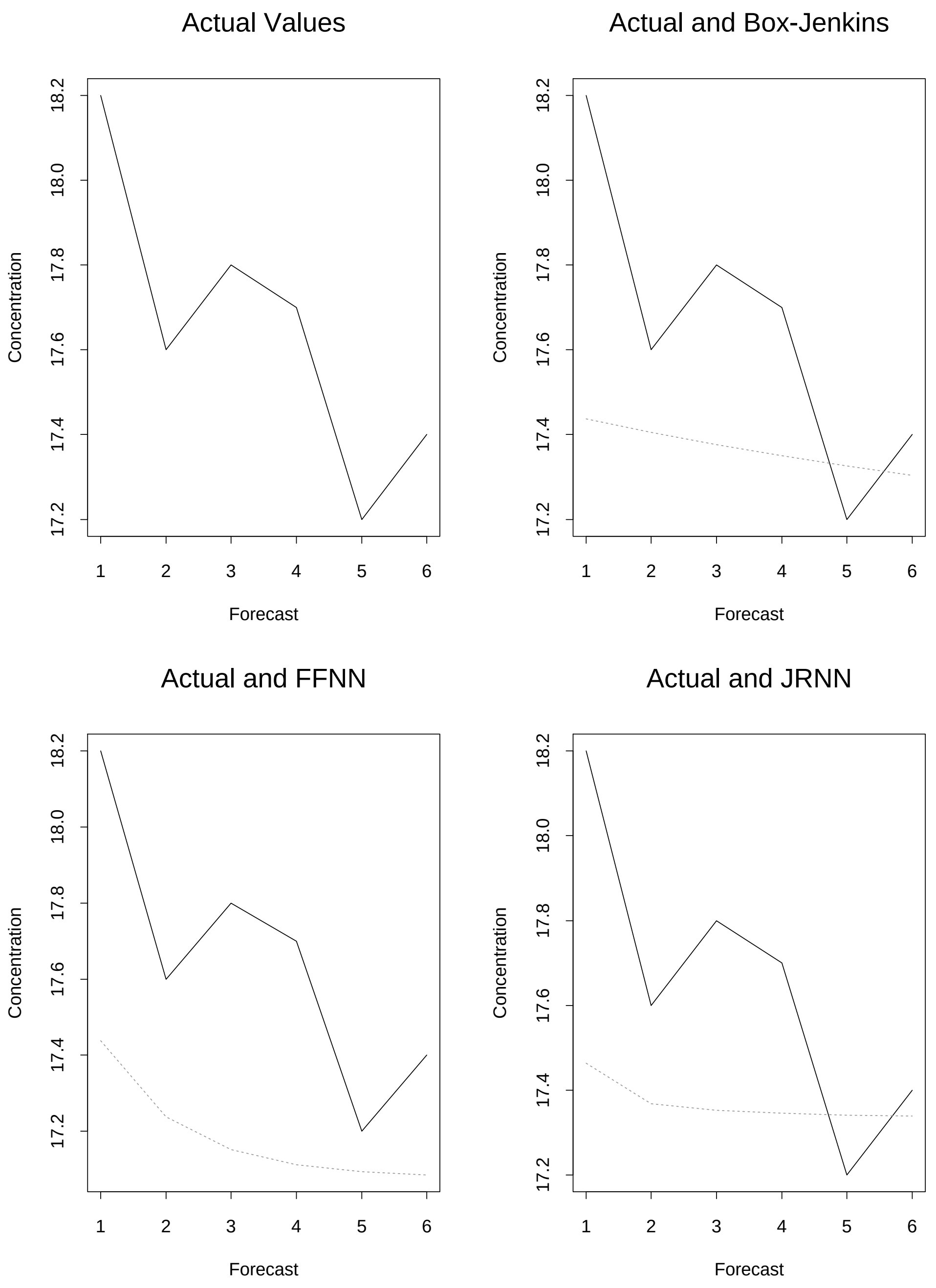

Figure 9: *Actual and forecasted values of last 6 observations from the Chemical series*

5. Conclusions and Future Work

This paper shows how some *ARMA* models can be implemented with neural networks. Most of the research regarding neural networks for time series forecasting has been done with feedforward networks which are analogous to a non-linear autoregressive model. By adding recurrence, it is possible to account for moving average structure inherent in a time series. Neural networks may perform marginally better, based on the FMSE, provided the moving average-like structure can be incorporated into the ANN, via recurrent networks or some other scheme. Such improvements are possible since ANNs can include non-linear relationships. However, this slight improvement does come at a cost in that the recurrent neural network takes considerably longer to train than a feedforward network or the time needed to estimate the linear model coefficients.

The literature on linear model selection is vast compared to that on choosing neural network architecture. As with conventional models, neural networks must also be constructed parsimoniously to avoid adding unnecessary noise into the system. Though much work has been done on architecture selection, it is essentially a trial and error process. By using the techniques for linear model selection, it should be possible to construct a neural network that will perform at least as well as the corresponding linear model. This architecture will not necessarily be the best, but will provide a good lower bound for further exploration.

Future research areas in this field include implementation of neural network architectures to account for higher order moving average structure and using statistical methods for detecting linear and non-linear relationships between variables to aid in architecture construction.

References

Bigus, J. P. (1996). *Data Mining with Neural Networks.* McGraw-Hill.

Box, G., Jenkins, G., & Reinsel, G. (1994). *Time Series Analysis Forecasting and Control* (Third edition). Prentice Hall : Englewood Cliffs, New Jersey.

Bulsari, A. B., & Saxen, H. (1993). A Recurrent Neural Network for Time-series Modelling. *Artificial Neural Networks and Genetic Algorithms – Proceedings of the International Conference in Innsbruck, Austria.*

Cheng, B., & Titterington, D. M. (1994). Neural Networks: A Review from a Statistical Perspective (with discussion). *Statistical Science*, *9*(1), 2–54.

Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent Neural Networks and Robust Time Series Prediction. *IEEE Transactions on Neural Networks*, *5*(2), 240–254.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*, 179–211.

Faraway, J., & Chatfield, C. (1995). Time Series Forecasting with Neural Networks: A Case Study. *Research Report 95-06 of the Statistics Group, University of Bath.*

Freeman, J. A. (1994). *Simulating Neural Networks with Mathematica.* Addison-Wesley Publishing Company.

Haykin, S. (1994). *Neural Networks : A Comprehensive Foundation.* Prentice Hall.

Hill, T., O'Connor, M., & Remus, W. (1996). Neural Network Models for Time Series Forecasts. *Management Science*, *42*(7), 1082–1092.

Hinton, G. E. (1992). How Neural Networks Learn from Experience. *Scientific American.*

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, *2*, 359–366.

Jordan, M. I. (1986). Serial Order: A Parallel Distributed Processing Approach. Tech. rep. Report 86-104, Institute of Cognitive Science.

Lippmann, R. P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, *4*, 4 – 22.

McDonnell, J. R., & Waagen, D. (1994). Evolving Recurrent Perceptrons for Time-Series Modeling. *IEEE Transactions on Neural Networks*, *5*(1).

Parzen, E. (1983). ARARMA Models for Time Series Analysis and Forecasting. *Journal of Forecasting*, *1*, 67 – 82.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, *323*, 533–536.

Sarle, W. S. (1994). Neural Networks and Statistical Models. *Proceedings of the Nineteenth Annual SAS Users Group International Conference.*

Tang, Z., de Almeida, C., & Fishwick, P. A. (1991). Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology. *Simulation*, *57*(5), 303–310.

Tong, H. (1990). *Non-linear Time Series.* Oxford University Press.