ELSEVIER

International Journal of Forecasting 16 (2000) 509–515

international journal of forecasting

www.elsevier.com/locate/ijforecast

Automatic neural network modeling for univariate time series

Sandy D. Balkin[a,*], J. Keith Ord[b]

[a]*Ernst & Young LLP, 1225 Connecticut Avenue, NW, Washington, DC 20037, USA*
[b]*McDonough School of Business, 320 Old North, Georgetown University, Washington, DC 20057, USA*

Abstract

Artificial neural networks (ANNs) are an information processing paradigm inspired by the way the brain processes information. Using neural networks requires the investigator to make decisions concerning the architecture or structure used. ANNs are known to be universal function approximators and are capable of exploiting nonlinear relationships between variables. This method, called *Automated ANNs*, is an attempt to develop an automatic procedure for selecting the architecture of an artificial neural network for forecasting purposes. It was entered into the M-3 Time Series Competition. Results show that ANNs compete well with the other methods investigated, but may produce poor results if used under certain conditions.

Keywords: Artificial neural networks; Automated ANNs; Architecture

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 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. In statistical parlance, the ANN network corresponds to a nonlinear model and the learning process to parameter estimation. For an introduction to ANNs see, for example, Hinton (1992) or Lippmann (1987), and for a review from a statistical perspective, see Cheng and Titterington (1994).

Recently, ANNs have been investigated as a tool for time series forecasting. A survey of the literature is given in Zhang, Patuwo, and Hu (1998). The most popular class, used exclusively in this study, is the multilayer perceptron, a feedforward network trained by backpropagation. This class of network consists of an input

*Corresponding author.

E-mail addresses: sandy.balkin@ey.com (S.D. Balkin), ordk@msb.edu (J. . Ord).

0169-2070/00/$ – see front matter
PII: S0169-2070(00)00072-8

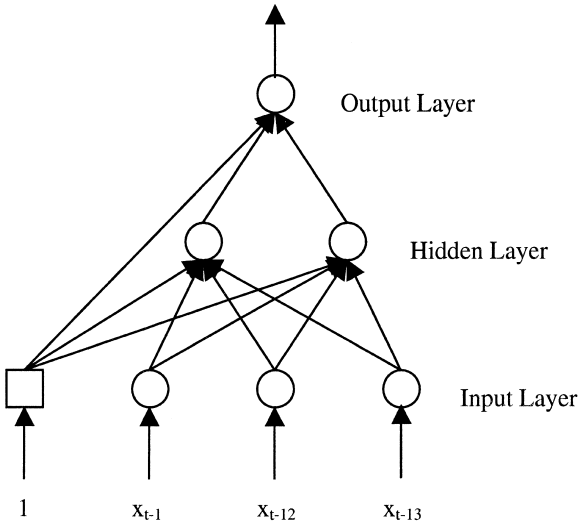

Fig. 1. Multilayer perceptron.

layer, a number of hidden layers and an output layer. Fig. 1 is an example of an ANN of this type with three neurons in the input layer, a single hidden layer with two neurons, and one neuron in the output layer. Using this type of ANN requires the investigator to make certain decisions concerning the structure, known as the *architecture* of the network. Neural networks are known to be universal function approximators (Hornik, Stinchcombe, & White, 1989). This property does not necessarily hold for time series where forecasting involves extrapolation. It holds when an unknown function is defined over a finite range of its arguments. The property may be of limited value in forecasting where:

1. processes are stochastic so that, at best, only the signal can be estimated;
2. prediction necessarily involves moving beyond the time range of historical inputs.

In general, 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 in the time series being forecast.

For a feedforward neural network with one hidden layer, the prediction equation, given by Faraway and Chatfield (1998), for computing forecasts 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 Fig. 1 is $NN[1, 12, 13; 2]$. The functional form may be written as

$$\hat{x}_t = \phi_o\left(w_{co} + \sum_h w_{ho}\phi_h\left(w_{ch} + \sum_i w_{ih}x_{t-j_i}\right)\right) \quad (1)$$

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 hidden 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, $y = \phi(x) = 1/(1 + e^{-x})$ is widely used in the ANN literature. Faraway and Chatfield (1998) recommend that for forecasting applications ϕ_h be logistic and ϕ_0 be the identify function. Forecasts are generated iteratively by performing successive one-step ahead forecasts using previous forecasts as estimates of observables.

2. Description of method

Our goal is to develop a set of rules on which to base the design decisions. Our primary concern is to make the model selection process data dependent. We tried to follow the paradigm used in more traditional forecasting methods

while not inhibiting the flexibility of the neural network method. The following is a detailed explanation of the decision-making process we implemented.

Since the parameters of the ANN are estimated by least squares, efficient estimates result only when the error terms are independent and have equal variances. If the process is intrinsically non-stationary, the error terms may well be dependent and have different variances.

- For the purposes of this exercise, we decided not to apply differencing, since proponents of ANN forecasting seem to regard such a step as unnecessary. This question is explored in greater detail elsewhere (Balkin, 1999, 2000).
- Thus, the first step is to determine if a natural log transformation of the series is necessary. This step is accomplished using the likelihood ratio. Let

$$w^2 = \frac{1}{n}\sum_i (x_i - \bar{x})^2$$

and

$$s^2 = \frac{1}{n}\sum_i \left(\log(x_i) - \frac{1}{n}\sum_i \log(x)\right)^2.$$

The likelihood is greater for the transformed series than the raw series when

$$\log(s^2) > \left(\log(w^2) + \frac{2}{n}\sum_i \log(x_i)\right);$$

if the inequality holds, take the log transform. When necessary, making the transformation should yield more efficient estimates since the ANN is fitted using the ordinary least squares criterion which requires constant error variance over time for efficient estimation.
- Next, we need to select the inputs used to train the network. The inputs to the network are chosen using a forward stepwise regression procedure with critical F-value equal to 2. The possible lags are chosen based on the time resolution of the series shown in Table 1. The inputs to the neural network are then chosen by considering all models that resulted in an F-statistic greater than 4.0 and choosing the one with the greatest number of included lags. For quarterly and other types of series, if the series length is less than 20, we only investigate lags 1–4. This rule is not parsimonious in terms of the number of lags, but our empirical studies suggest than an ANN may perform better with more lags than would be usual for an ARIMA model. However, Faraway and Chatfield (1998) present evidence suggesting one may easily go too far in that regard.

Table 1
Lags used as inputs based on time resolution

Time unit	Yearly	Quarterly	Monthly	Other
Lags	1–4	1–6	1–15	1–6

- The neural network is now trained on the inputs with the number of hidden nodes in the single hidden layer ranging from 1 to the number of inputs. The best network is chosen based on the minimum GCV value as described in Nychka, Bailey, Ellner, Haaland, and O'Connell (1996) where

$$\mathrm{GCV}(c) = \frac{1}{n} \times \mathrm{RSS}/(1 - p \times c/n)^2$$

where p denotes the number of parameters fitted and c is a cost coefficient, set equal to 2 in this study. Thus, the number of hidden nodes is limited even further for small series to ensure that enough degrees of freedom exist to estimate all the model parameters.
- Then, a simple linear auto-regression is fit with the same lags as variables selected by the GCV criterion. The linear AR model, ANN model and Naive (random walk) models are compared using the SBC criter-

ion. The one with smallest value is chosen to generate the forecasts. The comparison allows us to see whether a neural network is even necessary. For if the series is truly linear and a nonlinear model is used, the extra structure will add noise to the system and distort the forecasts.

- Forecasts are now generated from the selected model in an iterative fashion. A reverse transformation is applied, if necessary.

3. Method analysis

Estimating the weights of (or *training*) the neural network requires that the training data be representative of the process being forecast. The more training data, the more the network can learn about the series. How well a neural network does in forecasting stems from this idea. Thus, this method will not perform well for extremely short series or a series that is not representative of the process being modeled. An instance of the latter would be a series that requires intervention analysis to account for some uncharacteristic movements in the past.

A majority of the research on neural networks concerns classification and pattern recognition (cf. Cheng & Titterington, 1994). When ANNs are described as being robust to noisy data, it is with reference to classification and pattern recognition. Typically, either the noise level is low or the data set is very large. It is still under investigation when and whether these characteristics apply to time series forecasting as well.

4. Numerical example #1

As an example of how this method works, consider series N0702 from the M-3 Competition data set, which is a quarterly microeconomic series with 37 observations. The first step is to determine if a log transform is required. We first determine that $s^2 = 33\,669.8$ and $w^2 = 0.002815279$ and test whether

$$\log(33\,669.8) > \log(0.002815279) + \frac{2}{37} \times 301.2136$$
$$10.4244 > 10.4091$$

These values correspond to a likelihood ratio test marginally in favor of a logarithmic transformation. Next, we determine which lags to use as inputs. Since this series is quarterly, lags 1 through 6 are considered as candidates. We then run a forward stepwise regression. Lags 1, 2, 4 and 5 are chosen to be the inputs to the neural network. We now train the neural networks with one and two nodes in the hidden layer. The results are given in Table 2, which shows that model 1 with one node in the hidden layer is the preferred neural network model.

We now fit a linear regression model with the same lags and compare its SBC with that of the neural network. The SBC for the linear model is -76 while that of the neural network is -296. The Naive model has a SBC equal to -117. These results indicate that the neural network model provides a better fit for this series and will be used to generate the forecasts. The actual series and forecasts are shown in Fig. 2.

Table 2
Neural network fit statistics

Model	No. hidden units	DF model	DF residuals	Root MSE	GCV(1)	GCV(2)
1	1	7	25	0.10930	0.015290	0.02949
2	2	13	19	0.07042	0.008353	0.08376

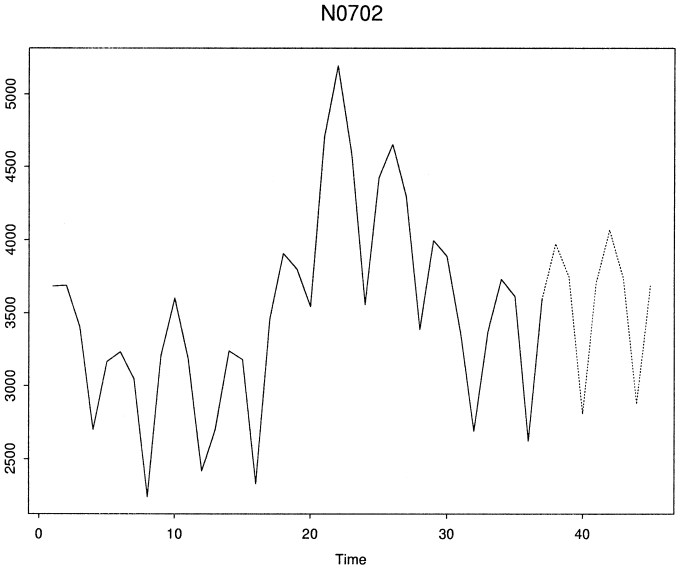

Fig. 2. Series N0702 and forecasts.

5. Numerical example #2

As an example of how this method may fail to provide reasonable forecasts, consider series N1386, from the M-3 Competition data set, which is a quarterly demographic data with 36 observations. The original series is shown in Fig. 3.

Our analysis indicates that a log transformation is desirable and we select lags 1, 3, 4, 5 and 6 as inputs to the neural network. A neural network model with one node in the hidden layer is found to be the best. The SBC for the ANN is -180 whereas that for the linear model is -76 and -64 for the Naive model. Thus, we find that the neural network model provides a better fit for this series and is used to generate the forecasts. The actual series and forecasts are shown in Fig. 4. It is obvious that these forecasts are characteristically different than the data used to generate them. Thus, in this example, the neural network model produced very poor forecasts. A possible reason for this is the ANN model is particularly susceptible to outliers or level shifts that occur at the very end of the series and project unwarranted rapid exponential growth.

6. Conclusion

Our paradigm attempts to provide an intelligent choice of inputs using recognized statistical procedures. An ANN typically has more parameters than most time series models. Therefore, it is expected that they will perform better on longer series with stable patterns for training. By construction, we would expect the ANN to provide superior forecasts when the process is nonlinear. The reason for using a complex method must be that a process contains elements not captured by simple forecasting methods, notably nonlinearity in the case of ANNs. For an ANN to outperform simple methods, we must have a sufficiently long series to detect the nonlinearity and to provide reliable estimates of

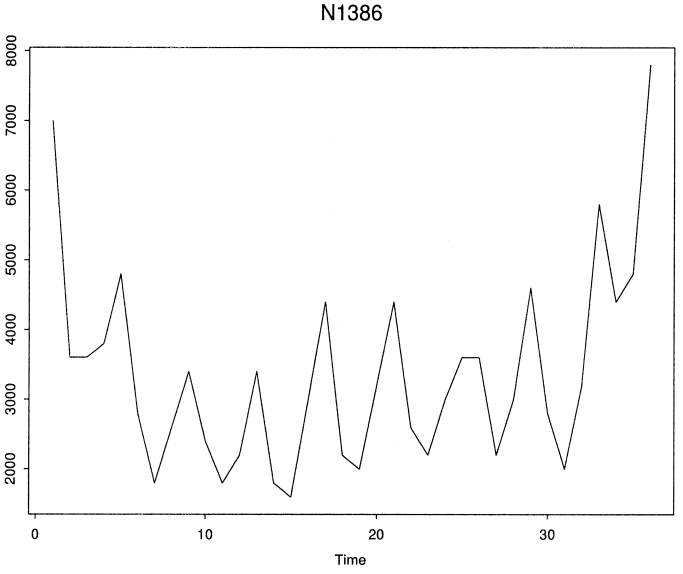

Fig. 3. Series N1386.

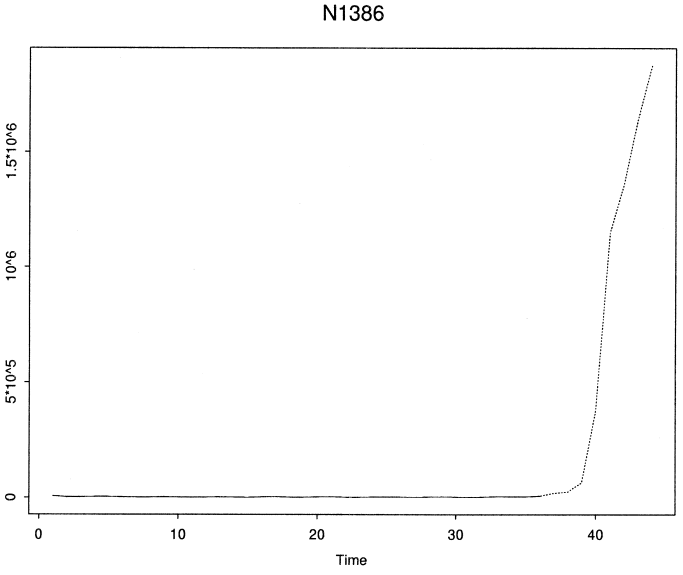

Fig. 4. Series N1386 and forecasts.

the parameters. When these conditions are satisfied, we can expect ANN to outperform simpler methods. Conversely, a simple method like exponential smoothing may well be adequate for forecasting most series, particularly when the series are short or have low information content. Results for the M3 and other competitions are generalized over a large number of series. So, simple methods may produce overall better results, but the complex models will perform better for those series with the relevant structure. Such distinctions require that we

analyze each series individually. This requirement comes as no surprise and is the reason that methods such as exponential smoothing are often used when automatic forecasting without user intervention is necessary.

Acknowledgements

The software we used for fitting the neural networks for this competition is described in Nychka et al. (1996) and we thank them for making their work publicly available. Earlier work used software described in Venables and Ripley (1994).

References

Balkin, S. D. (1999). *Stationarity Concerns When Forecasting using Neural Networks*, INFORMS Presentation, Philadelphia, PA.

Balkin, S. D. (2000). *A Statistical Implementation of Feedforward Neural Networks for Univariate Time Series Forecasting*, The Pennsylvania State University, Department of Management Science & Information Systems, Unpublished Ph.D. dissertation.

Cheng, B., & Titterington, D. M. (1994). Neural networks: a review from a statistical perspective. *Statistical Science 9*(1), 2–54.

Faraway, J., & Chatfield, C. (1998). Time series forecasting with neural networks: a comparative study using the airline data. *Applied Statistics 47*(2), 231–250.

Hinton, G. (1992). How neural networks learn from experience. *Scientific America*, 145–151, September.

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

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Mag. 4*, 4–22, April.

Nychka, D., Bailey, B., Ellner, S., Haaland, P., & O'Connell, M. (1996). *FUNFITS: data analysis and statistical tools for estimating functions,* Department of Statistics, North Carolina State University.

Venables, W. N., & Ripley, B. D. (1994). *Modern Applied Statistics with S-Plus*, Springer-Verlag, New York.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting 14*(1), 35–62.

Biographies: Sandy BALKIN is a statistical consultant with the Policy Economics and Quantitative Analysis group at Ernst & Young LLP in Washington, DC. He recently received his Ph.D. in Business Administration from the Pennsylvania State University. He specializes in forecasting and applied and computational statistics.

Keith ORD is Professor of Decision Sciences in the McDonough School of Business at Georgetown University. He is co-author of the latest editions of *Kendall's Advanced Theory of Statistics*. His research interests include time series and forecasting, and inventory modeling. He is an editor of the *International Journal of Forecasting*.