Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais (SBRN), Vol. 3, No. 1, pp. 40–49, 2005

Misspecified Neural Network Models and Linear Time Series Forecasting

Francisco Carlos de A. Pinto[1] and Marcelo C. Medeiros[2]
[1]Department of Statistics, Universidade Federal Fluminense
[2]Department of Economics, Pontifical Catholic University of Rio de Janeiro

Abstract

This paper studies the performance of neural networks estimated with Bayesian regularization to model and forecast time series where the data generating process is in fact a linear autoregression. A simulation experiment is carried out to compare the forecasts made by a correctly linear model and neural networks.

1 Introduction

The last two decades have witnessed a vast development of nonlinear time series techniques. Among the large amount of new methodologies, Feedforward Neural Networks (NN) form an important class of nonlinear models that has attracted considerable attention in the literature. The use of these models is mainly motivated by a mathematical result stating that NN models are a universal approximator to any Borel-measurable function to any given degree of accuracy; see, for example, [1, 2, 3, 4, 5, 6]. One central topic in the NN literature is how to select the variables and the number of hidden units in the model. Usually, this is done by some "rule of thumb". A vast number of models with different combinations of variables and number of hidden units are estimated and the one with the best performance according to some known criterion is chosen as the final specification. Several alternatives to this "rule of thumb" have appeared in the literature; see, for example, [7] and the references therein. One strategy that turned out to be quite successful in a number of applications is Bayesian regularization, proposed by [8, 9]. The fundamental idea is to find a balance between the number of parameters and goodness of fit by penalizing large models. The objective function is modified in such a way that the estimation algorithm effectively prunes the network by driving irrelevant parameter estimates to zero during the estimation process.

The goal of this paper is to extend the study of [10], verifying the effects of forecasting linear time series by neural network models estimated with Bayesian regularization. A simulation experiment is designed to compare the forecasting performance of neural networks and linear autoregressive models.

The plan of the paper is as follows. Section 2 describes Bayesian regularization and Section 3 discusses the issues related to forecasting from neural network models. A simulation study is carried out in Section 4. Section 5 contains concluding remarks.

2 Bayesian Regularization

Consider that the target time series $\{y_t\}$ is generated by the following stochastic process

$$y_t = T(\mathbf{x}_t; \mathbf{\Theta}) + \varepsilon_t, \tag{1}$$

where $T(\mathbf{x}_t; \mathbf{\Theta})$ is an unknown nonlinear function of the vector of variables $\mathbf{x}_t$ defined by the parameter vector $\mathbf{\Theta}$, and $\{\varepsilon_t\}$ is assumed to be a sequence of independent, normally distributed random variables with zero mean and finite variance, σ^2. Assume that $\mathbf{x}_t \in \mathbb{R}^q$ is formed by lagged values of y_t.

The goal of modeling techniques based on neural networks is to approximate (1) by the following nonlinear specification

$$y_t = G(\mathbf{z}_t; \boldsymbol{\psi}) = \lambda_0 + \sum_{i=1}^{h} \lambda_i F(\boldsymbol{\omega}_i' \mathbf{z}_t - \beta_i) + u_t, \tag{2}$$

where $G(\mathbf{z}_t;\boldsymbol{\psi})$ is a nonlinear function of $\mathbf{z}_t$, defined by the parameter vector $\boldsymbol{\psi} = [\boldsymbol{\lambda}', \boldsymbol{\omega}_1', \ldots, \boldsymbol{\omega}_h', \boldsymbol{\beta}']$, where $\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_h]'$, $\boldsymbol{\omega}_i = [\omega_{1i}, \ldots, \omega_{pi}]'$, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_h]'$. The vector of variables $\mathbf{z}_t \in \mathbb{R}^p$ is formed by lagged values of y_t and its elements are called *input variables*.

Usually, $\boldsymbol{\psi}$ is estimated by nonlinear least-squares

$$\hat{\boldsymbol{\psi}} = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \tilde{Q}_T(\boldsymbol{\psi}) = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{t=1}^{N} (u_t)^2 = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{t=1}^{N} (y_t - G(\mathbf{z}_t;\boldsymbol{\psi}))^2 , \tag{3}$$

and the estimated residuals $\hat{u}_t = y_t - G(\mathbf{z}_t;\hat{\boldsymbol{\psi}})$ is a good approximation to the true error term ε_t in (1). In most applications a simple gradient descent algorithm (backpropagation) is used to estimate $\boldsymbol{\psi}$. However, the estimation of $\boldsymbol{\psi}$ is usually not easy [11].

Approximating (1) by (2) poses two main problems. First, the true vector of variables $\mathbf{x}_t$ is not known in advance and the modeler has to determine which variables should be included in $\mathbf{z}_t$. The second problem is related to the selection of the number of hidden units in (2). Selecting a small number of hidden units leads to a poor approximation of the true data generating process. On the other hand, a model with a large number of hidden units may be overfitted, generating bad forecasts (poor generalization). In most neural network applications, it is customary to select the variables and the number of hidden units of the neural network approximation using some "rule of thumb". A vast number of models with different combinations of variables and number of hidden units are estimated and the one with the best performance according to some known criterion is chosen as the final specification. Several alternatives to this "rule of thumb" have appeared in the literature. The simplest one is the so-called *early stopping*. The key idea is to split the available data into three subsets. The first subset is used to estimate the parameters. The second subset is called the validation set. The error on the validation set is monitored during the estimation process. When the network begins to overfit the data, the error on the validation set typically begins to rise. When the validation error increases for a specified number of iterations, the estimation process is discontinued, and the parameters estimated at the minimum of the validation error serve as final estimates. The test set is not used for estimation, but it is saved for comparing different models. A large number of different specifications are estimated and compared by means of the out-of-sample performance. The model with the best forecasting performance is chosen as the final specification.

Pruning is another popular technique. The objective of pruning is to find the smallest network that fits the data well and produces good forecasts. The main idea is to start with a large network and sequentially reducing its size by removing some network connections. For a general survey on pruning see [12] and[13].

Another successful methodology, that is the main interest of this paper, is the Bayesian regularization approach proposed by [8, 9]. The fundamental idea is to find a balance between the number of parameters and goodness of fit by penalizing large models. The objective function is modified in such a way that the estimation algorithm effectively prunes the network by driving irrelevant parameter estimates to zero during the estimation process. The parameter vector $\boldsymbol{\psi}$ is estimated as

$$\hat{\boldsymbol{\psi}} = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \tilde{Q}_N(\boldsymbol{\psi}) = \underset{\boldsymbol{\psi}}{\operatorname{argmin}} \left(\eta Q_N(\boldsymbol{\psi}) + \gamma Q_N^*(\boldsymbol{\psi})\right), \tag{4}$$

where $Q_N(\boldsymbol{\psi}) = \sum_{t=1}^{N} (y_t - G(\mathbf{z}_t;\boldsymbol{\psi}))^2$, $Q_N^*(\boldsymbol{\psi})$ is the *regularization* or *penalty* term, and $\eta, \gamma > 0$ are *objective function* or *regularization* parameters. The usual penalty is the sum of squared parameters

$$Q_T^*(\boldsymbol{\psi}) = \sum_{j=0}^{p} \lambda_j^2 + \sum_{i=1}^{h} \beta_i^2 + \sum_{j=1}^{p} \sum_{i=1}^{h} \omega_{ji}^2. \tag{5}$$

The forecasting ability of the neural network model depends crucially on the values of η and γ, especially in small samples. The relative size of the objective function parameters determines the emphasis for the estimation process. If $\eta >> \gamma$, then the estimation algorithm will derive errors smaller, and the network may still overfit. If $\eta << \gamma$, estimation will emphasize parameter size reduction at the expense of network errors, thus producing a smoother function of the input variables. The main problem with implementing regularization is setting the correct values for the objective function parameters. One approach to determine the optimal objective function parameters is the Bayesian

framework, where the parameters of the network are assumed to be random variables with well-specified distributions. The objective function parameters are related to the unknown variances associated with these distributions and is estimated with statistical techniques. [14] give a detailed discussion of the use of Bayesian regularization in combination with the Levenberg-Marquardt optimization algorithm. The main advantage of this method is that even if the neural network model is over-parametrized, the irrelevant parameter estimates are likely to be close to zero and the model behaves like a small network.

Let $\mathbf{D} = (\mathbf{y}, \mathbf{Z})$ represent the data set, where $\mathbf{y} = [y_1, \ldots, y_N]'$ and $\mathbf{Z} = [\mathbf{z}_1 \ldots \mathbf{z}_N]'$. M a particular neural network model. After the data is taken, the density function for the parameters is updated according to Bayes' rule

$$P\left(\boldsymbol{\psi}|\mathbf{D}, \eta, \gamma, M\right) = \frac{P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right) P\left(\boldsymbol{\psi}|\gamma, M\right)}{P\left(\mathbf{D}|\eta, \gamma, M\right)}, \tag{6}$$

where $P\left(\boldsymbol{\psi}|\gamma, M\right)$ is the prior density, which represents our knowledge of the parameters before any data is collected, $P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right)$ is the likelihood function, which is the probability of the data occurring given the parameters and $P\left(\mathbf{D}|\eta, \gamma, M\right)$ is a normalization factor, which guarantees that the total probability is 1.

If the distribution of ε_t and the prior distribution for the parameters are both Gaussian, then $P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right)$ and $P\left(\boldsymbol{\psi}|\eta, M\right)$ are written as

$$P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right) = \left(\frac{\pi}{\eta}\right)^{-\frac{N}{2}} \exp(-\eta Q_N(\boldsymbol{\psi})) \tag{7}$$

and

$$P\left(\boldsymbol{\psi}|\eta, M\right) = \left(\frac{\pi}{\gamma}\right)^{-\frac{L}{2}} \exp(-\gamma Q_N^*(\boldsymbol{\psi})), \tag{8}$$

where $L = (p+2) \times h + 1$ is the total number of parameters in the neural network model. Substituting (8) in (7), we get

$$P\left(\boldsymbol{\psi}|\mathbf{D}, \eta, \gamma, M\right) = \frac{\left(\frac{\pi}{\eta}\right)^{-\frac{N}{2}} \left(\frac{\pi}{\gamma}\right)^{-\frac{L}{2}} \exp\left[-\left(\eta Q_N(\boldsymbol{\psi}) + \gamma Q_N^*(\boldsymbol{\psi})\right)\right]}{\text{Normalization Factor}} = Z(\eta, \gamma) \exp(-\tilde{Q}_N(\boldsymbol{\psi})). \tag{9}$$

In this Bayesian framework, the optimal parameters should maximize the posterior probability $P\left(\boldsymbol{\psi}|\mathbf{D}, \eta, \gamma, M\right)$, which is equivalent to minimizing the regularized objective function given in (4). The regularization parameters are optimized by applying Bayes' rule

$$P\left(\eta, \gamma|\mathbf{D}, M\right) = \frac{P\left(\mathbf{D}|\eta, \gamma, M\right) P\left(\eta, \gamma|M\right)}{P\left(\mathbf{D}|M\right)}. \tag{10}$$

Assuming a uniform prior density $P\left(\eta, \gamma|M\right)$ for the regularization parameters, then maximizing the posterior is achieved by maximizing the likelihood function $P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right)$. Since all probabilities have a Gaussian form, the normalization factor is expressed as

$$P\left(\mathbf{D}|\eta, \gamma, M\right) = \frac{P\left(\mathbf{D}|\boldsymbol{\psi}, \eta, M\right) P\left(\boldsymbol{\psi}|\gamma, M\right)}{P\left(\boldsymbol{\psi}|\mathbf{D}, \eta, \gamma, M\right)} = \left(\frac{\pi}{\eta}\right)^{-\frac{N}{2}} \left(\frac{\pi}{\gamma}\right)^{-\frac{L}{2}} Z^{-1}(\eta, \gamma). \tag{11}$$

Since the objective function is quadratic in a small area surrounding a minimum point, we can expand $\tilde{Q}_N(\boldsymbol{\psi})$ in a Taylor series around the minimum point of the posterior density, where the gradient is zero. Solving for the normalizing constant yields

$$Z(\eta, \gamma) = (2\pi)^{\frac{L}{2}} \left[\det\left(\mathbf{H}\right)^{-1}\right]^{1/2} \exp(-\tilde{Q}_N(\boldsymbol{\psi})), \tag{12}$$

where $\mathbf{H}$ is the Hessian matrix of the objective function. Inserting (11) into (10) we solve for the optimal values for η and γ at the minimum point. We do this by taking the derivative with respect to the log of (11) at set them equal to zero. This yields

$$\hat{\gamma} = \frac{\kappa}{2Q_N^*(\boldsymbol{\psi})} \tag{13}$$

and

$$\hat{\eta} = \frac{N - \kappa}{2Q_N(\boldsymbol{\psi})}, \tag{14}$$

where $\kappa = L - 2\gamma \text{trace}(\mathbf{H})^{-1}$ is called the effective number of parameters.

Following [14], here are the steps required for Bayesian optimization of the regularization parameters, with the Gauss-Newton approximation to the Hessian matrix:

1. Initialize η, γ, and the network parameters by the Nguyen-Widrow rule [15]. After the first estimation step, the objective function parameters will recover from the initial setting.

2. Take one step of the Levenberg-Marquardt algorithm to minimize the objective function $\tilde{Q}_N(\boldsymbol{\psi})$.

3. Compute the effective number of parameters $\kappa = L - 2\gamma \text{trace}(\mathbf{H})^{-1}$ making use of the Gauss-Newton approximation to the Hessian matrix available in the Levenberg-Marquardt optimization algorithm: $\mathbf{H} = \nabla^2 \tilde{Q}_N(\boldsymbol{\psi}) \approx 2\gamma \mathbf{J}'\mathbf{J} + 2\eta \mathbf{I}_N$, where $\mathbf{J}$ is the Jacobian matrix of the estimation set errors.

4. Compute new estimates for the objective function parameters.

5. Now iterate steps 1 through 3 until convergence.

3 Forecasting with Neural Network Models

Multi-step forecasting with nonlinear models is more challenging than forecasting with linear models. See, for example, [16, Section 8.1] for a general discussion.

Consider the simple nonlinear model defined as

$$y_t = G(y_{t-1}; \boldsymbol{\psi}) + u_t, \tag{15}$$

where $G(\cdot)$ is a nonlinear function with parameter vector $\boldsymbol{\psi}$. The term u_t is an independent identically distributed random variable with zero mean and finite variance. The history of the process up to time t is called $\mathcal{F}_t$.

Due the fact that $E(u_{t+1}|\mathcal{F}_t) = 0$, the optimal 1-step-ahead forecast of y_{t+1} is given by

$$\hat{y}_{t+1|t} = E(y_{t+1}|\mathcal{F}_t) = G(y_t; \boldsymbol{\psi}), \tag{16}$$

which is equivalent to the optimal 1-step-ahead forecast when $G(\cdot)$ is linear.

For multi-step forecasts, the problem is much more complicated. For 2-step-ahead the optimal forecast is given by

$$\hat{y}_{t+2|t} = E(y_{t+2}|\mathcal{F}_t) = E(G(y_{t+1}; \boldsymbol{\psi})|\mathcal{F}_t)) = \int_{-\infty}^{\infty} G(y_{t+1}; \boldsymbol{\psi}) f(u_{t+1}) du_{t+1}, \tag{17}$$

where $f(u_{t+1})$ is the density of u_{t+1}. Usually the expression (17) is approximated by numerical techniques, such as, for example, monte-carlo or bootstrap.

The monte-carlo method is a simple simulation technique for obtaining multi-step forecasts. For model (15), the k-step-ahead forecast is defined as

$$\hat{y}_{t+k|t} = \frac{1}{M} \sum_{i=1}^{M} \hat{y}^{(i)}_{t+k|t}, \tag{18}$$

where M is the number of replications and

$$\hat{y}^{(i)}_{t+k|t} = G(\hat{y}_{t+k-1|t}); \boldsymbol{\psi}) + \xi^{(i)}_{t+k|t}. \tag{19}$$

$\xi^{(i)}_{t+k|t}$ is a random number drawn from a normal distribution with the same mean and standard deviation as the in-sample estimated residuals.

4 Monte Carlo Simulation

In this section we report the results of a simulation study designed to find out the behavior of neural networks estimated with Bayesian regularization when the main concern is multi-step forecasting. We simulate 500 replications of several linear autoregressive models of order p, AR(p). At each replication we discard the first 500 observations to avoid any initialization effects. The simulated models are defined as

- Model 1: AR(1) without drift
$$y_t = \phi y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2), \tag{20}$$
where $\phi = 0.65, 0.75, 0.85, 0.95$.

- Model 2: AR(1) with drift
$$y_t = 0.5 + \phi y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2), \tag{21}$$
where $\phi = 0.65, 0.75, 0.85, 0.95$.

- Model 3: AR(2)
$$y_t = 0.24 + 0.6y_{t-1} - 0.08y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{22}$$

- Model 4: AR(2)
$$y_t = 0.00125 + 1.9y_{t-1} - 0.9025y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{23}$$

- Model 5: AR(2)
$$y_t = 0.015 + 1.35y_{t-1} - 0.38y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{24}$$

- Model 6: AR(2)
$$y_t = 0.49 + 0.6y_{t-1} - 0.58y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{25}$$

- Model 7: AR(2)
$$y_t = 0.29 + 1.4y_{t-1} - 0.98y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{26}$$

- Model 8: AR(3)
$$y_t = 0.0000625 + 2.85y_{t-1} - 2.7075y_{t-2} + 0.8574y_{t-3} + \varepsilon_t, \quad \varepsilon_t \sim \text{NID}(0, 0.5^2). \tag{27}$$

It is worth mentioning that Model 3 has the following two real roots: 0.2 and 0.4. Model 4 has two equal roots that are close to unit (0.95). The roots of Model 5 are 0.4 and 0.95. Models 6 and 7 have complex roots. Finally, the roots of Model 8 are all 0.95.

The forecasting experiment can be viewed of consisting of the following steps.

1. For each monte-carlo iteration
   (a) Generate a time series by the process defined by Models 1–8 with N observations, $N = 150, 550$.
   (b) Split the sample into two subsamples: the estimation set ($t = 1, \ldots, t_0$) and the forecasting set ($t = t_0 + 1, \ldots, N$), where $t_0 = 100, 500$.
   (c) Estimate the parameters of a neural network model with 5 hidden units and the first six lags as input variables and a first order linear autoregressive model with drift using only the estimation set.
   (d) For $t = t_0, \ldots, N - 4$, compute the k-steps-ahead out-of-sample forecasts, $k = 1, \ldots, 4$, denoted by $\hat{y}_{t+k|t}$, and the associated forecast errors $\hat{u}_{t+k|t}$. Multi-step forecasts for the neural network models are obtained by monte-carlo simulation with 500 iterations.

Table 1: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Model 1 (100 observations).

Horizon	$\phi = 0.65$				$\phi = 0.75$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5292 (0.0694)	0.4252 (0.0590)	0.5031 (0.0533)	0.4036 (0.0449)	0.5372 (0.0792)	0.4308 (0.0647)	0.5058 (0.0558)	0.4056 (0.0465)
2	0.6275 (0.0916)	0.5059 (0.0762)	0.6004 (0.0750)	0.4822 (0.0620)	0.6731 (0.1078)	0.5426 (0.0894)	0.6352 (0.0861)	0.5124 (0.0717)
3	0.6628 (0.1071)	0.5360 (0.0891)	0.6378 (0.0894)	0.5133 (0.0732)	0.7358 (0.1260)	0.5942 (0.1067)	0.6974 (0.1056)	0.5631 (0.0886)
4	0.6762 (0.1171)	0.5478 (0.0985)	0.6541 (0.0981)	0.5276 (0.0819)	0.7662 (0.1388)	0.6210 (0.1180)	0.7302 (0.1197)	0.5915 (0.1006)

Horizon	$\phi = 0.85$				$\phi = 0.95$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5535 (0.0987)	0.4437 (0.0808)	0.5059 (0.0556)	0.4060 (0.0480)	0.6362 (0.2570)	0.5128 (0.2109)	0.5144 (0.0752)	0.4141 (0.0648)
2	0.7340 (0.1531)	0.5925 (0.1287)	0.6686 (0.0978)	0.5392 (0.0833)	0.8979 (0.3809)	0.7305 (0.3208)	0.7237 (0.1409)	0.5841 (0.1246)
3	0.8366 (0.1832)	0.6803 (0.1569)	0.7679 (0.1322)	0.6222 (0.1135)	1.0707 (0.4573)	0.8794 (0.3974)	0.8741 (0.2033)	0.7121 (0.1833)
4	0.9015 (0.2069)	0.7361 (0.1790)	0.8318 (0.1587)	0.6770 (0.1376)	1.1952 (0.5120)	0.9886 (0.4529)	0.9924 (0.2593)	0.8134 (0.2335)

(e) For each forecasting horizon, compute the root mean square errors and the mean absolute errors defined as

$$RMSE(k) = \sqrt{\frac{1}{N - t_0 - 3} \sum_{t=N}^{N-4} \hat{u}_{t+k|t}^2}, \tag{28}$$

$$MAE(k) = \frac{1}{N - t_0 - 3} \sum_{t=t_0}^{N-4} |\hat{u}_{t+k|t}|. \tag{29}$$

2. Compute the mean and the standard deviation of the performance measures (28) and (29) for each forecast horizon over the 500 replications.

Tables 1 – 6 show the mean and the standard deviation (between parenthesis) of the RMSE and MAE over the 500 repetitions. Observing Tables 1 – 6 we can see that, except for Model 8 which has mixed results, the linear AR models produce forecasts with lower RMSE and MAE than the neural network model and that the difference increases as the roots of the simulated models tend to 1. When the sample size is increased the differences between the forecasts made by the linear and the neural network models are smaller.

5 Conclusions

This paper has addressed the issue of forecasting linear time series with neural networks estimated with Bayesian regularization. Bayesian regularization is a technique designed to avoid overfitting, where the fundamental idea is to find a balance between the number of parameters and the goodness of fit by penalizing large models. The objective function is modified in such a way that the estimation algorithm effectively prunes the network by driving irrelevant parameter estimates to zero during the estimation process. The main advantage of this method is that even if the ANN model is over-parametrized, the irrelevant parameter estimates are likely to be close to zero and the model behaves like a small network. A simulation study has been carried out to evaluate the performance of neural networks models in forecasting time series generated by a simple linear autoregressive models. The results were compared with the ones

Table 2: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Model 1 (500 observations).

Horizon	$\phi = 0.65$				$\phi = 0.75$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5045 (0.0554)	0.4047 (0.0448)	0.4984 (0.0533)	0.3999 (0.0434)	0.5092 (0.0561)	0.4082 (0.0467)	0.5028 (0.0519)	0.4030 (0.0434)
2	0.6028 (0.0781)	0.4854 (0.0661)	0.5952 (0.0745)	0.4792 (0.0634)	0.6324 (0.0870)	0.5091 (0.0725)	0.6237 (0.0799)	0.5019 (0.0667)
3	0.6360 (0.0934)	0.5127 (0.0787)	0.6282 (0.0899)	0.5060 (0.0759)	0.6919 (0.1084)	0.5585 (0.0915)	0.6829 (0.1012)	0.5511 (0.0854)
4	0.6470 (0.0998)	0.5211 (0.0836)	0.6406 (0.0969)	0.5157 (0.0812)	0.7217 (0.1216)	0.5855 (0.1046)	0.7129 (0.1148)	0.5775 (0.0983)

Horizon	$\phi = 0.85$				$\phi = 0.95$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5043 (0.0574)	0.4049 (0.0490)	0.4969 (0.0556)	0.3986 (0.0448)	0.5117 (0.0627)	0.4106 (0.0533)	0.5004 (0.0506)	0.4009 (0.0438)
2	0.6602 (0.0902)	0.5320 (0.0775)	0.6501 (0.0818)	0.5233 (0.0694)	0.7085 (0.1122)	0.5702 (0.0957)	0.6897 (0.0891)	0.5533 (0.0766)
3	0.7544 (0.1138)	0.6091 (0.0958)	0.7428 (0.1049)	0.5992 (0.0875)	0.8427 (0.1556)	0.6822 (0.1335)	0.8197 (0.1260)	0.6616 (0.1079)
4	0.8123 (0.1333)	0.6574 (0.1131)	0.8004 (0.1260)	0.6474 (0.1056)	0.9467 (0.1945)	0.7705 (0.1678)	0.9204 (0.1606)	0.7467 (0.1368)

Table 3: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Model 2 (100 observations).

Horizon	$\phi = 0.65$				$\phi = 0.75$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5280 (0.0700)	0.4234 (0.0567)	0.5044 (0.0522)	0.4046 (0.0451)	0.5365 (0.0779)	0.4303 (0.0634)	0.5079 (0.0553)	0.4068 (0.0469)
2	0.6276 (0.0925)	0.5048 (0.0779)	0.6017 (0.0760)	0.4829 (0.0641)	0.6708 (0.1090)	0.5401 (0.0928)	0.6390 (0.0868)	0.5138 (0.0742)
3	0.6627 (0.1095)	0.5357 (0.0934)	0.6366 (0.0897)	0.5117 (0.0769)	0.7335 (0.1356)	0.5948 (0.1178)	0.7032 (0.1119)	0.5666 (0.0941)
4	0.6734 (0.1160)	0.5443 (0.0994)	0.6506 (0.0970)	0.5240 (0.0831)	0.7640 (0.1522)	0.6201 (0.1332)	0.7352 (0.1283)	0.5935 (0.1283)

Horizon	$\phi = 0.85$				$\phi = 0.95$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5558 (0.1078)	0.4463 (0.0882)	0.5072 (0.0570)	0.4083 (0.0475)	0.6173 (0.2501)	0.5071 (0.2160)	0.5071 (0.0618)	0.4067 (0.0530)
2	0.7346 (0.1504)	0.5936 (0.1280)	0.6719 (0.0970)	0.5418 (0.0809)	0.8673 (0.3627)	0.7090 (0.3185)	0.7090 (0.1167)	0.5708 (0.1017)
3	0.8362 (0.1813)	0.6778 (0.1564)	0.7697 (0.1332)	0.6213 (0.1116)	1.0335 (0.4335)	0.8557 (0.3876)	0.8557 (0.1690)	0.6937 (0.1506)
4	0.8994 (0.2045)	0.7325 (0.1784)	0.8347 (0.1608)	0.6759 (0.1368)	1.1573 (0.4687)	0.9714 (0.4289)	0.9714 (0.2184)	0.7919 (0.1977)

Table 4: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Model 2 (500 observations).

Horizon	$\phi = 0.65$				$\phi = 0.75$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5983 (0.0553)	0.4076 (0.0463)	0.5000 (0.0543)	0.4016 (0.0469)	0.5091 (0.0566)	0.4092 (0.0421)	0.4994 (0.0520)	0.4008 (0.0442)
2	0.6022 (0.0801)	0.4848 (0.0669)	0.5964 (0.0792)	0.4798 (0.0675)	0.6322 (0.0850)	0.5101 (0.0754)	0.6234 (0.0832)	0.5014 (0.0711)
3	0.6372 (0.0946)	0.5139 (0.0798)	0.6330 (0.0944)	0.5103 (0.0806)	0.6934 (0.1134)	0.5665 (0.0912)	0.6829 (0.1066)	0.5507 (0.0899)
4	0.6503 (0.1009)	0.5263 (0.0867)	0.6497 (0.1027)	0.5243 (0.0876)	0.7265 (0.1227)	0.5812 (0.1042)	0.7120 (0.1212)	0.5751 (0.1024)
Horizon	$\phi = 0.85$				$\phi = 0.95$			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5104 (0.0573)	0.4090 (0.0478)	0.4972 (0.0471)	0.3977 (0.0406)	0.5145 (0.0666)	0.4116 (0.0567)	0.5002 (0.0474)	0.4011 (0.0399)
2	0.6667 (0.0963)	0.5369 (0.0804)	0.6531 (0.0771)	0.5249 (0.0650)	0.7135 (0.1170)	0.5756 (0.0990)	0.6920 (0.0807)	0.5563 (0.0694)
3	0.7598 (0.1266)	0.6135 (0.1070)	0.7455 (0.1065)	0.6005 (0.0907)	0.8557 (0.1636)	0.6934 (0.1401)	0.8240 (0.1153)	0.6639 (0.0991)
4	0.8186 (0.1495)	0.6652 (0.1288)	0.8055 (0.1300)	0.6511 (0.1100)	0.9647 (0.2034)	0.7878 (0.1774)	0.9257 (0.1512)	0.7485 (0.1284)

Table 5: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Models 3–8 (100 observations).

Horizon	Model 3				Model 4			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5189 (0.0618)	0.4172 (0.0513)	0.5073 (0.0529)	0.4072 (0.0458)	1.9651 (2.8773)	1.5820 (2.4707)	0.5364 (0.0914)	0.4332 (0.0834)
2	0.6011 (0.0814)	0.4845 (0.0675)	0.5919 (0.0744)	0.4751 (0.0630)	4.1569 (5.1031)	3.3989 (4.4189)	1.1957 (0.2832)	0.9745 (0.2670)
3	0.6151 (0.0885)	0.4964 (0.0743)	0.6059 (0.0817)	0.4868 (0.0699)	6.2174 (6.5829)	5.1486 (5.7560)	1.9778 (0.5688)	1.6246 (0.5418)
4	0.6153 (0.0890)	0.4968 (0.0752)	0.6060 (0.0825)	0.4871 (0.0704)	8.1328 (7.6933)	6.8066 (6.7813)	2.8508 (0.9325)	2.3560 (0.8953)
Horizon	Model 5				Model 6			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.6916 (0.3865)	0.5566 (0.3118)	0.5182 (0.0716)	0.4160 (0.0621)	0.5296 (0.0677)	0.4254 (0.0568)	0.5005 (0.0523)	0.4011 (0.0438)
2	1.1676 (0.5846)	0.9507 (0.4952)	0.8907 (0.1738)	0.7196 (0.1553)	0.6119 (0.0884)	0.4929 (0.0738)	0.5797 (0.0717)	0.4656 (0.0615)
3	1.5200 (0.7072)	1.2503 (0.6188)	1.1925 (0.2889)	0.9684 (0.2619)	0.6191 (0.0835)	0.4983 (0.0698)	0.5897 (0.0701)	0.4738 (0.0598)
4	1.7926 (0.8195)	1.4856 (0.7331)	1.4338 (0.3989)	1.1731 (0.3657)	0.6611 (0.0955)	0.5320 (0.0794)	0.6341 (0.0844)	0.5089 (0.0705)
Horizon	Model 7				Model 8			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5930 (0.1680)	0.4758 (0.1359)	0.5071 (0.0525)	0.4073 (0.0449)	0.7188 (4.2593)	3.6404 (2.1008)	0.6144 (0.2561)	0.5051 (0.2390)
2	1.0260 (0.2945)	0.8285 (0.2446)	0.8821 (0.1115)	0.7101 (0.0945)	4.8425 (5.0854)	4.2710 (2.5056)	1.9890 (0.9988)	1.6543 (0.9381)
3	1.1729 (0.3297)	0.9511 (0.2788)	1.1019 (0.1522)	0.8238 (0.1290)	5.9655 (5.8393)	4.9437 (2.9207)	4.2947 (2.3927)	3.6055 (2.2590)
4	1.1684 (0.3441)	0.9458 (0.2868)	1.0198 (0.1508)	0.8239 (0.1280)	10.0769 (6.4011)	5.5956 (3.2818)	7.6322 (4.5525)	6.4619 (4.3178)

Table 6: Mean and standard deviation (between parenthesis) of the root mean square errors and the mean absolute errors of multi-step forecasts over 500 replications of Models 3–8 (500 observations).

Horizon	Model 3				Model 4			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5077 (0.0541)	0.4068 (0.0451)	0.4965 (0.0530)	0.3974 (0.0455)	0.6016 (0.4340)	0.4818 (0.3349)	0.4978 (0.0545)	0.4001 (0.0445)
2	0.5880 (0.0753)	0.4733 (0.0632)	0.5761 (0.0737)	0.4620 (0.0619)	1.3546 (1.1273)	1.0904 (0.8895)	1.0781 (0.1385)	0.8685 (0.1144)
3	0.6031 (0.0837)	0.4860 (0.0708)	0.5920 (0.0829)	0.4751 (0.0696)	2.2292 (1.9252)	1.8036 (1.5493)	1.7397 (0.2627)	1.4048 (0.2193)
4	0.6050 (0.0849)	0.4882 (0.0720)	0.5933 (0.0841)	0.4762 (0.0705)	3.1667 (2.7296)	2.5779 (2.2348)	2.4549 (0.4217)	1.9851 (0.3552)
Horizon	Model 5				Model 6			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5172 (0.0758)	0.4154 (0.0619)	0.4953 (0.0528)	0.3977 (0.0454)	0.5069 (0.0556)	0.4072 (0.0465)	0.4968 (0.0512)	0.3982 (0.0436)
2	0.8758 (0.1641)	0.7068 (0.1377)	0.8339 (0.1076)	0.6695 (0.0910)	0.5881 (0.0753)	0.4740 (0.0622)	0.5797 (0.0711)	0.4649 (0.0608)
3	1.1578 (0.2509)	0.9397 (0.2165)	1.0982 (0.1665)	0.8860 (0.1437)	0.5974 (0.0718)	0.4817 (0.0595)	0.5878 (0.0689)	0.4711 (0.0590)
4	1.3732 (0.3326)	1.1189 (0.2938)	1.3064 (0.2258)	1.0573 (0.1954)	0.6451 (0.0836)	0.5214 (0.0709)	0.6324 (0.0825)	0.5084 (0.0698)
Horizon	Model 7				Model 8			
	Neural Network		Linear model		Neural Network		Linear model	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.5128 (0.0647)	0.4121 (0.0538)	0.4994 (0.0534)	0.4004 (0.0451)	2.9976 (1.6981)	1.5282 (0.8373)	0.5044 (0.0550)	0.4042 (0.0471)
2	0.8833 (0.1380)	0.7119 (0.1146)	0.8623 (0.1142)	0.6940 (0.0960)	3.5246 (2.0639)	1.7967 (1.0154)	1.5261 (0.1860)	1.2248 (0.1569)
3	1.0127 (0.1830)	0.8180 (0.1517)	0.9947 (0.1588)	0.8035 (0.1331)	4.0755 (2.4010)	2.0897 (1.1971)	3.1385 (0.4317)	2.5298 (0.3665)
4	1.0103 (0.1802)	0.8158 (0.1495)	0.9948 (0.1577)	0.8034 (0.1332)	4.5957 (2.6548)	2.3815 (1.3541)	2.3815 (0.8225)	4.3357 (0.7028)

obtained by a linear autoregressive model with drift. The main conclusion is that, in small samples, neural networks produces forecasts with larger RMSE and MAE statistics than the linear models, specially the the data generating process has roots near unit. When the sample size is increased the difference between linear and nonlinear forecasts becomes smaller.

References

[1] K. Funahashi. "On the Approximate Realization of Continuous Mappings by Neural Networks". *Neural Networks*, vol. 2, pp. 183–192, 1989.

[2] G. Cybenko. "Approximation by Superposition of Sigmoidal Functions". *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.

[3] K. Hornik, M. Stinchombe and H. White. "Multi-Layer Feedforward Networks are Universal Approximators". *Neural Networks*, vol. 2, pp. 359–366, 1989.

[4] K. Hornik, M. Stinchombe, and H. White. "Universal Approximation of an Unknown Mapping and its Derivatives Using Multi-Layer Feedforward Networks". *Neural Networks*, vol. 3, pp. 551–560, 1990.

[5] H. White. "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings". *Neural Networks*, vol. 3, pp. 535–550, 1990.

[6] A. R. Gallant and H. White. "On Learning the Derivatives of an Unknown Mapping with Multilayer Feedforward Networks". *Neural Networks*, vol. 5, pp. 129–138, 1992.

[7] M. C. Medeiros, T. Teräsvirta and G. Rech. "Building Neural Network Models for Time Series: A Statistical Approach". *Journal of Forecasting*, forthcoming.

[8] D. J. C. MacKay. "Bayesian Interpolation". *Neural Computation*, vol. 4, pp. 415–447, 1992.

[9] D. J. C. MacKay. "A practical Bayesian framework for Backpropagation Networks". *Neural Computation*, vol. 4, pp. 448–472, 1992.

[10] M. C. Medeiros and C. E. Pedreira. "What are the effects of forecasting linear time series with neural networks?" *Engineering Intelligent Systems*, vol. 9, pp. 237–242, 2001.

[11] J. Hush. "Training a Sigmoidal Node is Hard". *Neural Computation*, vol. 11, no. 5, pp. 1249–1260, 1999.

[12] R. Reed. "Prunning Algorithms – A Survey". *IEEE Transactions on Neural Networks*, vol. 4, pp. 740–747, 1993.

[13] U. Anders and O. Korn. "Model Selection in Neural Networks". *Neural Networks*, vol. 12, pp. 309–323, 1999.

[14] F. D. Foresee and M. . T. Hagan. "Gauss-Newton Approximation to Bayesian Regularization". In *IEEE International Conference on Neural Networks (Vol. 3)*, pp. 1930–1935, New York, 1997. IEEE.

[15] D. Nguyen and B. Widrow. "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights". In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pp. 21–26, 1990.

[16] C. W. J. Granger and T. Teräsvirta. *Modelling Nonlinear Economic Relationships*. Oxford University Press, Oxford, 1993.