Neural Networks for Time Series Forecasting: Practical Implications of Theoretical Results

Melinda Thielbar and D.A. Dickey

February 25, 2011

Research on the performance of neural networks in modeling nonlinear time series has produced mixed results. While neural networks have great potential because of their status as universal approximators (Hornik, Stinchcombe, and White 1989), their flexibility can lead to estimation problems. When Faraway and Chatfield (1998) used an autoregressive neural network to forecast airline data, they found that the neural networks they specified frequently would not converge. When they did converge, they failed to find the global minimum of the objective function. In some cases, neural networks that fit the in-sample data well performed poorly on hold-out samples. In conducting the NN3 competition, a time series forecasting competition designed to showcase autoregressive neural networks and other computationally-intensive methods of forecasting, standard methods such as ARIMA models still out-performed autoregressive neural networks (Crone *et*

al 2008).

A comparison of linear methods, smooth transition autoregressive methods, and autoregressive neural networks performed in Terasvirta (2005) may shed some light on neural network estimation problems and general poor performance. In Terasvirta (2005) it was discovered that when estimated without constraints, autoregressive neural networks tended to yield explosive forecasts, and only by hand-tuning the models or applying a post-estimation filter to the resulting parameter estimates could they remove the worst of the offenders. In addition, while some researchers claim that autoregressive neural networks could estimate trend and seasonality (Gorr 1994, Sharda and Patil 1992), in an empirical study on simulated series with seasonality and trends Zhang and Qi (2005) showed that the neural network performed much better after the series was adjusted for trends and seasonality. As Faraway and Chatfield (1998) discovered, the autoregressive neural network could not be treated as a "black box".

In recent years, some researchers have made an attempt to open the box and better understand the properties of autoregressive neural networks. In Trapletti *et al* (2000), Trappletti, Leisch, and Hornik showed that an autoregressive neural network is stationary and ergodic (under certain sufficient but not necessary regularity conditions). In Leoni (2009) sufficient conditions whereby the skeleton of an autoregressive neural network approaches a unique attraction point were defined.

We propose to build on the results in Trapletti *et al* (2000) and Leoni

(2009) by examining the practical aspects of forecasting with neural networks, including starting value selection, forecast performance, and the behavior of series generated from a neural network with known parameters. We begin by deriving some theoretical properties of an AR-NN with one lag. We then shift to simulated results and focus on an autoregressive neural network model with one lag and one hidden unit, where the noise term is distributed $N(0,1)$. We find that the general properties derived in the first section hold for our simple model, and that even when the AR-NN is reduced to its simplest form, the practical aspects of model estimation can still cause problems in using an AR-NN to forecast a nonlinear time series. We end with some general conclusions, including cautions about some of the pitfalls of AR-NN and recommendations for avoiding them.

1 Statement of the Problem

Consider the one-lag autoregressive neural network (AR-NN):

$$Y_t = \alpha_0 + \rho Y_{t-1} + \sum_{j=1}^{k} \lambda_j g(\gamma_j r_{t-1,j}) + e_t \tag{1}$$

where $r_{t-1,j} = Y_{t-1} - c_j$, Y_{t-1} is the first lag of Y_t and $\{c_1, ..., c_k\}$ are location parameters for the activation function g. The parameters $\{\gamma_1, ..., \gamma_k\}$ are slope parameters for the activation function and the vector $\lambda = \{\lambda_1, ..., \lambda_k\}$ is a vector of weights.

Let the activation function g be bounded and continuous, let ρ be such

that $|\rho| < 1$, and let the e_t be *iid* with probability distribution function that is positive everywhere in $(-\infty, \infty)$. These assumptions are necessary for the stability conditions shown in Trapletti *et al* (2000).

1.1 Process Skeleton and Attractors

In Tong (1993), it is suggested that simulation is often necessary for studying nonlinear time series, as general analytical results are difficult to obtain and often require restrictive assumptions. Even with modern computing power, simulations for AR-NN can quickly grow too large to be practical. The following results assume that the parameters for the AR-NN are given and examine the theoretical behavior of the series under the regularity conditions described above.

In Tong (1993), the behavior of a nonlinear time series is described in terms of the skeleton and the skeleton's equilibrium point(s). The skeleton, as described in Tong (1993), is the relationship between Y_t and Y_{t-1} when the noise term e_t is set identically equal to 0 for all t. Denote the skeleton of the series as S_t. The skeleton of the AR-NN described in (1) is:

$$S_t = \alpha_0 + \rho S_{t-1} + \sum_{j=1}^{k} \lambda_j g(\gamma_j r^s_{t-1,j}) \tag{2}$$

where $r^s_{t-1,j} = S_{t-1} - c_j$

The skeleton has also been called the *deterministic portion* of the random process.

We can rewrite S_t as:

$$S_t = (\alpha_0 + \sum_{j=1}^{k} \lambda_j g(\gamma_j r_{t-1,j}^s))(1 - \rho B)^{-1} \tag{3}$$

where B is the backshift operator.

This expresses the skeleton as a weighted sum of the previous value of the skeleton and the nonlinear operation on the previous value of the skeleton. Because g is a bounded function and $|\rho| < 1$, for t large enough, we can expect S_t to be bounded:

$$S_t \in [\min(\alpha_0 + \sum_{j=1}^{k} \lambda_j g(\gamma_j r_{t-1,j}^s))(1-\rho)^{-1}, \max(\alpha_0 + \sum_{j=1}^{k} \lambda_j g(\gamma_j r_{t-1,j}^s))(1-\rho)^{-1}] \tag{4}$$

within a finite number of steps.

If $|g| \leq 1$ (as with the logistic and hyperbolic tangent basis functions) then the above range becomes:

$$S_t \in \Big[\big(\alpha_0^* - \sum_{j=1}^{k} |\lambda_j|\big)\big(1-\rho\big)^{-1}, \big(\alpha_0^* + \sum_{j=1}^{k} |\lambda_j|\big)\big(1-\rho\big)^{-1}\Big]$$

We have assumed that the noise term e_t is iid, and therefore we know that for any $\epsilon > 0$, there exists M such that $P\Big(|e_t| \leq M\Big) > (1-\epsilon)$ i.e. that e_t is bounded in probability. We can therefore place e_t inside a finite range such that the probability of observing a value for e_t outside this range can be shrunk arbitrarily close to 0. This allows us to stay within the conditions

set in Trapletti *et al* (2000), yet have an expected range for the noise term.

The bounds on the S_t and e_t imply that there is a practical range for Y_t that depends on the activation function g and the parameters $\{\lambda_1, ...\lambda_k\}$, and M. For the normal distribution with mean 0, a popular default distributional assumption in the statistical literature, and a basis function g that is bounded such that $|g| < 1$, $M = 3\sigma$, and the range for Y_t is:

$$Y_t \in \left[\left(\alpha_0^* - \sum_{j=1}^{k} |\lambda_j|\right)\left(1-\rho\right)^{-1} - 3\sigma, \left(\alpha_0^* + \sum_{j=1}^{k} |\lambda_j|\right)\left(1-\rho\right)^{-1} + 3\sigma \right]$$

The practical range for Y_t may be further reduced by the presence of equilibria and whether those points are attractors for the series. We use the definitions of equilibrium and attraction point as defined in Tong (1993).

A point Y^* is an equilibrium point if it satisfies the condition:

$$Y^* = \alpha_0 + \rho Y^* + \sum_{j=1}^{k} \lambda_j g(\gamma_j(Y^* - c_j)) \tag{5}$$

i.e. if it is a point where the result from the skeleton is the same as the value going into the skeleton. Once S_t reaches Y^* it will remain at the same value for all $t \to \infty$. The long-run behavior of the AR-NN is dependent on the existence of Y^* and whether it is stable or unstable. We begin by deriving some basic properties of Y^* for a series that meets the stability conditions set forth in Trapletti *et al* (2000).

Theorem 1. *Suppose Y_t is an AR-NN with one lag and an arbitrary number*

of hidden units that meets the stability conditions in Trapletti et al *(2000), then if Y^* exists, it can be bounded by a finite range that depends on the weights of the activation function and the intercept term for the AR-NN.*

Proof: We can re-arrange equation (5) as follows:

$$Y^* = \left(\alpha_0^* + \sum_{j=1}^{k} \lambda_j g(\gamma_j(Y^* - c_j))\right)\left(1 - \rho\right)^{-1} \tag{6}$$

The equilibrium point Y^* then becomes the solution to equation (6), and must lie within the same bounds as set for the skeleton in (4).

The properties of the equilibrium, particularly whether it is unique and stable, now become critical to understanding the behavior of the AR-NN. We will use the definitions of stability found in Tong (1993).

- We say that Y^* is a *stable equilibrium* if there exists a neighborhood of Y^* such that $|Y_t - Y^*| \to 0$ as $t \to \infty$. We will call this kind of equilibrium an *attraction point* or an *attractor*.

- Y^* is an *unstable equilibrium* if no such neighborhood exists.

- Y^* is an equilibrium that is *globally and exponentially stable in the large* if it is unique, and $\exists\ K, c > 0$ such that $|Y_t - Y^*| \leq Ke^{-ct}|Y_0 - Y^*|\ \forall\ t$ for any starting point Y_0. We will call this kind of equilibrium a *global attraction point* or *global attractor*.

An attraction point, therefore, is a value to which the skeleton will converge if it becomes sufficiently close to Y^*. A global attraction point is an

attraction point to which the skeleton converges from any value on the real line, and the convergence is exponentially fast. Tong (1993) likens a global attraction point for a nonlinear time series with additive noise to the mean for a stationary linear time series. When the noise pushes the Y_t away from the global attractor, the skeleton draws it back exponentially fast.

Now we will show that if Y^* exists and is a global attractor, the range for Y_t depends on Y^*.

Theorem 2. *If Y^* exists and is globally and exponentially stable in the large, there is an expected range for Y_t based on the value of Y^* and the practical range for the noise term e_t.*

Proof: This result is an obvious consequence of the fact that the noise term e_t is additive and *iid*. If we allow S_t to represent the skeleton (i.e. the deterministic portion) of the series, then we can write any Y_t as:

$$Y_t = S_t + e_t$$

From the properties of a global attractor, we know that for any arbitrary Y_0, the following inequality holds:

$$|S_t - Y^*| \leq Ke^{-ct}|Y^* - Y_0|$$

By assumption, the noise term has a probability bound $|e_t| < M$. Therefore, we can say:

$$|S_t - Y^*| + |e_t| \leq Ke^{-ct}|Y_0 - Y^*| + M$$

$|Y_t - Y^*| = |S_t + e_t - Y^*| \leq |S_t - Y^*| + |e_t|$, and so

$$|Y_t - Y^*| \leq Ke^{-ct}|Y_0 - Y^*| + M$$

In the case of an attraction point that is globally and exponentially stable in the large, after the series process has been allowed to iterate for a sufficient number of time periods, we can expect Y_t to be inside the range $(Y^* - M, Y^* + M)$, where M depends on the distribution of the noise term e_t.

We have therefore shown that for any AR-NN of the form in equation (1), and a noise term that is *iid*, there is a practical range for Y_t. outside of which we can expect to observe few, if any, values for Y_t. If there is a global and exponentially stable attraction point, that range is $Y^* \pm M$, where Y^* is the global attractor and M is determined by the distribution of e_t. If there is no global attractor, then $Y_t \in (M_1^s - M, M_2^s + M)$, where M_1^s and M_2^s are the limits on the series skeleton.

We now consider the idea of *practical reducibility* and its implications for estimating the parameters of an AR-NN.

2 Practical Reducibility

The conditions for stability in Trapletti *et al* (2000) and Tong (1993) are shown using Markov theory. For an AR-NN with additive noise, when the

distribution of the noise term is positive everywhere in $(-\infty, \infty)$, it can be shown that the AR-NN forms an irreducible and aperiodic Markov chain. The chain is aperiodic because it does not cycle between a set of values at specified multiples of t. It is irreducible because it is impossible to reduce the range of Y_t from the entire real line $(-\infty, \infty)$ to a smaller finite set.

These proofs are possible because the noise term is additive. It does not depend on Y_t, and any value of e_t is possible. Therefore, even if the skeleton of the series converges to a limit cycle (which would make the series periodic), or to a single point (which would make the series reducible), the noise term ensures that Y_t is irreducible and aperiodic.

As discussed in the previous section, however, possible is not the same as likely, and in practice the sample size T is never infinity. We have shown that for a noise term that is *iid*, there is a likely range for Y_t based on the parameters of the AR-NN and the noise term e_t. The probability of observing a value for Y_t outside this range can be shrunk arbitrarily close to 0.

Recall that the activation function $g(\gamma_j r_{t-1,j})$ is a bounded and continuous function with $r_{t-1,j} = Y_{t-1} - c_j$, where c_j is a location parameter for the activation function. In order to estimate the parameters $\{c_1, ..., c_k\}$ we must observe values of Y_{t-1} that are near each c_j where $j \in \{1, ..., k\}$. All of the location parameters $\{c_1, ..., c_k\}$ must therefore be inside the practical range for Y_{t-1}. If there exists one c_j such that c_j is not in the practical range for Y_t, we say that the AR-NN is *not irreducible in a practical sense.*

It may be tempting to ignore the limited range for Y_t in applying AR-NN

to forecasting problems. It can be argued that the AR-NN is an approximation to an unknown functional form. Activation functions with estimated c_j that are outside the expected range for Y_t can be dropped from the model. These "problem" hidden units are unlikely to help explain or predict the relationship between Y_t and its lags, and could therefore be safely eliminated.

While this is true, it is also true that there is currently no method for deciding how many hidden units should be used in an AR-NN. It is also true that training an AR-NN is not a simple matter (Hush 1991), and it is possible for an estimation routine to produce an AR-NN with unacceptable values for c_j as the result of a local minimum. In the following pages, we use the above understanding of AR-NN to illustrate some of the pitfalls for training AR-NN and offer some guidelines for avoiding them, such as choosing appropriate starting values. We use the simplest possible AR-NN (a case with one lag and one hidden unit) as our example model and derive many of our practices from a simple re-parameterization that is outlined in the following section.

3 The Simplest Possible AR-NN

Consider the model:

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \lambda \tanh(\gamma(Y_{t-1} - c)) + \varepsilon_t \tag{7}$$

$$\varepsilon_t \sim N(0, \sigma)$$

This is the simplest case of an autoregressive neural network model. In neural network terminology, it has one hidden unit ($\lambda \tanh(\gamma(Y_{t-1} - c))$) and one shortcut connection ($\alpha_1 Y_{t-1}$). We use the tanh(.) function as our *activation function*, and while the hyperbolic tangent function is convenient for our purposes, there are many activation functions to choose from, including the logistic, the Gaussian radial basis, and the inverse tangent.

All moments of the noise term exist, and as long as the autoregressive portion ($Y_t - \alpha_0 - \alpha_1 Y_{t-1}$) is stationary, the mean, μ_y, exists. Taking expectations of both sides and rearranging gives us μ_Y as a function of the parameters and the expectation of the hyperbolic tangent function.

$$\begin{aligned} \mu_Y = E[Y_t] &= \alpha_0 + \alpha_1 \mu_Y + \lambda E[\tanh(\gamma(Y_{t-1} - c))] \\ \mu_Y(1 - \alpha_1) &= \alpha_0 + \lambda E[\tanh(\gamma(Y_{t-1} - c))] \\ \mu_Y &= \Big(\alpha_0 + \lambda E[\tanh(\gamma(Y_{t-1} - c))]\Big)\Big(1 - \alpha_1\Big)^{-1} \end{aligned}$$

The expectation of the hyperbolic tangent function is intractable analytically, but since it is the expectation of a bounded function, and Y_t has a noise distribution that is $N(0, \sigma)$, we know it exists.

Now, subtract the mean from both sides, divide by σ, and express the series in terms of $y_{t-1} = \frac{Y_{t-1} - \mu_Y}{\sigma}$ and $y_t = \frac{Y_t - \mu_Y}{\sigma}$:

$$y_t = \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*)) + e_t \tag{8}$$

$$
\begin{aligned}
\text{where } e_t &\sim N(0,1) \\
\rho &= \alpha_1 \\
\lambda^* &= \frac{\lambda}{\sigma} \\
\gamma^* &= \gamma\sigma \\
c^* &= \frac{c-\mu_Y}{\sigma} \\
\alpha_0^* &= \frac{\alpha_0-(1-\rho)\mu_Y}{\sigma}
\end{aligned}
$$

The above parameterization expresses the location parameter c^* in terms of the number of standard deviations between the center point of the hyperbolic tangent function and the series mean. The location parameter α_0^* is in terms of the number of standard deviations between the intercept for an AR(1) model and the intercept for the nonlinear model. The weight on the activation function (λ^*) and the slope (γ^*) are expressed as multiples of the standard deviation of the unexplained variation.

One of the most difficult aspects of studying AR-NN is the lack of simulation results to test and illustrate theoretical conclusions. The above reparameterization can be used to determine appropriate ranges for simulated parameter values and reduce the design space to a few relevant points.

For our simulation, we limit ourselves to an e_t that is distributed $N(0,1)$. We are therefore able to limit the ranges of α_0^* and c^* to ± 4 (since most values for Y_t will be less than 4 standard deviations from the mean). For our

Factorial Design for Model Parameters

	Low	High	Increment	Number of Levels
$\alpha 0$	-4	4	2	5
ρ	0.2	0.8	0.2	4
λ^*	-8	8	2	8
γ^*	1	7	2	4
c^*	-4	4	2	5

Table 1: Each parameter combination can have up to 3 equilibria. If there are multiple attraction points, the equilibrium point for the series can depend on the starting value for y_t (i.e. y_0). To account for this, three different starting points were attempted for each combination $y_0 = c^* - 3$, $y_0 = c^*$, and $y_0 = c^* + 3$. This resulted in 9,600 different parameter/starting value combinations.

model, the limits on the skeleton described in section 1 are determined by the weight of the activation function λ. We therefore study values for λ^* in $[-8, 8]$, since 8 is the maximum distance between α_0^* and c^*. The parameter γ does not have an obvious set of limits. For this study, we set the limits (somewhat arbitrarily) to $\gamma \in [1, 7]$.

With these ranges in mind, we utilize a full factorial design where the parameter values are chosen as specified in Table 1. More parsimonious designs were considered, but because of the unknown nonlinear relationships between the series behavior and the parameter combinations, sparse designs were deemed infeasible.

For simplicity, we drop the $*$ notation and refer to the model equation as:

$$y_t = \alpha_0 + \rho y_{t-1} + \lambda \tanh\left(\gamma(y_{t-1} - c)\right) \tag{9}$$

4 The AR(1) Model vs AR-NN

In order to study the series generated from the AR-NN described in (9), we generate $S = 10$ series of errors from a $N(0, 1)$ distribution. Each series is length $T = 1000$, with a leading burn-in period of 500 (1500 periods total). We choose $T = 1000$ to ensure that there are enough observed values of Y_t to estimate the parameters. We choose a leading burn-in of 500 to ensure that the series is allowed to reach its steady state from its starting point Y_0.

Because the errors are *iid*, we are able to use the same 10 sequences of e_t for each parameter combination. The sequences, then, are blocks in the design of experiments sense, which allows us to reduce the extraneous noise in the simulation.

Producing estimated parameter values from this simulated data is more difficult than one would expect in this relatively simple case. Even when the true parameter values were used as starting values, the model often failed to converge. When the model did converge, some parameter estimates were unreasonable, with values near ± 100. Descriptive statistics, including graphs of the underlying functions and attraction point for the series, showed that while some series were following the expected functional form, others, because of the location of their attraction points, were not (Figure 3).

It is difficult to determine from the parameter values whether an AR-NN has an attraction point, whether that attraction point is stable, and whether the natural limits placed on Y_t are problematic for model estimation. We suspect that for some parameter combinations in our factorial design, the

AR-NN will produce data where the relationship between Y_t and Y_{t-1} is nearly linear rather than following the functional form of the AR-NN. We wish to distinguish these models from those series that exhibit nonlinear behavior.

In order to identify these problem parameter sets, we want to compare the results from a trained AR-NN to an AR(1) model, but this is impractical because the AR-NN frequently fails to converge for these data. We need a statistic that estimates the fit for the AR-NN and can be calculated without training the AR-NN model.

Therefore, assume that we are able to estimate the parameters of the autoregressive neural network perfectly. If that were true, our error sum of squares would be equal to:

$$SSE_{sp} = \sum_{t=2}^{1000} e_{st}^2$$

where the p subscript represents a "perfect model", and the s subscript represents a given sequence of errors ($s = 1, 2, ..., 10$) used to generate data from the AR-NN.

Now, for each simulated data set (10 data sets per parameter combination, 96,000 data sets in total), we estimate the parameters of an AR(1), i.e. a linear model with the form:

$$y_t = a_0 + a_1 y_{t-1} + e_t^*$$

where e_t^* will be distributed $N(0,1)$ if the true relationship between y_t and its first lag is a straight line.

For each parameter combination, then, we can calculate $\hat{\delta}_k$, the average difference in the MSEs as:

$$\begin{aligned} \hat{\delta}_k &= \frac{1}{10}\sum_{s=1}^{10}\left(MSE_{sp} - MSE_{sa}\right) \qquad (10) \\ MSE_{sp} &= \frac{1}{999-1}\sum_{t=2}^{1000} e_{st}^2 \\ MSE_{sa} &= \frac{1}{999-2}\sum_{t=2}^{1000}(y_t - (\hat{a}_0 + \hat{a}_1 y_{t-1}))^2 \end{aligned}$$

where the e_{st} are the true errors used to generate the series and k indexes the parameter combinations $k = 1, ..., 9600$.

Under the null hypothesis, we expect $\delta_k = E[\hat{\delta}_k]$ to be 0 and $\hat{\delta}_k$ to have an approximately normal distribution. Under the alternative hypothesis, we expect $\delta_k > 0$, where the magnitude of δ_k is an indicator of how poorly the AR(1) fits the generated data.

We know that the skeleton (S_t) is bounded: $S_t \in (\alpha_0 - |\lambda|, \alpha_0 + |\lambda|)$, and that the practical range for Y_t is $(\alpha_0 - |\lambda| - 4, \alpha_0 + |\lambda| + 4)$. If there exists Y^*, a global attractor for Y_t, the series may be further bounded to: $Y_t \in (Y^*-4, Y^*+4)$, (see section 1). If the location parameter c is outside this range, then we can assume that it will be difficult to estimate the parameters of the AR-NN, and in fact our small AR-NN will look more like an AR(1)

than a nonlinear time series.

To test these speculations, we examine the values of $\hat{\delta}_k$ in terms of the parameter values for λ, γ, and $|\alpha_0 - c|$.Recall that $\hat{\delta}_k$ is the mean difference between the MSE for an AR(1) model and the "true" MSE: the average of the squared errors that are used to generate the series (see equation (10)). The subscript k indexes the parameter combinations.

In Figure 1, each point represents a parameter combination. The $\hat{\delta}_k$ are plotted on the horizontal axis and the value of λ is plotted on the vertical axis. The colors of the points are determined by $|\alpha_0 - c|$, with light gray representing small values of $|\alpha_0 - c|$ (0-2) and black representing large values of $|\alpha_0 - c|$ (6 and up). The fit of the AR(1) as measured by $\hat{\delta}_k$ is much worse when $\lambda < 0$, but the fit of an AR(1) model is almost the same as that of our perfect model when $\lambda > 0$.

It is not surprising that the AR(1) fits much worse when $\lambda < 0$. Recall that the hyperbolic tangent function is an odd function, i.e. $-\lambda \tanh(\gamma(y_{t-1} - c)) = \lambda \tanh(-\gamma(y_{t-1} - c))$, and so we only consider positive values of γ and allow the sign of λ to vary. We also limit our simulation to positive values for ρ. In the simulation, therefore, the sign of λ determines the sign of the slope of the activation function for the neural network. If the activation function and the shortcut connection have the same sign, then they are more likely to be collinear, and an AR(1) model that compromises between them may fit almost as well as a nonlinear function that attempts to estimate both effects independently.

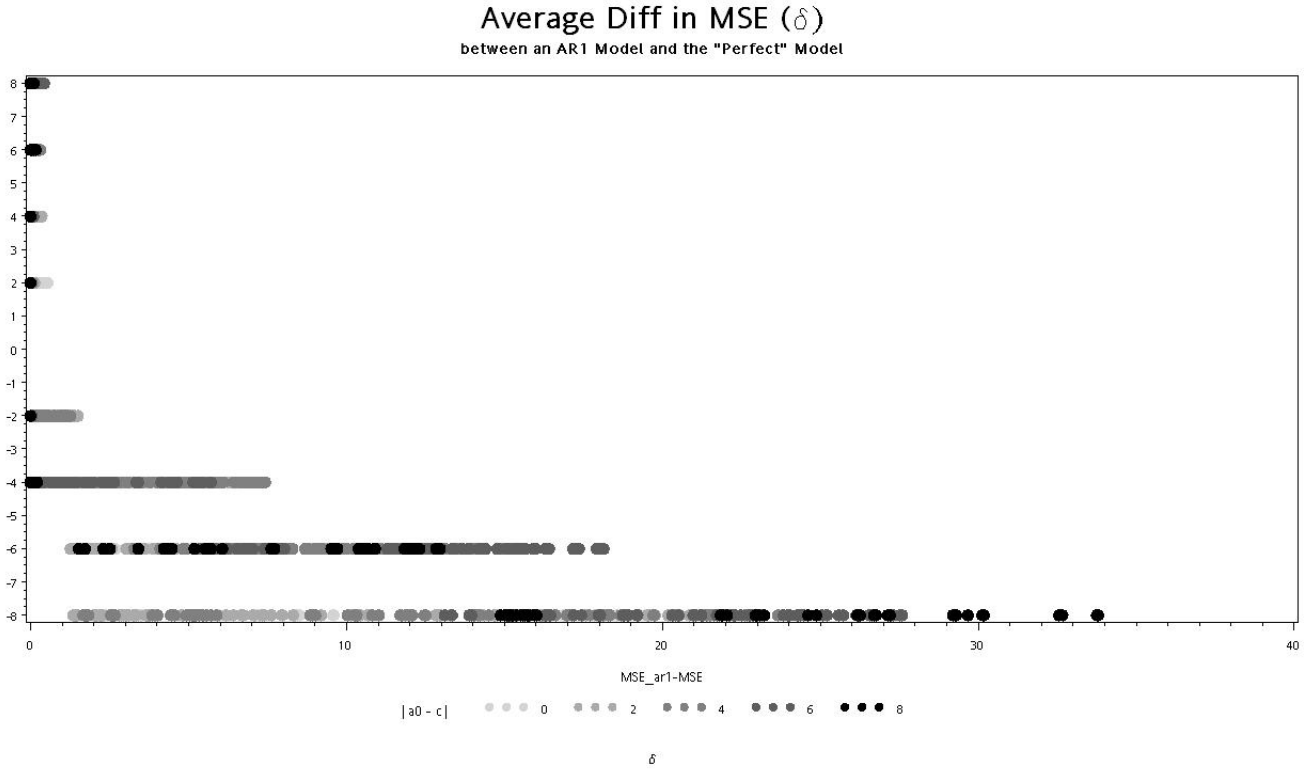

Figure 1: There is a striking contrast between the comparison of model fit for $\lambda > 0$ and $\lambda < 0$. The relationship between $\hat{\delta}_k$ and $|\alpha_0 - c|$ depends on the magnitude of λ. When $|\lambda|$ is large, location parameters that are far apart produce a model with nonlinear features. When λ is small, the relationship is reversed: small values of $|\alpha_0 - c|$ produce a model with nonlinear features.

When $|\lambda|$ is small, smaller values of $|\alpha_0 - c|$, are associated with larger values of $\hat{\delta}_k$, i.e. the AR(1) is a poor substitute for the nonlinear model when $|\lambda|$ is relatively small and the location parameters are close together. When $|\lambda|$ is large, we have the opposite relationship: For large $|\lambda|$, small values for $|\alpha_0 - c|$ are associated with small values for $\hat{\delta}_k$, i.e the AR(1) gets closer to fitting the true relationship between y_t and its first lag as the distance

between the location parameters increases (for large $|\lambda|$). This is consistent with our findings from section 1. The weight on the activation function (λ) determines the limit for the nonlinear function, and the acceptable range for $|\alpha_0 - c|$ depends upon the limit of the activation function and the practical range of the noise term.

Figure 2 shows the equation that generates the data and the resulting series, with y_t plotted on the vertical axis and y_{t-1} on the horizontal axis, for a set of parameters where $\lambda = -8$, $\alpha_0 = 4$, $c = -4$, $\gamma = 1$, and $\rho = 0.2$. These data scatter evenly on both sides of the location parameter for the hyperbolic tangent function and close to the location parameter as well. It should be easy to obtain parameter estimates for this AR-NN model. This is confirmed by the value of $\hat{\delta}_k$, which is 2.77, or nearly three times the magnitude of our chosen σ^2.

Figure 3 shows the data and underlying functions for the exact same set of parameters as Figure 2, except that λ is positive instead of negative (8 instead of -8). In this case, the series runs to an attractor that is far away from the location parameter of the hyperbolic tangent function. The hyperbolic tangent function adds little more than a constant to the generated data. It is not surprising that $\hat{\delta}_k < 0.0009$ for this series.

In Leoni (2009), it is shown that the presence of a global attraction point can be determined by the slope parameters alone for some AR-NN. The results above show that while this may be true, the distance between the location parameters relative to the weight of the activation function is im-

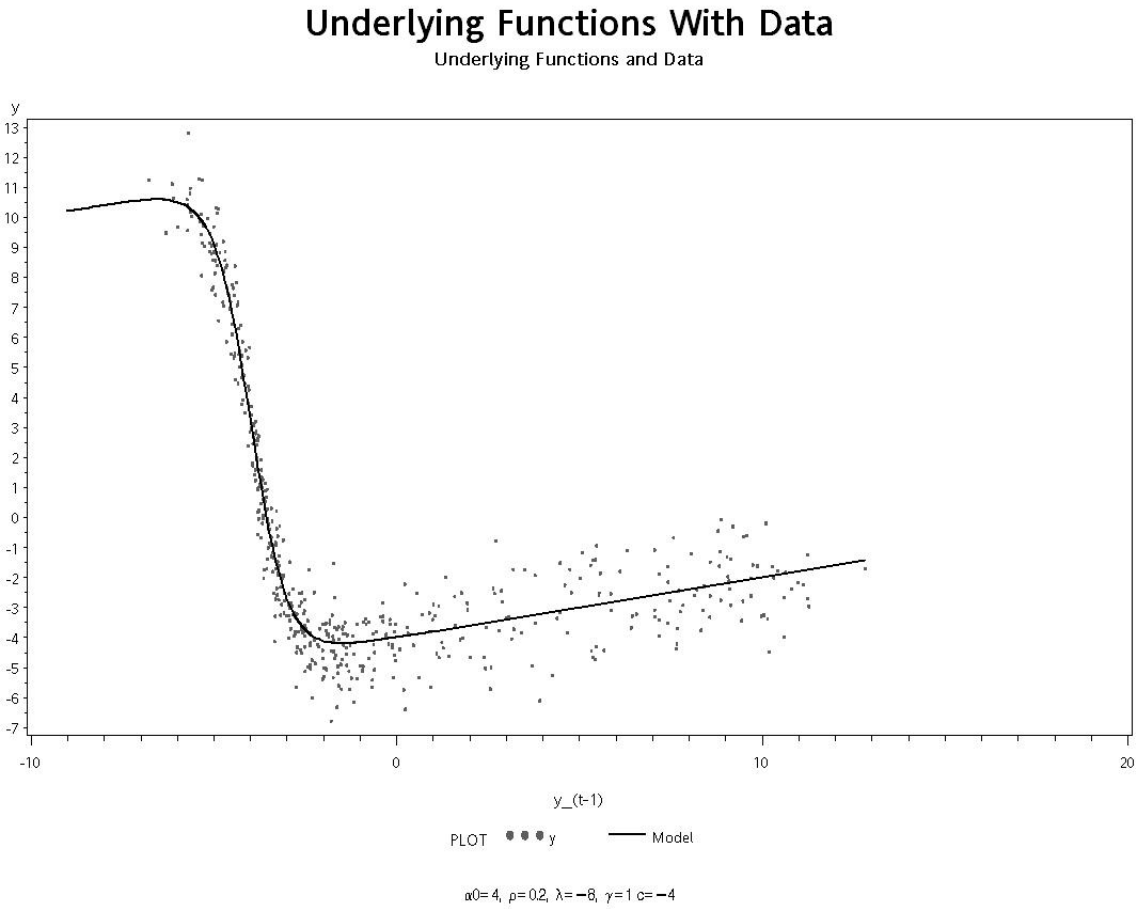

Figure 2: The generated data and the function that generated them are plotted together in order to show how the data relate to the underlying functions. In this case, the weight on the hyperbolic tangent function (λ) is negative and large in absolute value and the location parameters are far apart ($|\alpha_0 - c|$). The data follow the underlying model function closely, and the parameters for the nonlinear model are easy to estimate.

portant in determining whether it is possible to estimate the parameters of the neural network.

In later simulations, we choose to exclude parameter combinations where

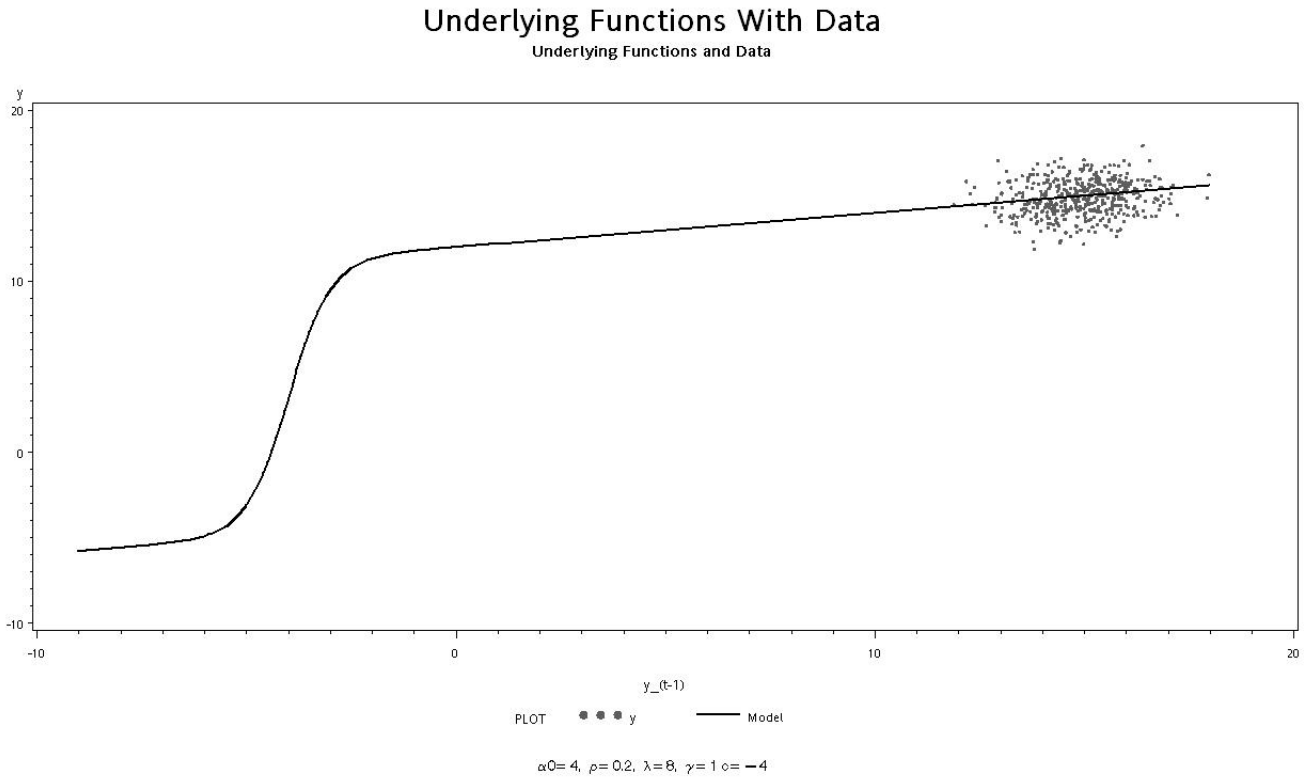

Figure 3: The generated data and the function that generated them are plotted together in order to show how the generated data relate to the underlying functions. In this case, the data cluster near an attractor that is on the flat portion of the hyperbolic tangent, and the function adds little more than a constant.

the AR(1) provides a good fit for the generated data. Because of the attraction points for these series, the data seem to follow a straight line, and it is unlikely that a researcher trying to forecast such a series would be tempted to estimate an autoregressive neural network. These series are also those most likely to exhibit estimation problems. We use the fact that $\hat{\delta}_k$ is the mean difference between the MSE estimated by the best-fitting AR(1) and

the true MSE for each series. For each parameter combination, we conduct a t-test on $\hat{\delta}_k$, where the null hypothesis is that $\delta_k \leq 0$ and the alternative is $\delta_k > 0$. The p-values for this one-sided hypothesis test are displayed in Figure 4. In later simulations, we eliminate parameter combinations where the p-value from this hypothesis test are > 0.05.

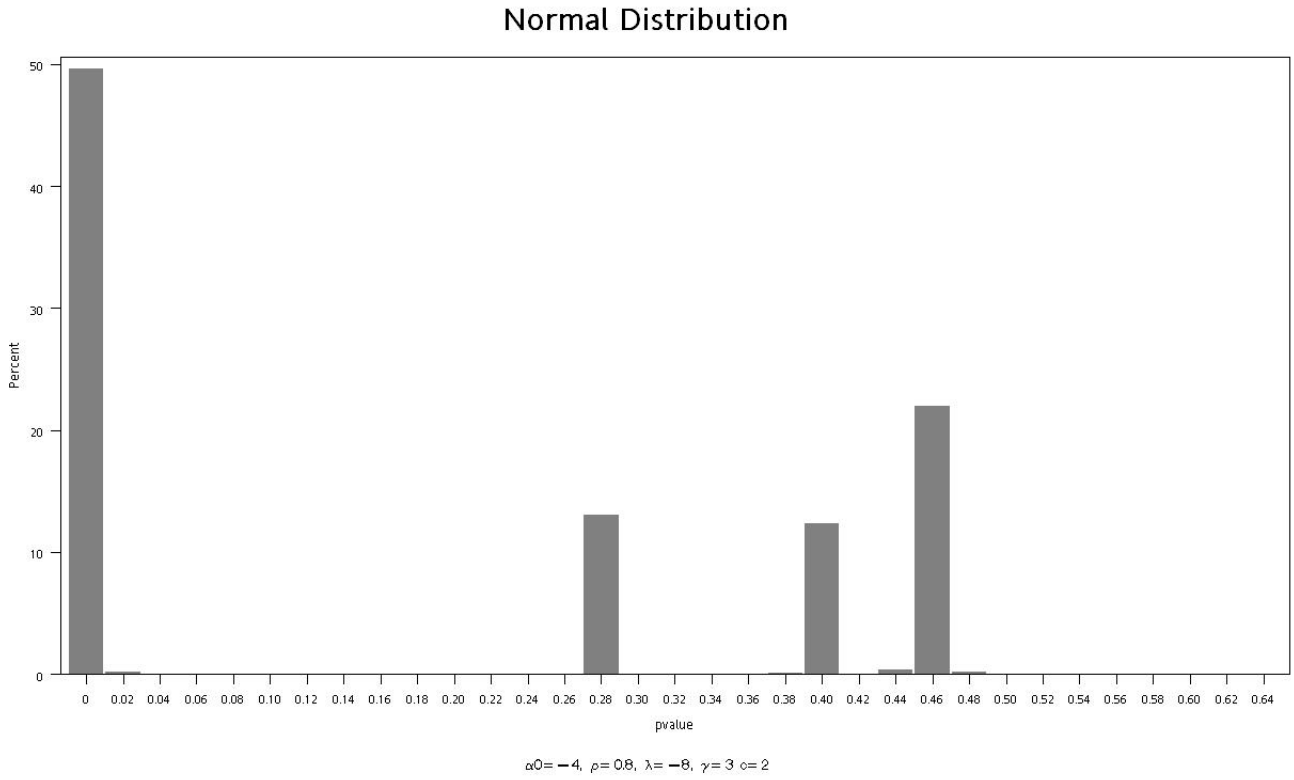

Figure 4: Distribution of p-values from a one-sided hypothesis test where the null hypothesis is $\delta_k \leq 0$. For the simulation, we use a cut-off of 0.05 and drop any parameter combination where the p-value from the one-sided hypothesis test is greater than 0.05.

Because failing to reject the null hypothesis eliminates a model from consideration, higher cutoffs including, $p < 0.1$ and $p < 0.3$, were considered.

It was found, however, that models with p-values between 0.05 and 0.2 were models where neither the AR(1) nor the AR-NN fit the data well.

4.1 The Importance of Noise

The simulations in this section are produced assuming that the noise term e_t is distributed $N(0, 1)$. The normal distribution is a popular default assumption in model estimation, but for a neural network model, it may not be the appropriate assumption. Recall that the proof in section 1 shows that for a bounded activation function (which is required in order for the AR-NN to be stationary) and additive noise that is *iid*, there is an acceptable range for the location parameters. If the location parameters are outside of the acceptable range, the series is not irreducible in a practical sense, and we cannot assume it is possible to estimate AR-NN parameters.

If e_t has a heavy-tailed distribution, the range restrictions on the location parameters can be loosened. To demonstrate, we perform the same simulation as outlined in section 4, except instead of using a noise term that is distributed $N(0, 1)$, we use the logistic distribution. The pdf of the logistic distribution is:

$$\frac{\mathrm{e}^{-(x-\mu)}}{s(1+\mathrm{e}^{-(x-\mu)/s})^2}$$

with scale parameter s and location parameter μ.

We choose s and μ so that the noise term is distributed with mean 0 and standard deviation 1. The logistic distribution, however, has more data in

the "tails", or at the extremes for e_t, which gives the series a wider range and should result in more series with nonlinear behavior than the original design. Figure 5 shows the p-values for δ_k when the same simulation is run using e_t generated from this distribution.

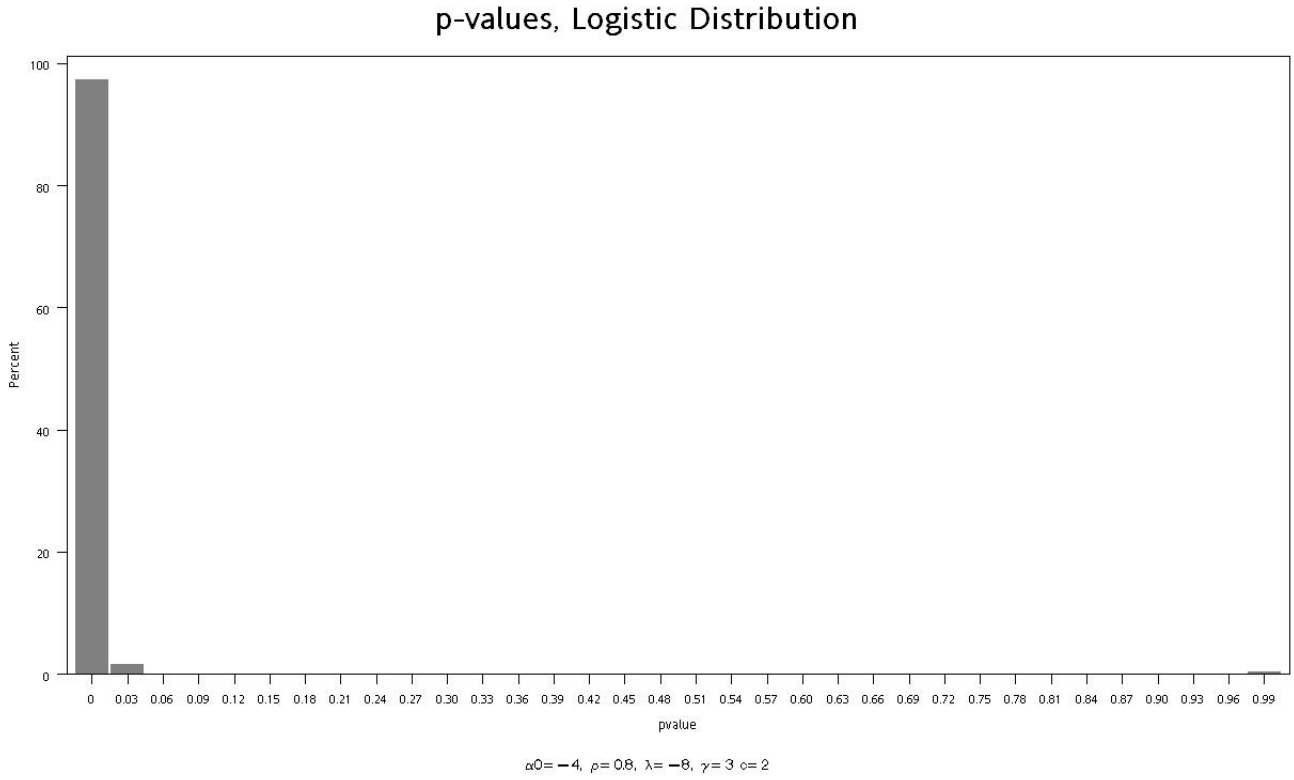

Figure 5: When the same simulation is run using a heavy-tailed distribution (in this case, the logistic) for e_t, the p-values are dramatically different.

In only a few cases were the p-values for the logistic distribution larger than our cut-off of 0.05, and all of these were parameter combinations where the generated data were very close to a straight line.

The heavy-tailed distribution may be a more realistic assumption for

applications such as economic data, where the outcome is expected to have a great deal of unexplained variation. The simulations for forecast performance (section 5) will include comparisons between simulations with normal noise and simulations with logistic noise.

5 Forecasting With Neural Networks

As pointed out in Gorr (1994), a neural network offers little insight to the underlying structure of the data. It is an approximation of an unknown relationship, and the parameter estimates are difficult to interpret. We therefore estimate parameters from our simulated series and evaluate how well the estimated model predicts its own data. We study two different different types of forecasts: a one-step-ahead forecast, where the most recent lag is used to predict the next value, and a twelve-period forecast horizon, mimicking a year-long forecast for a monthly time series. The simulation is limited to the 4800 parameter combinations identified in section 4 as those most likely to produce series with nonlinear properties.

For each generated series, we reserve the last twelve time periods as a holdback sample and use them to compare the out-of-sample performance for the AR-NN with the best-fitting AR(1) model and a naive random walk, where the next period forecast is the last observed values of y_t. The nonlinear model is estimated with nonlinear least squares, and the true parameter values are used as starting values. The parameters of the AR(1) are calculated using ordinary least squares. For both types of model, we omit the first observation of the series ($t = 1$) to mimic a true time series forecasting problem, where the lag for the first observation is unavailable. The fitting routine is Levenberg-Marquardt.

In most cases, the AR(1) model produces a very poor fit, which is not surprising since we have subset our parameter combinations to the 4800 we

consider most likely to generate data with nonlinear features (see section 4). There were also some cases where the nonlinear fitting routine reported a convergence failure, though these were rare and not consistent for any given parameter combination. When this happened, we used the results from the final iteration as our parameter estimates for the forecast. While this may seem counter-intuitive, we argue that it is a reasonable approximation of what occurs in the field. "Convergence" for a nonlinear model is often a judgment call. When working with a fitting routine programmed into a software package such as SAS or R, modelers will often work around convergence problems either by changing starting values (sometimes randomly or by a line search), changing the fitting method, or "loosening" convergence criteria. There are no standards or best practices for these procedures (Crone *et al* 2008). Because the starting values for the model are the true parameters, we argue that the last iteration from the fitting routine is as likely to be a true minimum for the objective function as any that is found by arbitrarily changing starting values or fitting criteria.

The estimation procedure for the neural network also sometimes reported convergence, but for parameter estimates that were unreasonable, with values for the location parameters near ± 90 and of opposite sign or values for ρ that were larger than 1 in absolute value. This problem is reminiscent of the troubles reported in Terasvirta (2005), where an "insanity filter" was applied to keep the AR-NN from generating unreasonable forecasts. It appears that unreasonable parameter values are possible, even when the architecture of the

AR-NN is known and the true parameter values are used as starting values for the nonlinear model. In our simulation, the unreasonable parameter estimates occurred for only 3% of the cases and not consistently for any parameter combination. These series were dropped before calculating fit statistics.

For each model, the forecasts are produced in one of two ways:

1. A point forecast, where the last lag (y_{t-1}) is used to predict the next value in the sequence (y_t). The function is then updated with the true value for the next period's forecast.

2. A bootstrap forecast, where at each point 500 different values for y_{t-1} are generated from the assumed distribution, and the forecast is an average of the forecasts from these generated values.

i.e. the bootstrap forecast is the result of:

$$\hat{y}_t = \frac{1}{500} \sum_{i=1}^{500} f(y_{t-1} + \varepsilon_i, \hat{\theta})$$

where f is the function that describes the neural network relationship, $\hat{\theta}$ is the vector of parameter estimates, and $\varepsilon_i \sim N(0, 1)$. In Terasvirta (2005) this procedure is recommended in order to stabilize forecasts from a neural network model.

Because the AR(1) is a linear model, the point forecast and the bootstrap forecasts will be the same, except for the bootstrapping error. There is no

reason to do both. In the tables, the bootstrap AR-NN predictions are labeled as BS. Point forecasts are labeled with a P.

A random-walk R-squared (RW R-squared) was used to evaluate the forecast performance on the holdout sample. The RW R-squared is calculated as follows:

$$\begin{aligned} RW\ R-squared &= 1 - \frac{SSE}{RW\ SSE} * \frac{T-2}{T} \\ RW\ SSE &= \sum_{t=2}^{T} (y_t - y_{t-1})^2 \end{aligned} \tag{11}$$

The RW R-squared can be thought of as a ratio of R-squares from a random walk forecast (where we simply use the last value of y_t as our forecast for the next value of y_t) and the model being evaluated. The RW R-squared is often a better measure of forecast performance than a standard R-squared, as it considers a more reasonable naive model than merely using the mean across all time periods. For each parameter combination, the RW R-squared was averaged over the 20 different sequences of errors. The descriptive statistics over all parameter combinations are shown in Table 7.

One of the interesting features of the analysis is the difference between the bootstrap estimates and the point estimates. The bootstrap estimates have a lower mean RW R-squared, and their range is much wider. Not only do the bootstrap estimates perform worse than the point estimates overall, there is a potential for disastrous performance (an RW R-squared that is less -1), where the bootstrap forecast performs worse than the naive model.

Descriptive Statistics of RW R-squared

In-Sample Performance

	Mean	Median	Std.	Min	Max
AR-NN (P)	0.846	0.954	0.233	-.117	0.992
AR-NN (BS)	0.714	0.830	0.314	-3.90	0.992
AR(1)	0.475	0.491	0.356	-.594	0.984

Out-of-Sample Performance

	Mean	Median	Std.	Min	Max
AR-NN (P)	0.911	0.981	0.137	0.116	0.998
AR-NN (BS)	0.877	0.956	0.164	-1.55	0.998
AR(1)	0.809	0.843	0.149	-.090	0.994

Table 2: Although it is recommended in Terasvirta (2005) that a bootstrap approach be used to stabilize AR-NN predictions, the point forecasts have better statistics overall for one-step-ahead forecasts.

The cases where the RW R-squared for the AR-NN was much higher than the RW R-squared for the AR(1) are, not surprisingly, cases where the values of λ are large in absolute value and negative, or when $|\lambda|$ is small and $|\alpha_0 - c|$ is small as well. These are the cases where the relationship between y_{t-1} and y_t is most obviously nonlinear (Figure 6).

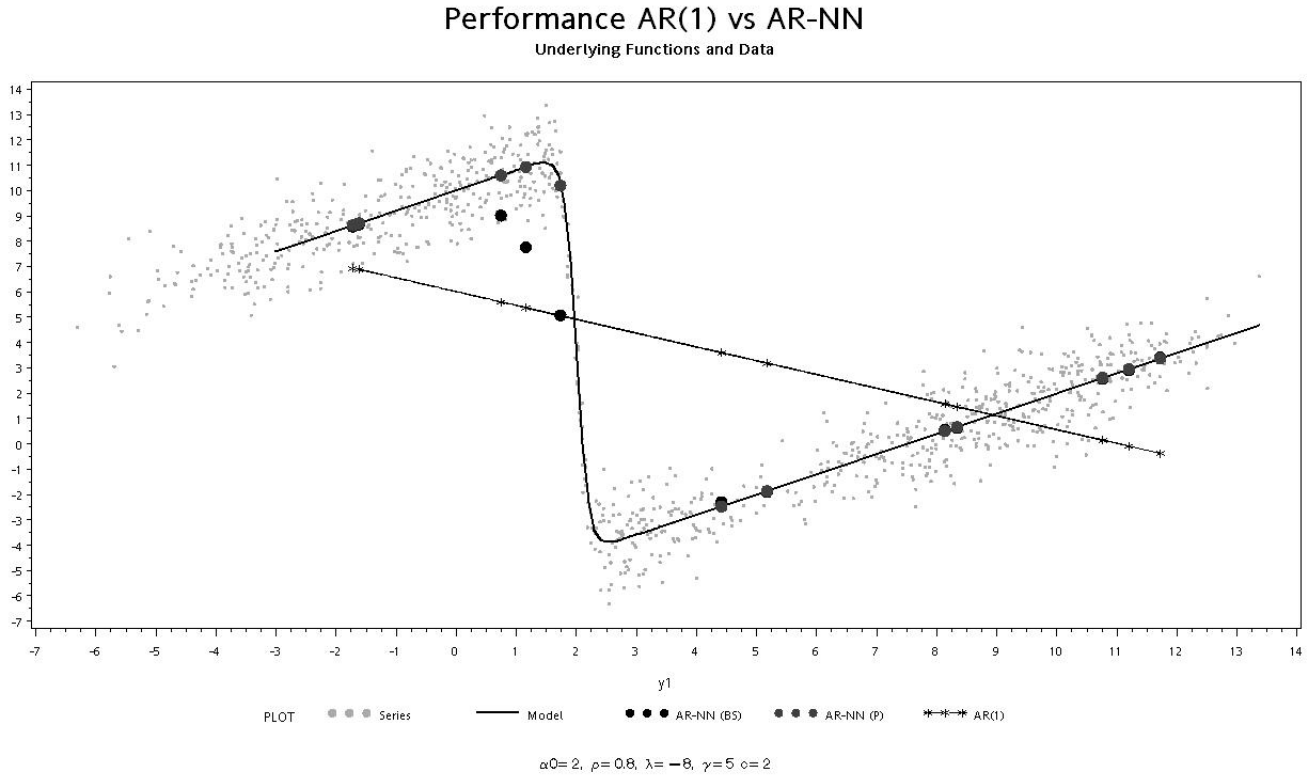

Figure 6: In the above graph, the straight line represents the estimated AR(1), and the curved line is the estimated nonlinear relationship between y_t and y_{t-1}. In this case, it is obvious that the nonlinear model is a better fit for the data, and the RW R-squared is close to 1. The bootstrap forecasts (black squares) are not exactly on the model function because they are an average of several forecasts for different possible values of y_{t-1}.

5.1 Prediction At Extremes

While the AR(1) model did not out-perform the AR-NN, the table shows that it did fairly well. In fact, the ratio of RW R-squares between the forecast for the AR(1) and the forecast for the AR-NN (point or bootstrap) hovers around 1.25, meaning that the AR-NN only beat the AR(1) model's performance by approximately 25% for most cases.

Because of the shape of the hyperbolic tangent function, the nonlinear features of the series are near the highest and lowest values of y_t. For some applications, such as energy demand and stock prices, it might be most important to have accurate forecasts for extremes. In that case, a more complicated model would be a better choice, even if its overall performance is similar to a simpler model. In Table 3, we compare the descriptive statistics for RW R-squared for different levels of y_t: above the series 90th percentile, below the series 10th percentile, and between the 90th and the 10th percentiles.

The table shows that while the AR(1) model has decent overall performance, it mostly does well when y_t is in the middle 80% of the data. The median of the RW R-squared for the point forecasts, however, remains consistently high across the three categories.

The bootstrap forecast does not do as well as the point forecast at extremes, which is not surprising since it is an average over a selection of possible values for y_{t-1} (high and low) over the assumed distribution.

Descriptive Statistics of RW R-squared
By Percentile of y_t

		Mean	Median	Std.	Min	Max
Lower 10%	AR-NN (P)	0.87	0.97	0.21	-0.04	1.00
	AR-NN (BS)	0.76	0.89	0.32	-2.70	0.99
	AR(1)	0.56	0.73	0.40	-1.22	0.99
Middle 80%	AR-NN (P)	0.89	0.97	0.16	0.17	1.00
	AR-NN (BS)	0.86	0.95	0.20	-4.90	1.00
	AR(1)	0.78	0.81	0.17	0.02	0.99
Upper 90%	AR-NN (P)	0.82	0.96	0.30	-0.33	1.00
	AR-NN (BS)	0.64	0.86	0.51	-6.67	0.99
	AR(1)	0.32	0.67	0.72	-3.14	0.98

Table 3: The AR-NN forecasts were better at extreme values of y_t. This could be useful in models where it is more important to be accurate when series values are high or low.

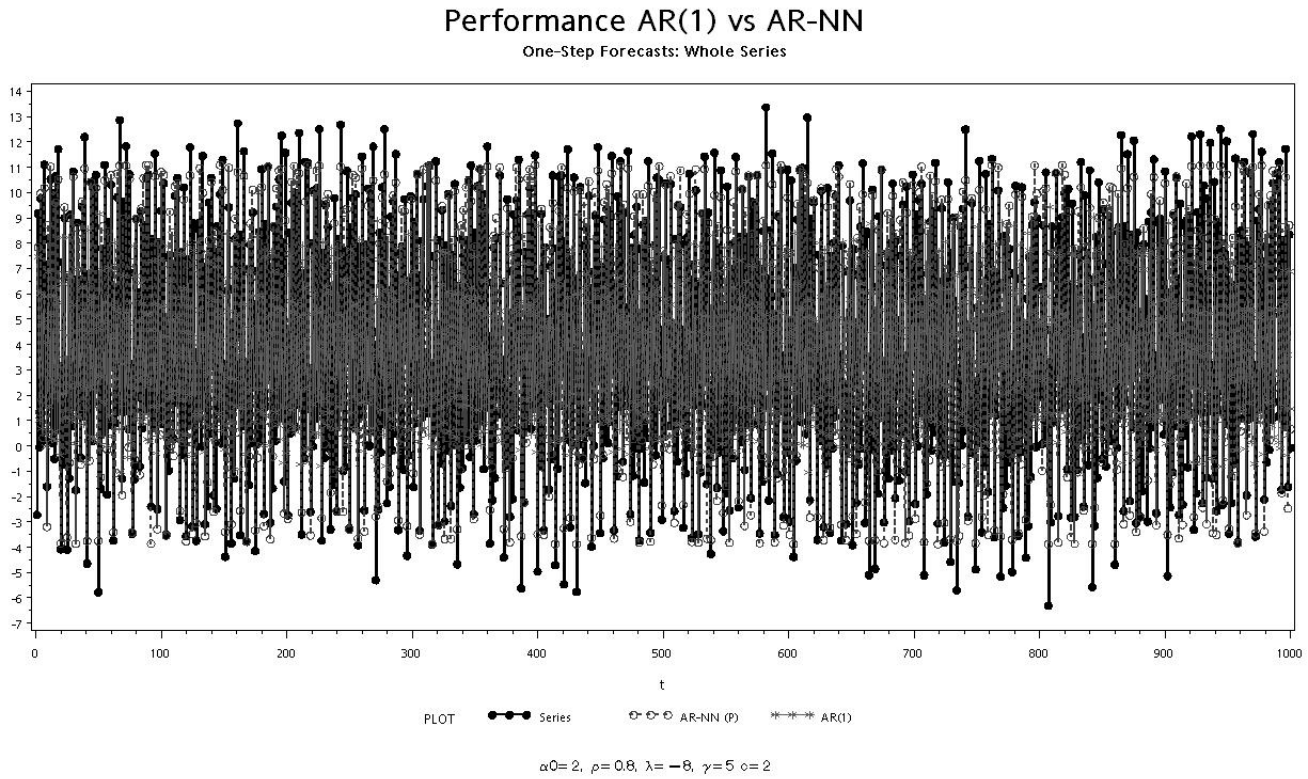

Figure 7: The AR(1) forecast (the gray line in the graph) tends to stay closer to the series mean than the AR-NN model. This means it is less able to adjust to dips and spikes that the AR-NN (open circles) can predict rather well. For the graph above, the AR-NN model performs much better (in terms of RW R-squared) than the linear model.

5.2 Long Horizon Forecasts

It has been hypothesized that even when a nonlinear time series model is not impressive for short forecasts, they could perform better at a long forecast horizon (Swanson and White 1997). In our study, it appears that the opposite has happened. Recall that a test sample of 12 time periods was withheld from each generated series. We now perform a 12-period forecast for both the estimated AR-NN and the estimated AR(1).

The 12-period forecasts were much worse, as measured by RW R-squared, than the one-step-ahead forecasts. For these long forecast horizons, however, the bootstrap forecasts were better than the point forecasts, with a higher mean and median, though the difference could be explained by random variation. (Table 4)

Examining the underlying model function and the generated data may shed some light on why the long horizon forecasts are performing so poorly. Figure 9 shows the actual model function, the estimated model function, the data, and three different kinds of forecasts: AR-NN (BS), AR-NN (P) and random walk. The horizontal line represents the random walk forecast and is simply the last value of y_t that is observed in the training data.

The forecast is actually an estimate of the nonlinear series skeleton, as defined in Tong (1993) and in section 1. For cases where the skeleton causes the series to cycle between two extremes, the forecast error will either be close to 0 (when the model guesses correctly which cluster the next value for y_t will fall into) or close to $\max(y_t) - \min(y_t)$ (when the model guesses incorrectly).

Descriptive Statistics of RW R-squared: 12-Step Forecasts

	In-Sample Performance				
Mean	Median	Std.	Min	Max	
AR-NN (P)	0.18	0.13	0.15	-0.11	0.54
AR-NN (BS)	0.26	0.28	0.11	-0.40	0.54
AR(1)	-0.06	0.39	0.95	-4.00	0.50
	Out-of-Sample Performance				
	Mean	Median	Std.	Min	Max
AR-NN (P)	0.38	0.50	0.61	-8.42	0.93
AR-NN (BS)	0.46	0.55	0.76	-24.06	0.93
AR(1)	-0.24	0.34	2.13	-54.74	0.92

Table 4: The RW R-squares for the 12-step-ahead forecasts were much worse than the one-step forecasts.

If the additive noise (e_t) is large enough compared to the range of the data, y_t can move out of phase with its skeleton. When this happens, the point forecast will be out of phase with the actual value of y_t and will remain out of phase until another value of e_t moves it back. This will leave us with a disproportionate number of residuals that are close to $\max(y_t) - \min(y_t)$.

We would expect the errors for the random walk forecast to also be close to 0 for some points and $\max(y_t) - \min(y_t)$ for the rest. If the last observed y_t happens to be in the center of the training data, however, the random walk forecast errors will be smaller than expected. This leads to a small denominator in equation (11) and therefore a small RW R-squared.

Meanwhile, the bootstrap estimates are produced by adding random noise

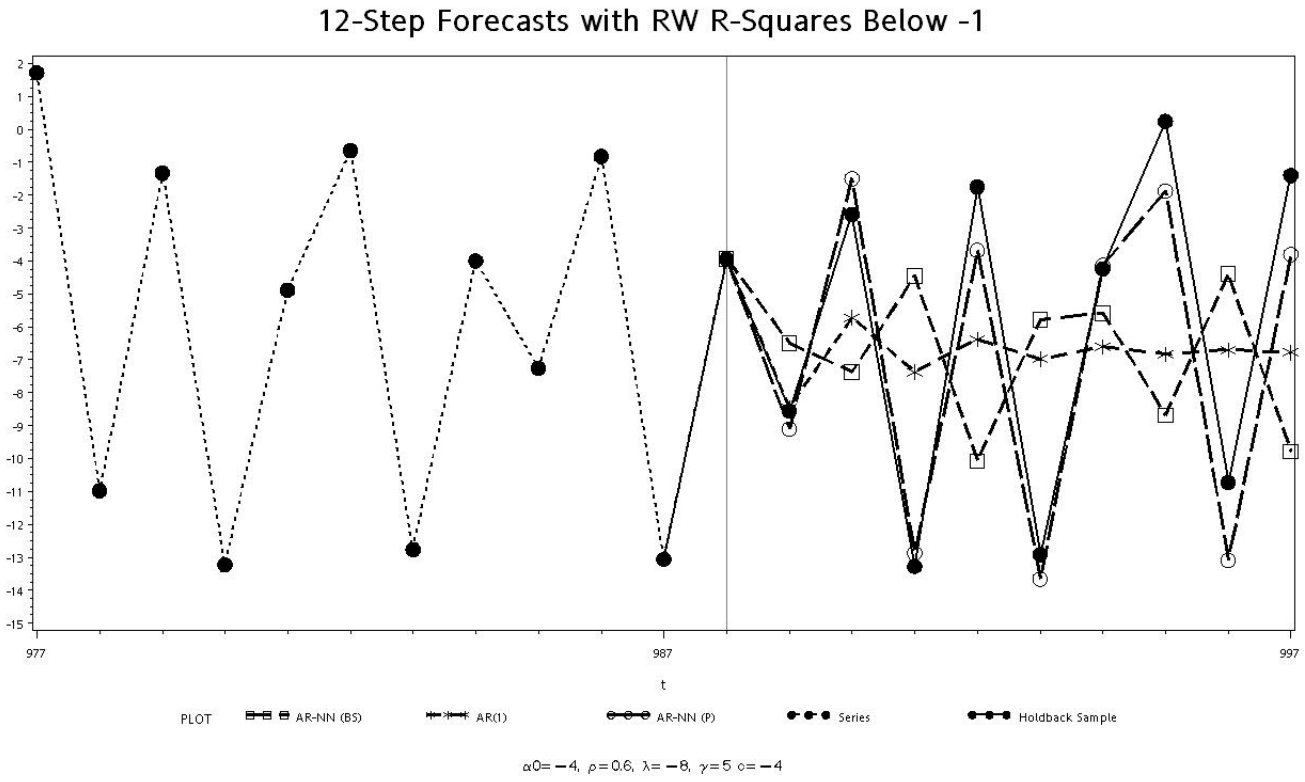

Figure 8: In the above graph, forecasts are represented with dashed lines. The estimated model relationship is really an estimate of the series skeleton. If the estimated skeleton has multiple attraction points, the noise can cause the series to move out of phase. While the AR-NN forecast may be near the actual series for the first few steps, it will eventually move in the opposite direction.

to the last observed value of y_t and generating many (in this case 500) possible forecasts. The bootstrap is then the average of the results, which pulls the estimates close to the center of the data. This is why the RW R-squares for the bootstrap forecasts are slightly better than point forecasts at long-horizons. Neither method, however, performs well, and these results show that forecasting with a neural network model can be problematic. As we see

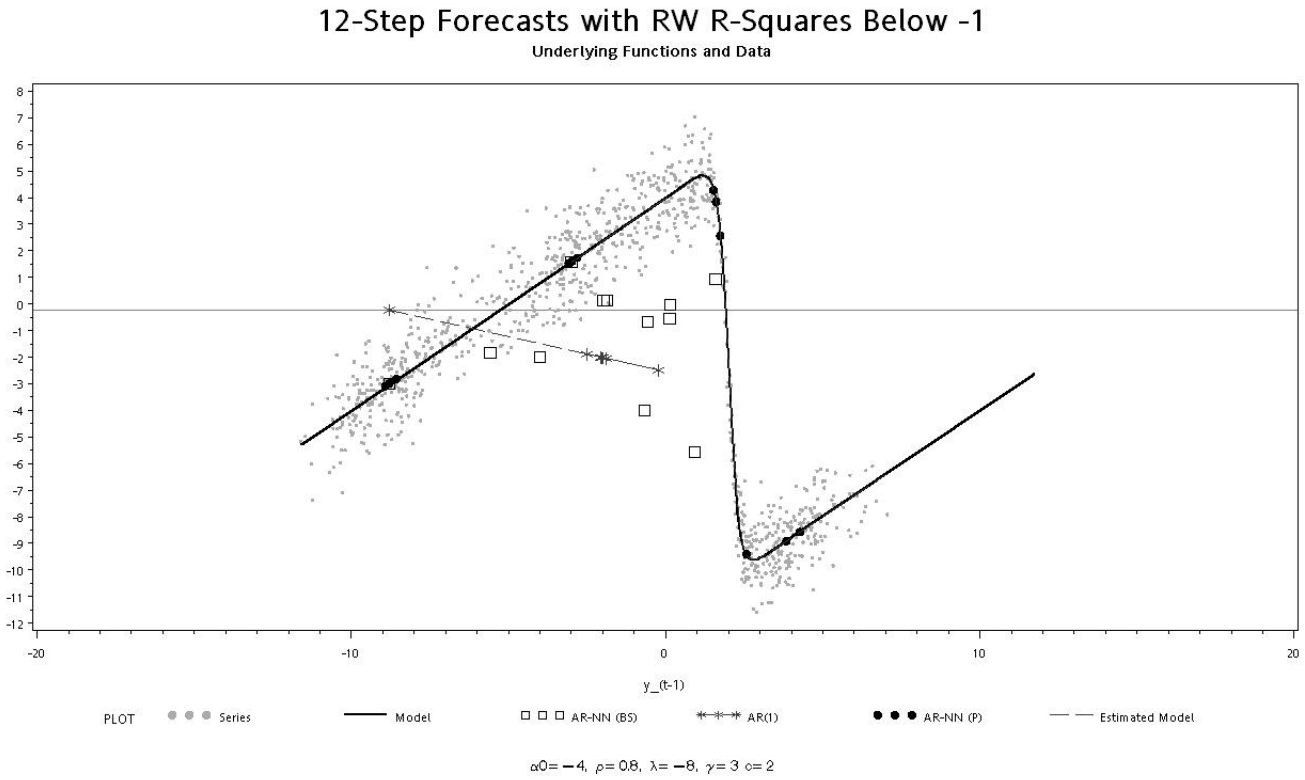

Figure 9: For this model, the parameter estimates were almost exactly equal to the true parameter values, yet the RW R-squares were very low for both point (black dots) and bootstrap (black squares) estimates. This is because the series is moving randomly between extremes.

in section 1, there must be enough random variation in the data to allow the series to visit all necessary elements of the solution space, yet the noise must not be the strongest underlying feature in the model.

5.3 Forecast Performance on Data Simulated with the Logistic Distribution

In section 4.1, we investigated changing the assumption that $e_t \sim N(0, 1)$ and instead used a logistic distribution with scale and location parameter chosen so that e_t has mean 0 and standard deviation 1. The logistic distribution is symmetric but heavy-tailed, resulting in a broader range of values for y_t. The results in that section showed that when the logistic distribution was used for y_t, almost all of the models exhibited nonlinear behavior, as measured by the difference between the true mean sum of squares (calculated from the errors used in the simulation) and the mean squared error from an estimated AR(1) model.

In Tables 5 and 6, we show the RW R-squared statistics for one-step and long-horizon forecasts when the data are simulated using a heavy-tailed distribution. In these tables, it is shown that the predictive performance of the AR-NN is very poor when the series is generated using a heavy-tailed distribution. This is not surprising. Assuming a heavy-tailed distribution for the noise term signifies that we are less certain about the value of y_t and expect more values that are far from what is predicted by the deterministic portion of the model. It is arguable that no model, even the correct one, would predict this data well.

One interesting difference between series simulated with the logistic distribution and series simulated with the normal distribution is the difference between the long forecast horizons and the short forecast horizons. For the

Descriptive Statistics of RW R-squared: One-Step-Ahead Forecasts
Logistic Distribution

	In-Sample Performance				
	Mean	Median	Std.	Min	Max
AR-NN (P)	0.347	0.264	0.254	-.030	0.818
AR-NN (BS)	0.338	0.258	0.249	-.032	0.809
AR(1)	0.241	0.184	0.234	-.098	0.724
	Out-of-Sample Performance				
	Mean	Median	Std.	Min	Max
AR-NN (P)	0.645	0.611	0.134	0.217	0.901
AR-NN (BS)	0.644	0.611	0.133	0.276	0.896
AR(1)	0.602	0.583	0.124	0.279	0.856

Table 5: When the logistic distribution is used to create the simulation data, the RW R-squares have much lower means and medians. The series is more likely to have some nonlinear features, allowing us to estimate the parameters, but the heavy tails of the logistic distribution mean there is more inherent noise.

data simulated with the normal distribution, the long-term forecasts were much worse than the one-step-ahead forecasts. For the logistic distribution, performance, although poor, was consistent whether we were predicting at long horizons or short horizons. This shows, as was shown in section 4, that the distributional assumptions are very important in nonlinear models. A small change from a normal to a symmetric heavy-tailed distribution makes a very large difference, even when the mean and variance of the noise term are the same.

Descriptive Statistics of RW R-squared: 12-Step Forecasts
Logistic Distribution

	In-Sample Performance				
Mean	Median	Std.	Min	Max	
AR-NN (P)	0.310	0.318	0.061	0.150	0.407
AR-NN (BS)	0.331	0.343	0.056	0.156	0.411
AR(1)	0.405	0.408	0.017	0.164	0.422

	Out-of-Sample Performance				
	Mean	Median	Std.	Min	Max
AR-NN (P)	0.428	0.546	0.859	-20.0	0.711
AR-NN (BS)	0.452	0.569	0.849	-19.7	0.744
AR(1)	0.533	0.628	0.558	-11.9	0.782

Table 6: When the logistic distribution is used to create the simulation data, the RW R-squares have much lower means and medians, but their minimum values are much higher. The series is more likely to have some nonlinear features, allowing us to estimate the parameters, but the heavy tails of the logistic distribution mean there is more inherent noise.

In this section we investigate how well a neural network model is able to forecast simulated data with known architecture. We use the actual parameter values as starting values in the nonlinear fitting routine and generate both a one-step-ahead and a twelve period forecast horizon, using both a point forecast and a bootstrap forecast as suggested in Terasvirta (2005). We find that the point forecast out-performs the bootstrap model at short and long horizons. Neither performed well in the twelve-period case. We have also shown that the assumptions about the noise term matter when investigating forecast performance. A symmetric heavy-tailed distribution with zero mean and a standard deviation of 1 produced very different results from a normal distribution with a zero mean and a standard deviation of 1.

In this investigation, our starting values give the model a performance "head start". In the next section, we examine different methods for choosing starting values for estimating a neural network model when we do not know the underlying parameters, which is the only case of practical interest.

6 Starting Value Selection

The AR-NN is, at base, a nonlinear model, and the estimates produced for nonlinear models are notoriously sensitive to starting values. It therefore makes sense to ask whether we can choose the starting values intelligently for our AR-NN so as to maximize our chances of producing a model that fits the data well. Other studies of neural networks for forecasting time

series either do not consider starting values at all or else use a grid search mechanism in order to fit the parameters.

Recall that our neural network model is:

$$y_t = \alpha_0 + \rho y_{t-1} + \lambda \tanh(\gamma(y_{t-1} - c)) + e_t$$

$$e_t \sim N(0, 1)$$

Even under the assumption that the architecture is known, if we use our factorial design as a guide, grid searching to produce starting values for this model would require us to try 3200 parameter combinations at the beginning of the estimation routine. If the grid search were guaranteed to produce the parameter estimates that minimized the objective function, the resources required would be justified, but this is not the case. In Hush (1991) it is shown that training a neural network model with a sigmoidal node is NP-hard, meaning that an exhaustive search of every possible parameter combination may be required in order to find the global minimum. If the parameter space is continuous, which must be the case in practice, then even our factorial design would not be adequate.

Further, in these results, we will show that a grid search on the ρ parameter actually produces worse estimates than simply choosing a value in the middle of the search space. Our data-based starting value routine, on the other hand, produces comparable results to using the true parameter values

as starting values.

6.1 Starting Value Simulation

As in the forecasting simulation, we produce 20 sequences of errors, which we use to generate 20 different series for each parameter combination. We then estimate a model from the generated data assuming that the architecture is known and using the following steps to determine start values:

1. Calculate the overall median of the series. This is the starting value for the location parameters α_0 and c.

2. Calculate the slope parameter from an AR(1) model. Call this number $\hat{\rho}$. The starting value for γ is $\text{sign}(\hat{\rho})$.

3. Calculate the largest deviation from the median of the series. This is the starting value for λ.

4. Set the starting value for ρ to 0.5.

The mean was also considered as a starting point for the location parameters, but because of the attraction point, many parameter combinations produced data that were skewed, which made the mean a poor measure of the distribution's center. The median's performance, therefore, was vastly superior to the mean (results not shown).

Our starting value routine produces convergence almost as often as using the true parameter values as the starting values. Not only that, but the

Model Convergence and Parameter Estimates

		How Start Values Were Determined			
		True		Start Value Routine	
Converged	Bad Parameter Est.	9,571	10.0%	6,018	6.3%
	Good Parameter Est	78,288	81.4%	81,974	85.3%
Did Not Converge	Bad Parameter Est	668	0.7%	365	0.4%
	Good Parameter Est	7,633	8.0%	7,803	8.1%
		96,160	100.0%	96,160	100.0%

Table 7: Of all the starting value routines that were tried, the Levenberg-Marquardt estimation method using the median as the starting value for the location parameters was obviously the best, arguably even better than using the true values of the parameter estimates.

starting value routine produces reasonable parameter estimates *more often* than using the true parameter values.

Recall that the ρ parameter must be bounded between (-1,1), or else the neural network model is not stationary (Trapletti *et al* 2000). A grid search for the starting value of ρ is therefore practical, and we consider a grid search for ρ instead of the arbitrary point starting value of 0.5. Table 8 shows the results from using a grid search for ρ while keeping the rest of the starting value routine the same. For this simulation, grid searching produced much worse results than simply choosing $\rho = 0.5$. This could be because having more starting values to choose from at the beginning increased the probability that the estimation routine would stop at a local minimum rather than the global minimum.

Convergence Rates over 20 Runs
Grid Search on ρ versus Point Starting Values
Reduced Simulation

		How Start Values Were Determined			
		Point Start for ρ		Grid Search on ρ	
Converged	Bad Parameter Est.	6,018	6.3%	4,797	5.0%
	Good Parameter Est	81,974	85.3%	54,666	56.9%
Did Not Converge	Bad Parameter Est	365	0.4%	381	0.4%
	Good Parameter Est	7,803	8.1%	36,697	37.8%
		96,160	100.0%	96,160	100.0%

Table 8: Grid searching for the starting values for ρ produced worse results than using a point value.

While model convergence is important, it is subjective. Different software packages use different criteria for whether a fitting routine has converged, and nearly all commercial packages allow the user to change these criteria as needed. It is therefore necessary to consider not only whether the software reports convergence, but whether the resulting model is able to perform in the required manner. For an AR-NN, the primary goal is forecasting, and therefore it is useful to compare the unexplained variation (i.e. the mean squared error) resulting from the different starting value methods.

We start by calculating the statistics for the MSE when the true parameter values are used as the starting values (Table 9). Right away, we see that software-reported convergence is not reliable. The MSEs calculated from the model where the true parameter values are used as the starting value are

Descriptive Statistics for Mean-Squared Error
True Values as Start

N	Mean	Median	Std	Min	Max
96,180	1.01	1.00	0.05	0.92	1.10

Table 9: When the true parameter values are used as the start values in the fitting routine, the mean-squared error from the resulting model is nearly always approximately 1, i.e. the expected MSE for our model where the noise term is distributed $N(0, 1)$.

approximately 1 in all cases, i.e. the expected value when our noise term is distributed $N(0, 1)$. In contrast, the default settings on a nonlinear fitting routine reported convergence for only 85% of the cases studied.

We now examine the MSE from a model estimated when we use the starting value routine under study and compare it to the MSE from a model produced when the true starting values are used as parameter values. Table 10 reports descriptive statistics for the difference between the MSEs across all parameter combinations.

As with the forecasting study, we exclude models where the parameter estimates were unreasonable ($|\rho| > 1$,$|\alpha_0| > 90, |\lambda| > 90$), or $|c| > 90$, which excluded approximately 3% of the replicates. The results from the comparison between the grid search and the point starting values for ρ were similar to results when the convergence was used as a measure of performance: the original routine worked best. In this case, however, using the true parameter estimates produced a model with the minimum MSE. The large maximum difference in model MSEs, even when the insanity filter is applied, shows

Descriptive Statistics for Mean-Squared Error
Reduced Simulation

	Difference in MSE Between True Parameters as Starting Values and Starting Value Routines				
Method	Mean	Median	Std	Min	Max
No Grid Search	4.94	0.00	38.92	-0.00	2,693.94
Grid Search	30.17	1.18	81.20	-0.00	3,976.46

Table 10: Because this is a predictive model, it is important to consider how well the estimated model performs. Here, we measure performance using the difference between the mean squared error for a model produced using the true parameters as starting values and the starting value method under study.

that it is possible to get unreasonable answers even when the starting values seem to be chosen reasonably. Fitting the nonlinear model, as pointed out in Hush (1991) and Faraway and Chatfield (1998), is a step that cannot be ignored.

7 Conclusions

In this paper, we focus on the existing theory for nonlinear time series as it relates to neural network models, and how that theory can be used as a guide in applying neural networks to time series forecasting. We have several results, including:

1. When the activation function is bounded, the process skeleton is also bounded, and for noise terms distributed *iid*, the series Y_t is bounded

in probability. This can have a non-trivial effect on whether the parameters of the AR-NN can be estimated. For simulation, this allows the researcher to limit the design space. For practical applications, the researcher can use the theoretical limits as a guide in selecting starting values.

2. The location of the neural network attraction point(s) is important in determining long-run behavior of the network. If the attraction point is such that the generated data are not near the location parameter(s) for the activation function (called c_j in this analysis), it is possible for the series data to cluster in a location where there is little information about the parameters of the nonlinear model. In these cases, the neural network we studied behaves like an AR(1).

3. It has been hypothesized that nonlinear time series may perform better than linear models at long forecast horizons, even when the linear model has better or equal one-step-ahead performance. For a noise term with a normal distribution our results show the opposite: The neural network models we studied had very poor performance at long forecast horizons. When a heavy-tailed distribution was substituted for the normal distribution, the long and short horizon forecasts had similar performance.

4. For the models studied in this research, bootstrap forecasts produced worse results than point forecasts.

5. Even when the architecture of the neural network is known, it is important to obtain good starting values for the estimation routine. For our experiment, grid searching, which is commonly used in the absence of another guide for starting values, produced worse results than using a simple point starting value. A fitting routine is not complete without a way to select starting values.

6. The distribution of the noise term matters. Heavy-tailed distributions, which may be more realistic for many applications, produce very different results than the normal distribution.

The authors recognize that the model studied in this research is a relatively simple neural network model. Most studies of neural network forecasting for time series focus on much more complex models with multiple lags and multiple hidden units. It is possible that the models studied in this research tended to behave like linear models because of the presence of only one hidden unit, and a more complex model might display more nonlinear properties. If that were the case, we would expect the nonlinear model to perform better than the linear model.

The relative simplicity of this model, however, does show some important weaknesses in neural network modeling. We would not expect the estimation problems to become easier as the model becomes more complex. Recall that Hush (1991) showed that training a neural network is NP-hard. The number of potential parameter combinations that must be considered for estimation

increases quickly as lags and hidden units are added.

There is also justification in the literature for what we have shown here. In Faraway and Chatfield (1998), it was found that the neural networks estimated for the airline data were frequently linear in most directions, even though many lags and hidden units were considered. In Trapletti *et al* (2000), there is a note that "In practice, the Hessian tends to be poorly behaved," indicating linear dependencies among the first derivatives of the neural network that would be explained by mostly linear behavior among the lags. In Zhang and Qi (2005), it was shown that neural networks produced forecasts that were orders of magnitude better when the correct de-trending and deseasoning techniques were applied. Many researchers who have attempted to use neural networks for forecasting have discovered problems similar to the ones we have found here.

It is also important to note that, while all of the series meet the conditions set in Trapletti *et al* (2000) for the parameters to be estimable, none of our models meet the conditions set in Leoni (2009) for an attraction point that is globally and exponentially stable in the large. Many of our series had multiple attraction points. Some have only one unstable equilibrium. Without a global attraction point, it is difficult to argue that the series will be inherently predictable, even if it is possible to estimate the parameters. The comparison made in Tong (1993) between the mean for a stationary linear time series and a nonlinear series with a global attraction point is well-taken. We are able to predict stationary linear time series because we

know that when a random shock pulls the series away from the mean, it will be more likely to return to the mean at the next time period. We know that we can predict a nonlinear time series with a unique and globally and exponentially stable attractor because when the random portion of the series pushes it away from the attractor, the deterministic portion of the series will pull it back.

If a series has multiple attraction points, we would expect to see the data cluster at different locations. If the attractors are "far" from each other, then the series will collect at one attraction point, which will be determined by the starting value. If the series attraction points are "close", then the series will move from one cluster to another. These movements are determined by the noise term, and are therefore unpredictable. The neural network is simply an estimate of the series skeleton, and even if the architecture is known and the parameter estimates are close to the true parameters, the noise term can cause the actual series to become out of phase with its skeleton. Forecasts for such a series would be very unreliable. This is exactly what we see in the forecasting chapter, and it is not surprising that these series are difficult to forecast beyond a few time periods, even when the model parameters are estimated with perfection.

Though Gorr (1994) points out that neural network models are primarily for forecasting, even forecasters need to understand the properties of the series under study. Much of neural network estimation is still a black box, and it is important to continue developing theory and general results that

can help us build better estimation techniques and understand the results from neural network models.

References

[1] Box, G.E.P, G.M Jenkins, and Reinsel, G.C. (1994) *Time Series Analysis, Forecasting and Control* (3rd edition) Prentice Hall.

[2] Crone, SF, Nikolopoulos, K, Hibon, M. (2008) "Automatic Modelling and Forecasting with Artificial Neural Networks: A forecasting competition evaluation". *Final Report for the IIF/SAS Grant 2005/6*.

[3] Dickey, D.A., Zhang Y. (2010) "Seasonal unit root tests in long periodicity cases". *Journal of the Korean Statistical Society* In press.

[4] Faraway, J. Chatfield, C. (1998) "Time Series Forecasting with Neural Networks: A Comparative Study Using the Airline Data." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. 47(2) pp. 231-250.

[5] Fuller, Wayne. (19996) *Introduction to Statistical Time Series* John Wiley and Sons Inc.

[6] Gorr, W.L. (1994) "Research prospective on neural network forecasting". *International Journal of Forecasting*. 10(1) pp. 1-4.

[7] Hippert, H.S., Pedreira C.E. (2001) "Neural networks for short-term load forecasting: a review and evaluation." *IEEE Transactions on Power Systems*. 16(1) pp. 44-55.

[8] Hornik, K.; Stinchcombe, M.; White, H. (1989) "Multilayer feedforward networks are universal approximators " *Neural Networks*. 2(5) pp. 359-366.

[9] Hush, D R. (1991) "Training a Sigmoidal Node is Hard". *Neural Computation*. 11 pp. 1249-1260.

[10] Medeiros M, Terasvirta, T, Rech, G. (2006) "Building Neural Network Models for Time Series: A Statistical Approach." *Journal of Forecasting*. 25(1) pp. 49-75.

[11] Leoni, P. (2009) "Long-Range Out-of-Sample Properties of Autoregressive Neural Networks." *Neural Computation*. 21(1) pp. 1-8.

[12] Luukkonen, R, Saikkonen,P, Terasvirta, T. (1988) "Testing linearity against smooth transition autoregressive models" *Biometrika* 75(3) pp. 491-499.

[13] Sarangapani, J. (2006) *Neural Network Control of Nonlienar Discrete-Time Systems*. Control Engineering Series. Taylor and Francis Group.

[14] Sharda, R. and Patil, R.B., (1992) "Connectionist approach to time series prediction: An empirical test." *Journal of Intelligent Manufacturing* 3(5) pp. 317-323.

[15] Swanson N.R., White H. (1997) "Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models." *International Journal of Forecasting* 13(4) pp.439-461.

[16] Terasvirta, T, Dick van Dijkb, D., Medeiros, M.C. (2005) "Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination" *International Journal of Forecasting* 21(4) pp. 755-774.

[17] Tong H (1993) *Nonlinear Time Series: A Dynamical Systems Approach*. Oxford University Press.

[18] Trapletti A, Leisch F, Hornik K. (2000) Stationary and integrated autoregressive neural network processes. *Neural Computation*. 12(10) pp. 2427-2450.

[19] White, Halbert. "Learning in Artificial Neural Networks: A Statistical Perspective". *Neural Computation*. 1(4) pp. 425-464.

[20] Wun, L.M , Hung, K. (1988) "A note on the distribution of mean square error of forecasting." *Communications in statistics. Theory and methods* 17(6) pp. 1929-1934.

[21] Zhang, G.,Patuwo B.E., Hu, Michael Y. (1998) "Forecasting with artificial neural networks: The state of the art". *International Journal of Forecasting*. 14(1) pp. 35-62.

[22] Zhang, G. ,Patuwo, B.E., Hu, M. Y. (2001) "A simulation study of artificial neural networks for nonlinear time-series forecasting". *Computers and Operations Research* 28(4) pp. 381-396.

[23] Zhang, G.P. Qi, M. (2005) "Neural network forecasting for seasonal and trend time series". *European Journal of Operational Research*. 160(2) pp. 501-514.