ABSTRACT

THIELBAR, MELINDA F. Neural Networks for Time Series Forecasting: Practical Implications of Theoretical Results. (Under the direction of David A. Dickey.)

Research on using autoregressive neural networks to forecast nonlinear time series has produced mixed results. While neural networks have been established as universal approximators, large-scale studies comparing neural network forecasts with simpler models have rarely shown better performance for the neural network model. In the best cases, the neural network models prevail only after careful tuning.

We examine the simplest case of an autoregressive neural network, where the current value of y_t is dependent on a function of one lag with one shortcut connection and one hidden unit. We find that even when data are generated from the autoregressive neural network, the location of the series attraction point often leads to data that exhibit little nonlinear behavior. We use these results as a guide in extending traditional theory on nonlinear time series models. Our added theory is used to select parameter values for a simulation and to generate starting values for training a neural network. Performance for different methods of estimating forecasts are compared. We find that even for our relatively simple neural network, where we know the correct number of hidden units, estimating the parameters is a nontrivial task, and forecasts should be approached with caution.

The one-lag, one hidden unit model is then applied to a time series from an experiment in engineering. We find that the methods developed in this paper work well for this data and have promise in applications where measurements are taken often using a computerized setup.

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

by
Melinda F. Thielbar

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina

2011

APPROVED BY:

Peter Bloomfield	John Monahan
C.T. Kelley	David A. Dickey Chair of Advisory Committee

DEDICATION

To the loving memory of my parents, Wayne and Mary Ann Thielbar.

BIOGRAPHY

Melinda Thielbar was born in a small town in central Missouri and received her undergraduate degree from the University of Missouri, Columbia. She holds a Master's degree in Economics, and her first research project was analyzing welfare-to-work programs using administrative databases maintained by the State of Missouri. She has worked as a teacher and a statistician in a variety of fields and is currently employed in the Advanced Analytics Lab at SAS Institute developing statistical methods for detecting fraud and financial crimes. She lives in Durham, North Carolina with her husband, video game writer Richard Dansky, and their three inevitable cats.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. David A. Dickey for his insight and support, as well as his patience and good humor throughout this process. I would also like to thank the members of my committee, Dr. Peter Bloomfield, Dr. John Monahan, and Dr. C.T. Kelley for their advice and comments at all phases of this project, as well as Amir Mosavi for providing data from his experiments.

This research is partly supported by a grant from the International Institute of Forecasters and SAS Institute.

TABLE OF CONTENTS

List of Tables . . . . . **vii**

List of Figures . . . . . **ix**

Chapter 1 Introduction **1**
1.1 Statement of the Problem 2
1.2 Thesis Organization 5

Chapter 2 Review of the Literature **6**
2.1 Basic Definitions 6
2.2 Threshold Models 7
2.3 Stationarity and Ergodicity of Threshold Models 7
2.4 Network Architecture 10
2.5 Estimation Difficulties 11
2.6 AR-NN Performance in Large-Scale Case Studies 13
2.7 Conclusion 15

Chapter 3 Model Estimation Problems **16**
3.1 Process Skeleton and Attraction Point 17
3.2 "Poorly-Behaved" Model Hessian 25
3.3 "Practical Reducibility" 30

Chapter 4 The AR(1) Model vs a "Perfect" Autoregressive Neural Network . **32**
4.1 The "Perfect" Model 32
4.2 Examining $\hat{\delta}_k$ in Terms of Parameter Values 33
4.3 Relationship Between λ and ρ 42
4.4 Simulation Reduction 44
4.5 Relationship Between Convergence and $\hat{\delta}_k$ 50
4.6 The Importance of Noise 51
4.7 Conclusions 54

Chapter 5 Forcasting With Neural Networks **55**
5.1 Forecasting with Autoregressive Neural Networks 55
5.2 Prediction At Extremes 63
5.3 Long Horizon Forecasts 64
5.4 Forecast Performance on Data Simulated with the Logistic Distribution 67
5.5 Conclusions 70

Chapter 6 Starting Value Selection **71**
6.1 Starting Value Estimation and Results 72
6.2 Convergence vs Performance 76
6.3 Conclusions 79

6.4 Recommendations for Starting Values 80

Chapter 7 An Analysis of Damage-Sensor Data 81
7.1 Automated Bridge Monitoring 81
7.2 Conclusions 88

Chapter 8 Conclusions and Suggestions for Future Research 89
8.1 Suggestions for Future Research 91

References 93

LIST OF TABLES

Table 3.1 Each parameter combination can have up to 3 equilibrium points. 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. 17

Table 3.2 Convergence Statistics for Different Estimation Methods 28

Table 4.1 For models where $p > 0.05$ (i.e. where the AR(1) model's fit is comparable to the "perfect" model), λ tends to be positive. The other parameters appear to be equally spaced along the ranges specified by the factorial design. 47

Table 4.2 For parameter combinations where $p < 0.05$ (i.e. where the AR(1) model fits the data poorly) λ tends to be negative. The other parameters appear to be equally spaced along the ranges specified by the factorial design. . . 48

Table 5.1 Although it is recommended in [17] that a bootstrap approach be used to stabilize AR-NN predictions, the point forecasts have better statistics overall for one-step-ahead forecasts. 57

Table 5.2 Although it is recommended that a bootstrap approach be used to stabilize AR-NN predictions, the point forecasts have better overall statistics and a narrower range. The bootstrap predictions have a potential for disastrous performance. 59

Table 5.3 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. 63

Table 5.4 The RW R-squares for the 12-step-ahead forecasts were much worse than the one-step forecasts. 64

Table 5.5 The RW R-squares for the 12-period bootstrap forecasts were negative in many cases, indicating that the random walk model was a better fit. The RW R-squares were slightly better for the 12-step bootstrap forecasts, though the bootstrap forecasts still had some cases where the performance was very bad (less than -10). 65

Table 5.6 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. 69

Table 5.7 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. 69

Table 6.1 When the mean is used as the starting value for the location parameters, approximately 70% of the runs result in convergence with reasonable parameter estimates. 73
Table 6.2 Although the noise term ε_t is normally distributed, the nonlinear nature of the deterministic potion of the equation will often as not produce a series that is skewed or bi-modal. 74
Table 6.3 Of all the starting value routines that were tried, the 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. 75
Table 6.4 Grid searching for the starting values for ρ produced worse results than using a point value. This can be because the objective function is not smooth; trying more options at the beginning increases the chances that the estimation routine will find a local minimum rather than the global minimum of the objective function. 75
Table 6.5 Gauss-Newton did not perform as well as Levenberg-Marquardt, which is consistent with results from previous chapters. 76
Table 6.6 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 $N(0, 1)$. 77
Table 6.7 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. The Levenberg-Marquardt estimation method with the median as a starting value for the location parameters still performs best, and the same method with a grid search on ρ still performs worst. Gauss-Newton, however, is a clear contender when performance is measured by Mean-Squared Error. 78

Table 7.1 Summary fit statistics for the AR-NN on data from the first sensor. 83
Table 7.2 The confidence limit for the slope parameter and the weight of the hyperbolic tangent function(γ and λ and γ, respectively) are positive, and their confidence limits do not cross 0. 83
Table 7.3 The summary statistics for the AR(1) on data from the first sensor. . . . 85
Table 7.4 The estimate for the slope parameter of the AR(1) is near 1. 85

LIST OF FIGURES

Figure 3.1 There can be up to 3 solutions for equation (3.2). 18
Figure 3.2 For some series, there is only one solution to equation (3.2), and therefore there is only one equilibrium point for the process skeleton. If the equilibrium is also an attraction point, the process skeleton will be drawn to that point from any starting value. 19
Figure 3.3 Not all equilibrium points are created equal. In many cases, the process skeleton will be repelled away from the equilibrium point after it gets "close". Such series result in limit cycles rather than one steady state solution. 20
Figure 3.4 The data generated from an autoregressive neural network is partially determined by the properties of the equilibrium point(s). If there are multiple attraction points, we can expect the data to cluster around these attraction points. In the figure above, the data are clustered around the two attractors. 21
Figure 3.5 A series with a unique, stable attraction point will have most of its data clustered around the attraction point. 22
Figure 3.6 A series with an unstable equilibrium point (an equilibrium that is not an attraction point) will still display nonlinear behavior, though data will be clustered near points in the limit cycle instead of equilibrium points. . 23
Figure 3.7 If the series attraction point is near the flat portion of the hyperbolic tangent function, the series data will be clustered near that point, and therefore it will be difficult to estimate the slope or location parameter from the generated data. 23
Figure 3.8 It is difficult to tell which sort of attractor we are dealing with based on the generated data. 24
Figure 3.9 Partial Derivatives $\gamma^* = 8$ and $\lambda^* = 1$. The derivative of the model function wrt γ^* and the derivative wrt λ^* are 0 for values of y_{t-1} that are farther than 1 unit away from c^*. 29
Figure 3.10 Partial Derivatives $\gamma^* = 1$ and $\lambda^* = 1$. When γ^* is small in absolute value, the derivatives are less extreme, and so there is a wider range of $y_{t-1} - c^*$ where the derivatives are nonzero. 29

Figure 4.1 There is a striking contrast between the comparison of model fit for $\lambda > 0$ and $\lambda < 0$. 35
Figure 4.2 For scaling purposes, this figure shows the $\hat{\delta}_k$ for data generated where $\lambda > 0$.Notice when the magnitude of λ is small, larger values for $\hat{\delta}_k$ are associated with smaller values of $|\alpha_0 - c|$, i.e. when the weight on the activation function is small in absolute value and positive, the location parameters of the activation function and the shortcut connection should be close together to produce a series with nonlinear features. 36

Figure 4.3 For scaling purposes, this figure shows the values of $\hat{\delta}_k$ for data generated where $\lambda < 0$. 37
Figure 4.4 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. 38
Figure 4.5 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 a boundary attractor and the hyperbolic tangent function adds little more than a constant. 39
Figure 4.6 The optimum value for $|\alpha_0 - c|$ depends on the magnitude of λ When $|\lambda|$ is large, location parameters that are close together generate series where the data cluster at opposite sides of the hyperbolic tangent function, with little information about the location parameter. 40
Figure 4.7 The optimum value for $|\alpha_0 - c|$ depends on the magnitude of λ. When $|\lambda|$ is small, location parameters that are close together generate series where the data follow the hyperbolic tangent function smoothly. Notice, however, that because $|\lambda|$ is smaller, these data do not cluster around the model function as tightly, producing a noisy fit around the function that generates the data. 41
Figure 4.8 When the slope of the hyperbolic tangent function and the shortcut connections have the same sign, the underlying model function is a weighted average of the shortcut connection and the activation function. As ρ increases, the underlying function is closer to a straight line, and therefore the generated data follow a straight line. 42
Figure 4.9 When the slope of the hyperbolic tangent function and the shortcut connections have opposite signs, the underlying model function is no longer a weighted average of the shortcut connection and the activation function. The value of $\hat{\delta}_k$ is less dependent on the value of ρ. 43
Figure 4.10 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. 44
Figure 4.11 Higher cutoffs for the p-value were considered. However parameter combinations that produced p-values between 0.1 and 0.3 were cases where neither the AR(1) nor the AR-NN fit the data well. 45
Figure 4.12 Higher cutoffs for the p-value were considered. However parameter combinations that produced p-values between 0.1 and 0.3 were cases where neither the AR(1) nor the AR-NN fit the data well. 46
Figure 4.13 Not surprisingly, the parameter combinations selected for the remaining simulations tend to have negative values for λ. 49
Figure 4.14 For the parameter combinations that produced small p-values, there is a relationship between λ and $|\alpha_0 - c|$. 49

Figure 4.15 The proportion of models converging without errors increases sharply as $\hat{\delta}_k$ increases. 50
Figure 4.16 Binning the observations shows a much sharper increase in the proportion of convergent models as $\hat{\delta}_k$ increases. 51
Figure 4.17 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. . . . 52
Figure 4.18 When the p-value from the logistic simulation was larger than 0.05, the relationship between y_t and y_{t-1} showed very little nonlinear behavior. . 53

Figure 5.1 Rw R-squared values for the AR(1) model. 58
Figure 5.2 The relationship between y_t and y_{t-1} for the parameter combinations where there is the most difference between the linear forecast and the AR-NN forecast is where the relationship is the most obviously nonlinear. 60
Figure 5.3 The AR(1) forecast 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 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. 61
Figure 5.4 There are parameter combinations where most of the variability in the series can be explained by an AR(1). These were cases where λ was small in absolute value. In general, values for λ that were large in absolute value and negative generated series where an AR(1) fit poorly and the AR-NN model produced high RW R-squareds. 62
Figure 5.5 The AR-NN forecast was near the actual series for the first few steps but then often started moving in the opposite direction. RW R-squares for these models were often very negative, below -1. 66
Figure 5.6 For this model, the parameter estimates were almost exactly equal to the true parameter values, yet the RW R-squares were very low, less than -1. This is because the series oscillates between extremes, and the noise term can cause the series to become out of phase with the forecast, so the forecast is moving in the opposite direction of the actual series. 67
Figure 5.7 For this model, the long-horizon forecast often becomes out of phase with the series. This leads to very poor measures of forecast performance by RW R-squared. 68

Figure 7.1 After the initial shock when the hydraulic cylinder is applied, the time series for the first sensor settles to a steady state with definite nonlinear features. 82
Figure 7.2 The point estimates for the autoregressive neural network follow the extremes in the holdout sample better than the AR(1). This is reflected by the difference in the out-of-sample RW R-squared statistics: 0.6140 for the AR(1) and 0.63796 for the AR-NN 84
Figure 7.3 The AR-NN follows the series quite well, despite extreme changes in the series values. 85

Figure 7.4 The point estimates for the autoregressive neural network follow the extremes in the holdout sample better than the AR(1). This is reflected by the difference in the out-of-sample RW R-squared statistics: 0.6140 for the AR(1) and 0.63796 for the AR-NN 86
Figure 7.5 At the beginning of the series, the AR-NN does a particularly good job of predicting the series. The early features of the data (before the steady state is reached) seem to be important for the AR-NN to perform better than the AR(1). 87

Chapter 1

Introduction

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 [8], their flexibility can lead to estimation problems. When Faraway and Chatfield [4] 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 holdout 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 [2].

A comparison of linear methods, smooth transition autoregressive methods, and autoregresive neural networks performed in [17] may shed some light on neural network estimation problems and general poor performance. In [17], Terasvirta *et al* (2005) 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 [6],[15], in an empirical study on simulated series with seasonality and trends Zhang and Qi [24] showed that the neural network performed much better after the series was adjusted for trends and seasonality. As Faraway and Chatfield [4] 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 [19] Trappletti, Leisch, and Hornik showed that an autoregressive neural network is stationary and ergodic (under certain sufficient but not necessary regularity conditions). In [11] Leoni defined sufficient conditions whereby the

skeleton of an autoregressive neural network approaches a unique attraction point.

We propose to build on the results in [19] and [11] 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 focus on simulated data and limit ourselves to an autoregressive neural network model with one lag, and one hidden unit, where the noise term is distributed $N(0,1)$.

1.1 Statement of the Problem

Consider the model:

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \lambda \tanh(\gamma(Y_{t-1} - c)) + \varepsilon_t \tag{1.1}$$

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

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 ($\alpha_1 Y_{t-1}$). The tanh(.) function in the above model is known as the *activation function*, and while the hyperbolic tangent function is convenient for our purposes, there are many activation functions to choose from, including the logistic function, the Gaussian radial basis function, and the inverse tangent.

Trapletti *et al* [19] shows that a neural network with additive noise is strictly stationary and ergodic if 1) the activation function is bounded, 2), the density function of the noise term (ε_t) is absolutely continuous with respect to Lebesgue measure and positive everywhere in $\mathbb{R}$, and 3) the characteristic roots of the linear portion ($Y_t - \alpha_0 - \alpha_1 Y_{t-1}$), i.e. $(1 - \alpha_1 B)(Y_t - \frac{\alpha_0}{1-\alpha_1})$ are not equal to 1. If we choose α_1 such that $|\alpha_1| < 1$, our neural network satisfies these conditions. It therefore has a mean, which we will call μ_Y.

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.

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

Notice if $\lambda = 0$, or if $E[\tanh(\gamma(Y_{t-1} - c))] = 0$, then equation (1.2) can be simplified to $\mu_y = \frac{\alpha_0}{1-\rho}$, which is the mean for an AR(1).

The expectation of the hyperbolic tangent function is intractable analytically, but we do know some facts about it.

First, notice that the hyperbolic tangent function is equal to 0 at $Y_{t-1} = c$ and that for any constant d, $\tanh(\gamma(Y_{t-1} - d - c)) = -\tanh(\gamma(Y_{t-1} + d - c))$. Therefore, if $c = \mu_Y$, then $E[\tanh(\gamma(Y_{t-1} - c))] = 0$ (This is true for $\varepsilon_t \sim N(0, \sigma^2)$, and true for any symmetric noise function centered at 0).

Further, because $|\tanh(\gamma(Y_{t-1} - c))| < 1 \; \forall \; Y_{t-1}$, we know that $|E[\tanh(\gamma(Y_{t-1} - c))]| < 1$, and as $c - \mu_Y \to \infty$, $E[\tanh(\gamma(Y_{t-1} - c))] \to 1$ if $\gamma > 0$ and $E[\tanh(\gamma(Y_{t-1} - c))] \to -1$ if $\gamma < 0$.

Now, subtract the mean from both sides and put the series in terms of $Y'_{t-1} = Y_{t-1} - \mu_Y$ and $Y'_t = Y_t - \mu_Y$:

$$Y_t - \mu_Y = \alpha_0 - \mu_Y + \alpha_1 Y_{t-1} - \alpha_1 \mu_Y + \alpha_1 \mu_Y + \lambda \tanh(\gamma(Y_{t-1} - c)) + \varepsilon_t \tag{1.2}$$

$$\begin{aligned} Y'_t &= \alpha_0 - \mu_Y + \alpha_1 \mu_Y + \alpha_1 Y'_{t-1} + \lambda \tanh(\gamma(Y_{t-1} - c)) + \varepsilon_t \\ &= \alpha_0 - \mu_Y + \alpha_1 \mu_Y + \alpha_1 Y'_{t-1} + \lambda \tanh(\gamma(Y'_{t-1} + \mu_Y - c)) + \varepsilon_t \end{aligned}$$

let $c' = c - \mu_Y$ and $\alpha'_0 = \alpha_0 - (1 - \alpha_1)\mu_Y$ then:

$$Y'_t = \alpha'_0 + \alpha_1 Y'_{t-1} + \lambda \tanh(\gamma(Y'_{t-1} - c')) + \varepsilon_t \tag{1.3}$$

Taking expectations,

$$\begin{aligned} E[Y'_t] &= \alpha'_0 + \alpha_1 E[Y'_{t-1}] + \lambda E[\tanh(\gamma(Y'_{t-1} - c'))] \\ 0 &= \alpha'_0 + 0 + \lambda E[\tanh(\gamma(Y'_{t-1} - c'))] \\ -\frac{\alpha'_0}{\lambda} &= E[\tanh(\gamma(Y'_{t-1} - c'))] \end{aligned}$$

We therefore know that $\alpha'_0 \in (-\lambda, \lambda)$.

$$Y'_t = \alpha'_0 + \alpha_1 Y'_{t-1} + \lambda \tanh(\gamma(Y'_{t-1} - c')) + \varepsilon_t$$

Now we divide by the series standard deviation:

$$\begin{aligned} Y_t' &= \alpha_0' + \alpha_1 Y_{t-1}' + \lambda \tanh(\gamma(Y_{t-1}' - c')) + \varepsilon_t \\ \frac{Y_t'}{\sigma} &= \frac{\alpha_0'}{\sigma} + \alpha_1 \frac{Y_{t-1}'}{\sigma} + \frac{\lambda}{\sigma} \tanh(\gamma(\frac{Y_{t-1}'}{\sigma}\sigma - c')) + \frac{\varepsilon_t}{\sigma} \end{aligned}$$

Let $y_t = \frac{Y_t - \mu_Y}{\sigma}$ and $y_{t-1} = \frac{Y_{t-1} - \mu_Y}{\sigma}$, then:

$$y_t = \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*)) + e_t \tag{1.4}$$

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

The above parameterization puts 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. Note that if $\lambda = 0$, α_0^* is the same as an intercept for an AR(1) model.

There are 5 challenges in estimating the model parameters:

1. Because the hyperbolic tangent function is odd ($(-\lambda \tanh(\gamma(y_{t-1} - c)) = \lambda \tanh(-\gamma(y_{t-1} - c))$) even this very simple version of an autoregressive neural network is not identifiable.

2. The high number of parameters requires a great many observations in order to obtain stable parameter estimates.

3. The recursive nature of the autoregressive neural network requires us to consider how the series *skeleton* will behave. The series skeleton, as described in [18] is the series behavior when e_t is set identically equal to $0 \; \forall \; t$. We will show that even for series with reasonable-seeming parameter values and initial values y_0, the skeleton of the process can converge to a location where there is little information about the hyperbolic tangent function and very little systematic nonlinear behavior.

4. The relative flatness of the hyperbolic tangent function for all except a limited range of y_{t-1} causes numerical problems in the model Hessian.

5. The model is nonlinear, and therefore any training algorithm must include well-defined starting values.

The first challenge can be solved by placing bounds on either λ^* or γ^* such that $\lambda^* > 0$ or $\gamma^* > 0$. The second is partly due to the functional form of the autoregressive neural network, and therefore we must restrict application to problems where there is sufficient data to take advantage of the model's flexibility. The final three challenges are the focus of this research, in sections that are organized as follows:

1.2 Thesis Organization

We propose to add to the existing literature in the following ways. First, while there is much theory for nonlinear time series [18], for neural network modeling of nonlinear time series [11], and [19], and much empirical work on the efficacy of neural networks for time series models[17], [4], and [24], little has been done to bridge the two. A careful examination of the theory as it relates to producing consistent estimates is a valuable first step in developing best practices. Using results from [11] and [19], we develop an alternate parameterization of a neural network model and examine some of the inherent properties of autoregressive neural networks that can lead to poor estimates.

Second, while much has been done to prove that a neural network converges to a steady state [11] and [19], very little has been done to show that steady state is desirable or that the estimated parameters are predictive. We will show that it is possible to write down a model whose skeleton converges to an attraction point where the neural network parameters cannot be estimated from the data. Further, evaluation of the model Hessian for a time series neural network with one hidden unit will show that there is a very narrow range of parameter values for which we can be sure that an estimation technique will reach a global minimum. These results serve as a guide for selecting starting values in estimation (a detail that is often ignored in the literature). Finally, we examine the forecast performance of an estimated autoregressive neural network on data generated from a simple neural network. We discover that even when the data are generated from a neural network with known parameters, the autoregressive neural network is not guaranteed to out-perform an AR(1) estimated from the same data.

Chapter 2

Review of the Literature

2.1 Basic Definitions

We restrict ourselves to the case of discrete time models with additive noise term ε_t, that is series described by:

$$\mathbf{Y_t} = f(\mathbf{Y_{t-1}}, \theta) + \varepsilon_\mathbf{t}$$

In [18] conditions are set whereby a discrete nonlinear time series is stationary and ergodic. Recall that if a time series Y_t is strictly stationary, then:

$$F(Y_{t_1}, ..., Y_{t_n}) = F(Y_{t_1+k}, ..., Y_{t_n+k}) \tag{2.1}$$

i.e., the probability function does not depend on time, but on the separation in time of the observations.

We say a series is weakly stationary if:

$$E[Y_{t_1}] = E[Y_{t_2}] = ... = E[Y_{t_k}] \text{ and } Cov(Y_{t_1}, Y_{t_2}) = Cov(Y_{t_1+k}, Y_{t_2+k}) \tag{2.2}$$

for all t_1, t_2 and $k \in \mathbf{Z}$[5]

We say that a time series is *ergodic in terms of the mean* if the mean squared error of the sample mean as an estimator of the population mean approaches 0 as $T \to \infty$, i.e. if observing an infinitely-long single realization of the series is equivalent to observing an infinite number of realizations of the series, for the purpose of estimating the mean [5].

2.2 Threshold Models

A threshold autoregressive model (TAR model) is a nonlinear time series model defined as:

$$Y_t = \alpha_0 + \pi \mathbf{Y_{t-p}} + \sum_{k=1}^{K} F(\theta_k, \mathbf{Y_{t-p}}) + \varepsilon_t \tag{2.3}$$

where $\mathbf{Y}_{t-p}$ is a vector of lagged values of Y_t and F is a bounded function in $\mathbf{Y}_{t-p}$.

There are many different types of threshold models, differentiated by the choice of F. For example, F may be a step function, the CDF of a Gaussian function, a logistic function, or a hyperbolic tangent function. The function F, then, allows us to specify two different intercepts: one where Y_{t-p} is higher than a specified level, and a different intercept when Y_{t-p} is below a specified level. Between these critical values, the series will display nonlinear properties if the transition function is smooth (as with the logistic, hyperbolic tangent, and Gaussian cdf functions), or it will have a discontinuity in the case where F is a step function. If $K > 1$, then we say the model has multiple regimes, i.e. more than one change in intercept. In cases where F is a smooth bounded function, we refer to the model as a Smooth Transition Autoregressive or STAR model. The STAR model with additive noise is equivalent to an auto-regressive neural network with a bounded and smooth activation function. There is also the LSTAR (Logistic Smooth Transition Autoregressive) model, a STAR model where the activation function is a logistic function.

Equation (2.3) is also the form for an autoregressive neural network, or AR-NN model, as defined in [19]. In [8] it is shown that a neural network is a universal approximator for any Borel-measurable function as $K \to \infty$, and in [20], White shows that is is possible to obtain consistent and asymptotically efficient estimates for the parameters of a neural network.

Because of these properties, Gorr [6] states that neural networks have great potential as predictors of nonlinear time series, including automatic detection of seasonality, trend breaks, and nonlinear relationships among the lags. Gorr [6] also cautions, however, that neural networks are complicated models and subject to all the problems forecasters have come to expect from complicated models, including overfitting and poor estimation due to the presence of local minima in objective functions. Gorr [6] also points out that neural network models offer little insight into the underlying structure of the data and are primarily useful in forecasts, and not as explanatory models.

2.3 Stationarity and Ergodicity of Threshold Models

In [18], conditions are set whereby a nonlinear time series with additive noise is stationary and ergodic. This is done using Markov chain theory.

A Markov chain is defined as a series Y_t where:

$$P(Y_{t+1} = y | Y_1 = y_1, Y_2 = y_2, ..., Y_t = y_t) = P(Y_{t+1} = y | Y_t = y_t) \tag{2.4}$$

i.e. the probability distribution of the next observation depends only on the value of the current observation. This can be extended to consider Y_{t-1} as a vector of lags of Y_t. The Markov chain is defined in terms of the probability measure of Y_t (usually denoted as μ_t), which is allowed to vary with time.

In the case of a Markov chain, the term *ergodic* refers to the convergence of the probability distribution. We say that a Markov Chain is ergodic if there exists a probability distribution π such that as $t \to \infty$, $\mu_t \to \pi$, regardless of the starting probability measure μ_0. We say that the Markov chain is *geometrically ergodic* if the convergence happens geometrically fast, i.e. if $\exists\ p \in (0,1)$ st $p^{-t}|\mu_t - \pi| \to 0$ as $t \to \infty$.

The distribution π is called the stationary distribution, and if π exists, the Markov chain is called stationary. If the Markov chain is stationary, we are able to assume that observing a chain as $t \to \infty$ is equivalent to observing the Markov chain for multiple starting probability distributions. Theorems describing the stationarity of Markov chains are called ergodic theorems.

In [18], a discrete time nonlinear time series with additive noise is treated as a Markov chain. Consider a time series in discrete time of the form:

$$\mathbf{y}_t = f(\mathbf{y}_{t-1}, \theta) + \varepsilon_t \tag{2.5}$$

Thus the chain is divided into a deterministic portion (f) and a stochastic portion (ε). The function f can also be called the *skeleton* of the time series, or the function describing the time series behavior when ε_t is set to 0 for all t. Both the inputs and the outputs can be vectors or single values, and the same is true for the noise term ε_t.

In [18], it is shown that a Markov chain described by 2.5 is geometrically ergodic if the following three conditions hold.

1. The $\mathbf{0}$ set is an equilibrium for f (i.e. $f(\mathbf{0}, \theta) = \mathbf{0}$) and it is *exponentially asymptotically stable in the large*, i.e. $\exists\ K$ and $c > 0$ such that if $\mathbf{y}_0$ is the starting value for the series, $\|\mathbf{y}_t\| \leq Ke^{-ct}\|\mathbf{y}_0\|$.

2. Either the noise term ε_t is *iid*, with a marginal distribution function that is absolutely continuous and has an everywhere positive probability density function with $E\|\varepsilon_t\| < \infty$. Or $\varepsilon_t = (e_t, 0, ..., 0)$, with e_t *iid*, each having a marginal distribution function that is absolutely continuous with an everywhere positive probability density function with $E\|e_t\| < \infty$.

3. $f(\mathbf{y}_{t-1}, \theta)$ is Lipschitz continuous.

These conditions are sufficient, but not necessary.

Condition 1 assures that the series is stable at and attracted to the origin. If the series were allowed to run without noise, $\mathbf{y}_t$ would eventually arrive at the 0 set and stay there. This is necessary to ensure that a suitable Lyapunov function can be found for f (For a full explanation of Lyapunov functions, and their relationship to autoregressive neural networks see [14] and [18]). Condition 2 assures that $\mathbf{y}_t$ will be allowed to visit all relevant portions of the space and that the noise term will not carry $\mathbf{y}_t$ so far from the origin that it cannot come back in a finite number of steps. Condition 3 assures that the deterministic portion of the series cannot send $\mathbf{y}_t$ so far away from the origin that it cannot return in a finite number of steps.

The fact that $\mathbf{0}$ must be an equilibrium set for f seems restrictive, but in the case where $\mathbf{y}_t$ is a scalar, as long as f has a finite equilibrium y^* that meets the second half of condition 1 (exponential asymptotic stability in the large), one can re-define f as $f - y^*$, and the condition becomes: $\exists\ K$ and $c > 0$ such that if y_0 is the starting value for the series, $|y_t - y^*| \leq Ke^{-ct}|y^* - y_0|$.

Tong [18] relates the need for stability at the origin to the well-known stationarity results for linear series. Consider:

$$Y_t = \alpha Y_{t-1} + \varepsilon_t$$

where $\varepsilon_t \sim iid(0, \sigma)$

We know that if $|\alpha| < 1$, the process is *stationary*, i.e. its probability function is the same regardless of the value of t. Tong [18] points out that the stationarity of the process depends on the deterministic portion's (αY_{t-1}) attraction to the origin. If $|\alpha| < 1$, then even when $|Y_{t-1}|$ is large, the pull towards 0 outweighs the noise term. When $|Y_{t-1}|$ is small, αY_{t-1} is also relatively small. It is therefore largely noise that moves the system away from the origin and largely the deterministic portion of the model that moves the series back toward the origin.

While conditions 1-3 are sufficient for stationarity, they are not necessary [19]. Condition 1, in particular (that the series not only have an equilibrium point, but that the equilibrium point be exponentially asymptotically stable in the large) is difficult to prove and impossible to show without additional assumptions about the form of f and the values of θ [11].

In [19] Markov chain theory is again used, this time applied to the case where f is a neural network of the following form:

$$y_t = f(\mathbf{y}_{t-1}, \theta) = \alpha_0 + \boldsymbol{\alpha}\mathbf{y}_{t-1} + \sum_{j=1}^{J} \lambda_j g(\mathbf{y}_{t-1}, \gamma_j) + \varepsilon_t \tag{2.6}$$

where $g(\mathbf{y}_{t-1}, \gamma_j)$ is a bounded and continuous activation function.

It is then shown that the following conditions are sufficient to show that (2.6) describes an irreducible and aperiodic Markov chain that is geometrically ergodic.

1. The activation function g is absolutely continuous and bounded
2. The shortcut connections $\boldsymbol{\alpha}\mathbf{y}_{t-1}$ do not contain a unit root,
3. ε_t is iid, with an absolutely continuous distribution function that is positive everywhere (same as Condition 2 from [18]).

From stationarity, it is then shown that the existence of moments for the series described by (2.6) is dependent on the existence of moments for ε_t. Moreover, if the second moment of ε_t exists (the first is already assumed to exist), and if the neural network contains no redundant hidden units, it is shown in [19] that it is possible to generate consistent and asymptotically efficient estimates for the series parameters.

The proof of stationarity in [19] is derived directly from the stationarity proof in [18], and while it is easy to show that an equilibrium point exists [11], it is difficult to prove that it is stable in the sense required by the proof in [18]. The existence and stability of equilibrium points and how they relate to the stationarity and ergodicity of a neural network is part of the subject of this thesis.

2.4 Network Architecture

In [8] Hornik, Stinchcombe, and White show that a single-layer neural network with a sufficient number of hidden units is able to approximate any nonlinear function to an arbitrary degree of accuracy. They show that this neural network representation exists and use this result to assert that a neural network's failure to approximate a function accurately must be due to inappropriate architecture (i.e. an incorrect number of hidden units), "inadequate learning" (i.e. estimation difficulties), or the lack of a systematic relationship between the predictor and the output variable.

The proof in [8] guarantees the existence of a universal approximator, and White [20] shows that the parameters of a neural network are the solutions to an optimization problem that minimizes an objective function based on the difference between the values predicted by the model and the actual values of Y_t. Neither result, however, guarantees that such an approximator can be found, nor does it guarantee that the parameters of the appropriate neural network are identified (i.e. whether a training algorithm will be able to distinguish one set of parameters from another). It also does not guarantee that the objective function of the neural network will be smooth and free of local minima.

In fact, Hush [9] shows that training a neural network with one hidden unit and a sigmoidal function is NP-Hard, meaning that determining the correct values of the parameters may (worst-case) require an exhaustive search of the entire parameter space.

It is also possible to write down equivalent versions of the same neural network model. Consider the following:

$$y_t = \alpha_0 + \alpha y_t - \lambda \tanh(\gamma_1 y_{t-1} + c)$$

This is a neural network with a single hidden unit and a sigmoidal activation function. Because of the sigmoidal shape of the hyperbolic tangent function. The equation above is equivalent to:

$$y_t = \alpha_0 + \alpha y_t + \lambda \tanh(-\gamma_1 y_{t-1} - c)$$

An estimation routine would not be able to distinguish between these two equivalent sets of parameter estimates.

It is also important to remember that the AR-NN structure is only an approximation of an underlying nonlinear relationship that is by definition unknown. Though we may use parameter estimation techniques and hidden unit selection algorithms that are driven by performance in predicting y_t, there is no way to know whether poor prediction performance is due to model specification, a failure to estimate the parameters (due to the objective function arriving at a local rather than global minimum), or the lack of a true nonlinear relationship between the lags and the outcome.

2.5 Estimation Difficulties

Faraway and Chatfield [4] use the airline data from Box *et al* [1] as a case study in forecasting times series with a neural network. They find that the forecasts from a neural network are sensitive to the number of hidden units selected and to the starting values chosen for the model. Many of their models fail to converge, and some that do converge produce forecasts that are clearly wrong [4]. Further, they find that transformations such as natural logs, dividing the entire series by a constant (100 in their case), or differencing the series to remove trend and seasonality produced much better estimates, though they show no theoretical justification for these methods. Further, the neural network forecasting literature of the time suggested that transformations were unnecessary [6].

Faraway and Chatfield conclude that forecasting with neural networks requires detailed knowledge of both time series forecasting and neural network modeling [4]. What is most interesting about Faraway and Chatfield's analysis, however, is their attempt to "open the black

box" of a neural network forecast and analyze the predictions created by the neural network model. Using a generalized additive model (GAM), they are able to plot the relationship between the chosen lags and the predicted value in different directions. They find that the neural network model they estimate for the airline data is linear in most directions, especially where the data is most dense. They conclude from this that the neural network is essentially a linear model with extra parameters and that, for the airline data at least, a neural network is not an appropriate forecasting model.

A neural network is a form of nonlinear model, and so Faraway and Chatfield's findings in [4] are not surprising in light of what we know about estimating nonlienar models. Unless the objective function is convex in all directions, it is quite possible for the estimator to become "stuck" at a local minimum rather than a global minimum. While trying different combinations of starting values can help alleviate this problem, there is no way to guarantee that the reported parameter estimates are the global minimum without searching the entire parameter space [9].

In their paper on the stationarity and ergodicity of an AR-NN, Trappletti *et al* [19] report that "in practice, the Hessian tends to be poorly behaved." The Hessian is the matrix of second derivatives of the objective function, and a singular or poorly-behaved Hessian often implies a linear dependency among the parameters. For the neural networks estimated in Faraway and Chatfield [4], where the neural network seems to be a straight line in many directions, the parameters of the transition function would show linear dependencies with the parameters of the shortcut connections (the linear portion of the neural network model).

In [24] Zhang and Qi investigate the claims that neural networks are capable of automatically producing models that adjust for seasonal variation and trends. In [24], a monthly time series is simulated with an intercept, a time trend, and a seasonal effect for each month. They then simulate 228 time periods (19 years of data) and use the last 12 periods as a holdout sample. The remaining time periods are used as a training and test set to estimate model parameters, as is standard practice in forecasting [24]. Forecasts generated from the original series are then compared to forecasts generated from pre-processed data that has been a) de-trended, b) de-seasoned, and c) both de-trended and de-seasoned.

The pre-processed data produce neural network models that out-perform models produced with the original data by orders of magnitude, with mean average percentage errors (MAPE) of 0.49 on the test data for pre-processed data vs 10.98 on raw data, with similar comparisons for other measures of model performance such as root MSE [24]. When compared to a standard ARIMA model, the autoregressive neural network showed superior out-of-sample performance in all cases. From this, Zhang and Qi conclude that when the data contain the specified seasonal effect, pre-processing improves the neural network's performance.

Neural networks were then compared to ARIMA models on actual data from ten time series with established trends and seasonal aspects. As with the simulated data, the AR-NN

performed poorly on the raw series, but well once the data had been de-trended and de-seasoned. The neural network also performed better than a traditional ARIMA model with a trend and seasonal effect that has been estimated on the original data. Zhang and Qi [24] conclude that AR-NN can produce better forecasts than an ARIMA model, but that they are worse at modeling trend and seasonality. They also point out that the lag structures and numbers of hidden units chosen by the modeling process are very different for different series that should have similar features, giving further evidence for known problems in specifying a neural network model on time series data [24].

2.6 AR-NN Performance in Large-Scale Case Studies

Large-scale experiments where AR-NN models are compared against linear models have been popular in the literature. In [16] AR-NN models are estimated for 9 quarterly econometric time series over a period of 30 years (1960-1993). The goal in [16] is to study the efficacy of neural network forecasting for real-time analysis, i.e. as a decision tool when decisions must be made for the next period using only data available up through the current period. To that end, the models they study are carried out on unadjusted and unrevised data on economic indicators such as unemployment and GDP.

The models are estimated using data from a rolling window of fixed size (40, 68, and 76 quarters). Model performance is compared for 1) re-estimating parameters but keeping the network architecture fixed, and 2) re-specifying the architecture and the parameters at each time period. A hypothesis test of whether there is a statistically significant difference in the root MSE of the overall out-of-sample forecasts is used to determine if one model performed significantly better than the other.

Overall, the AR-NN models perform well in [16]. In 5 of the 9 series studied, they performed significantly better than linear models and were worse for only one series. In the other cases, the difference in root MSE was not considered statistically significant. The AR-NN did not perform as well when the forecast horizon was extended to 4 periods (one year because the data are quarterly), out-performing the linear models in only 3 of the 9 cases, though in all of the remaining 6 cases, the difference between the AR-NN and linear model was not statistically significant. This is a disappointing result since it is often believed that one of the strengths of nonlinear models is the ability to forecast at long horizons with more accuracy than linear models.

A large-scale study of AR-NN and LSTAR (Logistic Smooth Transition Autoregressive) model performance verses standard linear models is also considered in [17]. As in [16] the out-of-sample performance is considered for many different econometric time series. Model parameters are also estimated based on a rolling window of data and updated each period.

Unlike in [16], the models in [17] are based only on lags of y_t, and the structure of the models is only allowed to change once per year. Different techniques for determining the number of hidden units are also considered: a "bottom-up" approach where hidden units are added as long as the errors from the model show nonlinear properties according to a statistical test described in [10], and a "pruning" approach that uses Bayesian regularization to take a large model and shrink some of the parameter estimates toward 0.

In [17] long-term forecasts for AR-NN and LSTAR models are estimated using a bootstrap method instead of simple point forecasts. The prediction is defined as

$$\begin{aligned} \hat{y}_{t+1} &= E[f(y_t, ..., y_{t-p})] \\ \hat{y}_{t+2} &= E[f(y_{t+1}, ..., y_{t-p+1})] \\ &\quad ...\text{and so on} \end{aligned} \tag{2.7}$$

where the expectation is taken over the probability distribution of $\varepsilon_{t+1}, \varepsilon_{t+2}, ...$ The numerical integration required for the above expectations is onerous, even for simple probability distributions and short forecasts, so the bootstrap method is used where 500 different forecast horizons are estimated by adding noise from the assumed distribution for the noise term to the predictors. The different forecasts are then averaged at each time period to arrive at one projected forecast [17].

It is found that AR-NN performed poorly compared to linear models. The neural networks estimated using a bottom-up approach also produced "explosive" forecasts, where the parameters for the linear portion gave characteristic equations with roots near 1. The LSTAR models did slightly better, as did models estimated with Bayesian regularization. It is concluded that Bayesian regularization is a superior method for determining parameter values for AR-NN but that the AR-NN may not be able to out-perform simpler methods.

There are differences in methodology between [17] and that may explain some of the findings. First, the authors in [16] include exogenous variables in their model. It is possible that the nonlinear features among the lags are not prevalent enough to warrant estimation with a neural network model. Also, [16] allows model structure to vary at every period, while the structure of the models in [17] are only changed every year. The main advantage of a neural network model is flexibility, and limiting the model's ability to adapt could have a strong effect on outcomes. We must also keep in mind that, while a neural network is a universal approximator, it is an approximation to an unknown nonlinear structure. The structural changes shown in [16] could be the result of a true nonlinear structure that is imperfectly modeled by the estimated AR-NN.

2.7 Conclusion

While early results on autoregressive neural networks seem promising, later studies reveal that neural networks can produce forecasts that fail to out-perform more simple models. In some cases, a neural network will produce disastrous forecasts. All researchers agree that neural network models require careful tuning and post-processing. There are no broadly-recognized best practices for this tuning, however [2], and as our research will show, researcher choices such as starting values and what constitutes "convergence" for parameter values have non-trivial effects on the forecasts generated. In addition, even models that are carefully-trained may converge with a singular Hessian, indicating uncertainty and instability in the parameter estimates [19].

Theoretical results on time series models and long-run behavior of already-trained autoregressive neural networks may shed some light on why researchers frequently have problems diagnosing model behavior and estimation problems that "converge" according to automated model diagnostics. In the next chapter, we attempt to used data generated from an autoregressive neural network with known parameters and a known number of hidden units, using the true parameter values as starting values for the estimation. We discover that even under these ideal circumstances, because of the location of the series attraction point, it can be difficult to estimate parameters with any degree of certainty.

Chapter 3

Model Estimation Problems

Recall that our reduced and re-parameterized model is as follows:

$$y_t = \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*)) + e_t \tag{3.1}$$

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

We generate data from this model using a set of parameter combinations from a full factorial design where the parameter values are chosen as specified in Table 3.1.

More parsimonious designs were considered, but because of the nonlinear (and unknown) relationships between the series mean and the equilibrium points, sparse designs were deemed infeasible.

We generate $S = 10$ series of errors of length $T = 1000$ from a $N(0,1)$, with a leading burn-in period of 500 (1500 periods total). Because the errors are *iid*, we are able to use the same sequence of e_t for each parameter combination. The sequences, then, are blocks in the design of experiments sense. This allows us to reduce the extraneous noise in the simulation by ensuring that each parameter combination is tested using the same sequences of e_t.

Table 3.1: Each parameter combination can have up to 3 equilibrium points. 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.

Factorial Design for Model Parameters

	Low	High	Increament	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

3.1 Process Skeleton and Attraction Point

In [18] the skeleton of a nonlinear time series is defined as the series behavior when e_t are set identically equal to 0. According to [18], the skeleton of any nonlinear time series with additive noise approaches a single point or a unique limit cycle as long as the function describing the series behavior is continuous and the noise term has a pdf that is positive for the entire real line. Our autoregressive neural network meets these conditions and therefore approaches a unique steady state given the starting value and the values of the parameters.

Define an *attraction point* for the skeleton as a point y^* where

$$y^* = \alpha_0 + \rho y^* + \lambda^* \tanh(\gamma^*(y^* - c^*))$$

If the skeleton reaches an attraction point, it will remain at that value for all following points in the skeleton. We can simplify the above equation somewhat as:

$$\begin{aligned} y^* &= \alpha_0^* + \rho y^* + \lambda^* \tanh(\gamma^*(y^* - c^*)) \\ y^* &= \big(\alpha_0^* + \lambda^* \tanh(\gamma^*(y^* - c^*))\big)\big(1 - \rho\big)^{-1} \end{aligned} \tag{3.2}$$

Notice that the left-hand side of equation (3.2), viewed as a function of y^* is the identity and thus graphs as a 45-degree line through the origin. The right-hand side is a function that is bounded and nearly flat for most of its domain (see Figure 3.1 for an example). Both are defined for the entire real line. There is, therefore, at least one point where the two lines intersect, and

therefore there must be at least one y^*.

The potential for multiple attraction points is discussed in [19] and [11]. It is possible for our series to have up to three equilibrium points, depending on slope parameters ρ and γ^* and the location parameters α_0^* and c^*. Figure 3.1 shows the underlying functions for a series with three equilibrium points. If more than one equilibrium point is also an attraction point, the attraction point of the process skeleton will depend on the starting value y_0.

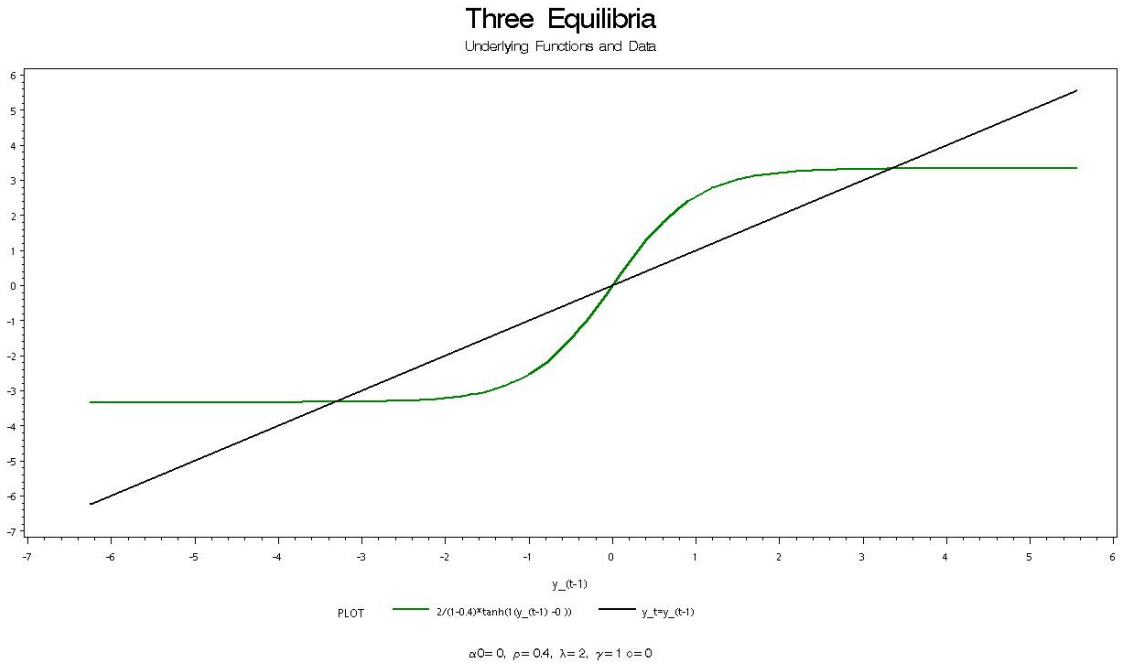


Figure 3.1: There can be up to 3 solutions for equation (3.2).

Equation (3.2), however, is only part of the story. Recall that the relationship between y_t and y_{t-1} is as follows:

$$y_t = \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*)) \tag{3.3}$$

which can also be written as:

$$y_t = \alpha_0^* + \rho y_{t-1} + (1-\rho)\frac{\lambda^*}{1-\rho} \tanh(\gamma^*(y_{t-1} - c^*)) \tag{3.4}$$

Each new y_t, then, is a weighted average between the previous y_t and the hyperbolic tangent function. We can think of equation (3.4) as a solution path to the point y^*.

Section 2.3 defines an equilibrium point as any solution y^* to equation (3.2). An equilibrium is an *attraction point* if there exists a neighborhood around y^* such that the skeleton of the process will move to y^* if y_0 is inside that neighborhood. An attractor is *globally and exponentially stable in the large* if the skeleton convergece to y^* for any $y_0 \in (\infty, -\infty)$, and the convergence is exponentially fast. Figure 3.2 shows the underlying functions and the convergence of the process skeleton for a series with one globally and exponentially stable attraction point.

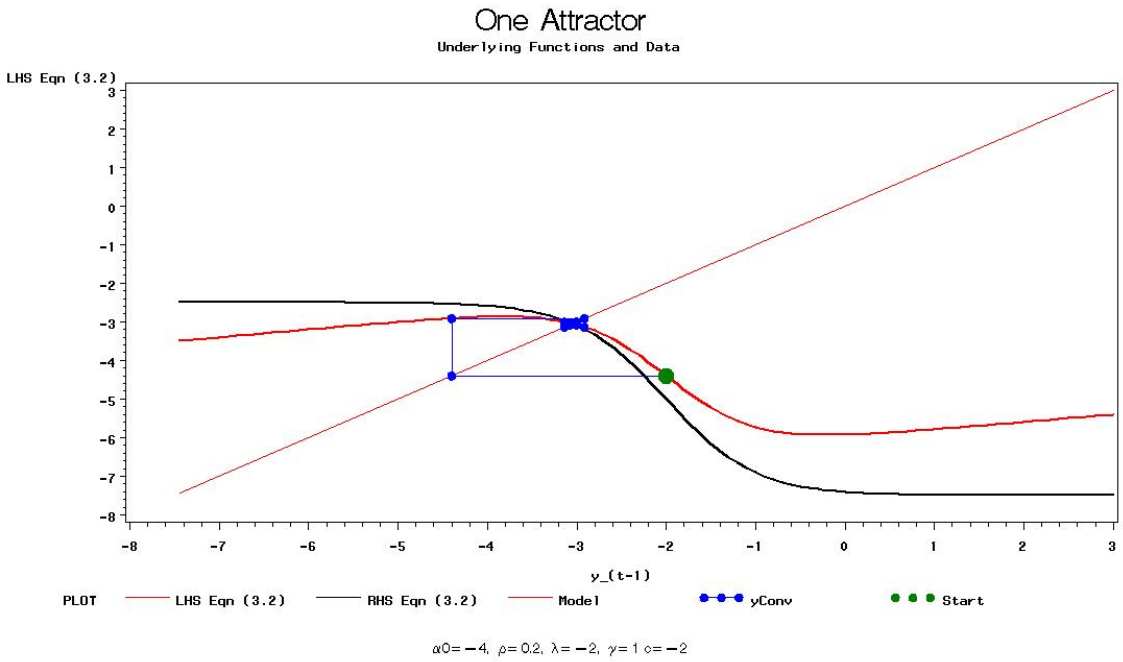


Figure 3.2: For some series, there is only one solution to equation (3.2), and therefore there is only one equilibrium point for the process skeleton. If the equilibrium is also an attraction point, the process skeleton will be drawn to that point from any starting value.

There is no guarantee that the series will converge to an equilibrium. In the case of an *unstable* equilibrium y^*, the skeleton will come close to y^* and then be repelled away (Figure 3.3). This results in a limit cycle where the skeleton moves away from the attraction point until it reaches the flat portion of the hyperbolic tangent function. Then it cycles back toward the intersection until it reaches a portion of the hyperbolic tangent that has sufficient slope to cause it to be repelled away. In this way, the skeleton revisits a finite number of points over and over.

The properties of the equilibrium point determines where most of the series data will fall, and it is therefore worthwhile to ask how the equilibrium point affects the generated data when

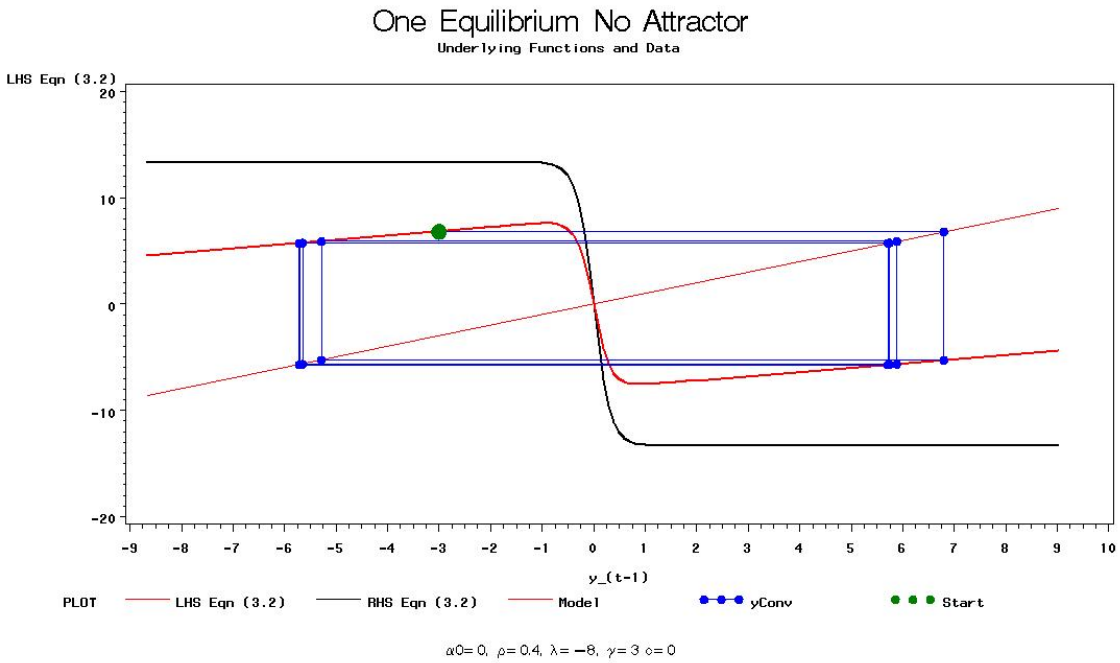


Figure 3.3: Not all equilibrium points are created equal. In many cases, the process skeleton will be repelled away from the equilibrium point after it gets "close". Such series result in limit cycles rather than one steady state solution.

we add e_t. Figures 3.4, 3.5, and 3.6 show scatter plots of generated data overlaid with the underlying functions and the initial sequence of y_t converging (or failing to converge) to the equilibrium point.

Whether the series converges to a limit cycle or one unique point, our main concern is whether the series converges to a location that allows us to estimate α_0^*, γ^*, λ^*, ρ and c^*. Unfortunately, convergence to a unique attractor does not guarantee a series with estimable parameters. It is quite possible (and indeed likely, as we will show below) to generate a series with an attraction point that is near the boundary of the hyperbolic tangent function. In these cases, the hyperbolic tangent function only adds a number that is close to $\pm\lambda$, acting as a constant in the functions generating the data. It is virtually impossible to estimate the location and the slope parameters for such a series. Figure 3.7 shows the relationship between equation (3.2) and the generated data for a series that displays this kind of behavior.

When plotted over time, data generated from the series illustrated in Figure 3.7 are nearly indistinguishable from data generated from the series shown in Figure 3.5. Figure 3.8 shows the series generated under both a stable center attractor and a boundary attractor. For each series, the shape over time is very much the same.

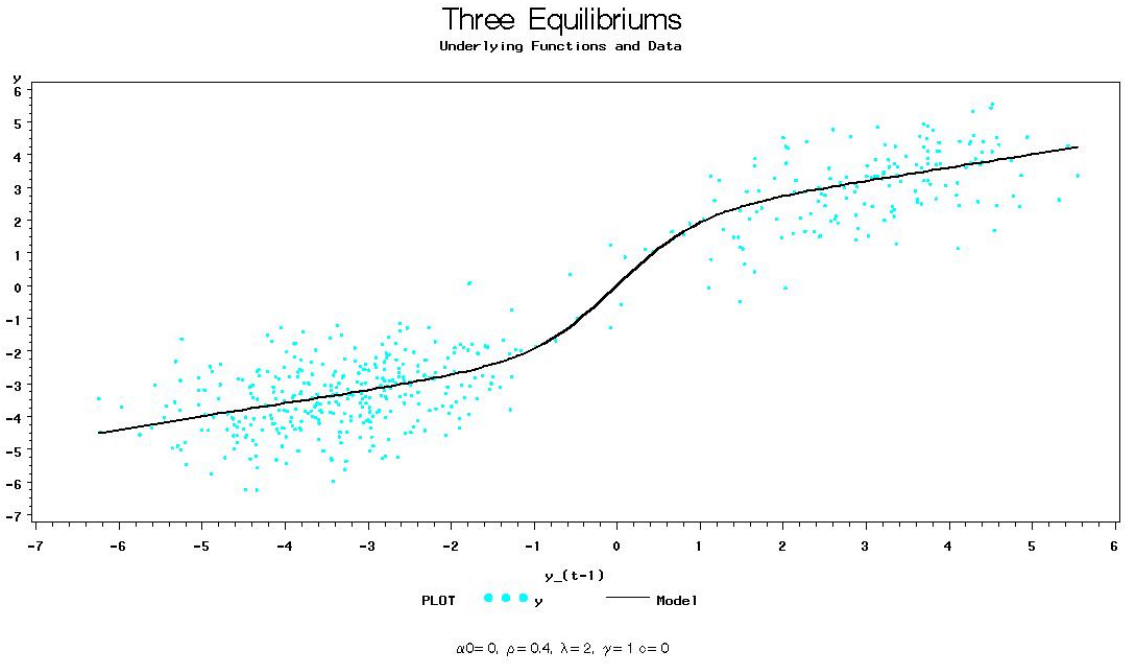


Figure 3.4: The data generated from an autoregressive neural network is partially determined by the properties of the equilibrium point(s). If there are multiple attraction points, we can expect the data to cluster around these attraction points. In the figure above, the data are clustered around the two attractors.

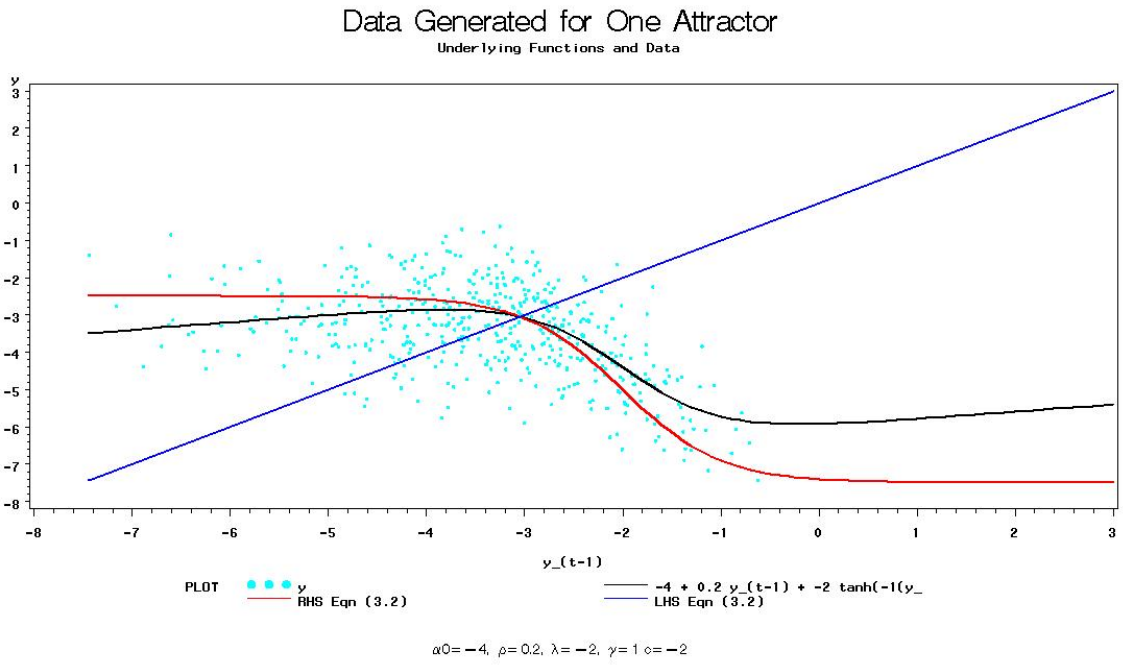


Figure 3.5: A series with a unique, stable attraction point will have most of its data clustered around the attraction point.

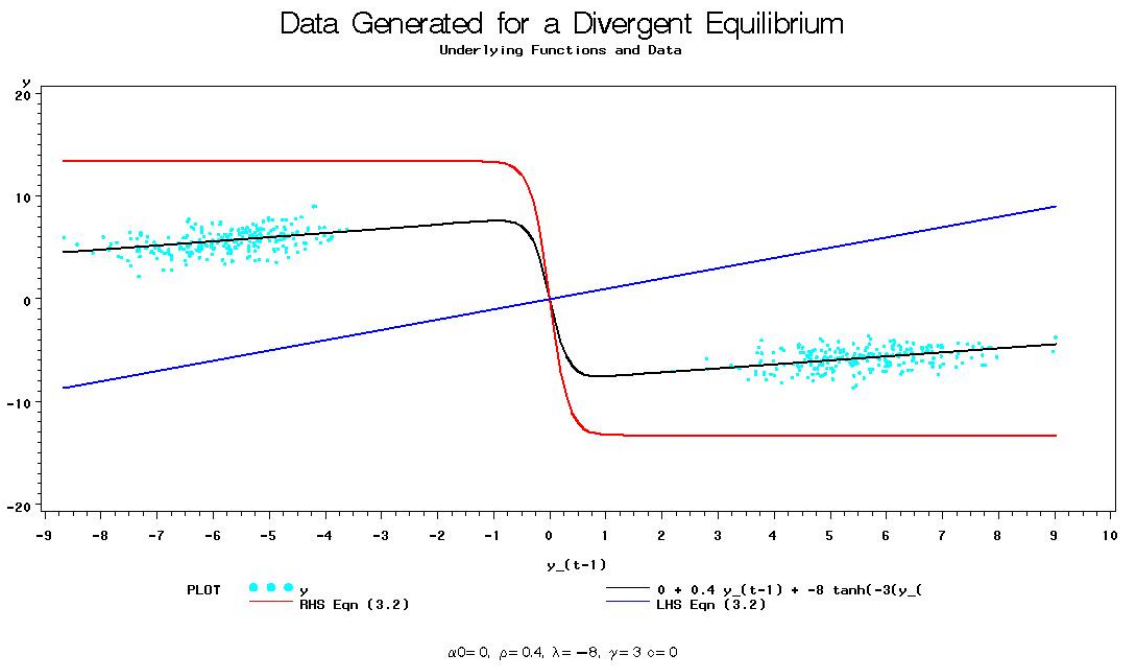


Figure 3.6: A series with an unstable equilibrium point (an equilibrium that is not an attraction point) will still display nonlinear behavior, though data will be clustered near points in the limit cycle instead of equilibrium points.

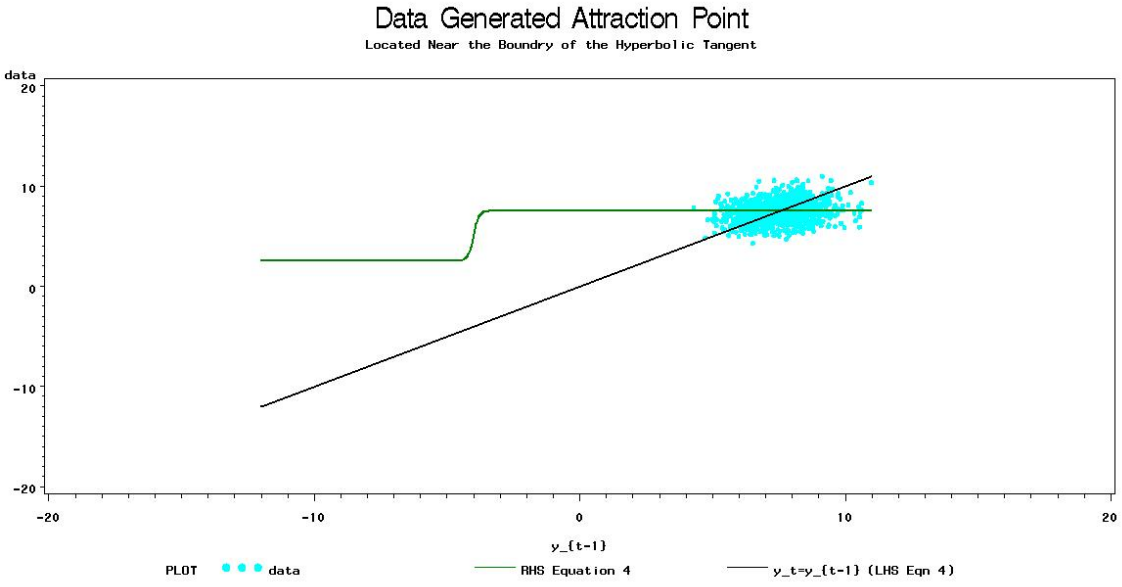


Figure 3.7: If the series attraction point is near the flat portion of the hyperbolic tangent function, the series data will be clustered near that point, and therefore it will be difficult to estimate the slope or location parameter from the generated data.

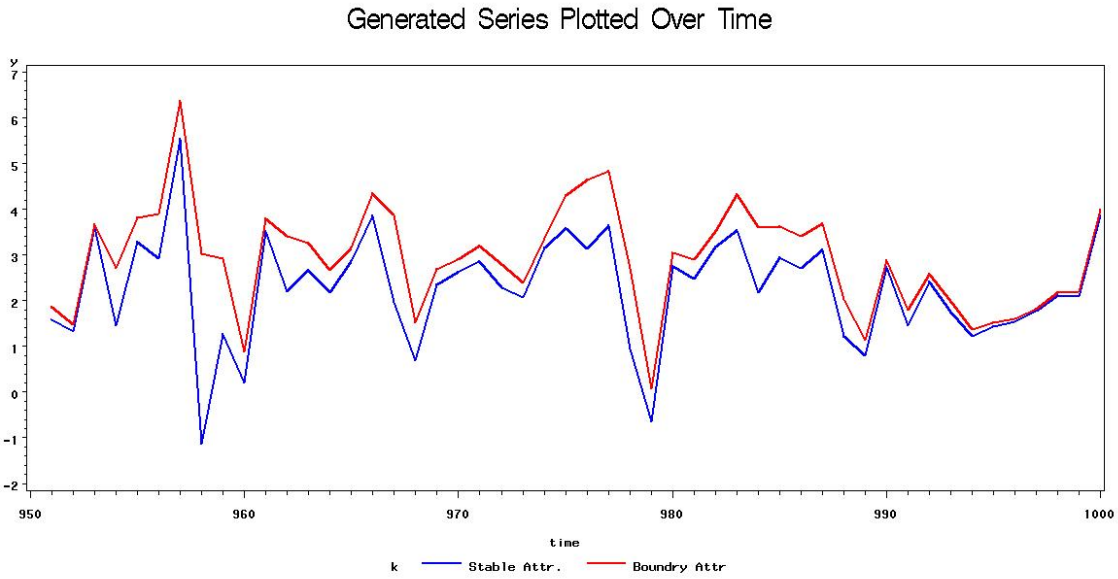


Figure 3.8: It is difficult to tell which sort of attractor we are dealing with based on the generated data.

The results above show that even when a time series is generated from an autoregressive neural network, the series may converge to an attractor located in a position where the generated data fails to display nonlinear properties. The next section describes the inherent difficulties in estimating an autoregressive neural network and how they are affected by the attractor (or lack thereof).

3.2 "Poorly-Behaved" Model Hessian

In [19], there is a mention that "in practice the Hessian tends to be poorly-behaved". In this section we examine the model Hessian and offer some explanations as to why this is the case.

Recall that our reduced and re-parameterized model is:

$$\begin{aligned} y_t &= \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*)) + e_t \\ e_t &\sim N(0,1) \end{aligned} \tag{3.5}$$

For $\theta = (\alpha_0^*, \rho, \lambda^*, \gamma^*, c^*)$, the gradient for this model is:

$$\begin{aligned} f_\theta(y_{t-1}, \theta) &= \big(1, y_{t-1}, \tanh(\gamma^*(y_{t-1} - c^*)), \lambda^*(y_{t-1} - c')\text{sech}^2(\gamma^*(y_{t-1} - c^*)), \\ &\quad -\lambda^*\gamma^*\text{sech}^2(\gamma^*(y_{t-1} - c^*)) \end{aligned}$$

(3.7)

We are training the model by minimizing the sum of squares error (SSE). For any parameter set θ, the error sum of squares is:

$$SSE = (Y - F(\theta))'(Y - F(\theta))$$

where $F(\theta)$ is the column vector: $f(y_{t-1}, \theta) = \alpha_0^* + \rho y_{t-1} + \lambda^* \tanh(\gamma^*(y_{t-1} - c^*))$, $t = 2, ..., T$.

Using the standard notation for optimization problems, we let $X(\theta)$ be the matrix of first partial derivatives evaluated at each y_{t-1} (so $X(\theta)$ is $p \times (T-1)$).

By Taylor Series, if θ^* is a local minimum, then,

$$\begin{aligned} \frac{\partial}{\partial\theta}(Y - F(\theta^*))'(Y - F(\theta^*)) &= \mathbf{0} \\ \text{i.e. } -2X(\theta^*)[Y - F(\theta^*)] &= \mathbf{0} \end{aligned}$$

where $X(\theta^*) = \frac{\partial}{\partial\theta}[y - F(\theta)]|_{\theta=\theta^*}$

Also, the second derivative of the objective function, the model Hessian, is positive semi-definite.

$$\begin{aligned} &\frac{\partial^2}{\partial\theta^{*2}}(Y - F(\theta^*))'(Y - F(\theta^*)) && \text{is positive semi-definite} \\ &\text{i.e. } -2\Big[H(\theta^*)] - X(\theta^*)X(\theta^*)'\Big] && \text{is positive semi-definite} \end{aligned} \tag{3.8}$$

where

$$H(\theta) = \begin{pmatrix} \sum_2^T \left[\frac{\partial^2}{\partial\theta_1^2} f(y_{t-1},\theta)(y_t - f(y_{t-1},\theta^*))\right] & \dots & \sum_2^T \left[\frac{\partial^2}{\partial\theta_1\partial\theta_p} f(y_{t-1},\theta)(y_t - f(y_{t-1},\theta^*))\right] \\ \vdots & \ddots & \vdots \\ \sum_2^T \left[\frac{\partial^2}{\partial\theta_p\partial\theta_1} f(y_{t-1},\theta)(y_t - f(y_{t-1},\theta^*))\right] & \dots & \sum_2^T \left[\frac{\partial^2}{\partial\theta_p^2} f(y_{t-1},\theta)(y_t - f(y_{t-1},\theta^*))\right] \end{pmatrix}$$

The Hessian of the SSE, then, is composed of two parts:

1. the sum of the cross products of the first derivative matrix
2. the sum of the cross products of the model second derivative matrix and the errors over t.

For a linear model, $H(\theta) = \mathbf{0}\ \forall\ \theta$, and $X(\theta)X(\theta)' = Y'_{t-1}Y_{t-1}$, which is always positive definite (unless all $Y_{t-1} = 0$). For our model, the matrix of second derivatives is as follows:

$H(\theta)|_{y_{t-1}} = f_{\theta\theta}(y_{t-1}, \theta) =$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & (.)\text{sech}^2(\gamma^*(.)) & -\gamma^*\text{sech}^2(\gamma^*(.)) \\ 0 & (.)\text{sech}^2(\gamma(.)) & -2\lambda^*(.)^2\text{sech}^2(.)\tanh(.) & -\lambda^*\text{sech}^2(.)[1-2\gamma^*(.)\tanh(.)] \\ 0 & -\gamma^*\text{sech}^2(.) & -\lambda^*\text{sech}^2(.)[1-2\gamma^*(.)\tanh(.)] & 2\lambda^*\gamma^*\text{sech}^2(.)\tanh(.) \end{pmatrix}$$

where (.) indicates $y_{t-1} - c^*$

Nonlinear estimation is different from linear estimation in that there is not a given equation that produces parameter estimates based on the data. Instead, all nonlinear fitting routines conduct a search where the algorithm starts with values specified by the modeler and then proceeds to search the parameter space for the global minimum of the objective function.

Table 3.2 lists four common search routines and their performance on a preliminary simulation with 10 sequences of errors and parameters chosen from the factorial design. The estimation methods listed in Table 3.2 differ in how the search is conducted. In each case, there is an "updating equation" that determines the next value for θ, given the current value. The estimation routine then re-calculates the objective function and determines whether it has improved (i.e. gotten smaller) since the last step. The search continues until successive steps fail to improve the value of the objective function, or until the algorithm has iterated a set (large) number of times.

When data are generated from our assumed model, using parameter combinations chosen from our factorial design and 10 different sequences of errors, the four estimation methods produced wildly different results. In our initial trials, models estimated with Gauss-Newton and Newton-Rhapson converged for approximately 60% of the trials. Models estimated with the Levenberg-Marquardt method converged in approximately 98% of the trials. Models estimated with the Gradient method converged in less than 1% of the trials.

In each case, the true parameter values were used as starting values for the estimation routine. It is clear that there are numerical issues that can prevent a nonlinear model estimated on these data from converging, even when the architecture and parameter values are known.

The poor performance of the gradient method leads one to suspect that there is a problem with the $X(\theta)X(\theta)'$ matrix, possibly a linear dependency among the derivatives. To investigate, we evaluate the gradient at the true parameter values on the series generated from those parameter combinations and investigate the largest condition index of $X(\theta)X(\theta)'$. A condition index of more than 200 indicates a linear dependency between the columns of $X(\theta)$. Of the 9,600 parameter combinations, approximately half had a condition index greater than 200–approximately the number with convergence errors.

In every case where the largest condition index was high, the dependency was between the

Table 3.2: Convergence Statistics for Different Estimation Methods

Solution Method	Percent Converged No Warnings	Update Equation
Gauss Newton	58%	$[X(\theta)X(\theta)']^{-1}X(\theta)[Y - F(\theta)]$
Levenberg-Marquart	98%	$\Big(X(\theta)X(\theta)' + \lambda diag[X(\theta)X(\theta)']\Big)^{-1}X(\theta)[Y - F(\theta)]$
Gradient (Steepest Descent)	$< 1\%$	$X(\theta)[Y - F(\theta)]$
Newton-Rhapson	60%	$G^{-1}X(\theta)'[Y - F(\theta)]$

derivative with respect to to γ^* and the derivative with respect to c^*. Inspection of (3.6) makes it clear why this is so. The partial derivative with respect to γ^* and the partial derivative with respect to c^* both involve $\text{sech}^2(\gamma^*(y_{t-1} - c^*))$. Figures 3.10 and 3.9 show the two derivatives plotted for different values of $(y_{t-1} - c^*)$.

The first partial derivatives with respect to λ^* and γ^* are 0 for all but a very narrow range of $y_{t-1} - c^*$. If all or most y_t are such that the first lag minus c^* is outside this range, then the first partial derivatives will be practically 0, and the cross products of the gradient will be 0 as well.

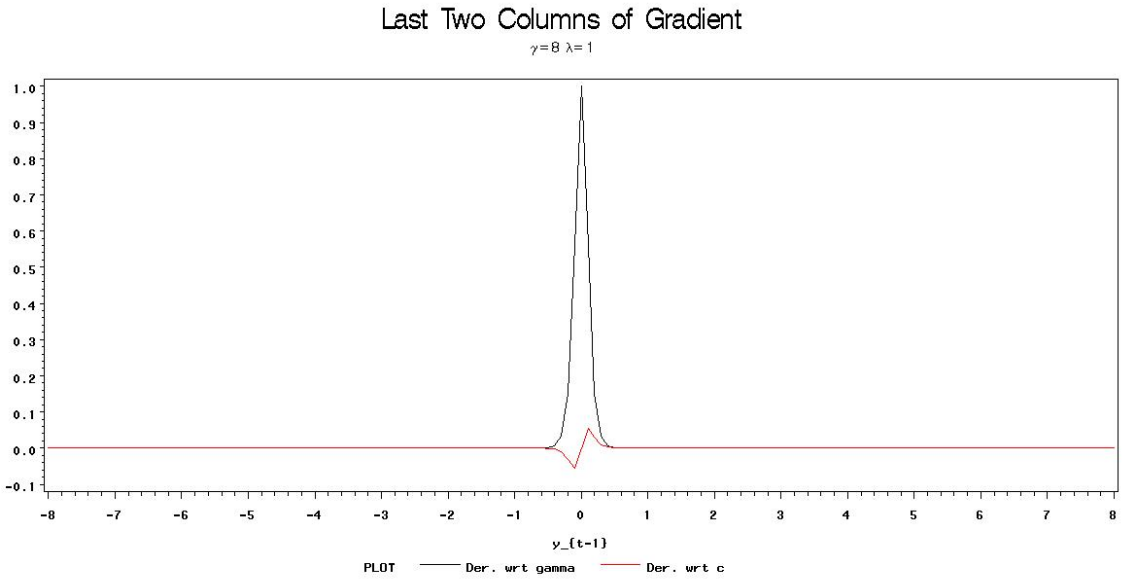


Figure 3.9: Partial Derivatives $\gamma^* = 8$ and $\lambda^* = 1$. The derivative of the model function wrt γ^* and the derivative wrt λ^* are 0 for values of y_{t-1} that are farther than 1 unit away from c^*.

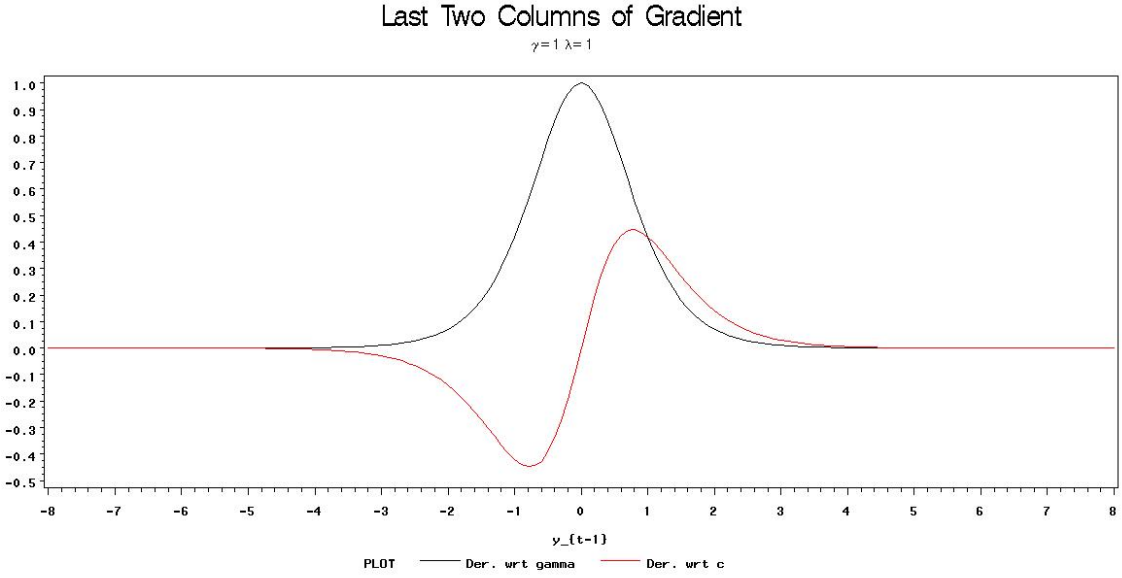


Figure 3.10: Partial Derivatives $\gamma^* = 1$ and $\lambda^* = 1$. When γ^* is small in absolute value, the derivatives are less extreme, and so there is a wider range of $y_{t-1} - c^*$ where the derivatives are nonzero.

3.3 "Practical Reducibility"

For parameterization (1.4), we set c^*, which is actually $c^* = \frac{c-\gamma\mu}{\sigma}$, and $\alpha_0^* = \frac{\alpha_0-\mu}{\sigma}$, where μ is the mean for the series Y_t, and σ is the standard deviation.

We can think of those location parameters, then, as the number of standard deviations between the mean of the series and the point where the tanh(.) function crosses 0 (in the case of c^*) and the number of standard deviations between the mean of the series and the point where the linear portion of the model crosses 0 (in the case of α_0^*). Setting these parameters to reasonable values (i.e. between ± 3, as our noise term is $N(0,1)$) seems to hold the series inside a set where it is possible to estimate the parameters no matter what the behavior of the skeleton–or forces the skeleton to behave in a way that makes the parameters possible to identify.

In contrast, however, it seems like if we set α_0^* and c^* from parameterization (1.4) separately, we can get a series where the parameters are difficult or impossible to estimate.

Consider the Markov theory used in [18] and [19] to prove a series is stationary and ergodic.

In [18] a Markov chain defined by

$$y_t = f(y_{t-1}) + e_t \tag{3.9}$$

is irreducible and aperiodic, provided that 1) $f(y_{t-1})$ is continuous, and 2) $e_t \sim$ IID with infinite support.

This is true for our series because the hyperbolic tangent function is continuous, and $e_t \sim N(0,1)$. Irreducibility and aperiodicity are key to proving stationarity and ergodicity, which lead to consistent estimates. In [19], the same idea is used: The AR-NN is turned into a Markov chain, shown to be irreducible and aperiodic (provided that the shortcut/direct connections/linear terms are stationary) and results in [18] are used to show that the AR-NN must be stationary and ergodic.

The additive noise with infinite support is key to irreducibility. If a Markov process is irreducible, it is possible to get to any portion of the state space from any other portion of the state space. If any value is possible for e_t, then even if $f(y_{t-1})$ is such that the skeleton of the process is cyclic, e_t, which is independent of the values of y_{t-1} will eventually cause y_t to leave the cycle. This allows the process to visit other parts of the solution space.

However, we know that "possible" is not the same as "likely", and for finite T, there are parts of the solution space we will visit infrequently if at all. If the parts of the solution space that the series fails to visit are uninteresting in terms of being able to estimate the parameters of the AR-NN, this is not a concern. However, if the unlikely portion of the solution space is critical to estimating the slope and location parameters of the transition functions, the Markov chain is not irreducible in a practical sense.

We have been able to parameterize the model so that $e_t \sim N(0,1)$. Now, add an intercept term to our model, as follows:

$$y_t = a_0 + f(y_{t-1}) + e_t \tag{3.10}$$

where $f(y_{t-1})$ is our usual hyperbolic tangent function with λ set equal to 1.

This yields a new Markov Chain:

$$\begin{aligned} g(y_{t-1}) &= a_0 + f(y_{t-1}) \\ y_t &= g(y_{t-1}) + e_t \end{aligned}$$

The function $g(y_{t-1})$ is a constant added to a continuous function, and therefore it is a continuous function. Because $f(y_{t-1})$ is the hyperbolic tangent function, it is bounded between -1 and 1. By the properties of the normal distribution, we expect e_t to be within (-3,3) for almost all of the observations. Therefore, the maximum practical range for y_t is $a0 \pm 4$ (the max/min of the hyperbolic tangent function plus the practical max/min of the noise term).

Recall that we have parametrized the hyperbolic tangent function as:

$$\tanh(\gamma^*(y_{t-1} - c^*))$$

Therefore, c^* is the location parameter of the hyperbolic tangent function, the point where the function crosses 0. From our earlier investigation of the hyperbolic tangent function, we know that the first partial derivatives with respect to γ^* and c^* are only non-zero within the range $y_{t-1} - c^* \in (-\frac{4}{\gamma^*}, \frac{4}{\gamma^*})$. If the linear term inside the hyperbolic tangent function $(\gamma * y_{t-1} - c)$ is such that $\frac{c^*}{\gamma^*}$ is outside the range $a_0 \pm 4$, then we cannot reasonably expect to observe values of y_{t-1} that will allow us to estimate the parameters γ^* and c^*.

This is just a re-statement of a feature of the series that we know intuitively must be true: the attraction point cannot be at a position on the flat part of the hyperbolic tangent function. The parametrization in (3.1) allows us to examine the location parameters of the nonlinear model in terms of their distance from the series mean. We therefore know that values for c^* and α_0^* that are larger than 4 in absolute value are not likely to produce series that display nonlinear properties.

Chapter 4

The AR(1) Model vs a "Perfect" Autoregressive Neural Network

4.1 The "Perfect" Model

We suspect that for some parameter combinations the AR-NN is producing data where $E[Y_t|Y_{t-1}]$ is a linear relationship rather than the nonlinear AR-NN. For these sets of parameter values an estimated AR(1) will fit the generated data as well as a trained AR-NN and with less noise. 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 (Section 3.2). We need a statistic that estimates the fit for the AR-NN that we can calculate 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 parameter combination, generate a data set from each error sequence (10 data sets per parameter combination, 96,000 data sets in total) and 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.

If our null hypothesis is that the true relationship between y_t and y_{t-1} is linear, then under the null, $\frac{1}{997}\sum_{t=2}^{1000}(e_t^*)^2$ is a reasonable estimate for the actual mean squared error (MSE) calculated from the the errors used to generate the data. For each parameter combination, then, we can calculate δ_k, the average difference in the MSE's as:

$$\begin{aligned} \hat{\delta}_k &= \frac{1}{10}\sum_{s=1}^{10} MSE_{sp} - MSE_{sa} \qquad (4.1) \\ MSE_{sp} &= \frac{1}{999-1}\sum_{t=2}^{1000} e_{st}^2 \\ MSE_{sa} &= \frac{1}{999-2}\sum_{t=2}^{1000}(y_t - (a_0 + a_1 y_{t-1}))^2 \end{aligned}$$

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 may be tempted to say that $\hat{\delta}_k$ should always positive, since the AR(1) should not be able to out-perform the correct model. In [21], however, the possibility of a larger MSE for a full model verses a reduced model is discussed, and an appropriate distribution for the MSE of a forecast is derived. In [21], it is shown that a reasonable null hypothesis for a statistic like δ_k is $H_0 : \delta_k \leq 0$,which allows for the fact that when the reduced model approximates the true relationship in the data better than the full model, the estimated reduced model may have a slightly smaller MSE than the full model.

Because we are not comparing a reduced and full model but the actual noise values from the simulation verses the mean squared error from an AR(1), the distribution for $\hat{\delta}_k$ is slightly different the one derived in [21], though in the simulation shown below, our results did include a few negative values for δ_k that were nonetheless close to 0. In the next section we investigate which parameter combinations produce large values of $\hat{\delta}_k$ and which produce $\hat{\delta}_k$ that are close to 0.

4.2 Examining $\hat{\delta}_k$ in Terms of Parameter Values

Recall that our reduced and re-parameterized model is:

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

Where the $*$ indicates that the parameters are re-centered to represent a distance from the series mean μ and scaled by the standard deviation of the error term, σ. For simplicity, from here forward we will drop the $*$ notation and refer to the model equation as:

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

We have speculated in Chapter 3 that the convergence problems for the Gauss-Newton and gradient methods are partly due to collinearity between the linear portion of the tanh(.) function and the linear shortcut connection. We hypothesize that when ρ (the slope parameter for the shortcut connection) and $\gamma\lambda$ (the slope parameter and weight of the activation function) have the same sign, we will either be stuck at a boundary attractor (Figure 4.5), or the generated data will be close to the center of the tanh(.), the portion of the hyperbolic tangent that is most like a straight line. In either of these two cases a straight line will explain the data as well as the nonlinear model function.

We further speculate that the difference in the location parameters (α_0 and c) is a secondary factor in producing series that do not exhibit nonlinear features. In Section 3.3 we show that when the activation function is bounded (a necessary condition for stationarity and ergodicity), and the model location parameters (α_0 and c) are far apart compared to the practical range of the deterministic portion of the model, the Markov chain formed by the discrete time series is not ergodic in a practical sense. Therefore, when $|\alpha_0 - c|$ is large (where "large" is determined by the assumptions on the noise term and the limit of the activation function) we would expect problems in estimating the series parameters.

To test these speculations, let us 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 (4.1)). The subscript k indexes the parameter combinations.

In Figure 4.1, each point represents a parameter combination. The $\hat{\delta}_k$ are plotted on the vertical axis and the value of λ is plotted on the horizontal axis. The colors of the points are determined by $|\alpha_0 - c|$, with light blue representing small values of $|\alpha_0 - c|$ and red representing larger values of $|\alpha_0 - c|$. 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 even function, i.e. $-\lambda \tanh\left(\gamma(y_{t-1} - c)\right) = \lambda \tanh\left(-\gamma(y_{t-1} - c)\right)$, 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

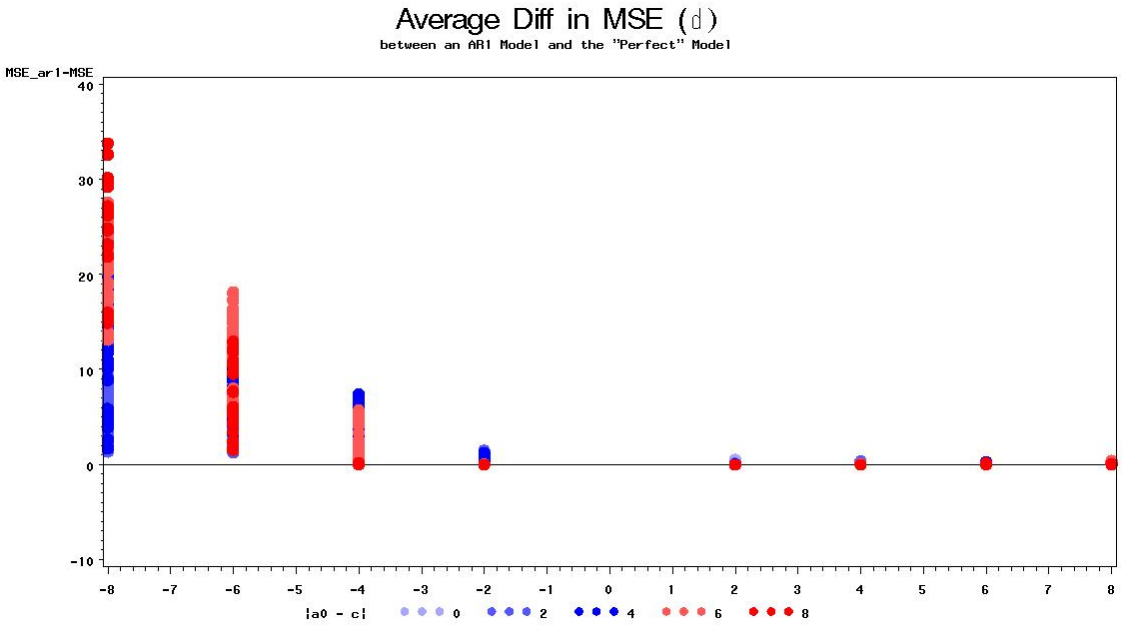


Figure 4.1: There is a striking contrast between the comparison of model fit for $\lambda > 0$ and $\lambda < 0$.

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.

The range of $\hat{\delta}_k$ is much greater for parameter combinations where $\lambda < 0$. Showing them on the same graph with parameter combinations where $\lambda > 0$ may be hiding additional patterns in the data. Figure 4.3 shows only the parameter combinations where $\lambda < 0$. Figure 4.2 shows only the parameter combinations where $\lambda > 0$. Separating the graph in this way reveals a relationship between the magnitude of λ and the value of $|\alpha_0 - c|$.

In both cases, 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 3.3. 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.

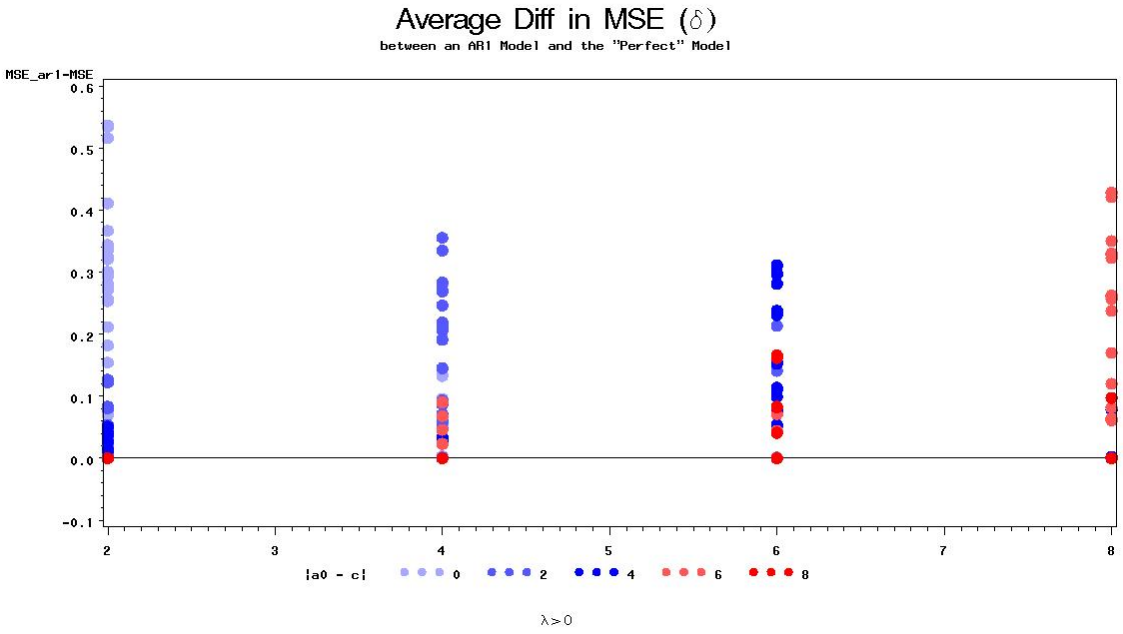


Figure 4.2: For scaling purposes, this figure shows the $\hat{\delta}_k$ for data generated where $\lambda > 0$.Notice when the magnitude of λ is small, larger values for $\hat{\delta}_k$ are associated with smaller values of $|\alpha_0 - c|$, i.e. when the weight on the activation function is small in absolute value and positive, the location parameters of the activation function and the shortcut connection should be close together to produce a series with nonlinear features.

Figure 4.4 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 $\lambda = -8$, $\alpha_0 = 4$, $c = -2$, $\gamma = 1$, and $\rho = 0.2$. These data scatter evenly on both sides of the location function for the hyperbolic tangent function and near the location parameter as well. It should be easy to obtain parameter estimates and AR-NN model from these data, and that 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 4.5 shows the data and underlying functions for the exact same set of parameters, as Figure 4.4 except that λ is positive instead of negative (set at 8 instead of -8). In this case, the series runs to a boundary attractor that is far away from the location parameter of the hyperbolic tangent function. The hyperbolic tangent function adds little more to the generated data than a constant. It is not surprising that $\hat{\delta}_k < 0.0009$ for this series.

The proof in [11] implies that the presence of a unique and globally stable attraction point depends only only the values of the slope parameters (ρ and γ) and the weight on the activation function (λ). The results above imply that while this may be true, the distance between the

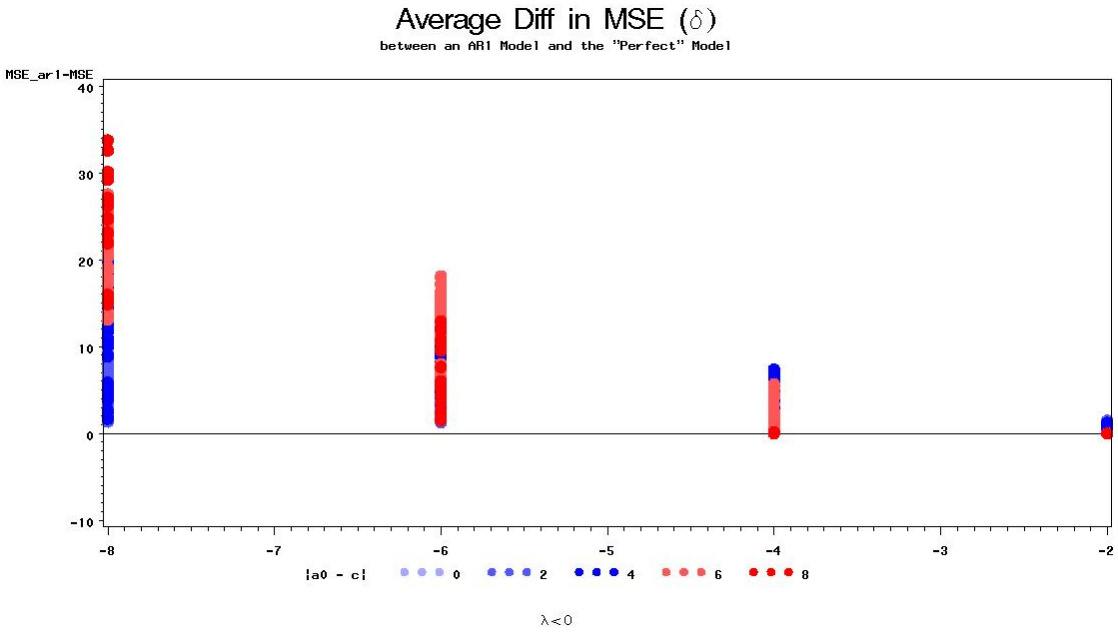


Figure 4.3: For scaling purposes, this figure shows the values of $\hat{\delta}_k$ for data generated where $\lambda < 0$.

location parameters, relative to the weight of the activation function, is important in determining whether it is possible to estimate the parameters of the neural network. We show here two examples where the location parameters are equal to each other to emphasize that the optimum distance between the location parameters (in terms of producing a series with nonlinear features) depends on the magnitude of λ.

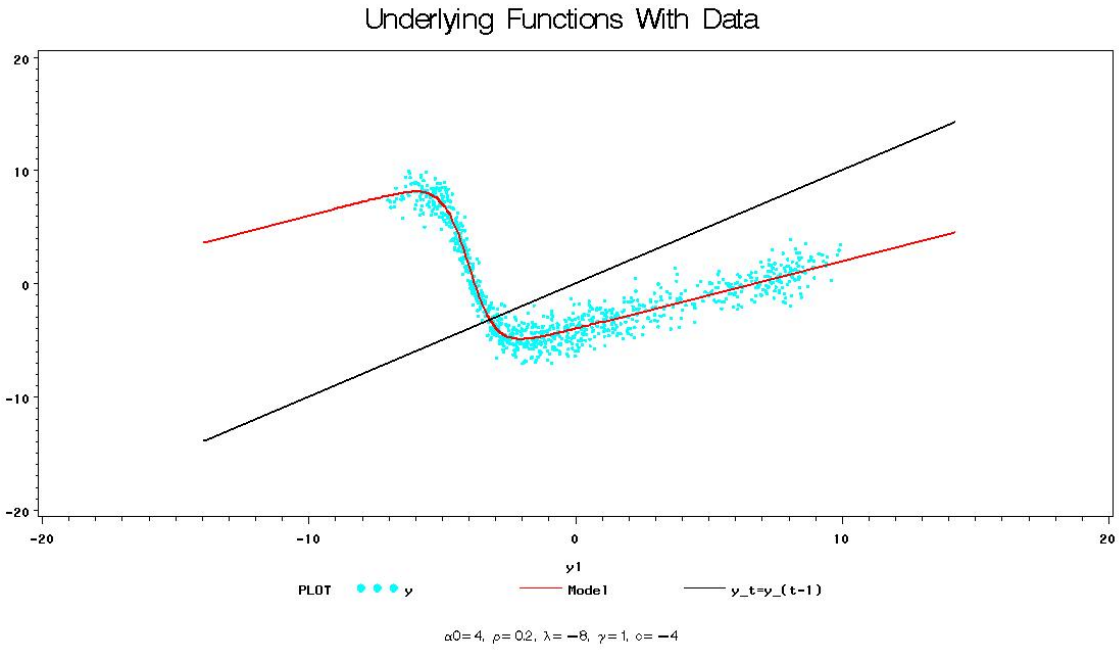


Figure 4.4: The generated data and the function that generated them are plotted together in order to show how the data relate to the underlying functions. In this case, the weight on the hyperbolic tangent function (λ) is negative and large in absolute value and the location parameters are far apart ($|\alpha_0 - c|$). The data follow the underlying model function closely, and the parameters for the nonlinear model are easy to estimate.

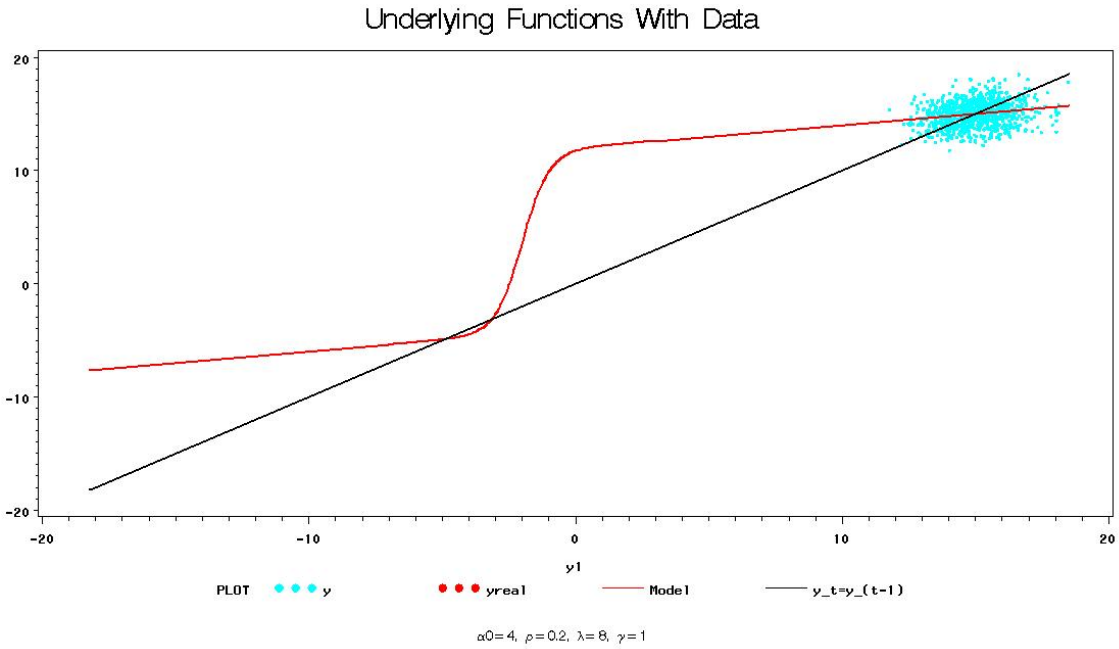


Figure 4.5: The generated data and the function that generated them are plotted together in order to show how the generated data relate to the underlying functions. In this case, the data cluster near a boundary attractor and the hyperbolic tangent function adds little more than a constant.

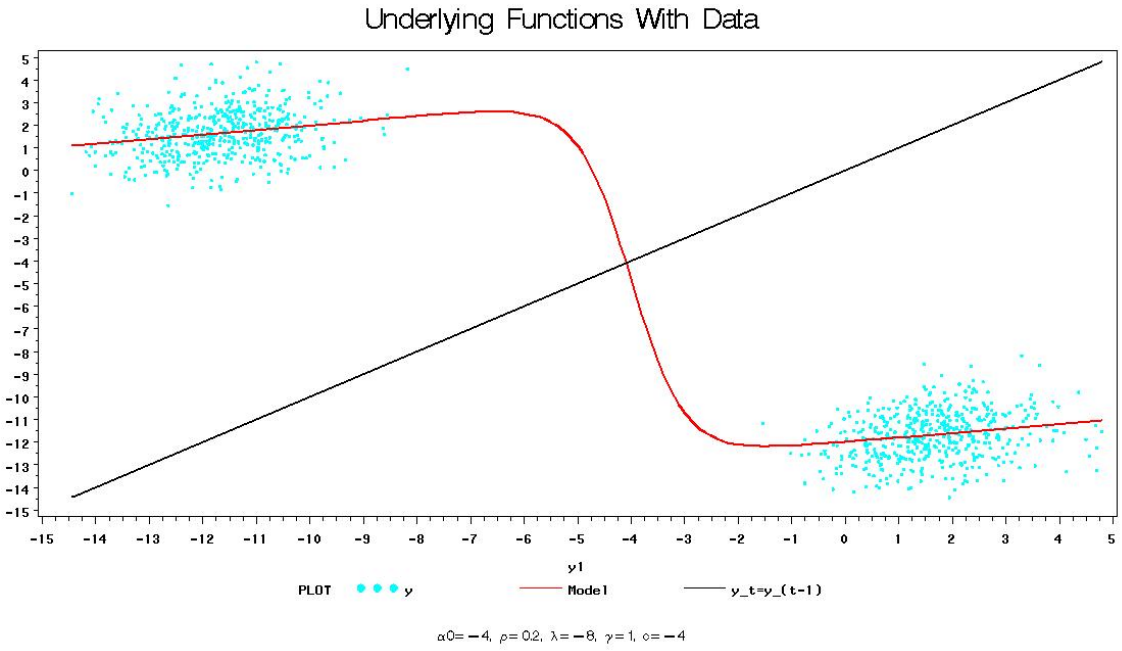


Figure 4.6: The optimum value for $|\alpha_0 - c|$ depends on the magnitude of λ When $|\lambda|$ is large, location parameters that are close together generate series where the data cluster at opposite sides of the hyperbolic tangent function, with little information about the location parameter.

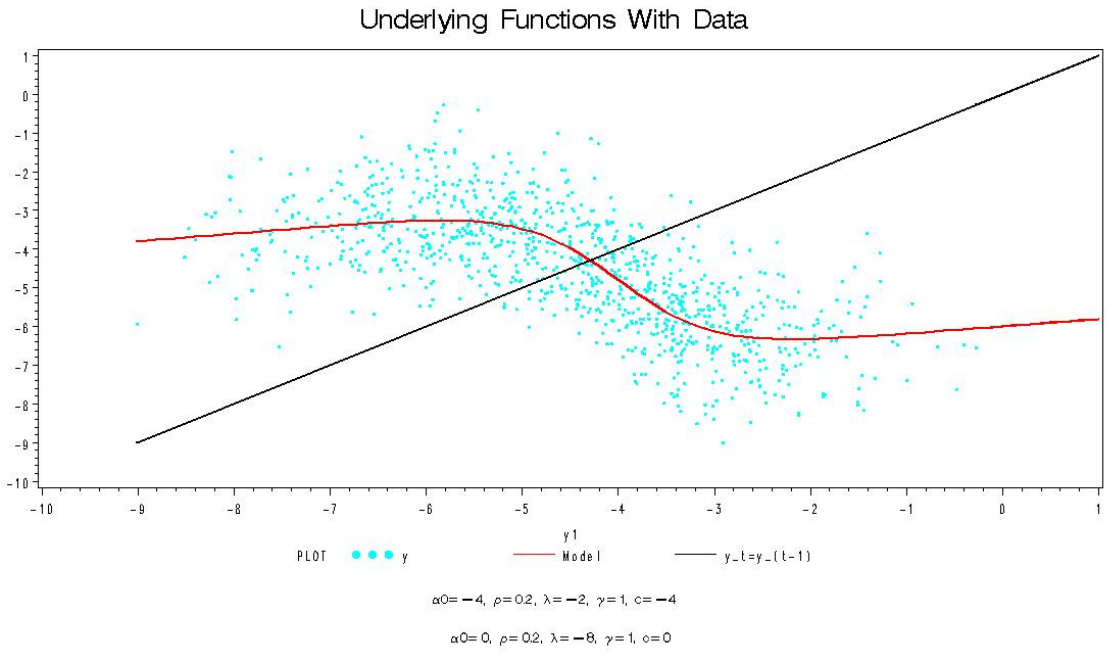


Figure 4.7: The optimum value for $|\alpha_0 - c|$ depends on the magnitude of λ. When $|\lambda|$ is small, location parameters that are close together generate series where the data follow the hyperbolic tangent function smoothly. Notice, however, that because $|\lambda|$ is smaller, these data do not cluster around the model function as tightly, producing a noisy fit around the function that generates the data.

4.3 Relationship Between λ and ρ

As we have shown in the previous section, when λ and ρ are of the same sign, there are collinearity problems with the nonlinear model. By the proof in [19], we know that $\rho \in (-1, 1)$ is required in order for the series to be stationary and ergodic. Therefore, when the hyperbolic tangent function and the shortcut connection have the same sign, we can think of the underlying function as a weighted average between them, where the magnitude of ρ determines the weight on the straight line, and as ρ increases, the underlying model function grows closer to a straight line and estimation of the parameters of the nonlinear function becomes more difficult. Figure 4.8 shows the relationship between $\hat{\delta}_k$ and ρ for $\lambda > 0$. Note that when the slope of the hyperbolic tangent function and the shortcut connection are the same sign, the range of $\hat{\delta}_k$ gets smaller, and δ_k are more consistently close to 0 as ρ increases.

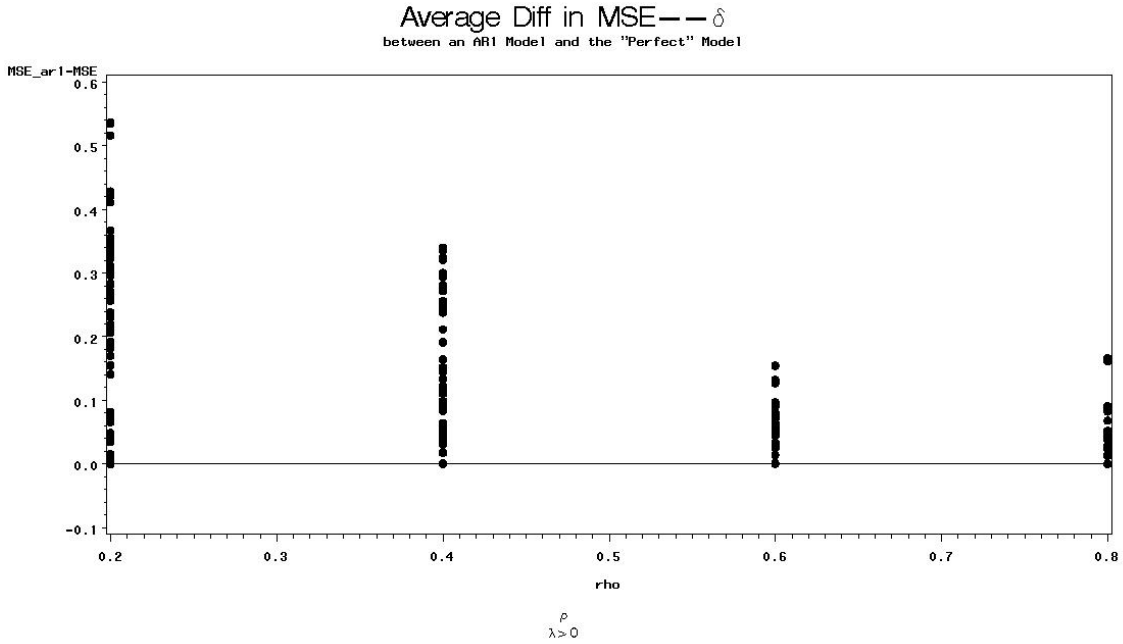


Figure 4.8: When the slope of the hyperbolic tangent function and the shortcut connections have the same sign, the underlying model function is a weighted average of the shortcut connection and the activation function. As ρ increases, the underlying function is closer to a straight line, and therefore the generated data follow a straight line.

When the slope of the hyperbolic tangent function has the opposite sign from ρ, the magnitude of ρ is not as important in determining the fit of the AR(1) model. This is because the model is no longer a weighted average between the hyperbolic tangent and the shortcut

connections. Figure 4.9 shows the relationship between ρ and $\hat{\delta}_k$ when $\lambda < 0$.

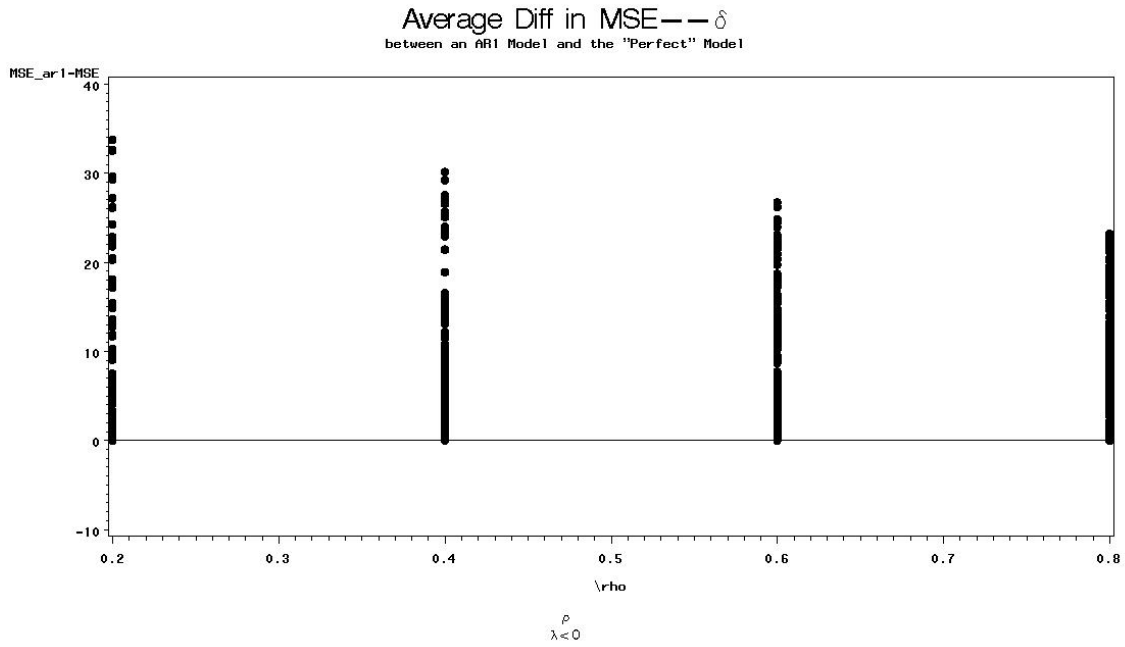


Figure 4.9: When the slope of the hyperbolic tangent function and the shortcut connections have opposite signs, the underlying model function is no longer a weighted average of the shortcut connection and the activation function. The value of $\hat{\delta}_k$ is less dependent on the value of ρ.

4.4 Simulation Reduction

In later simulations, we choose to exclude parameter combinations where 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 such as a singular Hessian matrix (see Chapter 3).

To reduce the number of series, 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 that the mean is $\delta_k > 0$. The p-values for this one-sided hypothesis test are displayed in Figure 4.10.

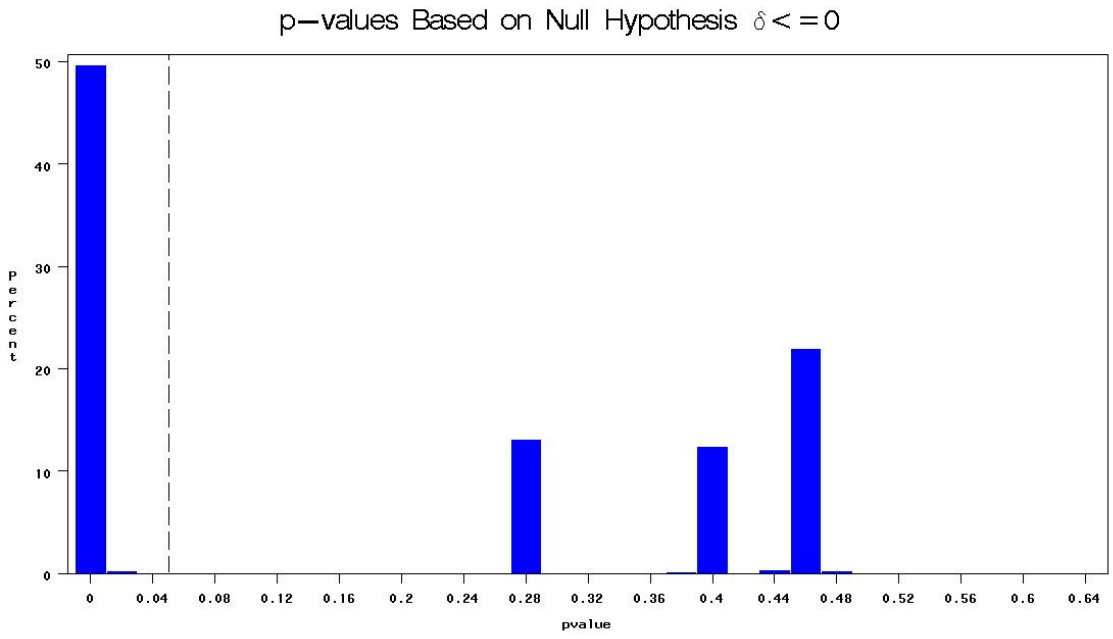


Figure 4.10: Distribution of p-values from a one-sided hypothesis test where the null hypothesis is $\delta_k \leq 0$. For the simulation, we use a cut-off of 0.05 and drop any parameter combination where the p-value from the one-sided hypothesis test is greater than 0.05.

In later simulations, where we evaluate the performance of the AR-NN model, we wish to consider only series where the generated data exhibit nonlinear features. We will therefore eliminate models where we fail to reject the null hypothesis $\delta_k \leq 0$ at the significance level

of 0.05, i.e. where there is no statistical evidence that the estimated unexplained variation from the AR(1), MSE_a, is larger than the true variation from the nonlinear model. Parameter combinations that failed to reject for this hypothesis test were dropped from the simulations shown in Chapters 5 and 6.

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. Figure 4.11 and Figure 4.12 show the generated data and the underlying model function for two of these parameter combinations.

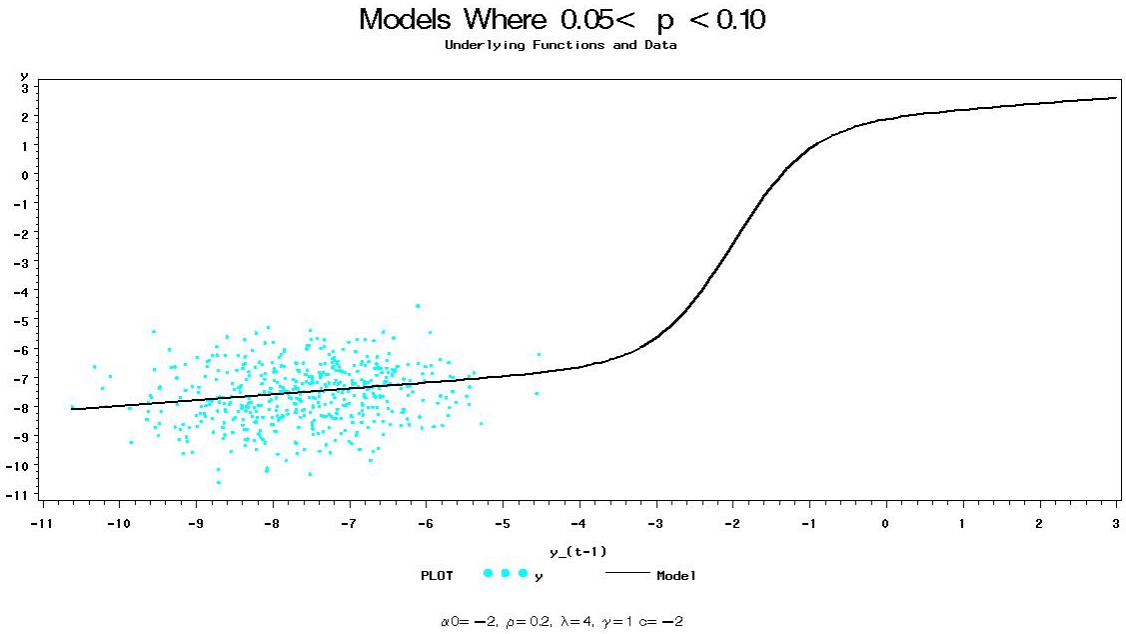


Figure 4.11: Higher cutoffs for the p-value were considered. However parameter combinations that produced p-values between 0.1 and 0.3 were cases where neither the AR(1) nor the AR-NN fit the data well.

Approximately 4800 parameter combinations produced p-values that were less than 0.05. Not surprisingly, the main difference between the parameter combinations chosen and those not chosen was the sign and magnitude of λ. Most of the parameter combinations chosen included values of λ that were less than 0 (Figure 4.13).

Investigating the parameters α_0, γ, and c showed counts that were equally spaced across the values of the factorial design for both the selected models and the dropped models (Table

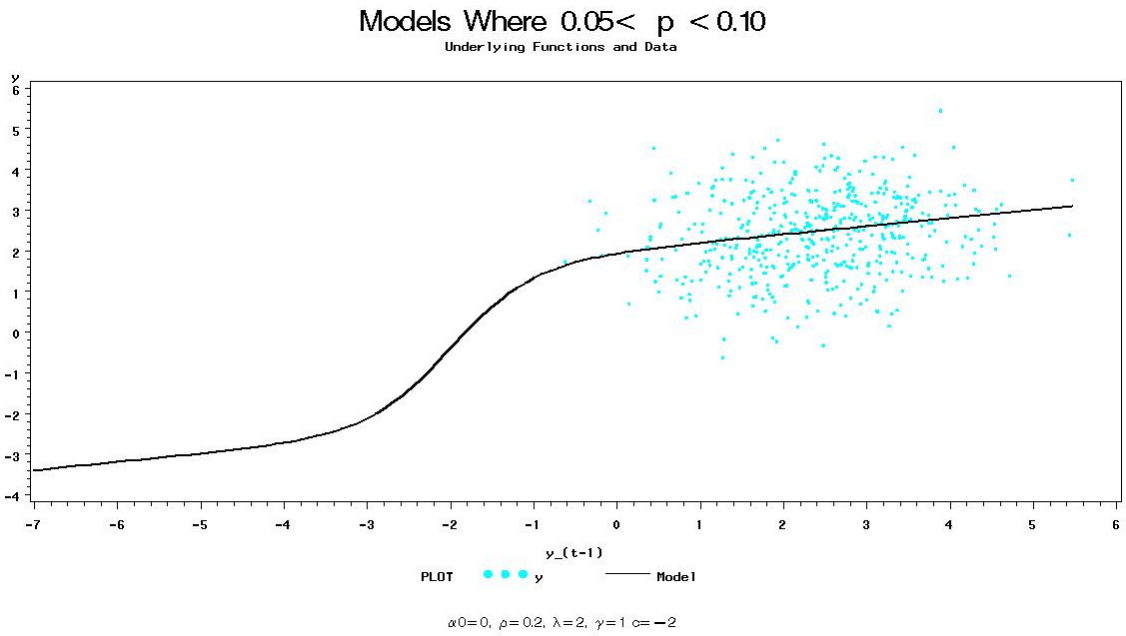


Figure 4.12: Higher cutoffs for the p-value were considered. However parameter combinations that produced p-values between 0.1 and 0.3 were cases where neither the AR(1) nor the AR-NN fit the data well.

4.1 and 4.2).

As before, we observe when $\hat{\delta}_k$ is small, λ (and hence $\lambda\gamma$) tends to be less than 0. Notice, however, that there are parameter combinations where $\hat{\delta}_k$ is well below 0.0426 and $\lambda\gamma > 0$.

If we examine the distributions of the parameter combinations for λ when $\hat{\delta}_k \geq 0$, we find that while most of the parameters have the same distribution as for $\hat{\delta}_k \leq 0$, the λ, as expected, is heavily skewed toward the positive values (Table 4.2).

However, from our observations in Figure 4.1, we expect a relationship between λ and $|\alpha_0 - c|$ for the models with p-values less than 0.05. Figure 4.14 shows that the mean of $|\alpha_0 - c|$ is indeed smaller when $|\lambda|$ is smaller.

Table 4.1: For models where $p > 0.05$ (i.e. where the AR(1) model's fit is comparable to the "perfect" model), λ tends to be positive. The other parameters appear to be equally spaced along the ranges specified by the factorial design.

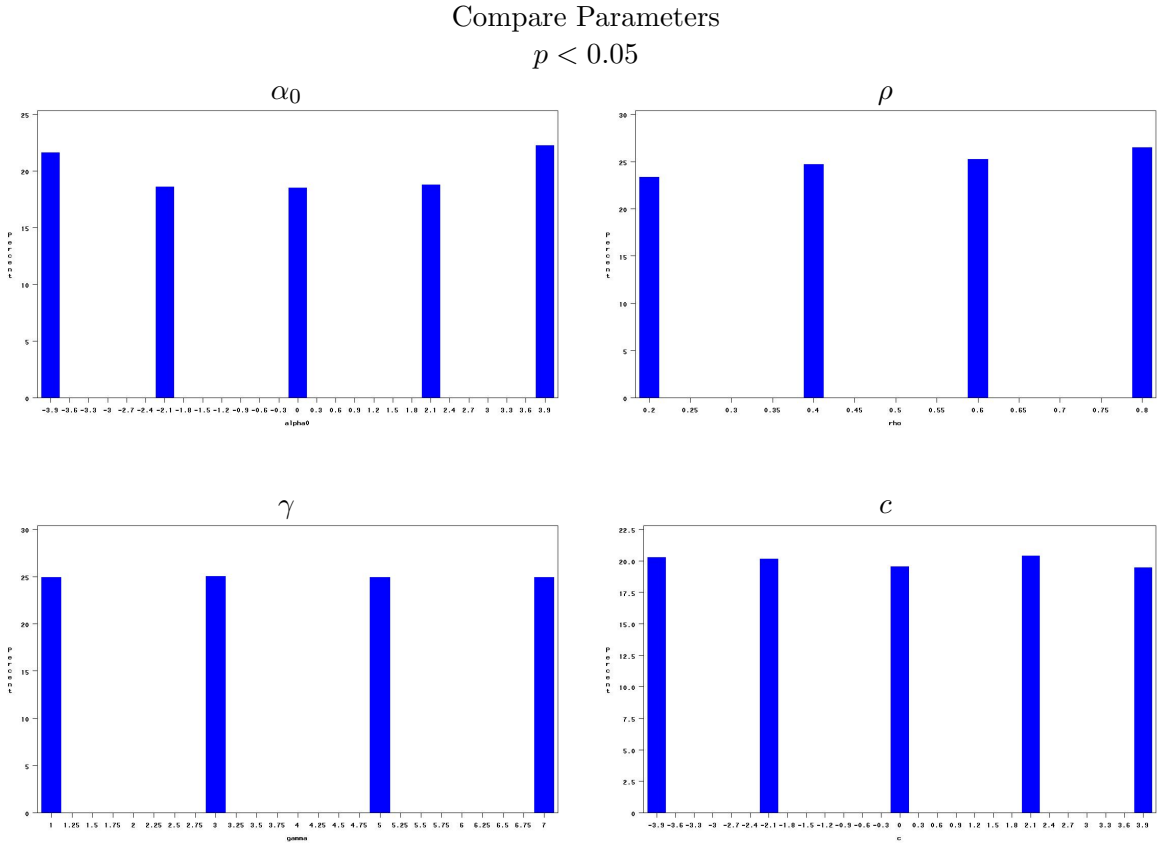
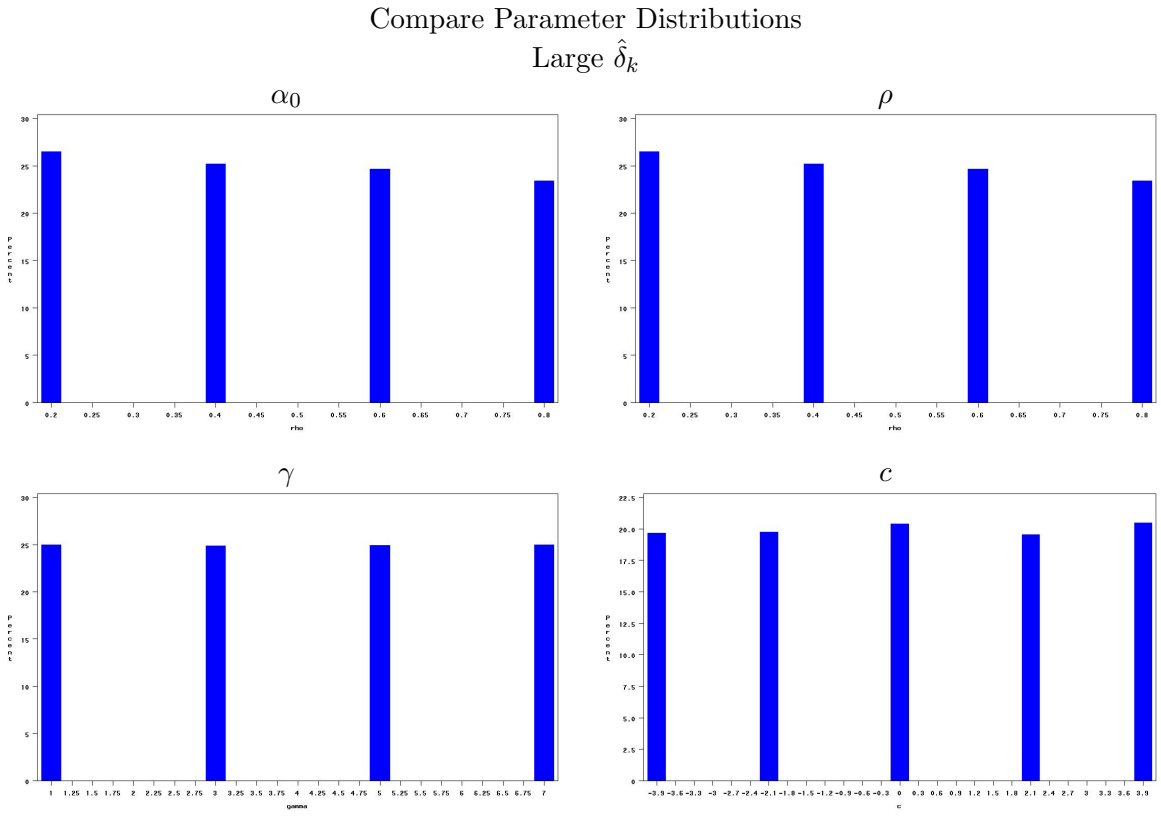


Table 4.2: For parameter combinations where $p < 0.05$ (i.e. where the AR(1) model fits the data poorly) λ tends to be negative. The other parameters appear to be equally spaced along the ranges specified by the factorial design.



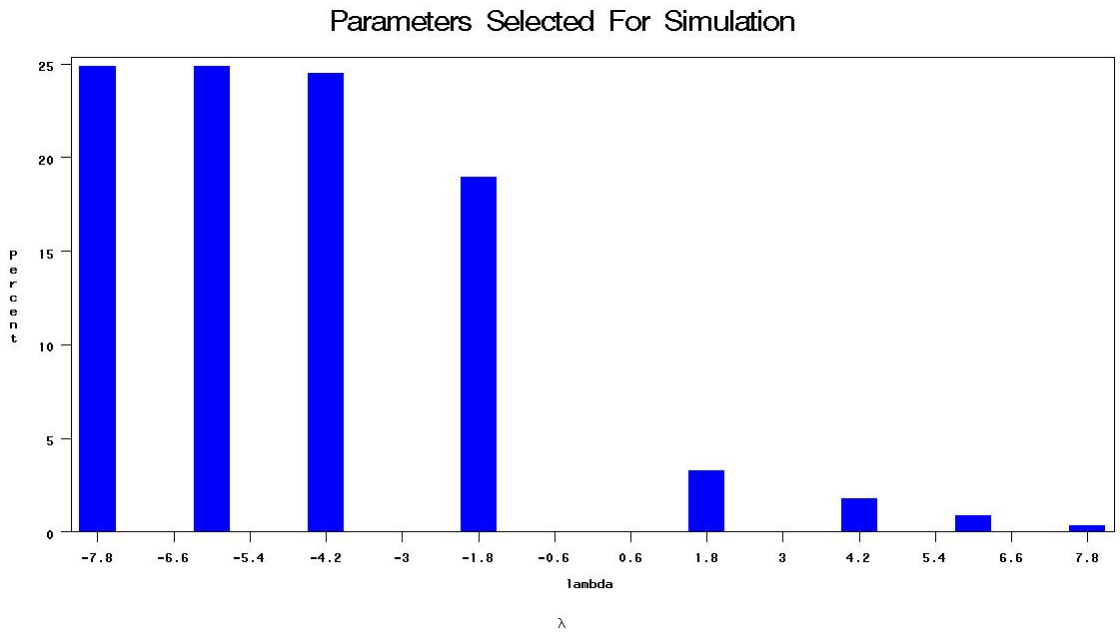


Figure 4.13: Not surprisingly, the parameter combinations selected for the remaining simulations tend to have negative values for λ.

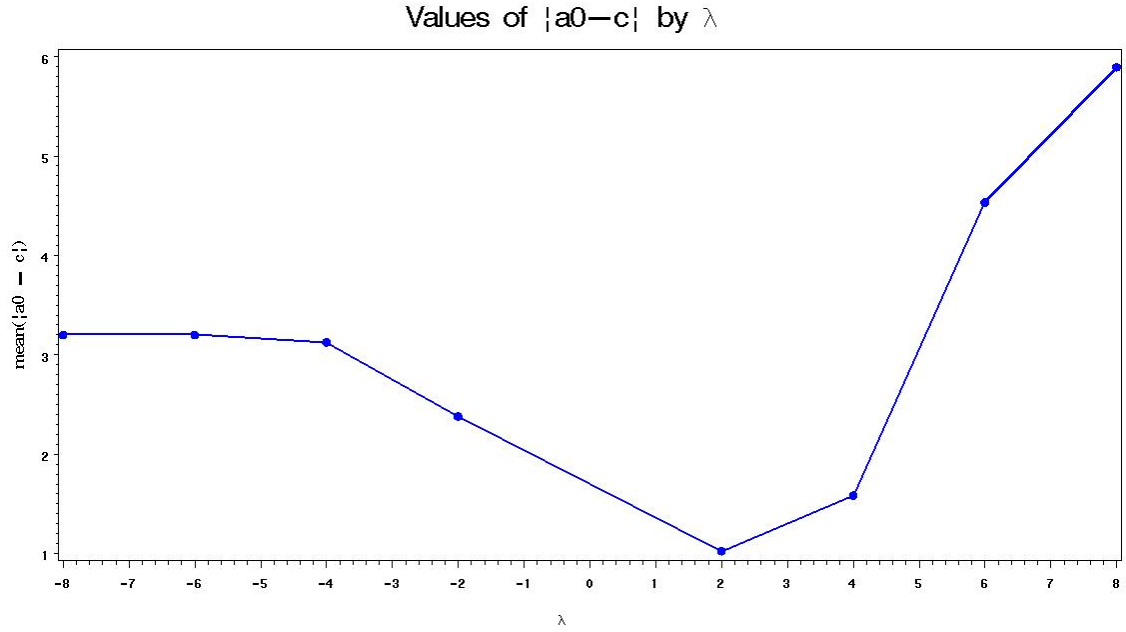


Figure 4.14: For the parameter combinations that produced small p-values, there is a relationship between λ and $|\alpha_0 - c|$.

4.5 Relationship Between Convergence and $\hat{\delta}_k$

If a linear model will fit as well as our nonlinear model, it is likely that when we try to estimate the nonlinear model, we will have convergence problems. Either the model will fail to converge, or the model will converge but with a singular Hessian matrix. In our 10 series of e_t, each of which is used to create a data set from each of the 9600 combinations of parameters, we reported estimation problems in approximately half the cases (results not shown). In all cases where SAS's PROC NLIN reported estimation problems, the reason given was a singular Hessian.

We now relate the probability of producing a singular Hessian to the value of $\hat{\delta}_k$. We calculate the number of series where the AR-NN converges without errors and with sensible parameter values for 20 trials. In this case, "sensible" is determined as $\hat{\rho} \in (-1, 1)$, $|\hat{\lambda}|$, $|\hat{\alpha}_0|$, $|\hat{\gamma}|$, and $|\hat{c}|$ all less than 90. We then sort the parameter combinations by the statistic $\hat{\delta}_k$ and calculate the cumulative average of the percent of models that converge. The results are shown in Figure 4.15. The probability of convergence without errors increases sharply as $\hat{\delta}_k$ increases.

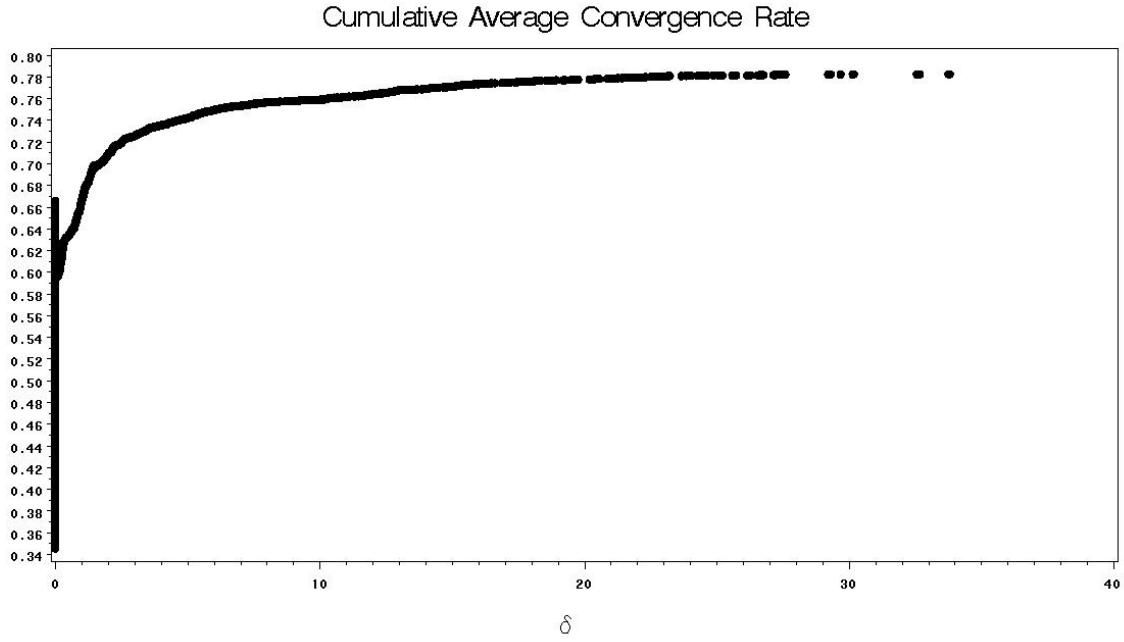


Figure 4.15: The proportion of models converging without errors increases sharply as $\hat{\delta}_k$ increases.

The cumulative probability may be misleading, since small values at the beginning are retained all the way across the data. Let us then bin the observations according to the values of

$\hat{\delta}_k$ and look at the mean probability of convergence without errors (singular Hessian or failure to attain sufficient reduction of sum of squared errors) for each bin (Figure 4.16). Although the relationship is not quite as smooth or dramatic; the graph shows the same relationship: small values of $\hat{\delta}_k$ are associated with lower probabilities of convergence.

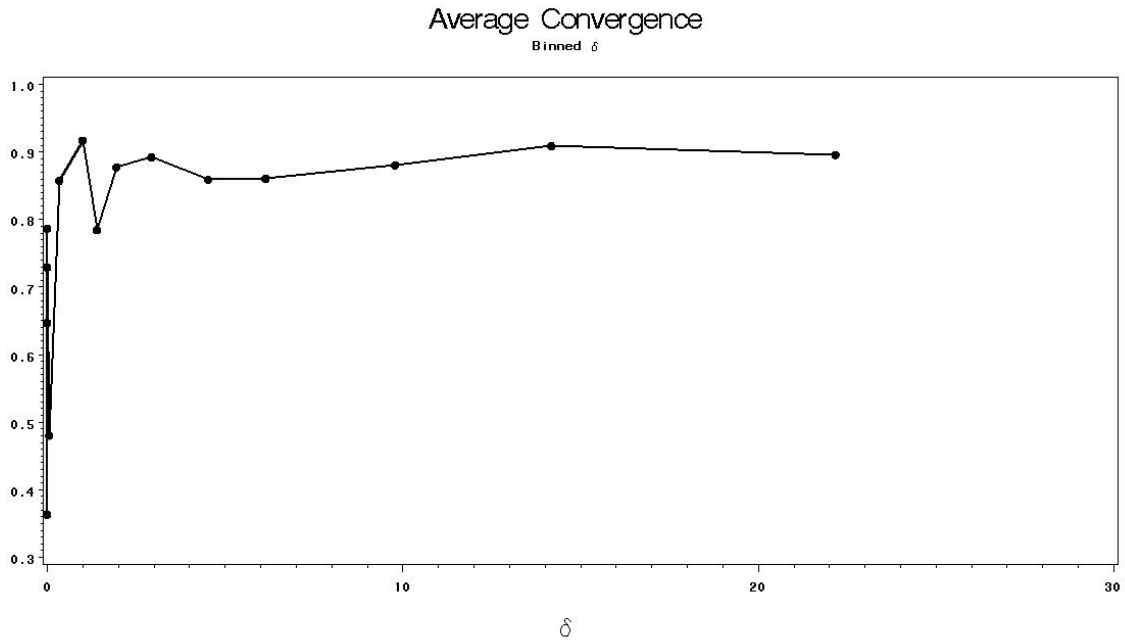


Figure 4.16: Binning the observations shows a much sharper increase in the proportion of convergent models as $\hat{\delta}_k$ increases.

4.6 The Importance of Noise

The simulations in this chapter 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 3.3 shows that for a bounded activation function (which is required in order for the AR-NN to be stationary) and additive noise that is distributed normally, 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 we assume that e_t has a heavy-tailed distribution, however, we can relax the range re-

strictions on the location parameters. To demonstrate, we perform the same simulation as outlined in Section 4.1, 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:

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

with scale parameter s and location parameter μ. As before, 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.

When the simulation is run for the same parameter combinations as with the normal distribution, we see a dramatic difference in the p-values for δ_k (Figure 4.17).

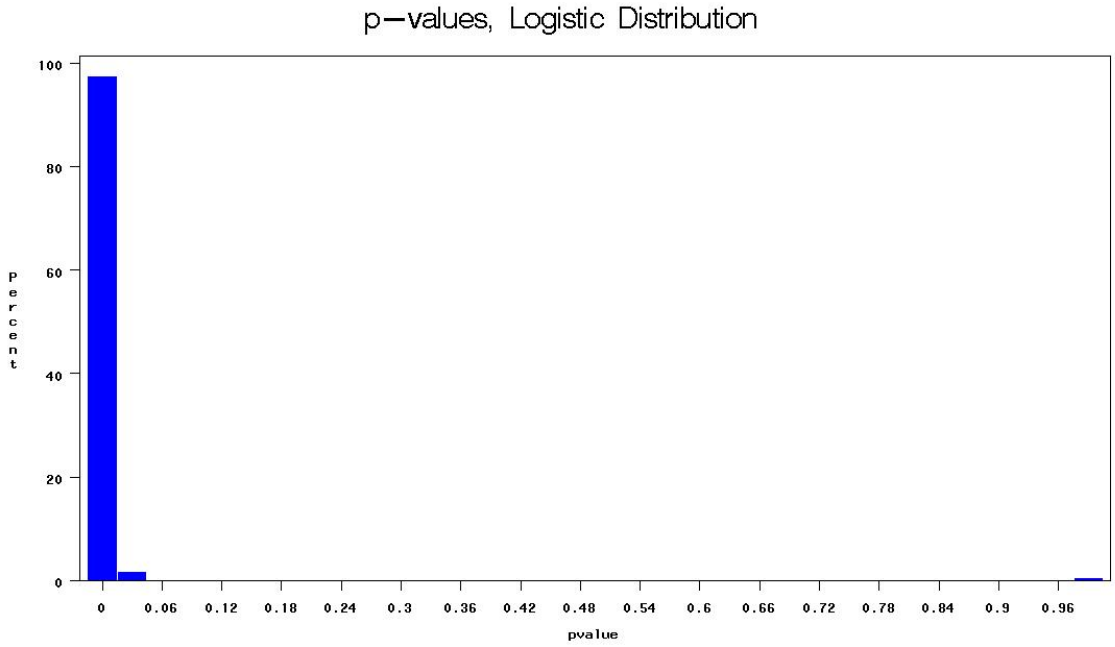


Figure 4.17: When the same simulation is run using a heavy-tailed distribution (in this case, the logistic) for e_t, the p-values are dramatically different.

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 (Figure 4.18).

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

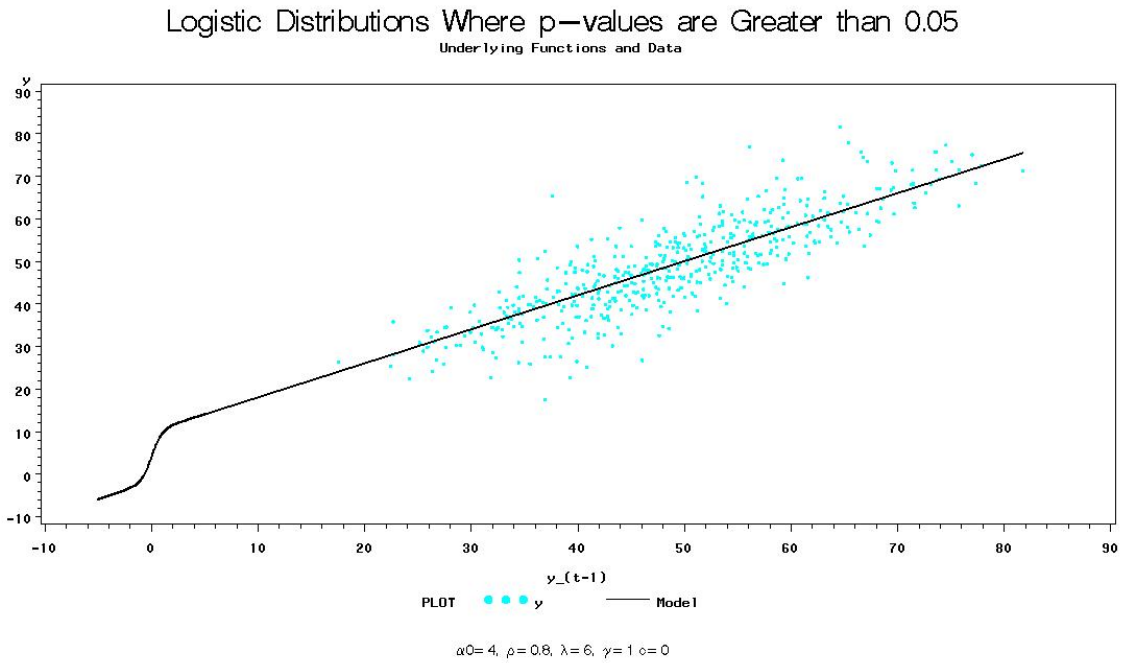


Figure 4.18: When the p-value from the logistic simulation was larger than 0.05, the relationship between y_t and y_{t-1} showed very little nonlinear behavior.

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

4.7 Conclusions

In approximately half the cases (4792), our factorial design led to a combination of parameters that produced simulated series where an AR(1) model fit the data as well as a model that assumes we were able to estimate, without error, the parameters that generated the series. We measure this with a statistic called $\hat{\delta}_k$, which is the average difference between the MSE for the perfect model and the MSE for an AR(1) model over 10 different series of errors. From inspection of the models where $\hat{\delta}_k$ was small, we are able to draw a few inferences.

1. If ρ and $\lambda\gamma$ have the same sign, the series is more likely to follow a straight line instead of having nonlinear features.

2. The distance between the location parameters $|\alpha_0 - c|$ determines whether the parameter combination generates a series that fails to display nonlinear features.

3. The $|\lambda|$ determines the permissible range of $|\alpha_0 - c|$. When $|\lambda|$ is large, large values of $|\alpha_0 - c|$ are associated with large δ_k, i.e. parameter combinations where the generated data follow the functional form or an AR-NN rather than an AR(1). When $|\lambda|$ is small, smaller values of $|\alpha_0 - c|$ are associated with larger values of δ_k.

4. The assumed distribution for e_t is very important in determining whether the resulting series displays nonlinear behavior. When a heavy-tailed distribution with the same mean and variance was used to generate the series, the results were dramatically different.

Because this is a time series analysis, the domain of the predictor is the same as the range of the outcome. It is therefore not surprising that the permissible range for $|\alpha_0 - c|$ is related to $|\lambda|$, which determines the range of the activation function. This is consistent with the results from section 3.3 and sheds more light on the results from [17], where a trained AR-NN performs poorly compared to linear models, and results from [4], where predictions generated from an AR-NN on actual data were linear in most directions.

In addition to the parameter combinations that did not generate data with nonlinear features, approximately half the models, 4,808, generated data that did exhibit nonlinear features. In the following sections we explore estimation and starting value techniques for these parameter combinations and examine forecasts generated for the nonlinear series.

Chapter 5

Forcasting With Neural Networks

5.1 Forecasting with Autoregressive Neural Networks

The primary purpose of an autoregressive neural network (AR-NN) is to forecast. As pointed out in [6], a neural network offers little insight to the underlying structure of the data. It is an approximator to an unknown relationship, and the parameter estimates are difficult to interpret, if they can be interpreted at all. If the AR-NN does not forecast well, there is little benefit to using it. We therefore create two different different types of forecasts for our simulated data: 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 Chapter 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, i.e. we assume that tomorrow will be the same as today. 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 Leveberg-Marquardt.

In most cases, the AR(1) model produces a very poor fit (see Figure 5.1), 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 4.4). 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 [2]. 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, with values for λ that were greater than 90, or values for ρ that were larger than 1 in absolute value. This problem is reminiscent of the troubles reported in [17], 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 case, the unreasonable parameter estimates only occurred for a few series and not consistently for any parameter combination. In the following analysis, we consider all forecasts (since it is often up to the researcher to identify a parameter estimate as "unreasonable") unless stated otherwise.

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 [17] this procedure is recommended in order to stabilize forecasts from a neural network model.

Table 5.1: Although it is recommended in [17] that a bootstrap approach be used to stabilize AR-NN predictions, the point forecasts have better statistics overall for one-step-ahead forecasts.

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

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:

$$\begin{aligned} RW\ R-squared &= 1 - \frac{SSE}{RW\ SSE} * \frac{T-2}{T} \\ RW\ SSE &= \sum_{t=2}^{T} (y_t - y_{t-1})^2 \end{aligned} \tag{5.1}$$

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 for a time series forecast 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 6.3.

As in [17], an insanity filter is applied to the forecasts so that models with obviously un-

reasonable parameter estimates are excluded. In our case, we chose to exclude models where $|\rho| > 1$, $|\hat{\alpha_0}| > 90$,$|\hat{c}| > 90$, and $|\hat{\lambda_0}| > 90$. This excludes approximately 3% of the resulting estimations, though no one parameter group was excluded for a large number of cases.

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.

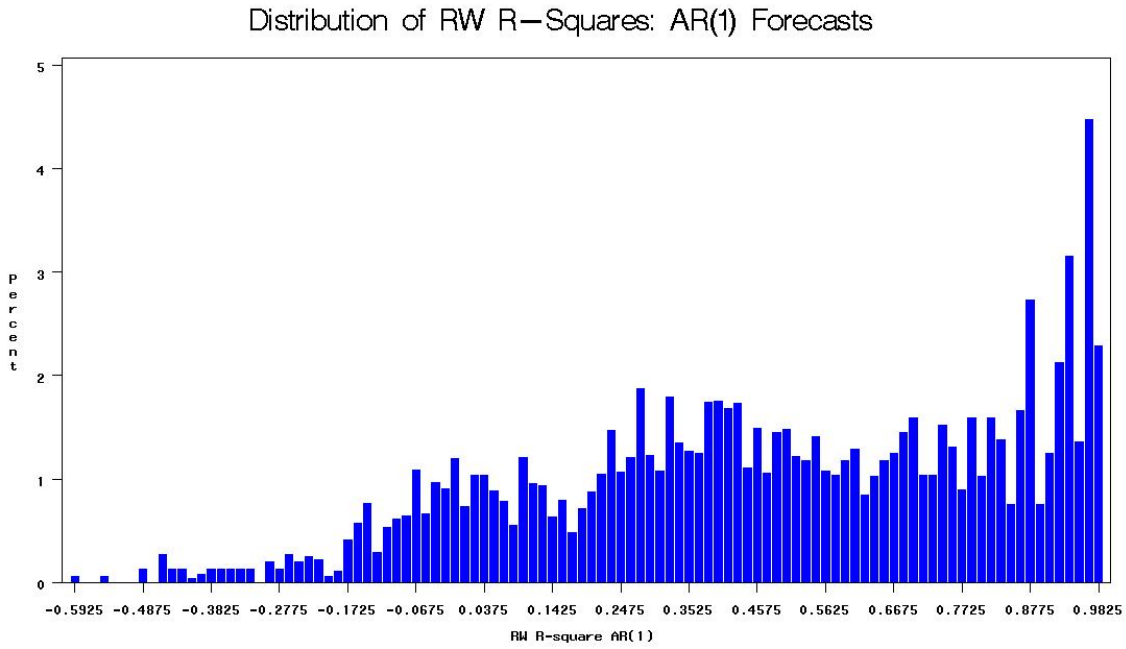
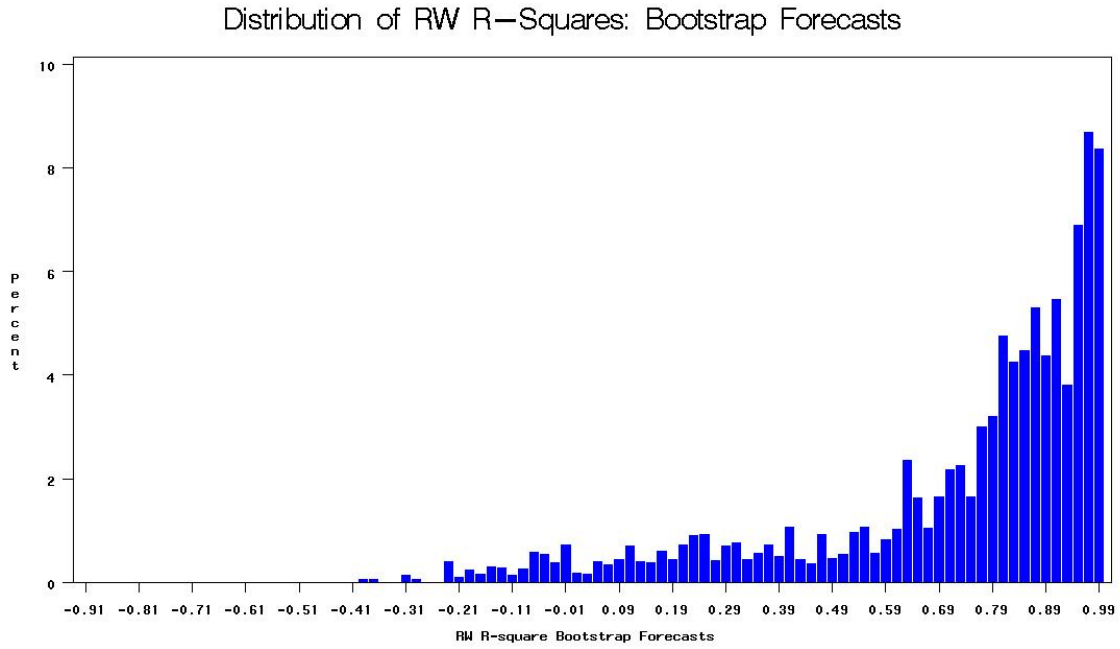


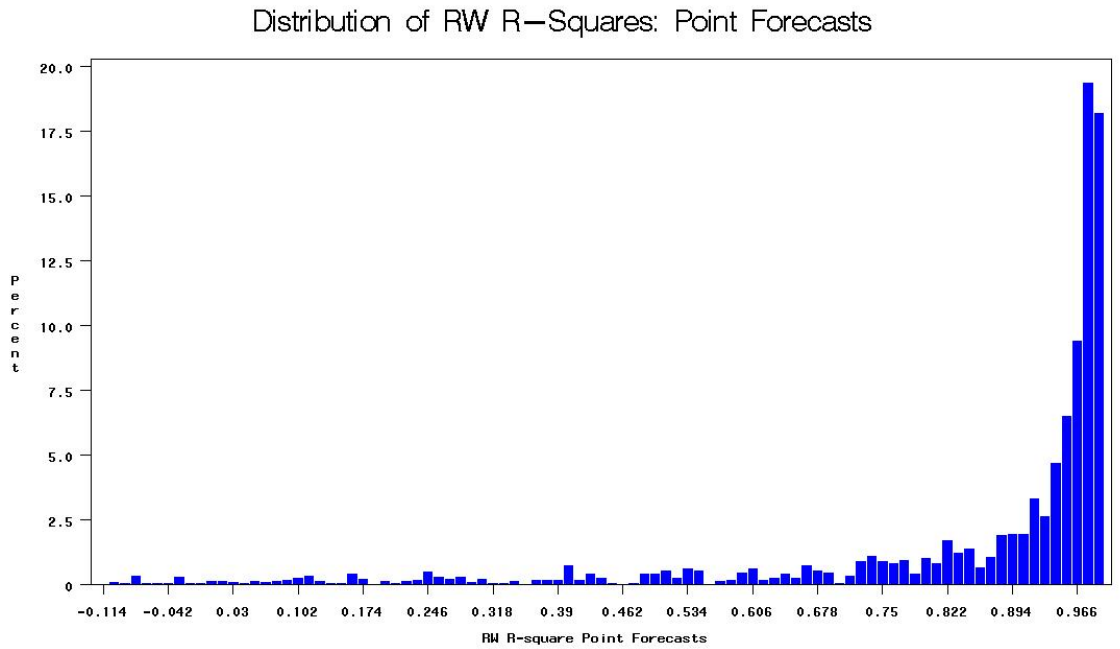
Figure 5.1: Rw R-squared values for the AR(1) model.

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, when $|\alpha_0 - c|$ is small as well. These are the cases where the relationship between y_{t-1} and y_t is most obviously nonlinear (Figure 5.2).

In a very few cases (approximately 20 of the 4800 parameter combinations), the AR(1) model had better out-of-sample performance than the AR-NN. However, these were parameter combinations where none of the models predicted well. The out of sample RW R-squares across all types of models (AR(1), AR-NN (BS) and AR-NN (P)) for this group ranged between 0.02

Table 5.2: Although it is recommended that a bootstrap approach be used to stabilize AR-NN predictions, the point forecasts have better overall statistics and a narrower range. The bootstrap predictions have a potential for disastrous performance.





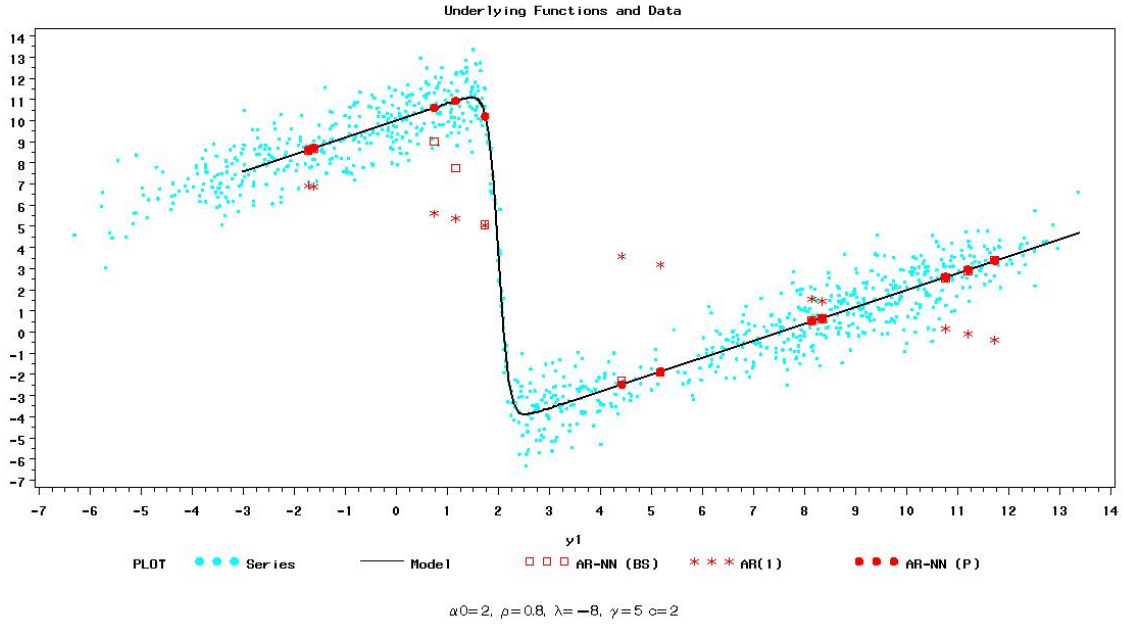


Figure 5.2: The relationship between y_t and y_{t-1} for the parameter combinations where there is the most difference between the linear forecast and the AR-NN forecast is where the relationship is the most obviously nonlinear.

and 0.2 (not shown).

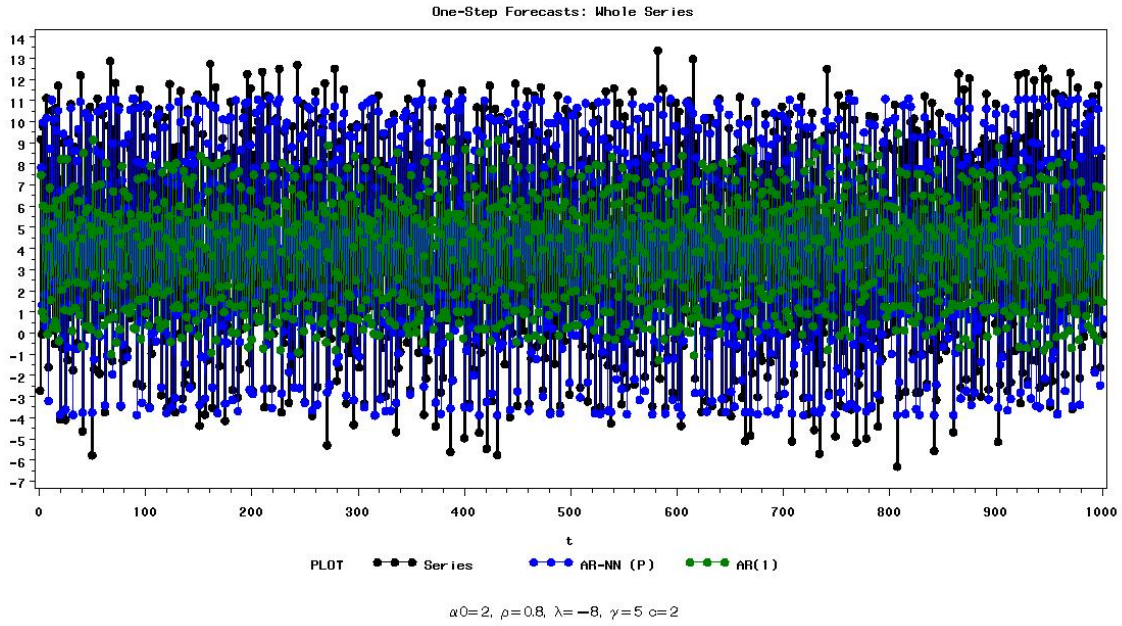


Figure 5.3: The AR(1) forecast tends to stay closer to the series mean than the AR-NN model. This means it is less able to adjust to dips and spikes that the AR-NN can predict rather well. For the graph above, the AR-NN model performs much better (in terms of RW R-squared) than the linear model.

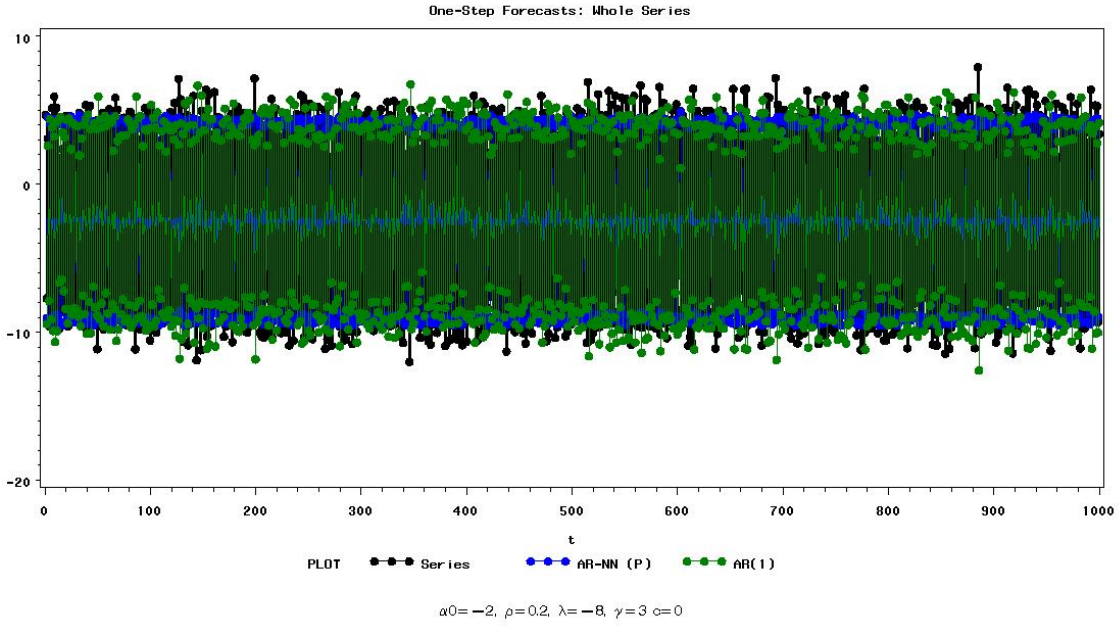


Figure 5.4: There are parameter combinations where most of the variability in the series can be explained by an AR(1). These were cases where λ was small in absolute value. In general, values for λ that were large in absolute value and negative generated series where an AR(1) fit poorly and the AR-NN model produced high RW R-squareds.

Table 5.3: 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.

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

5.2 Prediction At Extremes

While the AR(1) model did not out-perform the AR-NN, from the histogram (Figure 5.1) we can see 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 5.3, 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.

Table 5.4: The RW R-squares for the 12-step-ahead forecasts were much worse than the one-step forecasts.

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

5.3 Long Horizon Forecasts

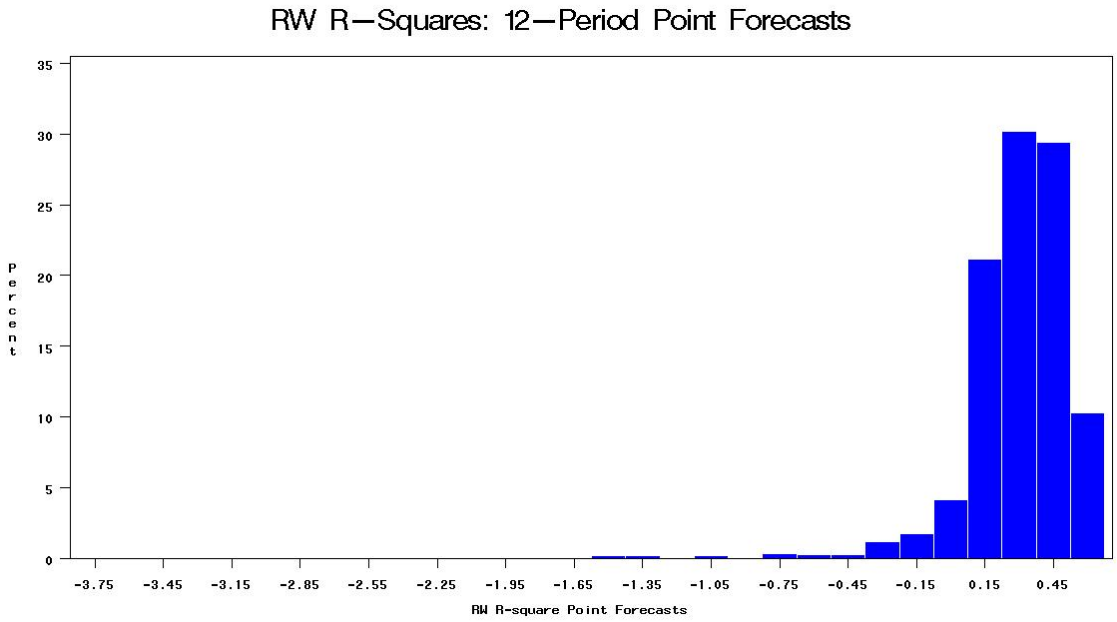
It has been hypothesized that even when a nonlinear time series model is not impressive for short forecasts, they could perform better at a long forecast horizons [16] In our study, it appears that the opposite has happened. Recall that a test sample of 12 time periods was withheld from each generated series. We now perform a 12-period forecast for both the estimated AR-NN and the estimated AR(1).

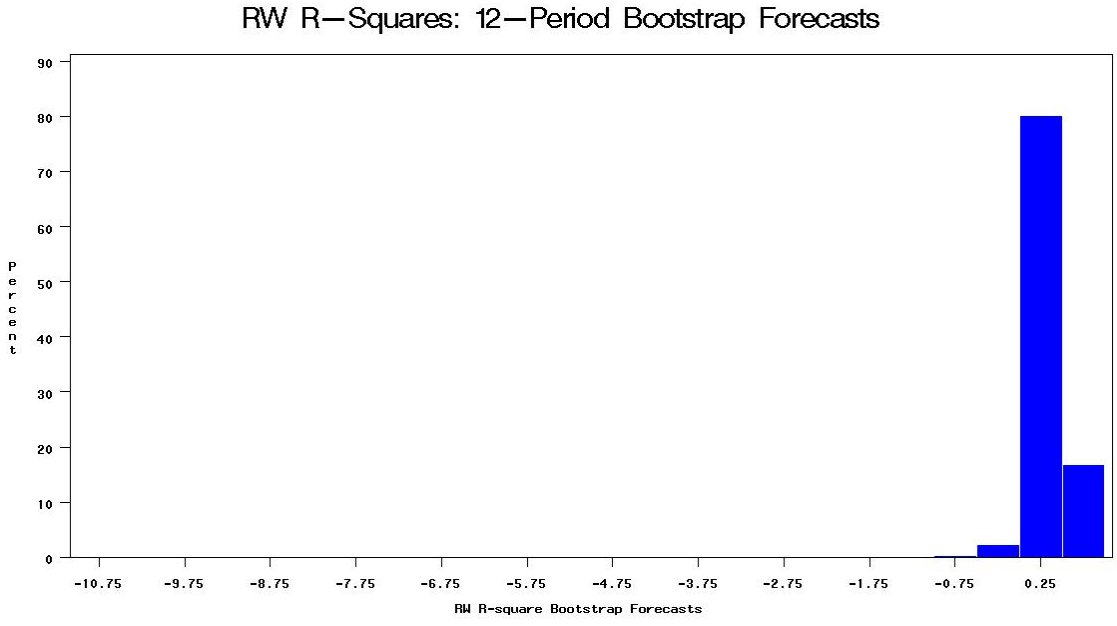
The 12-period forecasts were much worse, as measured by RW R-squared, than the one-step-ahead forecasts. For these long forecast horizons, however, the boot strap forecasts were better than the point forecasts, with a higher mean and median, though the difference could be explained by random variation. (Table 5.4)

Examining the underlying model function and the generated data may shed some light on why the long horizon forecasts are performing so poorly. Figure 5.6 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 [18] and in Section 2.3 of this thesis. 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

Table 5.5: The RW R-squares for the 12-period bootstrap forecasts were negative in many cases, indicating that the random walk model was a better fit. The RW R-squares were slightly better for the 12-step bootstrap forecasts, though the bootstrap forecasts still had some cases where the performance was very bad (less than -10).





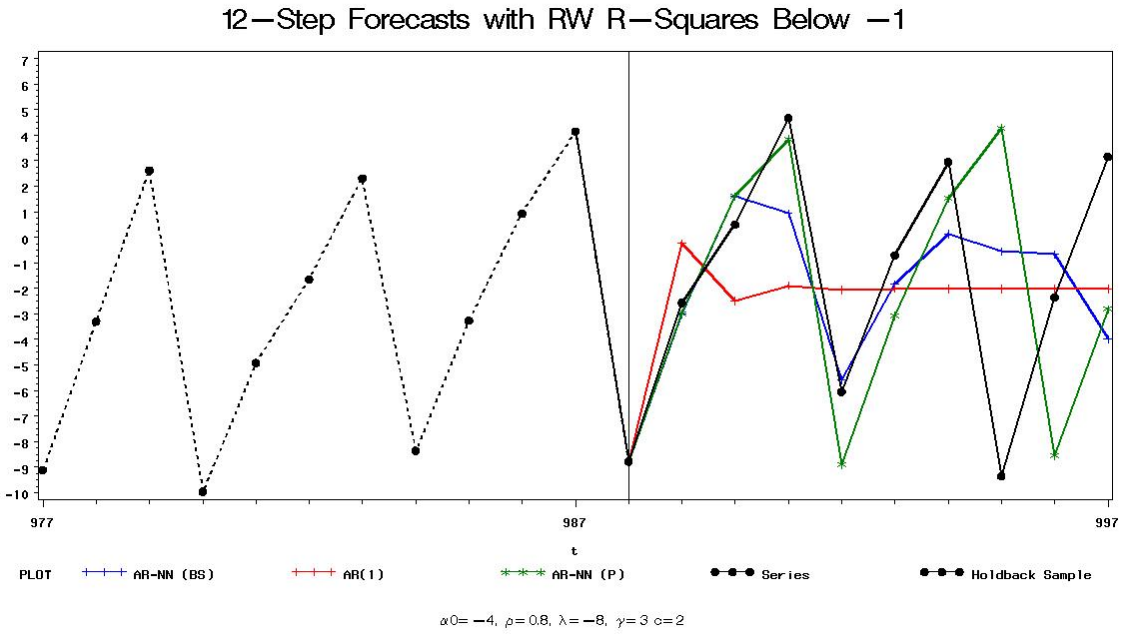


Figure 5.5: The AR-NN forecast was near the actual series for the first few steps but then often started moving in the opposite direction. RW R-squares for these models were often very negative, below -1.

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). 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 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 numerator in equation (5.1)and therefore a small RW R-squared.

Meanwhile, the bootstrap estimates are produced by adding random noise 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. On one hand, as we see in Section 3.3, there must be enough random variation in the data to allow the series to visit all necessary elements

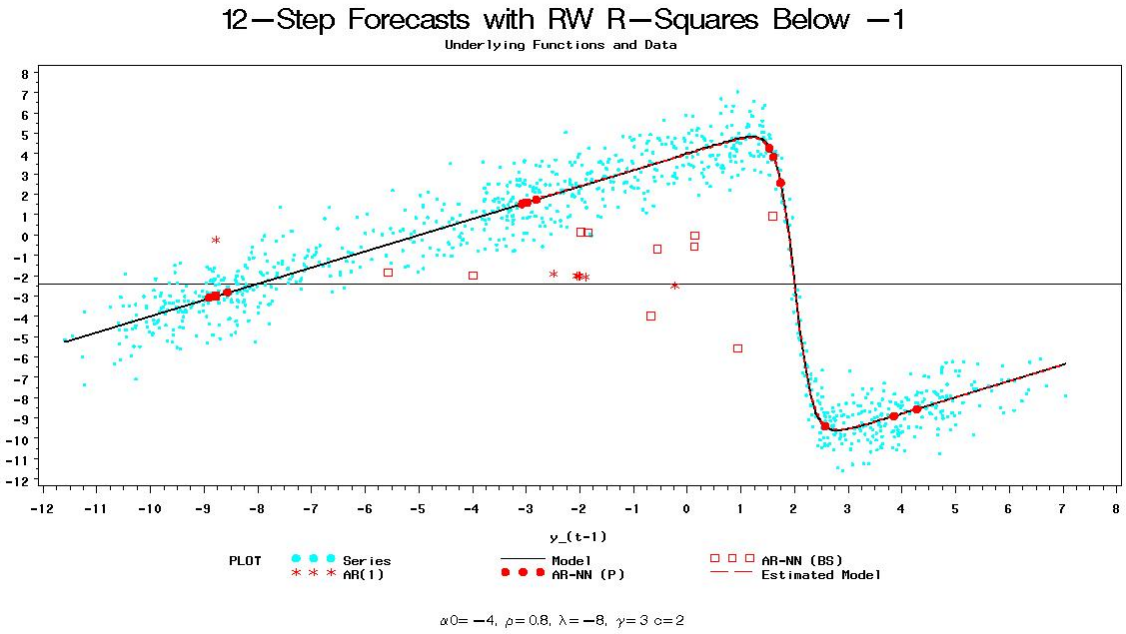


Figure 5.6: For this model, the parameter estimates were almost exactly equal to the true parameter values, yet the RW R-squares were very low, less than -1. This is because the series oscillates between extremes, and the noise term can cause the series to become out of phase with the forecast, so the forecast is moving in the opposite direction of the actual series.

of the solution space, yet the noise must not be the strongest underlying feature in the model.

5.4 Forecast Performance on Data Simulated with the Logistic Distribution

In Section 4.6, 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 chapter 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 5.6 and 5.7, 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 very poor when the series is generated using a heavy-tailed distribution. This is not surprising. Assuming a heavy-tailed

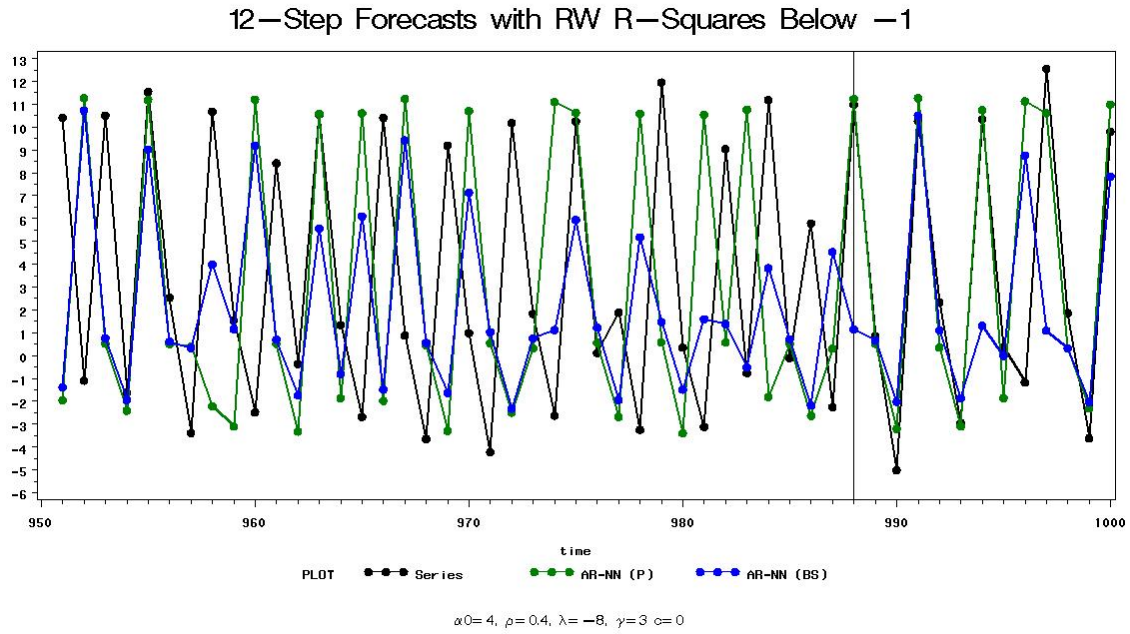


Figure 5.7: For this model, the long-horizon forecast often becomes out of phase with the series. This leads to very poor measures of forecast performance by RW R-squared.

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 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 Chapter 4, that the distributional assumptions are very important in nonlinear models. A small change from a symmetric 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.

Table 5.6: 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.

Descriptive Statistics of RW R-squared: One Step 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 5.7: 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.

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

5.5 Conclusions

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, using both a point forecast and a bootstrap forecast as suggested in [17]. We find that the point forecast outperforms the bootstrap model at short and long horizons. Neither model, however, 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 also give the model a performance "head start". In the next chapter, 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.

Chapter 6

Starting Value Selection

The AR-NN is, at base, a nonlinear model, and the estimates produced for a nonlinear model 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 9600 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. This is not, however, the case. In [9] it is shown that training a neural network model 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 on the ρ parameter actually produces worse estimates than simply choosing a value in the middle of the search space because the additional points increase the number of opportunities for the routine to find a local minimum rather than the global minimum of the objective function. 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 Estimation and Results

We assume that the architecture is known and use the following steps to determine start values for a series:

1. Calculate the overall mean 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 sign($\hat{\rho}$).

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

4. Set the starting value for ρ to 0.5.

The mean is used as an approximation for the center of the data. By our re-parameterization in (1.4), we see that it is possible to re-scale the series so that the location parameters for the series are in terms of their scaled distance from the mean. We have further shown in 4.2 that the distance between the location parameters helps determine whether the series will converge to an attraction point where the long-run behavior is nonlinear. Since we have not re-centered the data around the mean, (i.e. we have not subtracted μ_y from $y_t \ \forall \ t$), setting the location parameters equal to the sample mean is equivalent to setting them equal to each other and setting the location parameters in (1.4) equal to 0.

In Section 3.3, we show that λ is the practical limit for the series deviation from the location parameter α_0, and so the starting value for λ is the maximum deviation from the mean. The sign of γ is allowed to determine the sign of the slope of the activation function (since λ is positive), and ρ is simply given a "middle" starting value of 0.5.

We consider a model to be "converged" if the nonlinear fitting routine reports convergence. In addition to convergence, we investigate whether the model produced parameter estimates that are reasonable, i.e. where all parameters are less than 90 in absolute value. The Marquardt method is used as the fitting routine. Table shows the convergence rates for this set of starting values. For about 70% of our 96,160 runs (4808×20), the fitting routine reported convergence, and the parameter estimates were reasonable.

While the results from the initial fitting routine are good, they are not stellar, and an adjustment to the starting value routine might be in order. Recall that the nonlinear function does not necessarily produce data that is distributed symmetrically around the mean. It is quite

Table 6.1: When the mean is used as the starting value for the location parameters, approximately 70% of the runs result in convergence with reasonable parameter estimates.

Convergence Rates over 20 Runs
When Mean is Used as Starting Value For α_0 and c
Reduced Simulation

	Parameter Estimates		
Converged	Bad Parameter Est.	11,778	12.3%
	Good Parameter Est	68,202	70.9%
Did Not Converge	Bad Parameter Est	553	0.6%
	Good Parameter Est	15,647	16.3%
		96, 160	100.0%

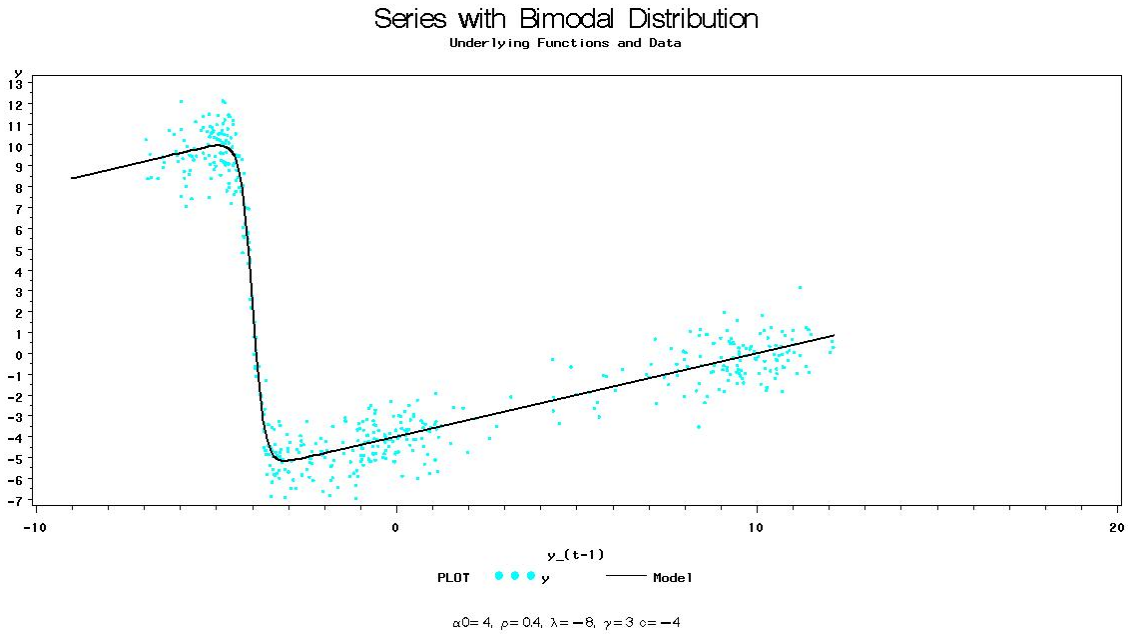
possible, in fact, to produce data that is heavily skewed or bimodal (Table 6.2). In both cases, the mean is a poor measure of the center of the distribution and therefore cannot be expected to be a reasonable starting value for the location parameter of of the hyperbolic tangent function. We therefore consider the same starting value routine as before, except that the median is chosen as the starting value for both location parameters, and the starting value for λ is the maximum deviation from the median.

When the median is used as the start value for the location parameter, the results improve significantly, so much that our starting value routine produces convergence almost as often as using the true parameter values as the starting values. Not only that, but the 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 [19]. 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.

In the previous chapter, we found that models estimated with Levenberg-Marquardt produced better forecasts than models estimated with Gauss-Newton. The Levenberg-Marquardt estimation method is a compromise between Gauss-Newton and the gradient/steepest descent method, with a parameter (called the Levenberg parameter) that controls the mixing between the two. Levenberg-Marquardt is frequently used when there are dependencies among the derivatives, which can often be the case in neural network models, as shown in [4] and in Chapter 3 of this thesis. In the literature review, we found that most researchers used Levenberg-

Table 6.2: Although the noise term ε_t is normally distributed, the nonlinear nature of the deterministic potion of the equation will often as not produce a series that is skewed or bimodal.



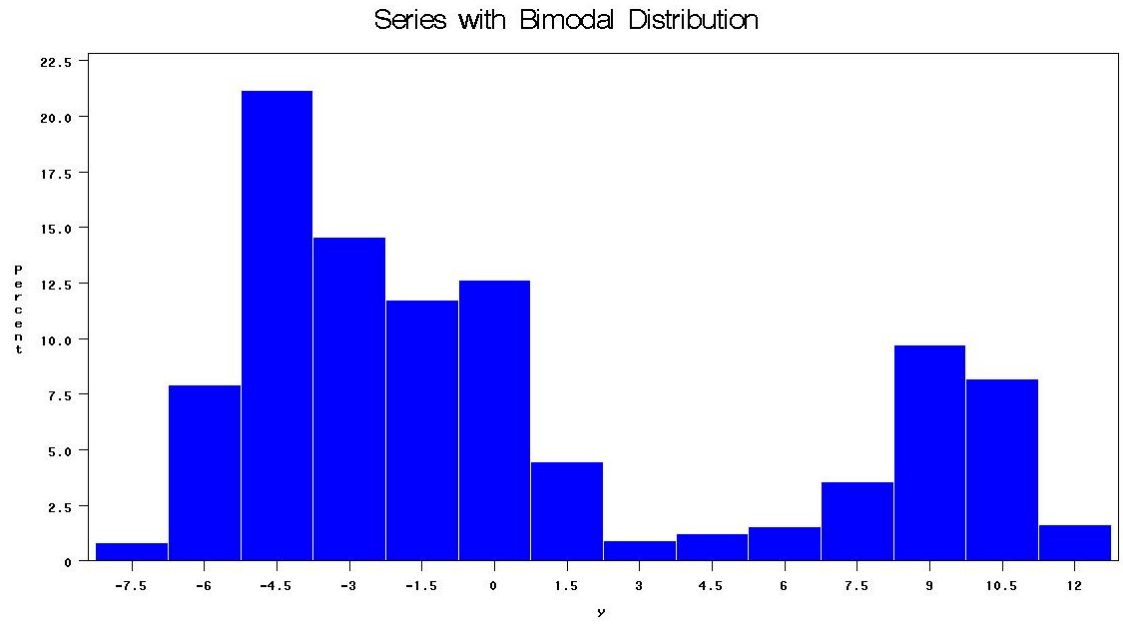


Table 6.3: Of all the starting value routines that were tried, the Marquardt estimation method using the median as the starting value for the location parameters was obviously the best, arguably even better than using the true values of the parameter estimates.

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 6.4: Grid searching for the starting values for ρ produced worse results than using a point value. This can be because the objective function is not smooth; trying more options at the beginning increases the chances that the estimation routine will find a local minimum rather than the global minimum of the objective function.

Convergence Rates over 20 Runs
Grid Search on ρ verses 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 6.5: Gauss-Newton did not perform as well as Levenberg-Marquardt, which is consistent with results from previous chapters.

Convergence Rates over 20 Runs
Levenberg-Marquardt verses Gauss-Newton
Reduced Simulation

		Estimation Method			
		Levenberg-Marquardt		Gauss-Newton	
Converged	Bad Parameter Est.	6,018	6.3%	11,778	12.3%
	Good Parameter Est	81,974	85.3%	68,202	70.9%
Did Not Converge	Bad Parameter Est	365	0.4%	533	0.6%
	Good Parameter Est	7,803	8.1%	15,647	16.3%
		96,160	100.0%	96,160	100.0%

Marquardt as their estimation method, when estimation was mentioned at all [10],[17]. Gauss-Newton is the default method for many commercial software packages, and therefore we also consider Gauss-Newton estimation. Results are shown in Table 6.5.

6.2 Convergence vs Performance

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. For comparison, we start by calculating the statistics for the MSE when the true parameter values are used as the starting values. The statistics are shown in Table 6.6. When the true parameter values are used as the starting value, no matter what estimation routine is used, the resulting MSE is approximately 1, i.e. the expected MSE with additive noise that is distributed $N(0, 1)$.

As in the forecasting example, we produce 20 sequences of errors, which we use to generate 20 different series for each parameter combination. We then examine the MSE from a model estimated when we use the starting value routine under study and compare it to the MSE from

Table 6.6: 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 $N(0, 1)$.

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

a model produced when the true starting values are used as parameter values. Table 6.7 reports descriptive statistics for the difference between the MSE's for each parameter combination.

As with the forecasting study, we exclude models where the parameter estimates were unreasonable ($|\rho| > 1$,$|\alpha_0| > 90, |\lambda| > 90$, or $|c| > 90$. As in the forecasting example, this excluded approximately 3% of the replicates.

The results from Table 6.7 illustrate two important principles for fitting nonlinear models. First, model convergence is subjective and not the only measure of the fitting routine's performance. Though the Levenberg-Marquardt estimation method with the median of y_t as a starting value for the location parameters "converged" more than any other starting value routine, including using the true parameter values, the resulting model rarely out-performed a model estimated using the true parameter values as start values. The Gauss-Newton estimation method, which appeared very poor when measured using convergence rate, was competitive with the Levenberg-Marquardt method when measured by model MSE.

Second, the large maximum difference in model MSE's, even when the insanity filter is applied, show 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 [9] and [4], is a step that cannot be ignored.

Table 6.7: 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. The Levenberg-Marquardt estimation method with the median as a starting value for the location parameters still performs best, and the same method with a grid search on ρ still performs worst. Gauss-Newton, however, is a clear contender when performance is measured by Mean-Squared Error.

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
Marquardt, Mean	8.17	0.21	41.54	-0.00	2617.86
Marquardt, Median	4.94	0.00	38.92	-0.00	2693.94
Gauss, Median	5.55	0.00	39.05	-0.00	2693.94
Marquardt, Median, Grid	30.17	1.18	81.20	-0.00	3976.46

6.3 Conclusions

Our results show that data-based starting values can produce estimates that are as good as or better than using the true parameters as starting values. Further, we see that grid-searching, far from a cure-all for parameter estimation, can actually produce worse results than an arbitrary point starting value for ρ. The parameter ρ has a known limit of $(-1, 1)$. It is difficult to argue that a grid search would be successful for a parameter without a known range when it does not even work for parameters with a known range.

For our simple model, the best estimation practices were as follows:

1. Use the median of y_t as the starting value for both α_0 and c.
2. Use $\max(|y_t - \text{med}(y_t)|)$ as the starting value for λ
3. Set $\gamma = \text{sign}(\hat{a}_1)$ where $\hat{a}_1$ is the slope parameter from an AR(1) model.
4. Let the starting value for ρ be 0.5, an arbitrary "middle" value for ρ.
5. Levenberg-Marquardt was the best estimation routine of those considered, which is not surprising given the results in Chapter 3.

We have seen why the median is a better starting value: it is a better measure for the middle of the distribution, and our understanding of attraction points explains why we are more likely to produce good parameter estimates when our location parameters are in the center of the distribution. The reasons for the poor performance of the grid search is less obvious. Intuitively, we would believe that more choices for the starting values of ρ would lead us to better answers for ρ and the rest of the parameters. The objective function for the neural network model, however, is not smooth, and the same problems that make good starting values important make trying many different starting values risky. Local minimums will cause the nonlinear fitting routine to stop in parts of the parameter space that are not optimal.

6.4 Recommendations for Starting Values

The results above show that starting values matter. Even a slight change from using the series mean to using the series median for the location parameters produced a major gain in the performance statistics. Grid searching on even one parameter produced worse results than proceeding from a sensible point starting value.

In general, it appears that the median works best as a measure of location and that the maximum deviation from the median works well as a starting value for λ. In absence of other information, starting the slope parameters of the function (γ and ρ) at theoretical medians for the parameters (in our case 1 and 0.5 respectively) works well.

We also see that no starting value routine works perfectly, not even using the true parameter values as starting values. In approximately 8% of the cases the model fails to converge no matter where we start. Some of these convergence failures are sensible: the data are clustered far away from the location parameters of the hyperbolic tangent function or show few nonlinear properties. Others simply show how difficult it is to estimate the parameters of a neural network model, even when the architecture is known.

Chapter 7

An Analysis of Damage-Sensor Data

Previous chapters have laid a groundwork for the practical aspects of estimating a neural network model using existing theory and appropriate extensions. Now, we show an application of these methods to sensor data used to detect structural damage in steel support beams for bridges. It is shown that even our simple AR-NN can be applied to a time series with definite nonlinear properties and achieve performance gains over a similar linear model.

7.1 Automated Bridge Monitoring

Currently, bridges are inspected visually, and inspectors must either rely on past knowledge of damage or finding damage that is visible to the eye. Researchers at the Structures Laboratory at North Carolina State University are working on methods to continuously monitor bridges through the use of electronic sensors that gather data on vibrations in the bridge's structure. They use an experimental setup where a steel beam is damaged and pressure from bridge traffic is simulated using a hydraulic cylinder. The goal is to identify vibration patterns associated with the damaged areas.

A forty-foot steel beam is set up in two continuous spans and supported by three pinned and roller supports, simulating the structure of a bridge. A hydraulic cylinder is used to exert downward pressure on the beam, and the resulting vibrations are measured through fifteen sensors placed at regular intervals. Measurements are taken every $3e^{-6}$ seconds. The vibration measurements for one sensor for time $t = 0$ through $t = 16$ (where time is measured in seconds) is shown in Figure 7.1.

At time 0, the pressure from the hydraulic cylinder is applied, which is responsible for the the large vibrations at the beginning of the series. After the initial shock, the series moves to a steady state, where the vibrations hover around the mean of the series (which is approximately 0), with periods of high fluctuations around the mean. These data are clearly nonlinear.

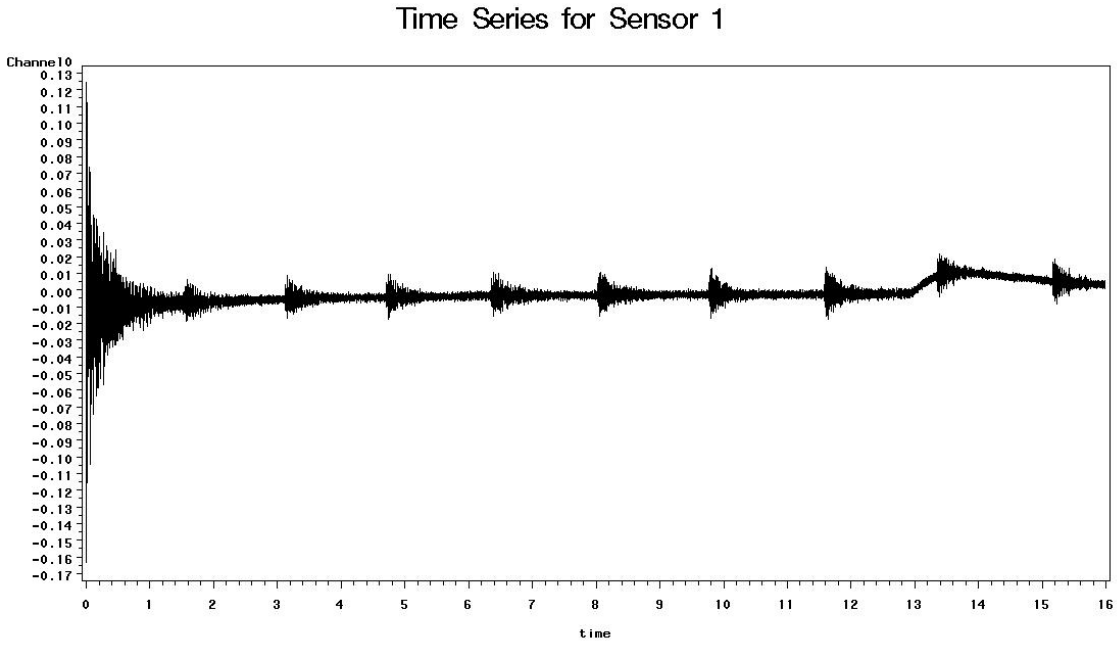


Figure 7.1: After the initial shock when the hydraulic cylinder is applied, the time series for the first sensor settles to a steady state with definite nonlinear features.

We estimate an autoregressive neural network (AR-NN) with one lag and one hidden unit. The parameters are estimated using the first 30,000 observations in the series, and the last 1600 are reserved as a holdout sample. We use the methods developed in Chapter 6 to calculate starting values for the model parameters. The resulting parameter estimates and model fit statistics are shown in Tables 7.1 and 7.2.

The slope parameters (ρ, λ, and γ) are positive. The confidence limits for the slope parameters of the hyperbolic tangent function (γ) and the weight (λ) do not cross 0, indicating that the hidden unit makes a non-trivial contribution in explaining the relationship between the current value for the vibration and the first lag.

The AR-NN performs well in both in-sample and out of sample forecasts. Figure 7.2 shows the point estimates for the AR-NN overlaid on the same graph as the actual series. The point estimates produce good forecasts, even at the beginning where the fluctuations in the values are extreme compared to the rest of the series.

One concern in time series forecasting is that the model may predict the data well overall but fail to catch extreme values, or even be a period behind the actual series. Figure 7.3 gives a closer look at the point estimates for the AR-NN verses the actual series. The AR-NN follows the actual series well, even at extreme values.

For comparison, we also estimate an AR(1) model from the same data. The AR(1) model

Table 7.1: Summary fit statistics for the AR-NN on data from the first sensor.

Parameter Estimates for the AR-NN
$y_t = \alpha_0 + \rho y_{t-1} + \lambda \tanh \gamma(y_{t-1} - c)$

Source	DF	Sum of Squares	Mean Square	Approx F Value	Pr > F
Model	4	1.2119	0.3030	33331.4	<.0001
Error	30368	0.2760	9.09E-6		
Corrected Total	30372	1.4879			

R-squared: 0.81 RW R-squared: 0.9927

Table 7.2: The confidence limit for the slope parameter and the weight of the hyperbolic tangent function(γ and λ and γ, respectively) are positive, and their confidence limits do not cross 0.

Parameter Estimates for AR-NN

Parameter	Estimate	Std Error	95% Confidence Limits	
α_0	-0.00243	0.000265	-0.00295	-0.00191
ρ	0.6509	0.0165	0.6186	0.6832
λ	0.0101	0.00132	0.00756	0.0127
γ	30.6051	2.8176	25.0823	36.1279
c	-0.00767	0.000703	-0.00905	-0.00629

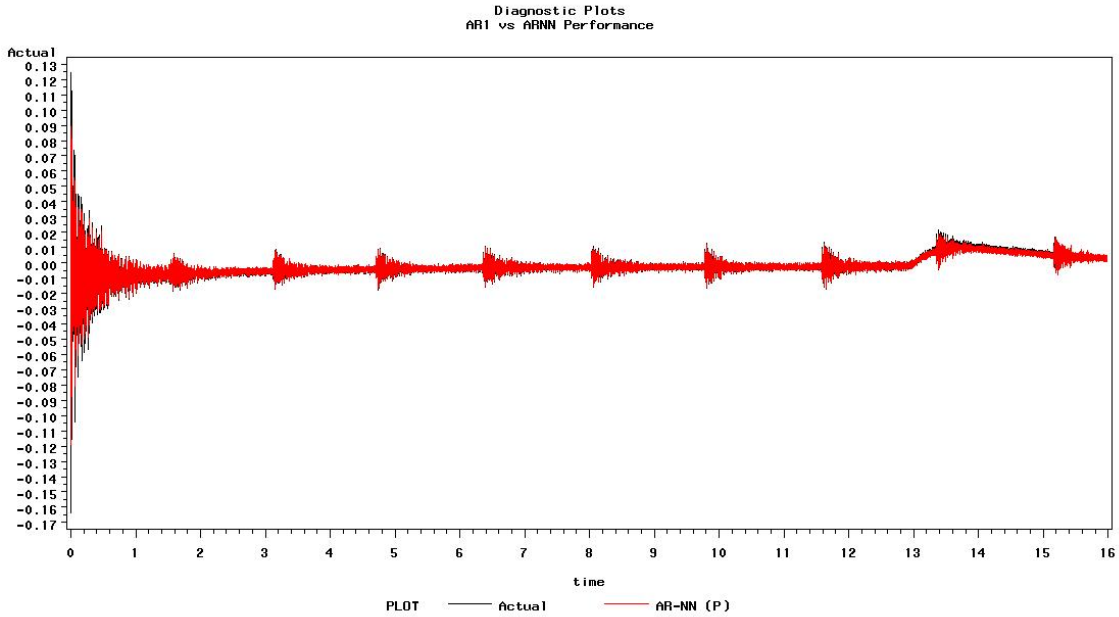


Figure 7.2: The point estimates for the autoregressive neural network follow the extremes in the holdout sample better than the AR(1). This is reflected by the difference in the out-of-sample RW R-squared statistics: 0.6140 for the AR(1) and 0.63796 for the AR-NN

produces an estimate for ρ that is approximately 0.89, though we are able to reject unit roots because of the large sample size.

The in-sample R-squared and RW R-squared statistics are slightly better for the AR-NN model, but only slightly. It is therefore worth asking whether the AR-NN model produces enough performance gain to justify its added complexity. Figure 7.4 shows a comparison of the AR-NN and the AR(1) plotted on the same graph as the actual series and the holdout sample. The training set (the part of the series used to estimate the parameters) is shown in gray, and the holdout sample is shown in black. The AR-NN has a slight edge over the AR(1) in out-of-sample performance, following the extreme values in the holdout sample slightly better than the AR(1). This is backed up by the out-of sample RW R-squared values, which were 0.6380 for the AR-NN and 0.6140 for the AR(1).

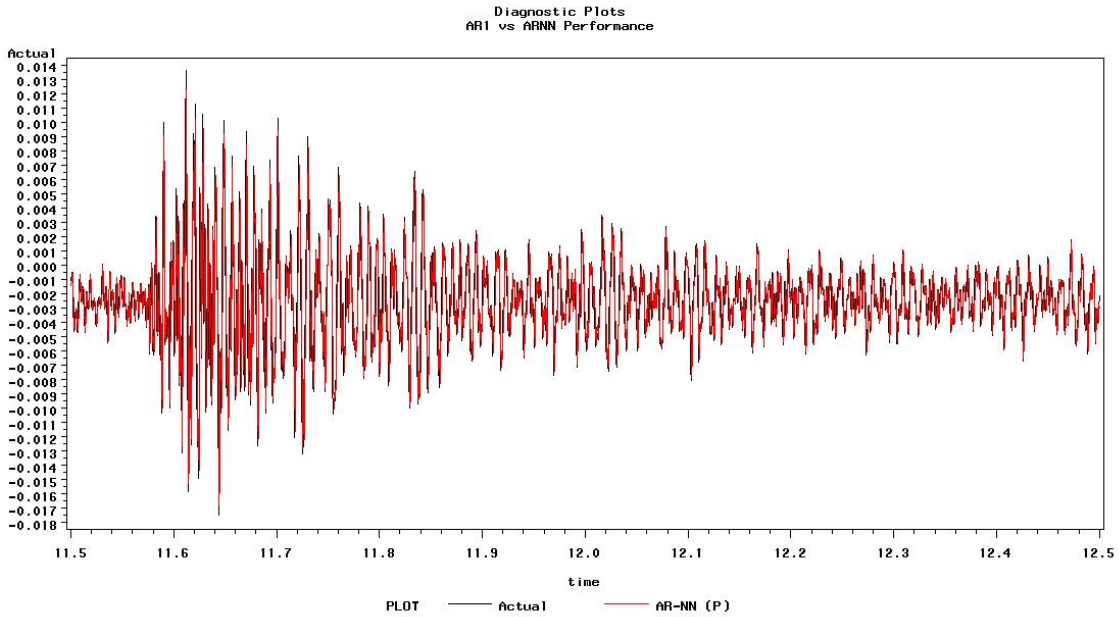


Figure 7.3: The AR-NN follows the series quite well, despite extreme changes in the series values.

Table 7.3: The summary statistics for the AR(1) on data from the first sensor.

Parameter Estimates for the AR(1)

Source	DF	Sum of Squares	Mean Square	Approx F Value	Pr > F
Model	1	1.20445	1.20445	129040	<.0001
Error	30371	0.28348	0.00000933		
Corrected Total	30372	1.48793			

R-squared: 0.81 RW R-squared: 0.9929

Table 7.4: The estimate for the slope parameter of the AR(1) is near 1.

Parameter Estimates for AR(1)

Parameter Variable	DF	Estimate	Standard Error	t Value	Pr > —t—
Intercept	1	-0.00027452	0.00001866	-14.71	<.0001
ρ	1	0.89486	0.00249	359.22	<.0001

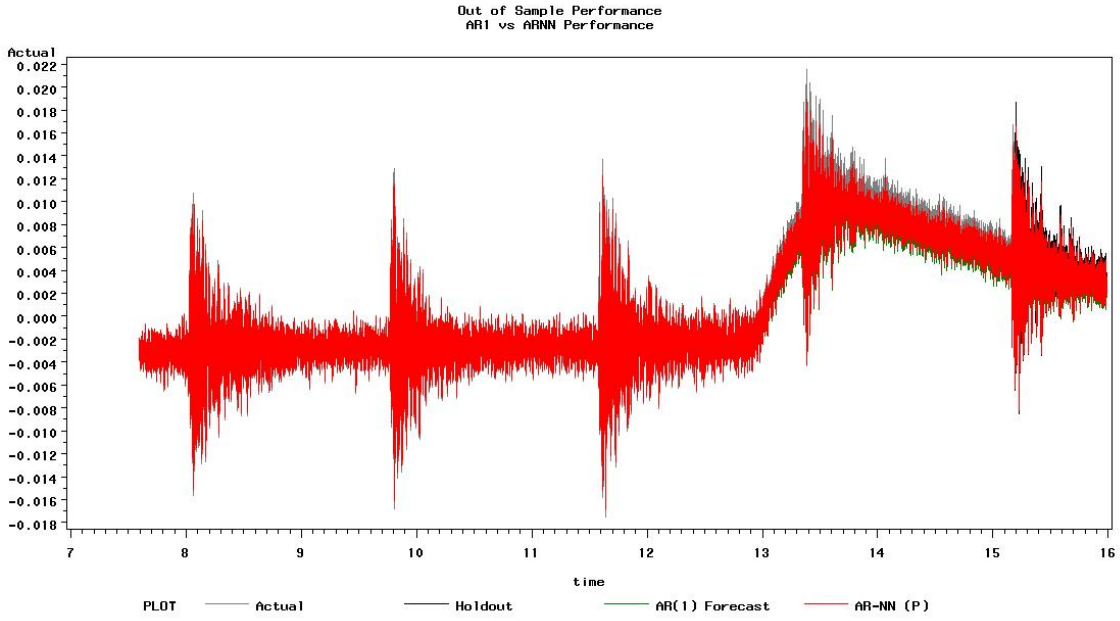


Figure 7.4: The point estimates for the autoregressive neural network follow the extremes in the holdout sample better than the AR(1). This is reflected by the difference in the out-of-sample RW R-squared statistics: 0.6140 for the AR(1) and 0.63796 for the AR-NN

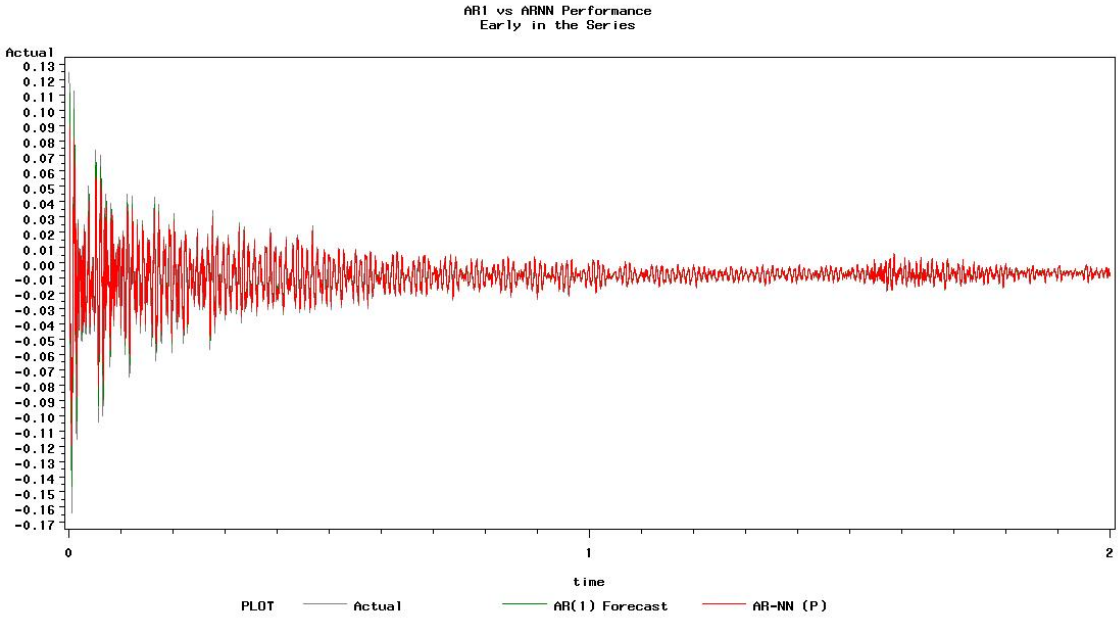


Figure 7.5: At the beginning of the series, the AR-NN does a particularly good job of predicting the series. The early features of the data (before the steady state is reached) seem to be important for the AR-NN to perform better than the AR(1).

7.2 Conclusions

The problems with estimating the model parameters found in simulation encourage us to proceed with caution in applying the AR-NN to actual data. We see here that there are nonlinear series for which even this relatively simple AR-NN can forecast the series behavior well. The model's poor performance on the de-trended version of the same series, and its poor performance when the early periods are eliminated, indicate that the results in [24] should not be followed blindly. It seems that de-trending data can remove the nonlinear features that the AR-NN is designed to estimate. This is also an encouraging results for the AR-NN model. Fluctuations that look like trends or seasonal variations that are actually due to an underlying nonlinearity in the time series may be modeled more accurately with the AR-NN. This is particularly true in physical phenomena like vibrations in a steel beam.

The large amount of data available in this series is also important. The nonlinear features for the vibration data, while apparently present, are slight. Without a large number of observations, it would be difficult to model them with any accuracy. For applications were data are collected continuously by an automated system, like our sensors, this model might be useful for estimating hard-to-identify nonlinear relationships.

Chapter 8

Conclusions and Suggestions for Future Research

This research is not intended to be the "last word" on neural network modeling but rather a collection of best practices and stepping stones to further research in this field. In this thesis, 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 from our simulations, including:

1. The location of the neural network attraction point(s) is important in determining long-run behavior of the network. If the attraction point is not near the location parameter for the hyperbolic tangent function (called c 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).

2. The boundedness of the activation function when the series is far from c and the fact that it is nearly a straight line when data from the series lie close to c causes linear dependencies among the first derivatives of the series. This can cause numerical problems in the model Hessian and makes parameter estimation difficult. It is suggested that analysis use the Levenberg-Marquardt estimation technique, as it is better-suited to models which have these problems.

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. Our results show the opposite: The neural network models we studied had very poor performance at long forecast horizons, even when the true network architecture was known.

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. Using the median instead of the mean for the starting values of the location parameter, which seemed a relatively minor change, produced large gains in the number of models that converged. A fitting routine for these models is not complete without a way to select starting values.

6. Even the simple neural network model studied here can have significant performance gains over an AR(1) when applied to data with strong nonlinear features. Neural networks can be a powerful tool for understanding nonlinear time series and forecasting data with nonlinear relationships among the lags.

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 [9] 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 [4], 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 [19], 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 most-linear behavior among the lags. In [24], it was shown that neural networks produced forecasts that were orders of magnitude better when the correct de-trending and de-seasoning 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 [19] for the parameters to be estimable, none of our models meet the conditions set in [18] for an attraction

point that is globally and exponentially stable in the large. Many of our series had multiple attraction points. Some have one unstable attraction point. Without a stable 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 2.3 between the mean for a stationary linear time series and a nonlinear series with a stable 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 points. 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 noise term can cause the series to move from one cluster to another. Finally, if the series does not have any stable attraction point, it will have a limit cycle. 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. In a series like this, forecasts 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.

8.1 Suggestions for Future Research

An obvious extension of this research is to develop results for more than one hidden unit and/or lag. Every added lag, however, adds at least two more parameters to the model, and every additional hidden unit adds at least five parameters to the model. For example, simulating a neural network with two hidden units using this design would mean considering 48,000 parameter combinations. Because of the unknown and nonlinear relationship between the simulated values of the parameters, it is difficult to reduce the simulation appropriately. Some of the parameter combinations could be eliminated with the techniques used in Chapter 4, but even those relatively simple calculations would not be trivial.

In [11], conditions are set where we can expect a neural network model to have a global and exponentially stable attractor. It would be interesting to perform the simulations in this thesis for parameter combinations that meet these conditions. Neural network forecasts for data simulated under these conditions may be more likely to out-perform linear forecasts estimated

from the same data. Because the ranges of the parameters allowed in [11] are much smaller than those studied here, it may also be possible to consider multiple hidden units and lags under these conditions.

Though [6] 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 better-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. 14.

[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, Jagannathan (2006) *Neural Network Control of Nonlinear Discrete-Time Systems*. Taylor and Francis.

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

[15] Sharda, R. and Patil, R.B., (1992) "Connectionist approach to time series prediction: An empirical test." *Journal of Intelligent Manufacturing* 3(5) pp. 317323.

[16] 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

[17] 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

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

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

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

[21] 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

[22] 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. 3562

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

[24] 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