WORKING PAPER

No. 00-02MIS November 2000

"Exploring the Need for Node Biases in Time Series Forecasting with Artificial Neural Networks"

By

Douglas M. Kline & Gerald Kohers
Dept. of Management & Marketing
Sam Houston State University
P.O. Box 2056
Huntsville, TX 77341 U.S.A.

The Working Papers series is published periodically by the Center for Business and Economic Development at Sam Houston State University, Huntsville, Texas. The series includes papers by members of the faculty of the College of Business Administration reporting on research progress. Copies are distributed to friends of the College of Business Administration. Additional copies may be obtained by contacting the Center for Business and Economic Development.

Exploring the Need for Node Biases in Time Series Forecasting with Artificial Neural Networks

Douglas M. Kline
Sam Houston State University
PO Box 2056
Huntsville, TX 77341
(936) 294-1842
fax: (936) 294-3957
douglaskline@acm.org

Gerald Kohers
Sam Houston State University
PO Box 2056
Huntsville, TX 77341
(936) 294-3883
fax: (936) 294-3957
mis_gk@shsu.edu

Exploring the Need for Node Biases in Artificial Neural Networks for Time Series Forecasting

Time Series Forecasting is an important problem area with applications in every business discipline and in engineering and the sciences. It involves predicting the value of a variable of interest based on historical values of this variable.

Artificial Neural Networks (ANNs) are a promising technique for time series forecasting. It's advantages are 1) the ability to model more complex, nonlinear series 2) no need to pre-specify a model for the data as in traditional methods 3) no assumptions are required of the data (such as normal distribution). In short, it is a general purpose technique which can be applied to arbitrary time series. ANNs have already been shown to equal and surpass traditional methods when there is sufficient data. However, the ANN architecture must be matched to the problem, and there are many choices to be made in the use of ANNs. This has led to inconsistent methodologies and mixed results regarding the efficacy of ANNs.

One choice in defining the ANN for forecasting regards the use of a node bias in the hidden nodes of the NN. Theoretical results show that node biases are not strictly necessary. However, most NN software includes them by default and practitioners routinely include them.

Node biases are important to study for several reasons. First, they are an additional parameter to the forecasting model. ANNs tend to be high in parameters relative to traditional methods, which can cause poor generalizability, especially in many data-poor forecasting tasks. Ideally, we would like to use an *appropriate* number of parameters to the problem at hand. When using node biases on every hidden node and adding hidden nodes to the architecture, we lose fine control over how many parameters are used. In a situation with four inputs and one output, each additional hidden node adds six additional parameters to the model. By controlling individual

hidden node biases, we gain much finer control over the number of parameters in the forecasting model.

Second, node biases actually influence the shape of the logistic activation function, which may affect the model's ability to recognize patterns, and certainly affects the shape of the training error surface. There is no consensus regarding the need for node biases. The theoretical results stating that NNs are universal function approximators do not rely on node biases, but there is no research to show the effect they have. Since node biases affect the training error surface, they may have an impact on the training scheme used to build the NN model. This can significantly impact both the time required to train the NN, and the quality of the resulting forecasts.

This paper uses the widely-studied Airline Data to explore the merits of node bias when forecasting using ANNs. The results should be important to practitioners, as well as those doing research in forecasting or ANNs, or forecasting with ANNs.

A. Literature Review

Among the many areas that Feed-Forward Artificial Neural Networks (ANNs) have been applied to is the area of time series forecasting (Zhang, et al. 1998, Hill, et al., 1996, Faraway & Chatfield, 1998, Zhang & Hu, 1998, Tang, et al., 1991, Tang & Fishwick, 1993, Zhang, et al. 1998). Not only has this technique shown great promise under certain circumstances (Tang & Fishwick, 1993, Tang, et al., 1991, Balkin & Ord, 1998, Zhang & Hu, 1998), but it also has desirable properties as a time series forecasting tool[1], such as the ability to model arbitrary non-linear functions (Hornik, 1991, Hornik, et al., 1989), few a priori assumptions (Cheng & Titterington, 1994), and the ability to generalize.

Unfortunately, the flexibility of neural networks has resulted in substantial variations in applied methodology. This has led to inconsistent results (Tang & Fishwick, 1993), unrealistic expectations, and the tendency to treat neural networks as "blackboxes." Furthermore, there are challenges specific to neural networks that must be addressed.

One of the challenges of using ANNs is that, compared to traditional time-series techniques, there are a relatively large number of parameters. For instance, consider a general model that applies to all forecasting techniques: $y_{t+1} = f(y_t, y_{t-1}, y_{t-2}, \ldots, y_{t-n}, p_1, p_2, \ldots, p_m)$. Here yt+1 represents the one step ahead future value to be forecasted based on the past values n+1 values, y_t, y_{t-1}, etc. The m model parameters, p, are those values that can be adjusted to make the model perform better. In simple traditional techniques, such as moving average and exponential smoothing, there is only one parameter, i.e. m=1. Even in relatively complex traditional techniques, the number of parameters is small, perhaps four or five. Furthermore, in traditional techniques it is simple to increase parameters one at a time to appropriately model the time series, thus "fine-tuning" the model to the time-series.

In general, as m increases, the model becomes more powerful and able to forecast more complex problems. However, the practitioner needs to have a sufficient number of data points in order to accurately estimate the parameters. In situations where there are too many parameters relative to the number of points, the model tends to "memorize" the past patterns and performs poorly on patterns that it has not seen before.

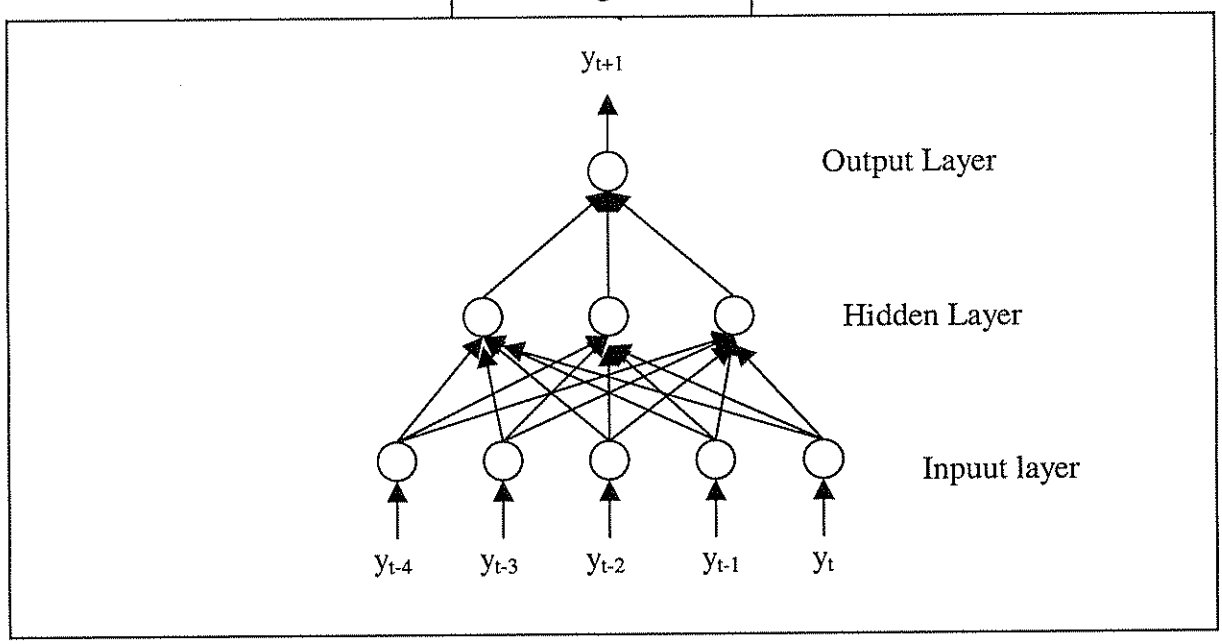
ANNs have two problems concerning parameters: 1) there are a relatively large number of parameters in a ANN model 2) it is difficult to fine-tune the parameters. Consider Figure 1,

---

[1] See Zhang, Patuwo, and Hu, 1998 for a good review of Neural Networks used for forecasting.

which would be a typical architecture for quarterly time series data. The input nodes are the last five quarters of data (the last four quarters plus the same quarter last year). The output node is the forecasted value. There is one parameter per arc (an arc weight), and one parameters for each hidden node (a node bias). So this basic ANN has m=21 parameters. Simplifying the ANN as much as possible (using only one hidden node) there are still m=7 parameters. Adding hidden nodes increases the number of parameters by seven.

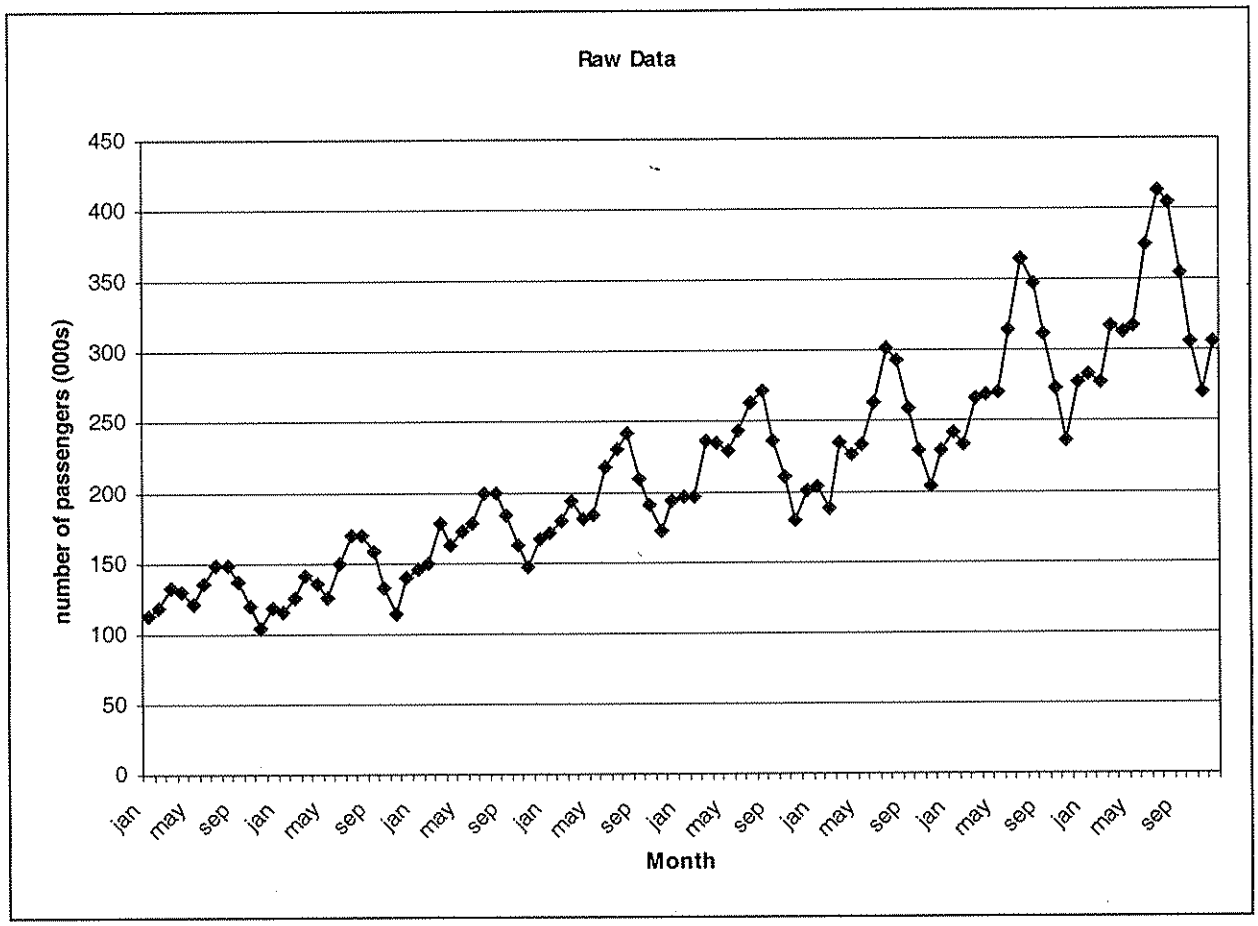
Figure 1



In summary, ANNs show great promise for time series forecasting, but there are problems appropriately tuning the model to the problem. This may, in part, account for the varied degrees of success reported in the literature.

Based on the theoretical research in Neural Networks (Hornik, et al., 1996, and Hornik, 1991), ANNs are universal function approximators under assumptions of sufficient data, training algorithms, and architecture. The theoretical research does not require that node biases be used to

Figure 2



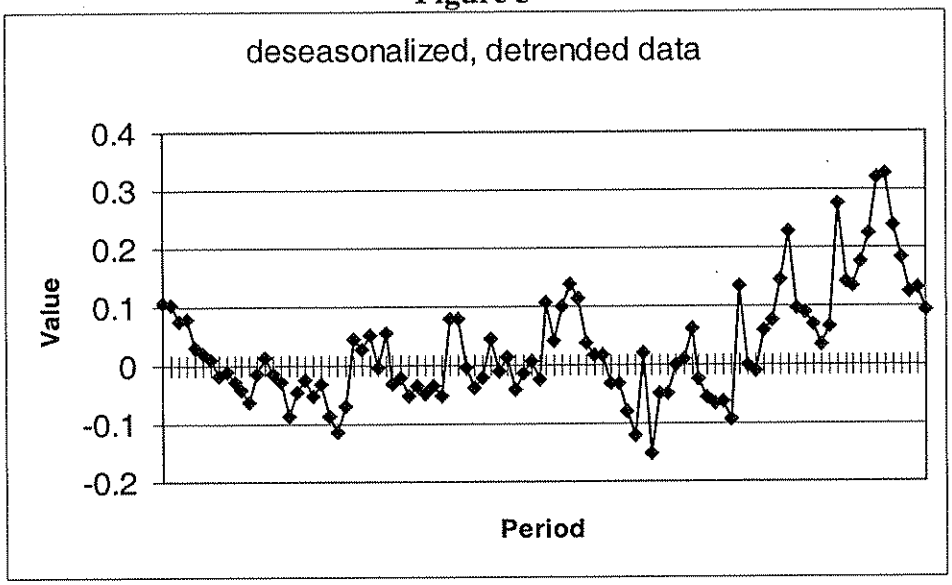
derive this result, even though the conventional use of ANNs includes node biases on every hidden node. By allowing control over the inclusion of node biases, one gains ability to "tune" the architecture by adjusting the number of parameters. Specifically, if the current architecture has five input nodes, one output node, and one hidden node (without node bias), it is possible to

increase the number of parameters in the model by one (by adding the node bias), rather than by seven (six arc weights and one node bias).

II. Methodology

The data to be used is the Airline Data (Box, et al., 1994). It consists of ten years of monthly data displaying monthly seasonality and a general upward trend. It represents the number of international airline passengers from January 1949 to December 1960. Figure 2 shows

Figure 3



the raw data. The data was first de-seasonalized using a multiplicative model, then de-trended using a linear model. Figure 3 shows the de-trended, de-seasonalized data that was presented to the neural network. We chose to de-trend and de-seasonalize, since there is no agreement on how these characteristics of a time series should be handled. The data shows obvious trend and seasonality, and the linear trend and multiplicative seasonality model fit the data well.

To study the effect of node bias, we perform a one-step-ahead forecast using one to four lags. This sets the number of outputs of the NN to one, and the number of inputs to between one and

four. Based on previous research, we assume that no more than two hidden nodes are necessary, and that the number of hidden nodes should not exceed the number of inputs. Furthermore, we allow each hidden node to have a bias or not. This leaves us with the following architectures to study in Table 1:

Table 1

Inputs	Hidden	Biases	Outputs	Parameters
1	0	0	1	1
1	1	0	1	2
1	1	1	1	3
2	0	0	1	2
2	1	0	1	3
2	1	1	1	4
2	2	0	1	6
2	2	1	1	7
2	2	2	1	8
3	0	0	1	3
3	1	0	1	4
3	1	1	1	5
3	2	0	1	8
3	2	1	1	9
3	2	2	1	10
4	0	0	1	4
4	1	0	1	5
4	1	1	1	6
4	2	0	1	10
4	2	1	1	11
4	2	2	1	12

Notice that there are several ways of accomplishing the same number of parameters. Furthermore, there are some parameter levels (such as 7) that can be accomplished in only one way. This is a significant advantage to the ability to control the use of node biases. Also note that the architectures with zero hidden nodes are traditional linear regressive models.

Based on the above combinations, we arranged the data into 92 patterns to be presented to each NN. Each pattern consists of four inputs and one output, in the form (lag4, lag3, lag2, lag1, target). In cases where the architecture does not require all four inputs, the most recent inputs are used. For instance, an architecture with only two inputs would use only lag1 and lag2. The same 92 patterns are presented to each NN for comparison purposes, even though some architectures might, in practice, allow for more data patterns. We use 48 patterns for training each NN, 24 for verification (to evaluate generalization), and 20 as a holdout. The holdout set consists of the last, most recent patterns, consistent with a forecasting task. The other 72 patterns are randomly assigned between the training set and verification set. In this way, the NN is presented recent patterns for training. The 48 training patterns gives us four patterns per parameter in the worst case (the 12 parameter model). The verification and holdout sets are large enough to sufficiently evaluate the models.

We use custom spreadsheets in MS Excel 2000 to model the NNs, and use the MSExcel Solver feature to perform training of the NNs. The Solver is a general optimization tool that has several options. We used the conjugate gradient algorithm with forward differencing, quadratic estimates, and automatic scaling. The Solver minimizes the SSE of the training set. Training a NN has been shown to be a nonlinear optimization problem solvable by nonlinear optimization algorithms. These algorithms have several advantages over the traditional back-propagation algorithms, such as speed, guaranteed convergence, and quality of the solution found. Furthermore, there is no need to set parameters such as training rate, momentum, etc.

Each NN architecture is trained 30 times from different random starting points, retaining the MSE of the training set, the verification set, and the holdout set. We also recorded the starting times of each training session, to informally track the time required for training. This is

not very accurate, given the need for screen updates, and the overall multi-tasking environment, but it can give us a general feel for the processing requirements for training each architecture.

III. Results

In a forecasting setting, one would like a NN model (architecture with determined parameters) that performs well on both the training and verification sets. In general, there is a trade-off. Learning the training set too well leads to poor performance on the verification set. Table 4 shows the Training and Verification MSE, by the number of parameters, for each trained NN from each random start. The results are as we expect. As the number of parameters increases, the training MSE, on average, decreases. However, the verification MSE reduces as the number of parameters increases to around three. It seems that between three and eight parameters offers the best generalizability, but certain numbers of parameters create a high variance in the MSE. This indicates a highly irregular training error surface with many local minima. In these circumstances, we might increase the number of random starts to ensure proper coverage of the training error surface.

Tables 5 through 9 show the training MSE vs. the verification MSE for 2 through 6 parameters, respectively. Plotting the MSEs in this manner allows us to see how each NN architecture allows for generalizability. In practice, we would like a NN that performs well on both sets, so we would choose a NN that is close to the bottom left corner of the graph. The graphs also show the different ways that each number of parameters is arrived at. This allows us to evaluate the efficacy of the use of node biases.

Figure 4
MSE by Number of Parameters

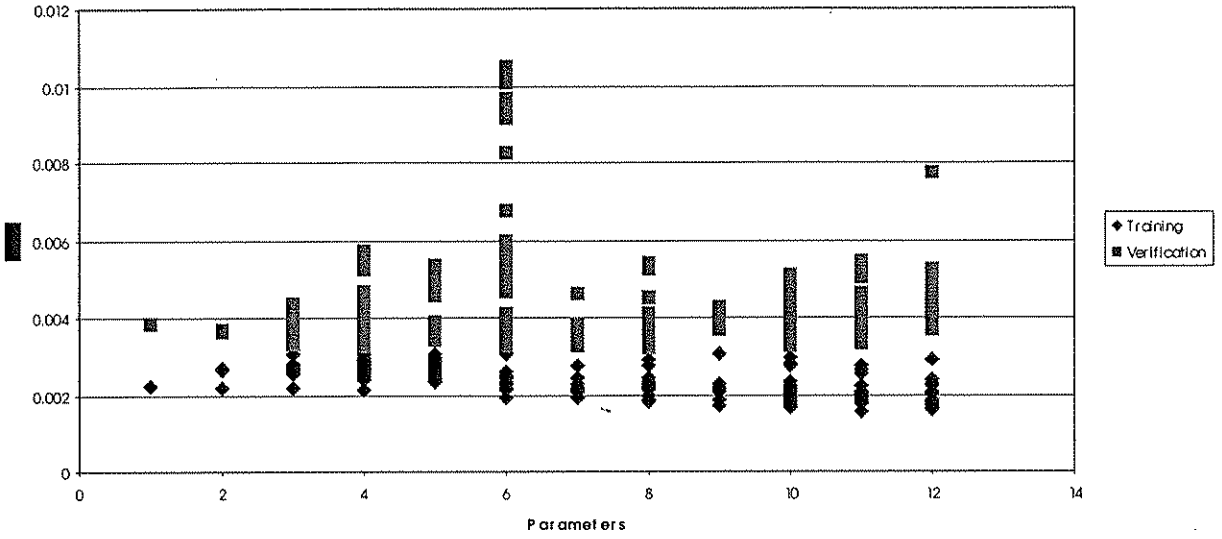


Figure 5
2 Parameter NN Generalizability

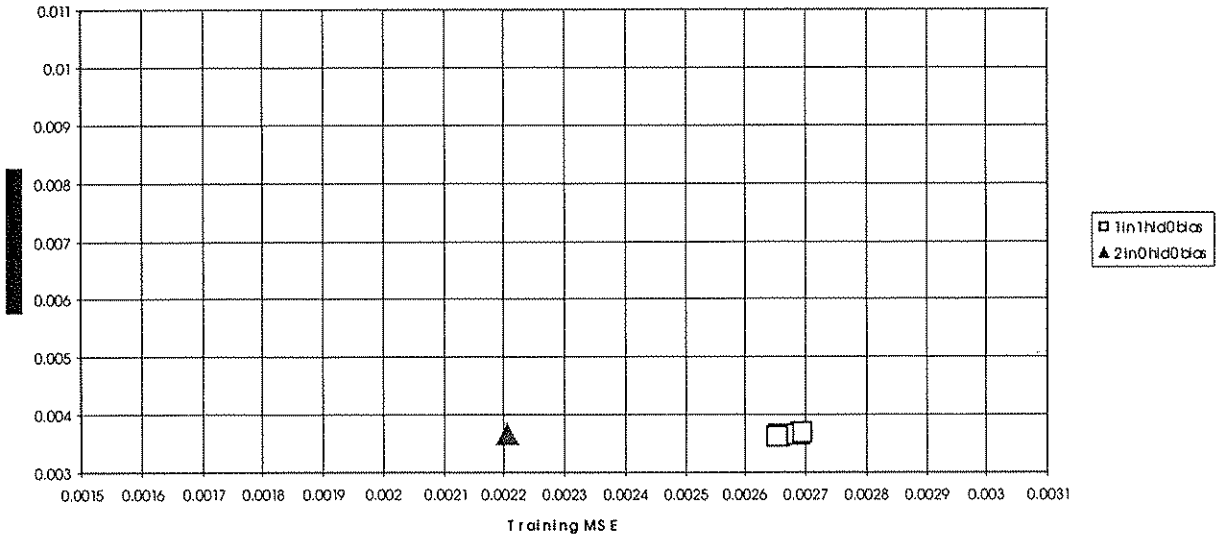


Figure 6
3 Parameter NN Generalizability

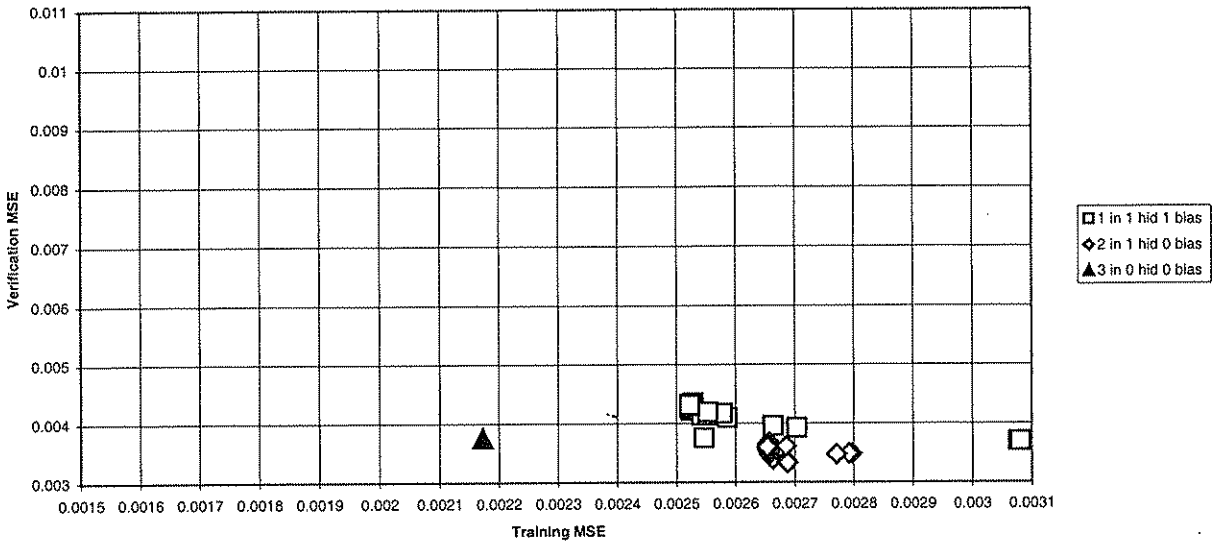


Figure 7
4 Parameter NN Generalizability

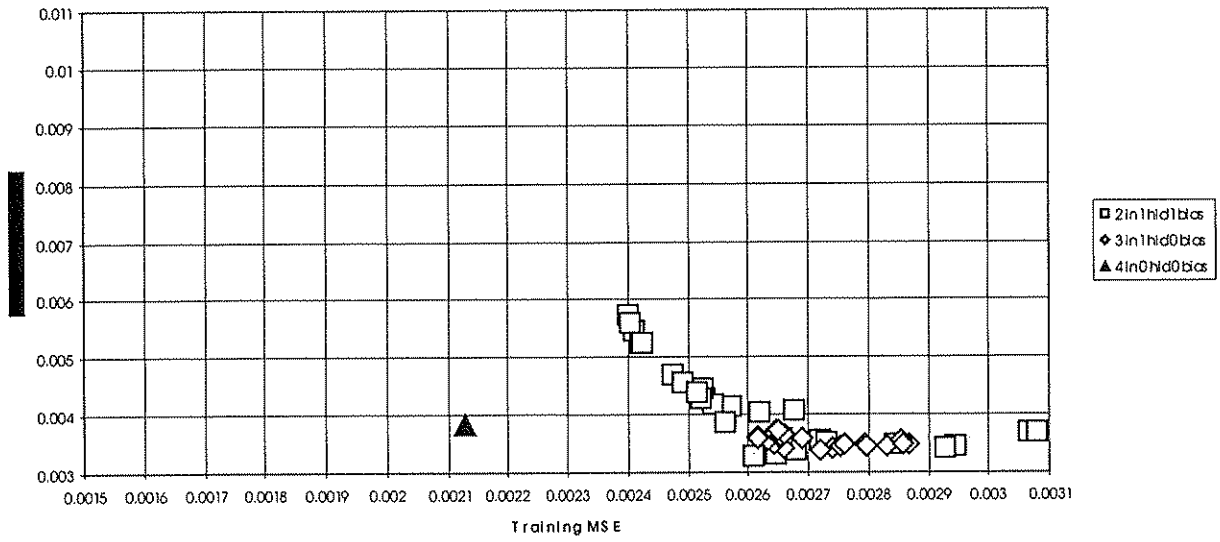


Figure 8
5 Parameter NN Generalizability

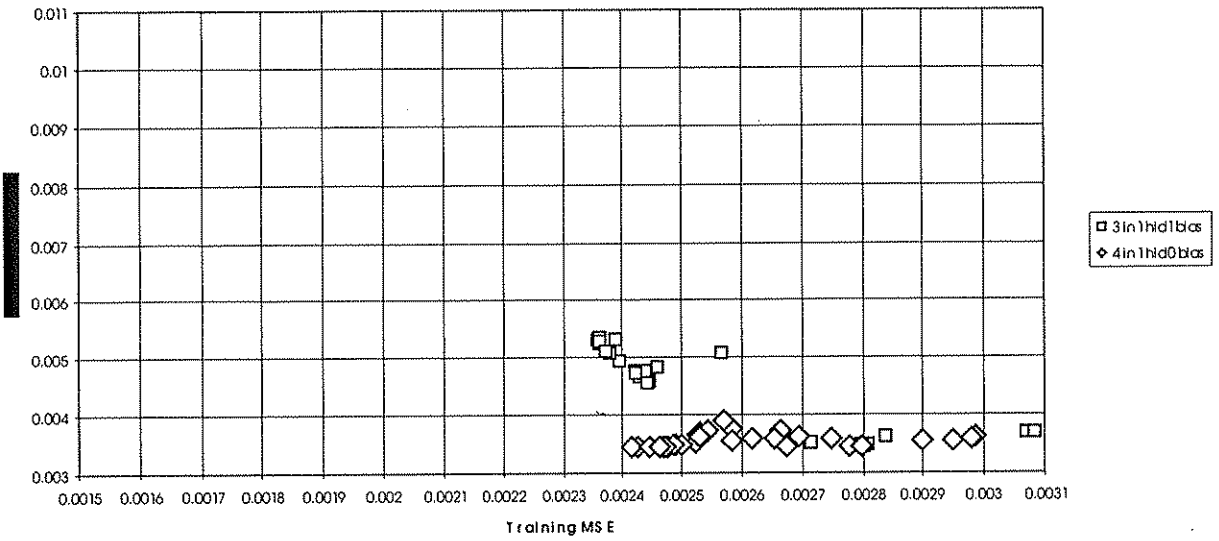
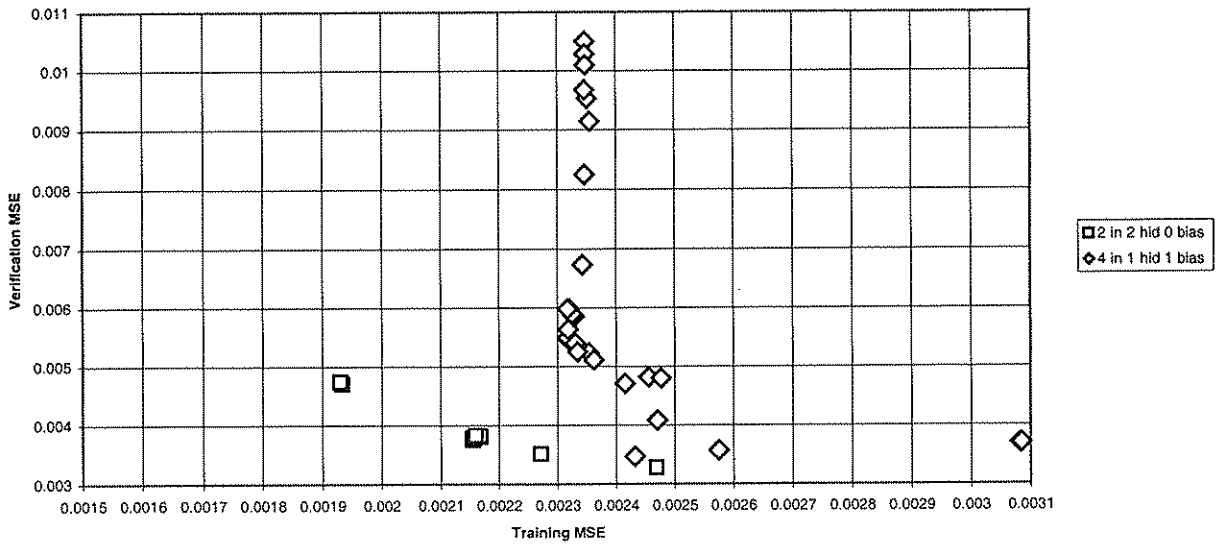


Figure 9
6 Parameter NN Generalizability



Several results are apparent from the graphs. First, as the number of parameters increases, the variance of the MSEs increases, and the trade-off between training MSE and verification MSE becomes apparent. Models that perform very well on the training set tend to perform less well on the verification set.

Second, the models that include node biases tend to have a higher variance in both training and verification MSEs than those without node biases. This might indicate that in training models with node biases, we may need to train more models using more random starts to achieve a high quality model.

In practice, we are faced with how to choose the best model for our forecast. A common heuristic is the "Keep the best" model. This is typically done by choosing the model that performs best on the verification set, or performs the best on the weighted average of the training and verification MSEs. Table 2 below summarizes the top ten models in terms of the lowest weighted average of training and verification MSEs. This table also includes the linear models, those with no hidden nodes, which were not in our top ten choices for models to keep. Notice that all of the top models have node biases and have a relatively large number of parameters.

Table 2

inputs	hidden	biases	parameters	Training MSE	Verification MSE	Weighted MSE	Holdout MSE
3	2	2	10	0.001724	0.003701	0.002383	0.07406
3	2	2	10	0.001945	0.003606	0.002499	0.163951
4	2	2	12	0.001717	0.004071	0.002502	0.109267
3	2	1	9	0.001689	0.004127	0.002502	0.24019
3	2	2	10	0.001746	0.004022	0.002504	0.094803
3	2	2	10	0.001793	0.003927	0.002504	0.023707
4	2	1	11	0.001741	0.004085	0.002523	0.038435
2	2	2	8	0.001986	0.003636	0.002536	0.187164
2	2	2	8	0.001979	0.003649	0.002536	0.377023
2	2	2	8	0.00198	0.003655	0.002538	0.557446

The mean of the training and verification MSE for the thirty random starts is given in Table 3. As could be seen in the Tables, the node biases reduce the training MSE, but do not always help in generalization – the verification MSE is not always reduced by adding node biases.

Table 3

			biases			
Inputs	hidden	Data	0	1	2	Grand Total
1	0	Average of Training	0.00224			0.00224
		Average of Verification	0.00382			0.00382
	1	Average of Training	0.00266	0.00259		0.00263
		Average of Verification	0.00364	0.00416		0.00390
1 Average of Training			0.00264	0.00259		0.00262
1 Average of Verification			0.00365	0.00416		0.00390
2	0	Average of Training	0.00221			0.00221
		Average of Verification	0.00367			0.00367
	1	Average of Training	0.00267	0.00263		0.00265
		Average of Verification	0.00354	0.00416		0.00385
	2	Average of Training	0.00216	0.00218	0.00212	0.00215
		Average of Verification	0.00381	0.00370	0.00378	0.00376
2 Average of Training			0.00241	0.00240	0.00212	0.00235
2 Average of Verification			0.00367	0.00393	0.00378	0.00380
3	0	Average of Training	0.00217			0.00217
		Average of Verification	0.00375			0.00375
	1	Average of Training	0.00269	0.00260		0.00265
		Average of Verification	0.00356	0.00448		0.00402
	2	Average of Training	0.00217	0.00214	0.00200	0.00211
		Average of Verification	0.00391	0.00386	0.00397	0.00391
3 Average of Training			0.00243	0.00237	0.00200	0.00232
3 Average of Verification			0.00374	0.00417	0.00397	0.00395
4	0	Average of Training	0.00213			0.00213
		Average of Verification	0.00383			0.00383
	1	Average of Training	0.00262	0.00244		0.00253
		Average of Verification	0.00356	0.00611		0.00484
	2	Average of Training	0.00214	0.00204	0.00199	0.00206
		Average of Verification	0.00405	0.00413	0.00437	0.00418
4 Average of Training			0.00237	0.00224	0.00199	0.00224
4 Average of Verification			0.00381	0.00512	0.00437	0.00444
Total Average of Training			0.00244	0.00238	0.00204	0.00234
Total Average of Verification			0.00373	0.00437	0.00404	0.00404

IV. Conclusion

This study has explored the use of node biases in neural networks used for forecasting. The results show that the node bias certainly has an effect on training. In cases where the NN architecture includes a node bias, the variance of the training and verification MSEs is higher than in equivalent parameter models. This might indicate a need to try more random starts in the training methodology.

It is unclear whether a node bias parameter has advantages over an arc weight parameter. The top models based on a weighting of verification MSE and training MSE all included node biases, but it might be that the task required a higher number of parameters than was handled with the non-node bias models.

From a practitioner standpoint, there are advantages to not using node bias. In many cases it is too computation intensive to use many random starts. In models without node bias fewer random starts should sufficiently cover the error space to ensure a quality model. Also, each additional hidden node added to the model adds fewer parameters, thus enabling more fine-tuning of the model to the problem. Finally, it should be clear that a combination of hidden nodes with biases and without is appropriate and very helpful in tuning the parameters to the problem at hand.

V. Bibiliography

Balkin, Sandy, and J. Keith Ord (1998) Automatic Neural Network Modeling for Univariate Time Series, *presentation at the Institute for Operations Research and the Management Sciences*, Seattle, October 1998.

Berardi, Victor L. (1998) Neural Network Ensembles for Posterior Probability Esitmation, *Proceedings of the 1998 Decisions Sciences Institute Annual Conference*, Las Vegas, NV, November 1998, pp. 1081-1083.

Box, G.E.P., B.M. Jenkins, and G.C. Reinsel (1994) Time Series Analysis, Forecasting and Control, 3rd edn. Englewood Cliffs: Prentice Hall.

Box, G.E.P., and Qwilym M. Jenkins (1976) Time Series Analysis: Forecasting and Control, Revised Edition. San Francisco, Holden-Day.

Burke, Laura I, and James P. Ignizio (1992) Neural Networks and Operations Research: An Overview, *Computers and Operations Research*, 19(3/4), pp. 179-189.

Chiang, W-C, TL Turban, and GW Baldridge (1996) A Neural Network Approach to Mutual Fund Net Asset Value Forecasting, *Omega, The International Journal of Management Science*, 24(2), pp. 205-215.

Cottrell, Marie, Bernard Girard, Yvonne Girard, Morgan Mangeas, and Corinne Muller (1995) Neural Modeling for Time Series: A Statistical Stepwise Method for Weight Elimination, *IEEE Transactions on Neural Networks*, 6(6), pp. 1355-1364.

Desai, Vijay S., and Rakesh Bharati (1998) The Efficacy of Neural Networks in Predicting Returns on Stock and Bond Indices, *Decision Sciences*, 29(2), pp. 405-425.

Faraway, Julian, and Chris Chatfield (1998) Time series forecasting with neural networks: a comparative study using the airline data, *Applied Statistics*, 47(2), pp. 231-250.

Hill, Tim, Marcus O'Connor, and William Remus (1996) Neural Network Models for Time Series Forecasts, *Management Science*, 42(7), pp. 1082-1092.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989) Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, 2, pp. 359-366.

Hornik, Kurt (1991) Approximation Capabilities of Multilayer Feedforward Networks, *Neural Networks*, 4, pp. 251-257.

Hung, Ming S., and James W. Denton (1993) Training neural networks with the GRG2 nonlinear optimizer, *European Journal of Operational Research*, 69, pp. 83-91.

Johnson, Richard A. and Dean W. Wichern (1992) *Applied Multivariate Statistical Analysis, third edition*. New Jersey, Prentice Hall, pp. 356-394.

Kline, Douglas M., and Gerald Kohers (1999) Evaluating Methodologies for Neural Network Multi-Step Time Series Forecasting. *Working paper No. 99-01MIS*, Center for Business and Economic Development, Sam Houston State University.

Kohers, Gerald, and Douglas M. Kline (1998) Evaluating Methods for Multi-Period Time Series Forecasting with Neural Networks. *Presentation at INFORMS National Meeting*, May 2-5, Cincinnatti, OH.

Makridakis, Spyros, Chris Chatfield, Michele Hibon, Michael Lawrence, Terence Mills, Keith Ord, and LeRoy F. Simmons (1993) The M2-Competition: A real-time judgmentally based forecasting study, *International Journal of Forecasting*, 9, pp. 5-22.

Schaffer, Cullen (1993) Selecting a Classification Method by Cross-Validation, *Machine Learning*, 13, pp. 135-143.

Tang, Zaiyong, Chrys de Almeida, and Paul A. Fishwick (1991) Time series forecasting using neural networks vs. Box-Jenkins methodology, *Simulation,* 57(5), pp. 303-310.

Tang, Zaiyong and Paul A. Fishwick (1993) Feedforward Neural Nets as Models for Time Series Forecasting, *ORSA Journal on Computing*, 5(4), pp. 374-385.

Wang, Shouhong (1995) The Unpredictability of Standard Back Propagation Neural Networks in Classification Applications, *Management Science,* 41(3), pp. 555-559.

Zell, Andreas, Gunter Mamier, Michael Vogt, Niels Mache, Ralf Hubner, Sven Doring, Kai-Uwe Herrmann, Tobias Soyez, Michael Schmalzl, Tilman Sommer, Artemis Hatzigeorgiou, Dietmar Posselt, Tobias Schreiner, Bernward Kett, Gianfranco Clemente, and Jes Wieland, *Stuttgart Neural Network Simulator (SNNS) User Manual*, Version 4.1, University of Stuttgart Institute for Parallel and Distributed High Performance Systems (IPVR), 1995, available at: http://www-ra.informatik.uni-tuebingen.de/SNNS/.

Zhang, Guoquinang, B. Eddy Patuwo, and Michael Y. Hu (1998) Forecasting with artificial neural networks: The state of the art, *International Journal of Forecasting*, 14, pp. 35-62.

Zhang, G. Peter, B. Eddy Patuwo, and Michael Y. Hu (1998) Nonlinear Time Series Forecasting with Artificial Neural Networks, *Proceedings of the 1998 Decisions Sciences Institute Annual Conference*, Las Vegas, NV, November 1998, pp. 1023-1025.

Zhang, Gioquinang, and Michael Y. Hu (1998) Neural Network Forecasting of the British Pound / US Dollar Exchange Rate, Omega, The International Journal of Management Science, 26(4), pp. 495-506.

Zhang, G. Peter, B.E. Patuwo, and M.Y. Hu (1998) Forecasting with Artificial Neural Networks: The State of the Art. *International Journal of Forecasting*, v. 14, pp. 35-62.

Zhang, G. Peter (1999) An Experimental Evaluation of Neural Networks for Linear Time Series Forecasting. *Proceedings of the 30th Annual Meeting of the Decision Sciences Institute*, Nov. 20-23, New Orleans, Louisiana.