A Delay Damage Model Selection Algorithm for NARX Neural Networks

Tsungnan Lin [1,2*], C. Lee Giles [1,3], Bill G. Horne [1], S.Y. Kung [2]

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
[2] Department of Electrical Engineering, Princeton University, Princeton, NJ 08540
[3] UMIACS, University of Maryland, College Park, MD 20742

U. of Maryland Technical Report CS-TR-3707 and UMIACS-TR-96-77

Abstract

Recurrent neural networks have become popular models for system identification and time series prediction. NARX (Nonlinear AutoRegressive models with eXogenous inputs) neural network models are a popular subclass of recurrent networks and have been used in many applications. Though embedded memory can be found in all recurrent network models, it is particularly prominent in NARX models.

We show that using intelligent memory order selection through pruning and good initial heuristics significantly improves the generalization and predictive performance of these nonlinear systems on problems as diverse as grammatical inference and time series prediction.

Keywords: Recurrent neural networks, tapped–delay lines, long–term dependencies, time series, automata, memory, temporal sequences, gradient descent training, latching, NARX networks, auto–regressive, pruning, embedding theory.

*Current address: Epson Palo Alto Laboratory, 3145 Porter Drive, Suite 104, Palo Alto, CA 94304

1 Introduction

NARX (Nonlinear AutoRegressive models with eXogenous inputs) recurrent neural architectures [6, 41], as opposed to other recurrent neural models, have limited feedback architectures which come only from the output neuron instead of from hidden neurons. It has been shown that in theory one can use NARX networks, rather than conventional recurrent networks, without any computational loss and that they are at least equivalent to Turing machines [53]. Not only are NARX neural networks computationally powerful in theory, but they have several advantages in practice. For example, it has been reported that gradient-descent learning can be more effective in NARX networks than in other recurrent architectures with "hidden states" [25].

Part of the reason can be attributed to the embedded memory of NARX networks. This embedded memory will appear as jump-ahead connections which provide shorter paths for propagating gradient information more efficiently when the networks are unfolded in time to backpropagate the error signal and thus reduce the network's sensitivity to the problem of *long-term dependencies* [34, 36].

Not only can the embedded memory reduce the sensitivity to long-term dependencies, but it also plays an important role in learning capability and generalization performance [35]. In particular, forecasting performance could be seriously deficient if a model's memory architecture is either inadequate or unnecessarily large. Therefore, choosing the appropriate memory architectures for a given task is a critical issue in NARX networks.

The problem of memory-order selection is analogous to that of choosing an optimal subset of regressors variables in statistical model building. In optimal subset selection, it is desired that the model includes as many regressors as possible so that the information content in these regressors will influence the predicted value of the dependent variable; on the other hand, it is also desired that the model includes as few regressors as possible because the variance of the model's predictions increases along with the increasing number of regressors [39].

According to the *embedding theorem* [45, 51, 57], the memory orders need to be large enough in order to provide a sufficient embedding. The problem of choosing the proper memory architecture corresponds to give a good representation of input data. A good representation can make useful information explicit and easy to extract. Two different representations can be equivalent in terms of expressive power, but may differ dramatically in the efficiency or effectiveness of problem-solving.

When there is no prior knowledge about the model of the underling process, traditional statistical tests can be used, for example, Akaike information criterion (AIC) [1] and the minimum description length principle (MDL) [50]. Such models are judged on their "goodness-of-fit", which is a function of the likelihood of the data given the hypothesized model and its associated degrees of freedom. Fogel [16] applied the modification of AIC to select a "best" network. However, the AIC method is complex and can be troubled by imprecision [52, 23].

Evolutionary Programming [2, 18] is another search mechanism. This algorithm operates on a population of models. Offspring models are created by randomly mutating parents models. Competition between offspring models for survival are judged according to the *fitness function*. Fogel [17] used evolutionary programming for order selections of linear models in a time series of ocean acoustic data. But the algorithm can be computationally expensive when the underling process is complex and nonlinear.

Alternatively, an adaptive algorithm which treats the delay operators as ordinary adjustable parameters can be a useful technique. This algorithm iteratively determines the memory of a model based on the gradient information. Originally proposed by Etter, it was used as a "adaptive delay filter", which included variable delays taps as well as variable gains, for modeling several sparse systems [15, 7]. Recently others [4, 13, 33, 14] have also extended neural networks to include adaptable time delays. Because the error function of the adaptable time delays depends on the autocorrelation function of input signals [15, 7], the gradient of the delay operator will depend on the derivative of input signals. However, a closed form of the derivative of the input signal can not always be determined in general. Therefore, there is no guarantee that such a modified algorithm for a nonliner model would converge to the optimum solution.

In this paper, we propose a pruning-based algorithm, the Delay Damage Algorithm, to determine the optimal memory-order of NARX and input time delay neural networks. This algorithm can also incorporate several useful heuristics, such as weight decay [29], which are used extensively in static networks to optimize the nonlinear function. [For a survey of pruning methods for feedforward neural networks, see [49].]

The procedure of the algorithm starts with a NARX network with enough degrees of freedom in both input and output memory, and then delete those memory orders with small sensitivity measure after training. After pruning, the network is retrained. Of course, this procedure can be iterated. This method should be contrasted to another recurrent neural network procedure [22] where nodes are pruned based on output values and is similar in spirit to the pruning method of Pederson and Hansen[47]. The sensitive measure of each memory order is calculated by estimating the second order derivative of the error function with respect to each memory order. Le Cun *et al.* [11] originally calculated the "saliency" by estimating the second order derivative for each weight. The success of their algorithm had been implemented in identification of handwritten ZIP-codes by pruning the weights of feedforward networks [11, 10].

2 NARX Neural Network

An important and useful class of discrete–time nonlinear systems is the *Nonlinear AutoRegressive with eXogenous inputs* (NARX) model [6, 32, 37, 54, 55]:

$$y(t) = f\left(u(t-D_u), \ldots, u(t-1), u(t), y(t-D_y), \ldots, y(t-1)\right), \tag{1}$$

where $u(t)$ and $y(t)$ represent input and output of the model at time t, D_u and D_y are the input-memory and output-memory order, and the function f is a nonlinear function. When the function f can be approximated by a Multilayer Perceptron (MLP), the resulting system is called a *NARX recurrent neural network* [6, 41]. Figure 1 shows a NARX networks with input-memory of order 2 and output-memory of order 3. It has been demonstrated that NARX neural networks are well suited for modeling several nonlinear systems such as heat exchangers [6], waste water treatment plants [54, 55], catalytic reforming systems in a petroleum refinery [55], nonlinear oscillations associated with multi–legged locomotion in biological systems [58], time series [9], and various artificial nonlinear systems [6, 41, 48].

When the output-memory order of NARX network is zero, a NARX network becomes a Time Delay Neural Network (TDNN) [30, 31, 60], which is simply a tapped delay line input into a MLP. In general the TDNN implements a function of the form:

$$y(t) = f\left(u(t-D_u), \ldots, u(t-1), u(t)\right). \tag{2}$$

Tapped delay lines can be implementations of *delay space embedding* and can form the basis of traditional statistical autoregressive (AR) models. In time series modeling, subset models are often desirable in the hope of capturing the global behavior of the data. A subset autoregressive (SAR) time series model is defined as:

$$y(t) = \sum_{i=1}^{m} a_{\alpha_i} y(t-\alpha_i) + u(t). \tag{3}$$

where a_{α_i} and α_i are the ith coefficient and ith order respectively, and $u(t)$ is the white noise innovation with zero mean. SAR models have demonstrated their long-term prediction capability in various applications [38, 42] and can easily be extended into nonlinear models. A nonlinear version of a SAR is a NSAR [1] .

A primary problem associated with the nonlinear subset model is how to optimally select the subset orders. Various methods have been suggested for the determination of the orders in linear case [64, 42]. In designing the nonlinear sparse models, we can determine the memory order by applying the Delay Damage Algorithm, described in the next section. The Delay Damage approach is based on the assumption that the memory order of an initial network should be given enough

[1] We use the term NSAR and not TDNN to differentiate between networks that are driven by previous values and those by external inputs. However, these distinctions are not always made nor are standard. For example, NARX networks have also been called NARMA (Nonlinear AutoRegressive Moving Average) networks. It would also be possible to refer to a NSAR as a NAR model.

degrees of freedom to learn the the given task in a reasonable amount of training time.

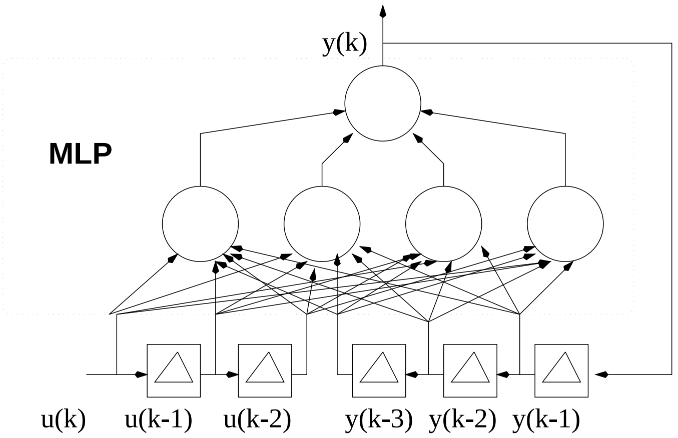

Figure 1: A NARX neural network with 2 input-memory taps and 3 output-memory taps.

3 Delay Damage Algorithm

The Delay Damage algorithm attempts to select optimal memory orders of NARX networks by calculating the sensitivity of the error to each memory order after the network is trained by gradient-based learning algorithm.

Several methods for sensitivity calculations have been proposed [40, 28, 24], for details see the survey paper by Reed [49]. Our method of calculating sensitivity is based on evaluating the second order derivative of the cost function with respect to each memory order [47].

We assume the cost function (E) is the mean-squared error:

$$E = \frac{1}{2} \sum_{p} \sum_{\tau=t_0}^{t_1} (d^p(t) - o^p(t))^2, \tag{4}$$

where p and τ denote the pattern index and time index respectively. The MLP of a NARX network can be described as:

$$x_i(t) = f\left(a_i(t)\right), \tag{5}$$

$$a_i(t) = \sum_j w_{ij} x_j(t), \tag{6}$$

where $x_i(t)$ is the output of hidden node i, $a_i(t)$ is the weighted-sum input, f is the nonlinear function, and w_{ij} is the real valued connection weight from node j to node i.

By the chain rule, the first-order derivative of E with respect to output node i at time $t = t_1$ is given as

$$\frac{\partial E}{\partial a_k(t_1)} = f_k'(a_k(t_1))e_k(t_1), \tag{7}$$

where $e_k(t_1)$ is the error between the output node and the target output. The gradient information of the hidden nodes can be obtained by backpropagating the gradient information from the output node. For the earlier time steps, there will be *an injection error* from the target output into the output node. Thus, not only is the gradient information determined by a backward pass through the unrolled network, but the injection errors are also taken into account in reverse order. The error signal of output node k at time $t_0 < \tau < t_1$ will become:

$$\frac{\partial E}{\partial a_k(\tau)} = f_k'(a_k(\tau))\left(e_k(\tau) + \sum_j w_{jk}\frac{\partial E}{\partial a_k(\tau+1)}\right). \tag{8}$$

Differentiating the first-order derivative once more yields the second order information. For the output node k at the last time step t_1, the second order information can be described by

$$\frac{\partial^2 E}{\partial a_k^2(t_1)} = f'(a_k(t_1))^2 - e_k(t_1)f''(a_K(t_1)). \tag{9}$$

The Levenberg-Marquardt approximation was used by Le Cun *et al.* to drop the second term of Equation 9. This will result in the same order of complexity as computing the first-order derivatives and non-negative quantities. The second order information of the hidden nodes at time t_1 can be calculated by the backpropagation procedure. Proceeding to the earlier time steps, the effect of the injection error should be taken into account. Therefore, the second order gradient of the output node k at interval time $t_0 < \tau < t_1$ will become:

$$\frac{\partial^2 E}{\partial a_k^2(\tau)} = f'(a_k(\tau)^2)\left(1 + \sum_j w_{jk}^2 \frac{\partial^2 E}{\partial a_k^2(\tau+1)}\right). \tag{10}$$

The procedure is computed down to $t_0 + 1$.

In order to compute the second-order derivative with respect to each memory order of a NARX neural network, the function f of each memory order is assumed to be linear. Therefore, the sensitivity of input-memory order $u(k-n)$ and output-memory order $y(k-m)$ can be described respectively as:

$$\frac{\partial^2 E}{\partial u(k-n)^2} = \sum_p \sum_\tau \left((u^p(t-n))^2 \sum_j w_{jn}^2 \frac{\partial^2 E}{\partial \left(a_j^p(\tau)\right)^2} \right) \tag{11}$$

$$\frac{\partial^2 E}{\partial y(k-m)^2} = \sum_p \sum_\tau \left((y^p(t-m))^2 \sum_j w_{jm}^2 \frac{\partial^2 E}{\partial \left(a_j^p(\tau)\right)^2} \right). \tag{12}$$

Once the sensitivity of each node is evaluated, nodes are pruned based on a predescribed sensitivity threshold or ratio and retraining occurs. Pruning stops when the a certain error, total or otherwise, is reached.

4 Experimental Results

Here we discuss experimental results for a grammatical inference problem and time series prediction problems, sun spot data and laser intensity data from the Santa Fe time series competition. The grammatical inference problem will use a NARX neural network whereas the time series problems will use degenerate forms of the NARX network, the NSARs. We also give a brief introduction to the theory of dynamic embedding before discussing the results of time series prediction.

In order to also optimize the architecture of the MLP of a NARX network or NSAR, several methods of weight-elimination [5, 29, 44, 61, 63] can be incorporated into the training algorithm. In the following experiments, networks are trained using weight decay [29]. All experiments were trained using Back-Propagation Through Time (BPTT).

4.1 Grammatical Inference: Learning A 512-state Finite Memory Machine

NARX networks have been shown to be able to simulate and learn a class of finite state machines [8, 21], called respectively *definite and finite memory machines*. When being trained on strings which are encoded as temporal sequences, NARX networks are able to "learn" rather large (hundreds to thousands of states) machines provided that they have enough memory and the logic implementation is not too complex. However, the generalization performance and the size of extracted machines are also found to be very sensitive to the memory order selections in NARX networks [35]. The purpose of this experiment is to see how the delay damage algorithm can improve the generalization performance of NARX networks with unnecessary memory structures.

In this experiment, the finite memory machine has 512 states. The machine has input order of 5 and output order of 4. Its transition function can be described as the simple logic function:

$$y(k) = \bar{u}(k-5)\bar{u}(k) + \bar{u}(k-5)y(k-4) + u(k)u(k-5)\bar{y}(k-4) \tag{13}$$

where y and u represent output and input respectively, and $\bar{x}$ represents the complement of x. The FSM is shown in Figure 2. The *depth*, d, of the machine is 9. The training set was 300 strings randomly chosen from the complete set. The complete set, which consists of all strings of length from 1 to $d+1$ (10 in this case), are shown to be able to sufficiently identify a finite memory machine with depth d [20] . The strings were encoded such that input values of 0s and 1s and target output labels "negative" and "positive" corresponded to floating point values of 0.0 and 1.0 respectively.

Initially, before pruning the NARX networks were chosen to have 4 hidden nodes, 10 input taps, and 10 output taps. The number of weights was 91. The memory order of the neural network was chosen to be large enough to make sure the architecture had enough degrees of freedom to learn the large machine within a reasonable amount of time. The networks were trained with Back Propagation Through Time (BPTT) algorithm at the learning rate of 0.1 and weight decay of 0.001. The training time was set to 5000 epochs. For more details, see [20, 21]. For each of 50 experiments, the weights were randomly initialized within the range of $[-0.5, 0.5]$.

The average training time was approximately 600 epochs. After training, the trained networks were tested on the remaining strings of the complete set. A zero error rate showed that the networks had learned the complete set. However, when trained networks were tested on the strings of length longer than 10, the number of errors was no longer zero and plotted in the Figure 3 as a function of the length of testing strings. Note that the performance of NARX networks can be strongly dependent on the selection of memory order [35]. In particular the new testing set consisted of 250 positive strings and 250 negative strings from length 20 to 150 in increments of 10.

Applying the delay damage algorithm described in Section 3 to prune the taps with small sensitivity measure always resulted in the minimal NARX architectures with 5 input memory orders and 4 output memory orders. Furthermore, out of 50 pruning runs, the minimal NARX architectures always contain the proper subset of $u(t)$, $u(t-5)$, and $y(t-4)$. The number of weights of pruned networks could be reduced to anywhere from 20 to 30 depending on the number of remaining memory orders. The time to retrain the pruned networks took on the average 70 epochs. Furthermore, the generalization error on the remaining strings of the complete set and the strings randomly generated from length 20 to 150 was reduced to *zero*.

4.2 The Theory of Embedding

In order to clarify the results of our experiments on time series prediction we give a brief introduction to the theory of embedding, for more details see [51]. The *state* of a deterministic dynamic system is the information necessary to determine the entire future evolution of the system. A time series

Figure 2: A 512 state finite memory machine with input order 5 and output order 4.

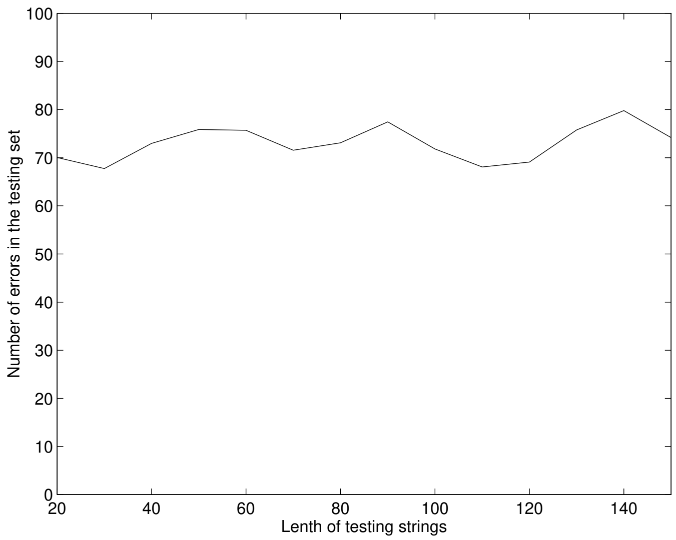

Figure 3: Number of errors in the testing set, which consists 250 positive strings and 250 negative strings from length 20 to 150, as a function of the length of the strings for a NARX neural network trained on the data from the 512 state finite memory machine.

is a set of measures of an observable quantity of the system over time. An observable quantity is a function only of the state of the underlying system. The observations , $y(n)$, are a projection of the multivariate state space of the system onto the one dimensional space. In order to do prediction, we need to reconstruct as well as possible the state space of the system using the information in $y(n)$.

The embedding theorem motivates the technique of using time-delay coordinate reconstruction in reproducing the phase space of an observed dynamical system. A collection of time lags in a vector space of d dimensions,

$$\mathbf{s}(t) = \Big[y(t), y(t-T), y(t-2T), \cdots, y(t-(d-1)T)\Big]\,, \tag{14}$$

will provide sufficient information to reconstruct the states of the dynamical system. The purpose of time-delay embedding is to unfold the projection back to a multivariate state space that is representative of the original system [43, 46, 57]. It was shown that if the dynamical system and the observed quantity were generic, then the delay coordinate map from a d-dimensional smooth compact manifold to $2d+1$-dimensional reconstruction space was a *diffeomorphism* (one-to-one differential mapping). The theorem was further refined by Sauer *et al.* [51] such that a measured quantity led to a one-to-one delay coordinate map as long as the reconstruction dimension was greater than twice the box-counting dimension of the attractor. The embedding theorem provides a *sufficient* condition for choosing the embedded dimension d_E large enough so that the projection is theoretically able to construct the original state space. Once a large enough $d = d_E$ has been obtained, any $d \geq d_E$ will also provide an embedding.

The predictive relationship between the current state $\mathbf{s}(t)$ and the next value of time series can be expressed as

$$\begin{aligned} y(t+1) &= f\left(\mathbf{s}(t)\right) \\ &= f\left(y(t), y(t-T), y(t-2T), \cdots, y(t-(d-1)T)\right). \end{aligned} \tag{15}$$

The embedding theorem provides a theoretical framework for nonlinear time series prediction. Once an embedding dimension is chosen, one remaining task is to approximate the mapping function f. It has been shown that a feedforward neural network with enough neurons is capable of approximating any nonlinear function to an arbitrary degree of accuracy [12, 19, 26, 27]. Neural networks thus can provide a good approximation to the function f. These arguments provide the basic motivation for the use of NARX networks to the nonlinear time series prediction.

It is still an open question as to how to optimally choose the embedding dimension d and the delay time T. The embedding theorem only guarantees that a sufficiently large embedding dimension d_E will be capable of unfolding the state space. It does not describe how complicated the mapping function will be. One may be able to find a minimum embedding dimension to unfold the state space, but the mapping function may be too complicated to approximate. For example, two different embedded dimensions can be equivalent in terms of repressive power, but the nonlinear function may dramatically differ in the efficiency of approximation. It should be noted that when T is chosen, the embedding representation will always be equally spaced. This rules out the possibility of any unequally spaced representation of the embedding dimension.

In the following experiments, we will use the delay damage algorithm to prune the taps of a neural network. After pruning, the networks will be retrained, at least twice. The final network architecture ends up providing a unequally spaced temporal representation of the state space. These sparse delay architectures can be regarded as the nonlinear versions of SAR models (NSARs) and the subset orders of the NSAR provide the embedding coordinates to reconstruct the mapping function. The time lags in the initial embedding vector can be equally or unequally spaced as long as the embedding dimension is large enough to unfold the state space.

4.3 Prediction of Sunspot Data

Sunspots are dark botches on the sun and yearly averages have been recorded since 1700. The series is shown in Figure 4 and has served as a benchmark for time series prediction problems. Several researchers have tested the prediction ability of neural networks on this data. For example, Weigend *et al.* [61] trained a NSAR (or TDNN) network with an embedded dimension of 12 with 8 hidden neurons and pruned the weights by adding a complexity term to the cost function. They were able to reduce a network with 8 hidden nodes to 3. The embedding dimension remained the same. Svarer *et al.* [56] pruned the weights using a second order sensitivity measure. Our approach is to prune the orders of networks directly and use the weight decay technique to optimize the nonlinear mapping function.

For comparison we treat the data in the same way as Weigend *et al.* [61]. and partition it into a training set from 1700 through 1920 and a testing set from 1921 to 1979. The data set was scaled to be in the range of $[0, 1.0]$. We tried various architectures with various number of hidden nodes and different embedding dimensions. The initial embedding dimension was usually close to 14. However, the largest initial order depth was 33. The networks were trained as a one-step ahead predictors at the learning rate of 0.1 and weight decay of 0.001. After pruning the taps, the networks were retrained.

Our cost function was the normalized mean squared error defined as:

$$\begin{aligned} NMSE(N) &= \frac{\sum_{k\in S}(target_k - prediction_k)^2}{\sum_{k\in S}(target_k - mean_S)^2} \\ &= \frac{1}{\sigma^2}\frac{1}{N}\sum_{k\in S}(x_k - \hat{x_k})^2, \end{aligned} \tag{16}$$

where $k = 1 \cdots N$ enumerates the point in the data set S, and $mean_S$ and σ^2 denote the sample average and sample variance of the target value in S. This measure removes the data's dependence on dynamic range and size of the data set. To make comparisons with previous work, the variance was normalized with the same value of 1535 [61]. Training was stopped when the normalized mean squared error was less than 0.1.

Reported in table 1 are our two best results from the architectures denoted as B and C. The final number of hidden nodes were respectively 6 and 3. The final subset orders of the two networks after pruning consisted of $(x_t, x_{t-1}, x_{t-9}, x_{t-11}, x_{t-13}, x_{t-14}, x_{t-31}, x_{t-33})$ and $(x_t, x_{t-1}, x_{t-2}, x_{t-8}, x_{t-10}, x_{t-11}, x_{t-19}, x_{t-22})$ respectively. Table 1 lists the one-step ahead prediction performance

Network	*NMSE*			*No. of parameters*
	Train (1700-1920)	Test (1921-1955)	Test (1956-1979)	
A (Weigend *et al.*)	0.082	0.086	0.35	43
B	0.094	0.1065	0.449	69
C	0.0966	0.1013	0.4033	31
D (Svarer *et al.*)	0.090	0.082	0.35	14

Table 1: Normalized MSE for one-step ahead prediction on the sunspot data.

of various models on the data set. Comparing the NMSE, the single-step prediction qualities of different models are quite comparable.

We also compared long-term prediction results to one another. To do the long-term prediction, the predicted output is fed back into the neural network. Hence, the inputs consists of predicted value as opposed to actual data of the origin time series. We used the *average relative I-times iterated prediction variance* proposed by Weigend *et al.* [61] as a performance measure of the long-term behavior of the models. The *average relative I-times iterated prediction variance* is defined as:

$$arv(iterated, I) = \frac{1}{\sigma^2}\frac{1}{M}\sum_{t=1}^{M}(x_t - \hat{x}_{t,I})^2, \tag{17}$$

where x_t is the real data point at time t, M the difference between the period of the prediction and I, and $\hat{x}_{t,I}$ is the predicted value for time t after I iterations. Recall that I is the number of time steps into the future of the prediction. The average relative iterated prediction variance is shown in Figure 5 and went through from 1921 to 1979. From Figure 5, it can be seen that the networks B and C which have sparsely connected tapped delay lines start to perform better as I is increased. Note that the arv is greater than 1.0, since the variance of the original data 1920 through 1979 is fairly large. If we normalize the arv to the variance of the original data set, the results are shown in Figure 6. Again the networks B and C perform better as I is increased.

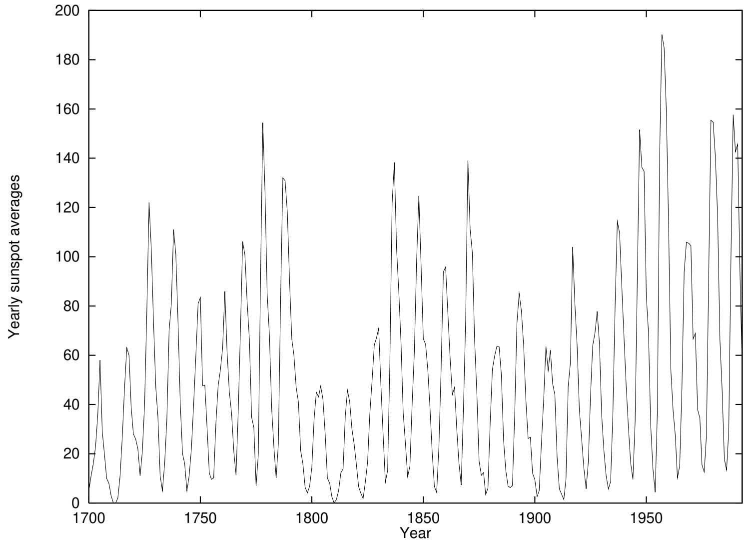

Figure 4: Sunspot data for years 1700 to 1979.

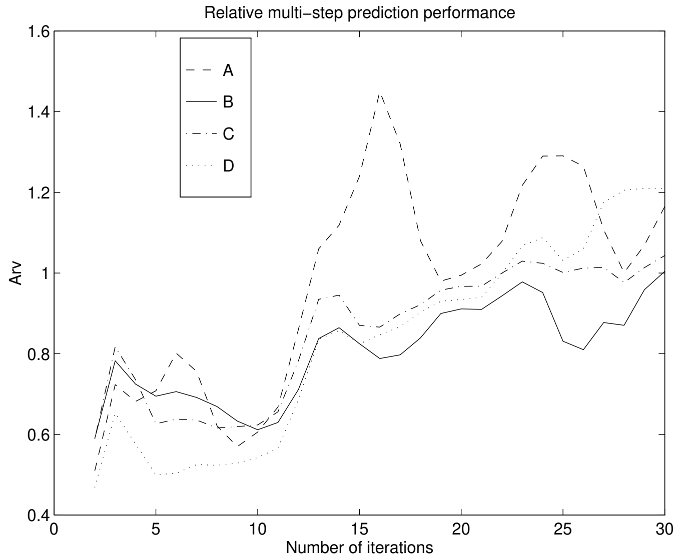

Figure 5: Relative multi-step prediction performance. These results were normalized to a variance of 1535.

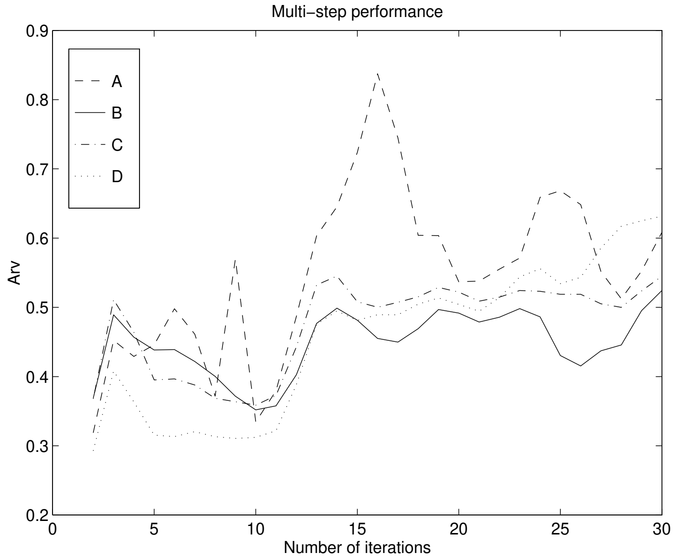

Figure 6: Relative multi-step prediction performance. The results were normalized to the variance of the original data set.

4.4 Prediction of Laser Data

In order to explore the capability of capturing the global behavior of NSAR neural networks, we also tested them on the laser data the Santa Fe competition. The data set consists of laser intensity collected from a laboratory experiment [62]. Although deterministic, its behavior is chaotic as seen in Figure 7. Figure 8 shows the normalized autocorrelation function of the first 1000 training data points.

The networks were trained as a one-step ahead predictor using the first 1000 points. The data was scaled to zero mean and unit variance. We give the results of our best performing neural network. The network was chosen to have two hidden layers. The first hidden layer had 5 neurons and the second layer had 3 neurons. We used the hyperbolic tangent function as the nonlinear function in the two hidden layers. The output layer had only one output neuron with linear function. We chose the embedding dimension to be 25. However, the depth of the tapped delay line was very long (up to 500). We used the last 12 subsets, 3 subsets from the past 100 data points (i.e., x_{t-99},x_{t-100},

x_{t-101}), and 10 subsets from the past 445 data points (i.e., $x_{t-445},\cdots,x_{t-455}$). We heuristically chose the memory order by observing that the autocorrelation function of the training data set was peaked in these areas. The total number of weights was 152.

In this experiment, we used the adaptive learning rate algorithm and batch data presentation. The algorithm works as follows. The cost function is monitored during training. If the increase of the cost function at time t is not larger than the product of the cost of previous time step and *the cost bonus*, κ, the learning rate is increased by ten percent until the learning rate reaches the maximum learning rate, 1.0, and the weights are updated. If the increase of the cost function at the present time step is larger than the product of the cost bonus and the cost of earlier time step, the weight vector of previous time step is restored and the learning rate decreases by half until it reaches the minimum learning rate (10^{-8}), and the weights are updated according to the smaller learning rate and the gradient information of one time step ahead. Of course, this requires one to keep track of the weight vector and the gradient information of previous time steps. In our experiment, the initial learning rate was 0.01 and the cost bonus, κ, was 0.001. The network was trained with the previously discussed mean squared error cost function in the batch mode (i.e. errors were accumulated until the end of the data). When the average squared error is less than 0.004, training was stopped and pruning commenced. This procedure was performed twice.

The trained and pruned neural network NSAR gave some interesting long-term predictive behavior of the time series. The network was able to predict 1000 data points using closed-loop iteration. Figure 9 shows the 500 steps iterated prediction from time $t = 1000$. Figure 10 shows the 1000 step iterated prediction; note that the network was able to predict three collapses and there were only two collapses in the training data set. We pruned 5 taps of the original network when trained on the first 1000 data points. The final architecture had only 127 weights.

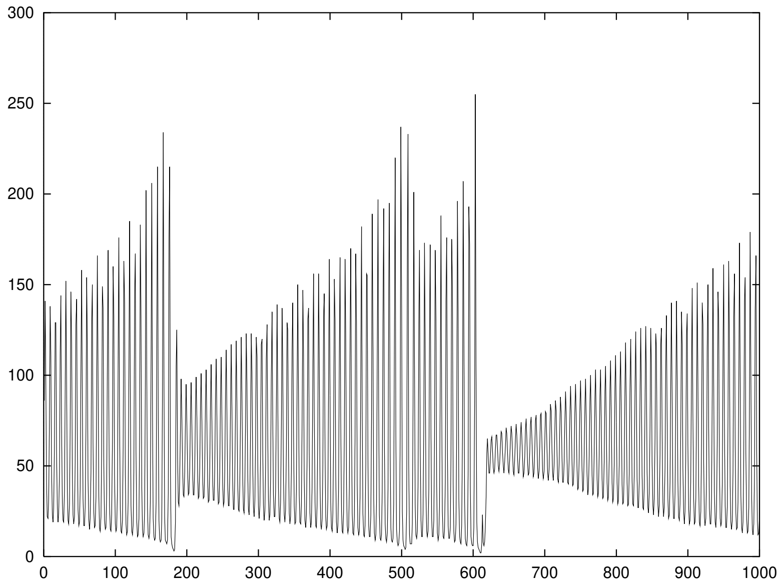

Figure 7: Original 1000 laser data.

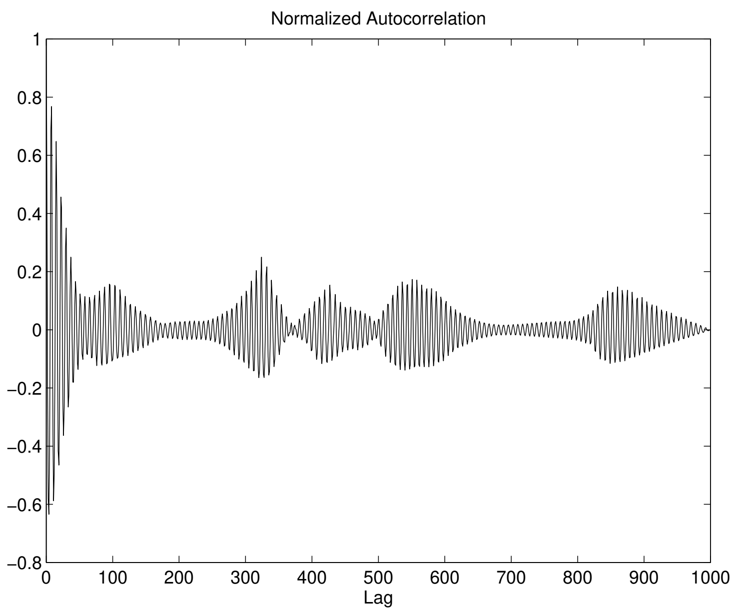

Figure 8: Normalized autocorrelation function of the 1000 laser data.

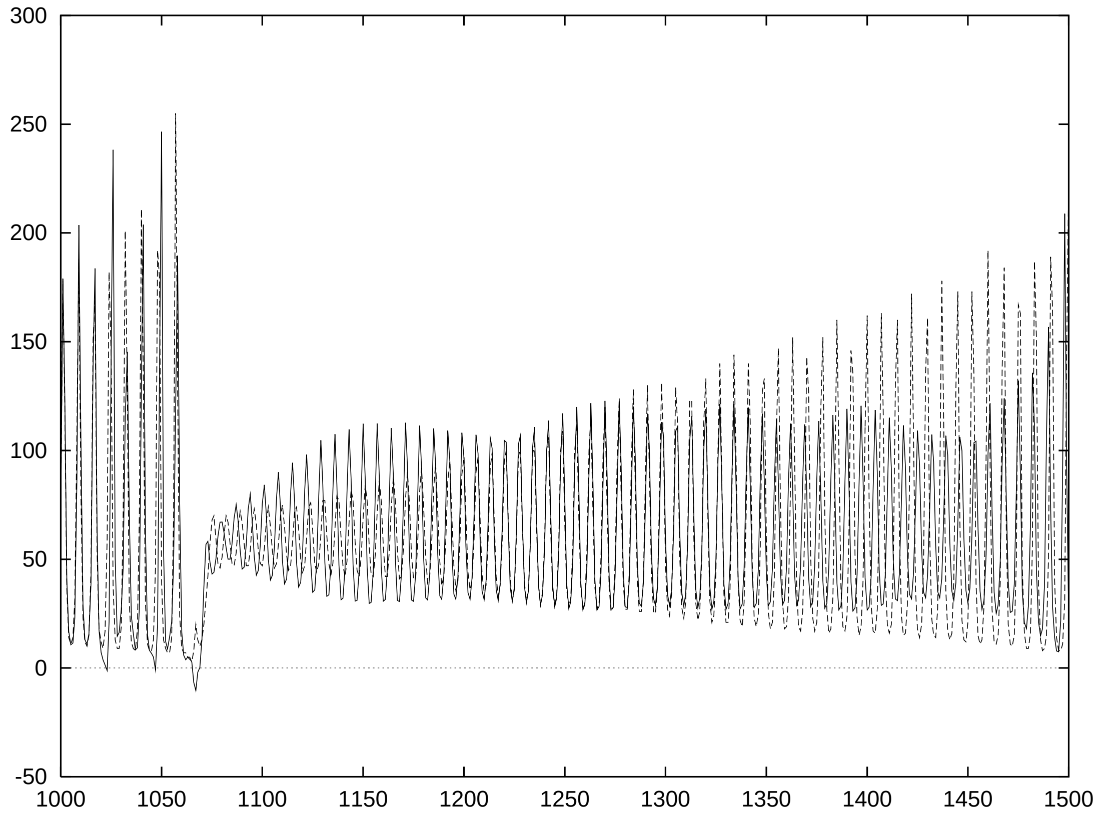

Figure 9: The solid line represents 500 data points predicted by the pruned network when using closed loop iteration.

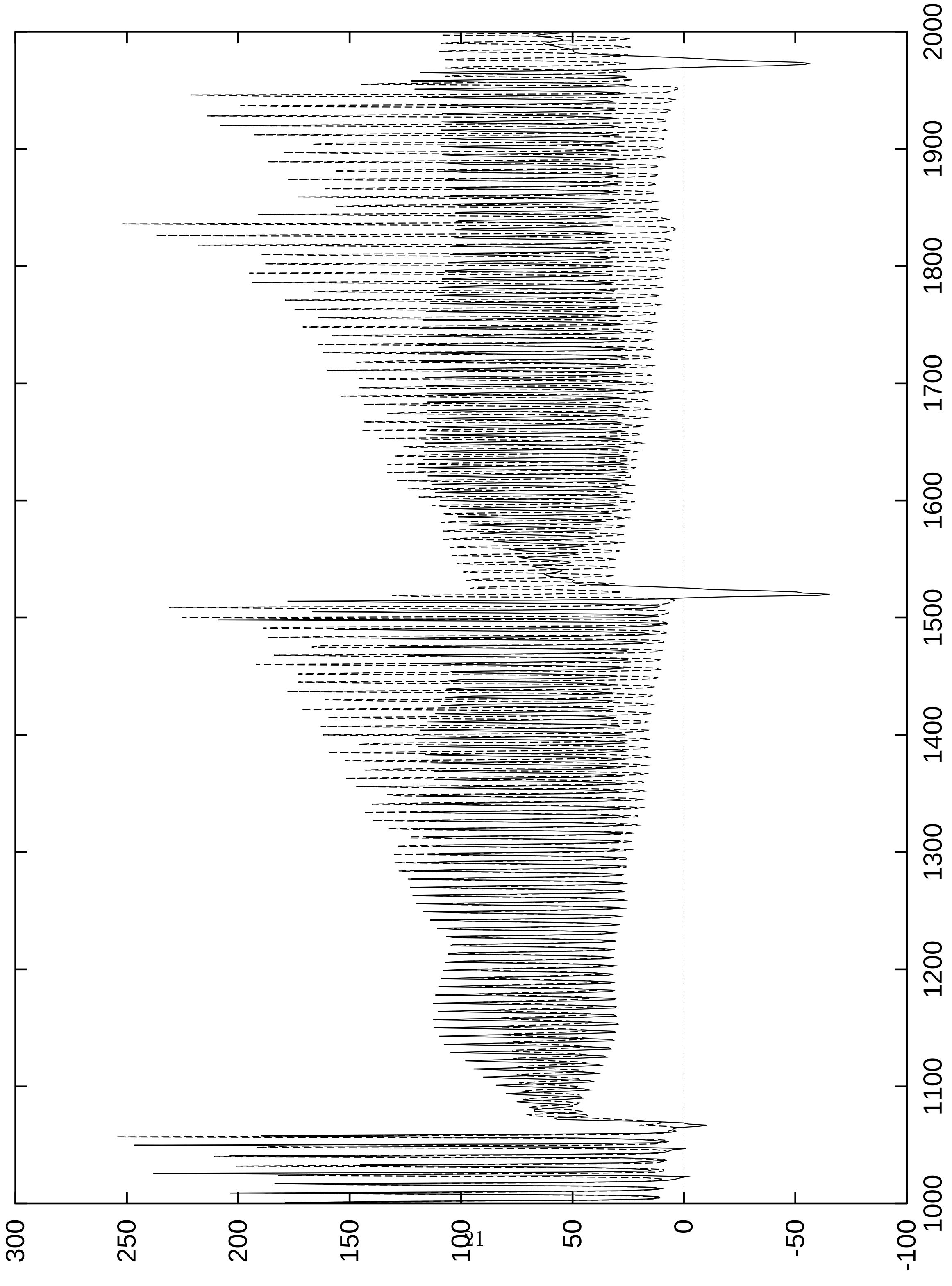

Figure 10: The solid line represents 1000 data points predicted by the pruned network when using closed loop iteration.

5 Conclusion

Determining of the proper architecture of a dynamical network is a difficult yet critical task. Not only can the nonlinear architectures affect the performance, but the memory architecture of dynamical model can have a significant impact on its dynamical behavior.

In this paper, we proposed a pruning-based algorithm to determine the embedded memory order of NARX recurrent neural networks and degenerate forms such as NSAR networks. This algorithm can also incorporate several useful heuristics, e.g. weight decay, which have been used extensively in static networks to optimize the nonlinear function. We also show that this algorithm can demonstrate improved performance on both nonlinear predictions and grammatical inference tasks. Furthermore, optimizing the memory architecture and the nonlinear function through pruning often results in sparsely connected architectures but with long time windows which are able to model the global features of the underlying system quite efficiently.

Though the embedding theorem demonstrates that the order of embedded memory should be large enough in order to provide a guarantee of forming a diffeomorphism mapping, choosing the proper memory order can be a challenge. Choosing different embedding memory architectures corresponds to giving different representations of the state space of the underlying system. The major issue is that such a representation plays an important role in solving problems. A good representation can make useful information explicit and easy to extract. In this work we only explored classic tapped delay memory structures. It would be interesting to see if similar results could be achieved for other memory models[3, 59]. However, a minimal representation does not necessarily mean a good representation. Two different representations can be equivalent in terms of expressive power, but may make a great difference in efficiency and effectiveness of problem-solving.

6 Acknowledgements

We would like to acknowledge useful discussions with G. Flake, D. Hush and J. Principe.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE trans. Automatic Control*, AC 19:716–723, 1974.

[2] W. Atmar. Notes on the simulation of evolution. *IEEE Transactions on Neural Networks*, 5(1):130–148, 1994.

[3] A.D. Back and A.C. Tsoi. A comparison of discrete-time operator models for nonlinear system identification. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 883–890. MIT Press, 1995.

[4] U. Bodenhausen and A. Waibel. The tempo 2 algorithm: Adjusting time-delays by supervised learning. In *Advances in Neural Information Processing Systems 3*, pages 155–161. Morgan Kaufmann, San Mateo, CA, 1991.

[5] Y. Cauvin. A back-propagation algorithm with optimal use of hidden units. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 519–526, 1989.

[6] S. Chen, S.A. Billings, and P.M. Grant. Non–linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990.

[7] Yung-Fu Cheng and Delores M. Etter. Analysis of an adaptive technique for modeling sparse systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(2):254–264, 1989.

[8] D.S. Clouse, C.L. Giles, B.G. Horne, and G.W. Cottrell. Time-delay neural networks: Representation and induction of finite state machines. *IEEE Transactions on Neural Networks*. Accepted.

[9] J. Connor, L.E. Atlas, and D.R. Martin. Recurrent networks and NARMA modeling. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 301–308, 1992.

[10] Y. Le Cun, J. S. Denker, and S. A. Solla. Handwritten digit recognition with a backpropagation network. In *Advances in Neural Information Processing Systems*, volume 2, pages 396–404, 1990.

[11] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, CA, 1990. Morgan Kaufmann Publishers.

[12] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[13] J. E. Dayhoff D.-T. Lin and P. A. Ligomenides. A learning algorithm for adaptive time-delays in a temporal neural network. Technical Report SRC-TR-92-59, Systems Research Center, University of Maryland, College Park, Maryland, 1992.

[14] Shawn P. Day and Michael R. Davenport. Continuous-time temporal back-propagation with adaptable time delays. *IEEE Transactions on Neural Networks*, 4(2):348–354, 1993.

[15] D. M. Etter and S. D. Stearns. Adaptive estimation of time delays in sampled data systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(3):582–587, 1981.

[16] David B. Fogel. An information criterion for optimal neural network selection. *IEEE Transactions on Neural Networks*, 2(5):490–497, 1991.

[17] David B. Fogel. Using evolutionary programming for modeling: An ocean acoustic example. *IEEE Journal of Oceanic Engineering*, 17(4):333–340, 1992.

[18] David B. Fogel. Evolutionary programming: An introduction and some current directions. *Statistics and Computing*, 4:113–129, 1994.

[19] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[20] C. L. Giles, B. G. Horne, and T. Lin. Learning a class of large finite state machines with a recurrent neural network. Technical Report UMIACS–TR–94–94 and CS–TR–3328, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, 1994.

[21] C.L. Giles, B.G. Horne, and T. Lin. Learning a class of large finite state machines with a recurrent neural network. *Neural Networks*, 8(9):1359–1365, 1995.

[22] C.L. Giles and C.W. Omlin. Pruning recurrent neural networks for improved generalization performance. *IEEE Transactions on Neural Networks*, 5(5):848–851, 1994.

[23] E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *J. Royal Stat. Soc. B.*, 41:190–195, 1979.

[24] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S.J. Hanson, J.D. Cowan, and C.L. Giles, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

[25] B.G. Horne and C.L. Giles. An experimental comparison of recurrent neural networks. In *Advances in Neural Information Processing Systems 7*, pages 697–704. MIT Press, 1995.

[26] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[27] B. Irie and S. Miyake. Capabilities of three–layered perceptrons. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 641–648, San Diego, CA, 1988. IEEE.

[28] E.D. Karnin. A simple procedure for pruning back–propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[29] A. Krogh and J.A. Hertz. A simple weight decay can improve generalization. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957, 1992.

[30] K.J. Lang, A.H. Waibel, and G.E. Hinton. A time–delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–44, 1990.

[31] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and signal modeling. Technical Report LA–UR–87–2662, Los Alamos National Laboratories, Los Alamos, New Mexico, 1987.

[32] I.J. Leontaritis and S.A. Billings. Input–output parametric models for non–linear systems: Part I: deterministic non–linear systems. *International Journal of Control*, 41(2):303–328, 1985.

[33] D.T. Lin, J.E. Dayhoff, and P.A. Ligomenides. Trajectory production with the adaptive time-delay neural network. *Neural Networks*, 8(3):447–461, 1995.

[34] T. Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

[35] Tsungnan Lin, B.G. Horne, and C.L. Giles. How memory orders effect the performance of narx networks. Technical Report UMIACS-TR-96-76 and CS-TR-3706, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, 1996.

[36] Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies is not as difficult with narx recurrent neural networks. In *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, 1996.

[37] L. Ljung. *System identification : Theory for the user*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[38] G. A. N. Mbamalu and M. E. El-Hawary. Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation. *IEEE Transactions on Power Systems*, 8(1):343–348, 1993.

[39] D.C. Montgomery and E.A. Peck. *Introduction to Linear Regression Analysis*. Wiley, New York, 1982.

[40] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Connection Science*, 11:3–26, 1989.

[41] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–27, 1990.

[42] Gil Nave and Arnon Cohen. Ecg compression using long-term prediction. *IEEE Transactions on Biomedical Engineering*, 40(9):877–885, 1993.

[43] R. Ma né. On the dimension of the compact invariant sets of certain nonlinear maps. In *Lecture Notes in Math. No. 898*, page 230. Springer-Verlag, 1981.

[44] S.J. Nowlan and G.E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992.

[45] Edward Ott, Tim Sauer, and James Yorke. *Coping with Chaos*. John Wiley and Sons, Inc, New York, NY, 1994.

[46] N. Packard, J. Crutchfield, D. Farmer, and R. Shaw. Geometry from a time series. *Physical Review Letter*, 45:712–715, 1980.

[47] M.W. Pedersen and L.K. Hansen. Recurrent networks: Second order properties and pruning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, 1995.

[48] S.-Z. Qin, H.-T. Su, and T.J. McAvoy. Comparison of four neural net learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, 3(1):122–130, 1992.

[49] Russel Reed. Pruning algorithms— a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

[50] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, IT-30:629–636, 1984.

[51] T. Sauer, J. Yorke, and M. Casdagli. Embedology. *Journal of Statistical Physics*, 65:579–616, 1991.

[52] R. Shibata. Various model selection techniques in time series analysis. In *Handbood of statistics*, volume 5, pages 179–187. Eds. Amsterdam: Elsevier, 1985.

[53] H.T. Siegelmann, B.G. Horne, and C.L. Giles. Computational capabilities of recurrent narx neural networks. *IEEE Trans. on Systems, Man and Cybernetics*, 1997. In press.

[54] H.-T. Su and T.J. McAvoy. Identification of chemical processes using recurrent networks. In *Proceedings of the American Controls Conference*, volume 3, pages 2314–2319, 1991.

[55] H.-T. Su, T.J. McAvoy, and P. Werbos. Long–term predictions of chemical processes using recurrent neural networks: A parallel training approach. *Industrial Engineering and Chemical Research*, 31:1338–1352, 1992.

[56] C. Svarer, L. K. Hansen, and J. Larsen. On design and evaluation of tapped-dealy neural architectures. In *IEEE International Conf. on Neural Networks*, pages 46–51, 1992.

[57] F. Takens. Detecting strange attractors in turbulence. In *Lecture Notes in Math. No. 898*, pages 366–381. Springer-Verlag, 1981.

[58] S.T. Venkataraman. On encoding nonlinear oscillations in neural networks for locomotion. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pages 14–20, 1994.

[59] Bert de Vries and Jose Principe. The gamma model - a new neural network for temporal processing. *Neural Networks*, 5(4):565–576, 1992.

[60] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time–delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

[61] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. Prediction the future: A connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

[62] A.S. Weigend and N.A. Gershenfeld. *Time Series Prediction: Forecasting the future and understanding the past*. Addison–Wesley, 1994.

[63] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting sunspots and exchange rates with connectionist networks. In M. Casdagli and S. Eubank, editors, *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, volume 12. Addison–Wesley, 1991.

[64] Gwo-Hsing Yu and Yow-Chano Lin. A methodology for selecting subset autoregressive time series models. *Journal of Time Series Analysis*, 12(4):363–373, 1989.