A Delay Damage Model Selection Algorithm for NARX Neural Networks

Tsung-Nan Lin, *Member, IEEE*, C. Lee Giles, *Fellow, IEEE,* Bill G. Horne, *Member, IEEE*, and Sun-Yuan Kung, *Fellow, IEEE*

Abstract— Recurrent neural networks have become popular models for system identification and time series prediction. Nonlinear autoregressive models with exogenous inputs (NARX) neural network models are a popular subclass of recurrent networks and have been used in many applications. Although 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.

Index Terms— Automata, autoregressive, embedding theory, gradient descent training, latching, long-term dependencies, memory, NARX networks, pruning, recurrent neural networks, tapped-delay lines, temporal sequences, time series.

I. INTRODUCTION

NONLINEAR autoregressive models with exogenous inputs (NARX) recurrent neural architectures [6], [44], as opposed to other recurrent neural models, have limited feedback architectures that 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 [56]. 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" [27].

Part of the reason can be attributed to the embedded memory of NARX networks. This embedded memory will appear as jump-ahead connections that 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* [36], [38]. Recently, it has been shown

that such embedded memory can help gradient-based learning in other recurrent neural network architectures [22].

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 [37]. In particular, forecasting performance could be seriously deficient if a model's memory architecture is either too little or too much memory. 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 [41].

According to the *embedding theorem* [48], [54], [60], 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 giving 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 underlying process, traditional statistical tests can be used, for example, Akaike information criterion (AIC) [1] and the minimum description length (MDL) principle [53]. 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 [25], [55]. Such model complexity and regularization methods are readily used for nonlinear models such as neural networks; see, for example [24], [42], and [67].

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 mod-

els in a time series of ocean acoustic data, but the algorithm can be computationally expensive when the underlying process is complex and nonlinear.

Alternatively, an adaptive algorithm that 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 an "adaptive delay filter," which included variable delays taps as well as variable gains, for modeling several sparse systems [7], [15]. Recently, others [4], [13], [14], [35] 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 [7], [15], 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 cannot always be determined in general. Therefore, there is no guarantee that such a modified algorithm for a nonlinear 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 [31], which are used extensively in static networks to optimize the nonlinear function. (For a survey of pruning methods for feedforward neural networks, see [52].)

The procedure of the algorithm starts with a NARX network with enough degrees of freedom in both input and output memory or taps and then deleting 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 with other recurrent neural network pruning procedures where recurrent nodes are pruned based on output values [23] and where second-order methods are used to prune input taps and single order feedback taps for fully recurrent neural networks [50]. 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 [10], [11].

II. NARX NEURAL NETWORK

An important and useful class of discrete-time nonlinear systems is the NARX model [6], [34], [39], [57], [58]

$$y(t) = f[u(t-D_u), \ldots, u(t-1), u(t)$$
$$\cdot y(t-D_y), \ldots, y(t-1)] \quad (1)$$

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 function f can be approximated by a multilayer perceptron (MLP), the resulting system is called a *NARX recurrent neural network* [6], [44]. Fig. 1 shows a NARX network with input memory of order 2 and output memory of order 3. It has been
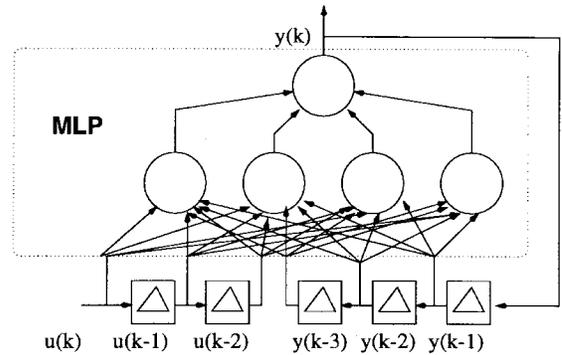
Fig. 1. NARX neural network with two input-memory taps and three output-memory taps.

demonstrated that NARX neural networks are well suited for modeling several nonlinear systems such as heat exchangers [6], waste water treatment plants [57], [58], catalytic reforming systems in a petroleum refinery [58], nonlinear oscillations associated with multilegged locomotion in biological systems [61], time series [9], and various artificial nonlinear systems [6], [44], [51].

When the output-memory order of NARX network is zero, a NARX network becomes a time delay neural network (TDNN) [32], [33], [63], which is simply a tapped delay line input into a MLP. In general, the TDNN implements a function of the form

$$y(t) = f[u(t-D_u), \cdots, u(t-1), u(t)]. \quad (2)$$

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)[1] time series model is defined as

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

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 [40], [45] and can easily be extended into nonlinear models. A nonlinear version of a SAR is the NSAR.[2]

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 the linear case [45], [69]. In designing the nonlinear sparse models, we can determine the memory order by applying the delay damage algorithm, which is described in the next section. The delay damage approach is based on the assumption that the memory order of an initial network should

---

[1] This does *not* mean synthetic aperture radar.

[2] 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 they standard. For example, NARX networks have also been called nonlinear autoregressive moving average (NARMA) networks. It would also be possible to refer to a NSAR as an NAR model.

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

III. 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 [26], [30], [43]; for details, see the survey paper by Reed [52]. Our method of calculating sensitivity is based on evaluating the second-order derivative of the cost function with respect to each memory order [50].

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

$$E = \tfrac{1}{2} \sum_p \sum_{\tau=t_0}^{t_1} [d^p(t) - o^p(t)]^2 \qquad (4)$$

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

$$x_i(t) = f[a_i(t)] \qquad (5)$$
$$a_i(t) = \sum_j w_{ij} x_j(t) \qquad (6)$$

where

$x_i(t)$ output of hidden node i;
$a_i(t)$ weighted-sum input;
f nonlinear function;
w_{ij} 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

$$\frac{\partial E}{\partial a_k(t_1)} = f_k'[a_k(t_1)]e_k(t_1) \qquad (7)$$

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 (*the injection error*) of output node k at time $t_0 < \tau < t_1$ will become

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

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

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

The Levenberg–Marquardt approximation was used by Le Cun *et al.* to drop the second term of (9). This will result
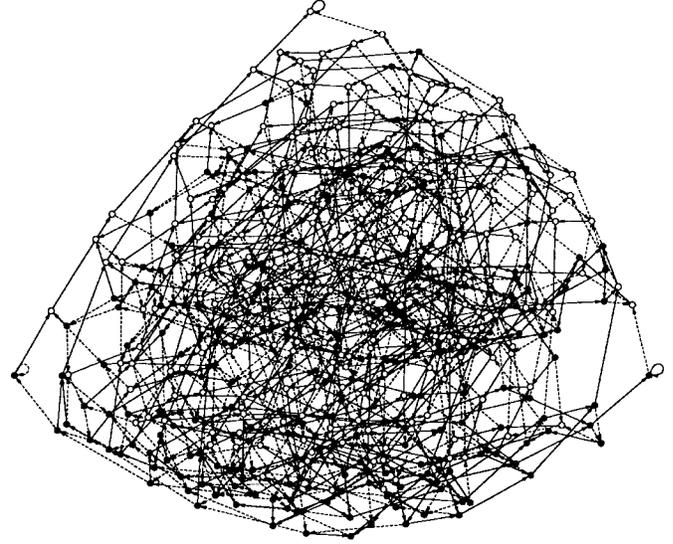
Fig. 2. A 512-state finite memory machine with input order 5 and output order 4.

in the same order of complexity as computing the first-order derivatives and nonnegative 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

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

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 can be defined such that there is a virtual layer of linear neurons between the memory taps and the first hidden layer. Each memory tap connects to each linear neuron with a fixed weight 1.0. The adjustable weights that previously connect memory taps and the nonlinear neurons will connect the virtual linear node and the nonlinear neurons. Therefore, the functionality of these two networks is the same. The error signal of each memory tap is propagated through the linear neuron with fixed weight 1.0. Therefore, the sensitivity of input-memory order $u(k - n)$ and output-memory order $y(k - m)$ can be described, respectively, as

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

$$\frac{\partial^2 E}{\partial y(k-m)^2} =$$
$$\sum_p \sum_\tau \left\{ [y^p(t-m)]^2 \sum_j w_{jm}^2 \frac{\partial^2 E}{\partial [a_j^p(\tau)]^2} \right\}. \qquad (12)$$
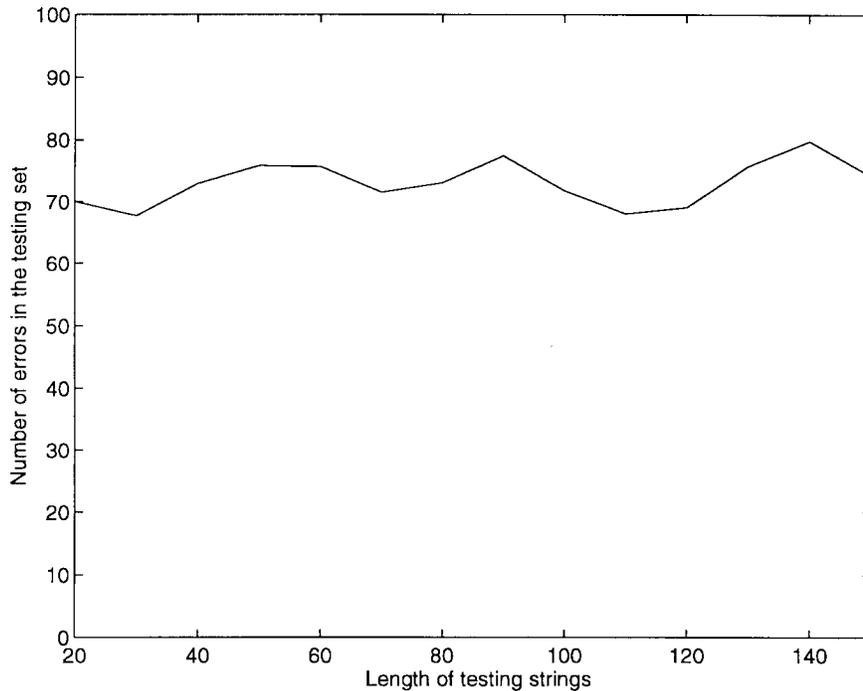
Fig. 3.   Number of errors in the testing set, which consists of 250 positive strings and 250 negative strings from length 20–150, as a function of the length of the strings for a NARX neural network trained on the data from the 512-state finite memory machine.

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 a certain error, total or otherwise, is reached.

IV. 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 NSAR's. 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], [31], [47], [64], [66] can be incorporated into the training algorithm. In the following experiments, networks are trained using weight decay [31]. All experiments were trained using back-propagation through time (BPTT) [68].

A. 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 that 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 [37]. 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

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

where y and u represent output and input, respectively, and $\bar{x}$ represents the complement of x. The FSM is shown in Fig. 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 $d+1$ (ten 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 0's and 1's 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 four hidden nodes, ten input taps, and ten 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 the 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] and [21]. For each of 50 experiments, the weights were randomly initialized within the range of [−0.5, 0.5].
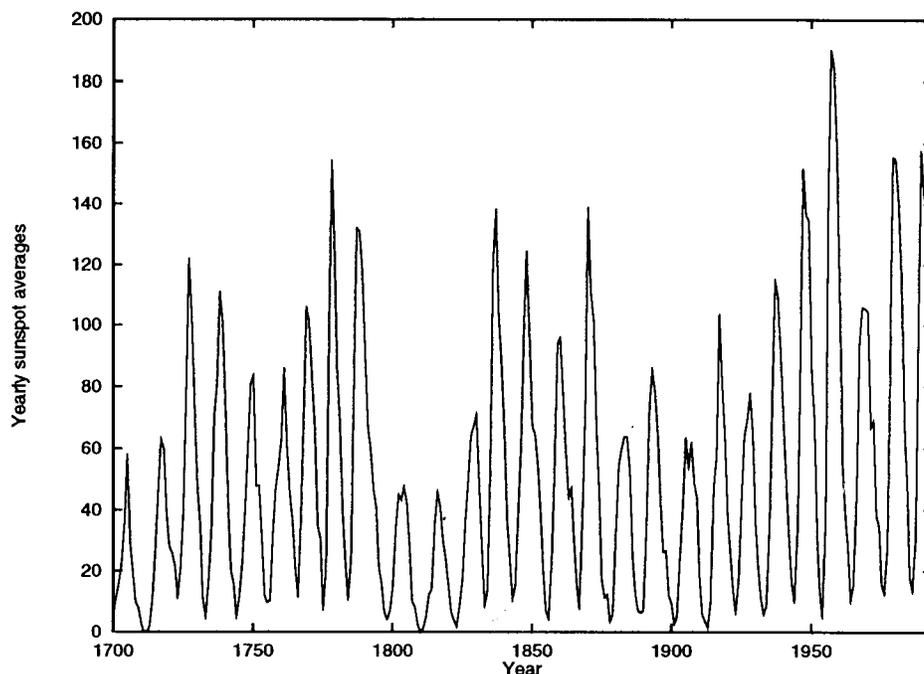
Fig. 4. Sunspot data for years 1700 to 1979.

TABLE I
NORMALIZED MSE FOR ONE-STEP AHEAD PREDICTION ON THE SUNSPOT DATA

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

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 ten, the number of errors was no longer zero and plotted in the Fig. 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 [37]. 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 III to prune the taps with small sensitivity measure always resulted in the minimal NARX architectures with five input memory orders and four 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 20–30, depending on the number of remaining memory orders. The time to retrain the pruned networks took, on 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*.

B. 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 [54]. The *state* of a deterministic dynamic system is the information necessary to determine the entire future evolution of the system. A time series 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 (1-D) 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

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

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 [46], [49], [60]. 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.* [54] 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.
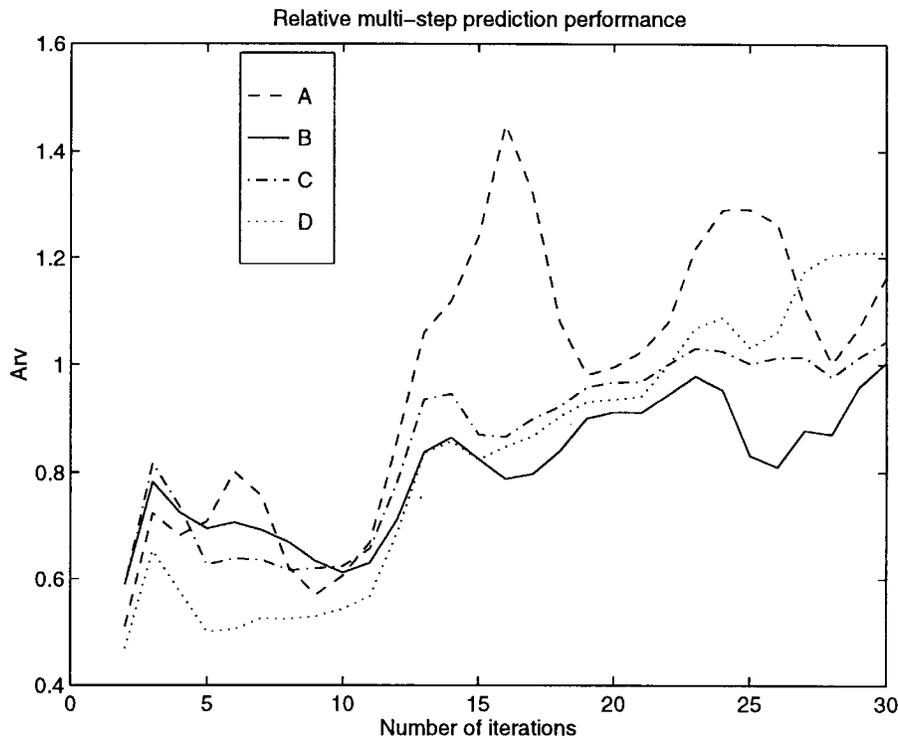
Fig. 5. Relative multistep prediction performance where results were normalized to a variance of 1535. Curves A and D are from previous work, and curves B and C are from the pruned networks; see Table I for more details. Both pruned network models perform better as the number of future predictions increase.
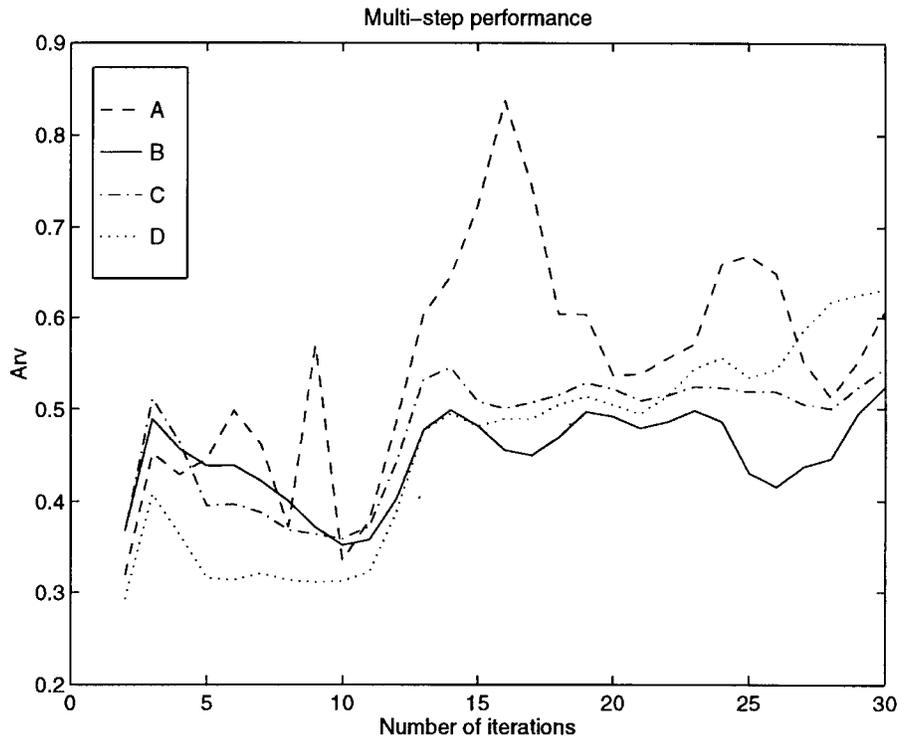
Fig. 6. Relative multistep prediction performance where results were normalized to the variance of the original data set. Curves A and D are from previous work, and curves B and C are from the pruned networks; see Table I for more details. Both pruned network models perform better as the number of future predictions increase.

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

$$y(t+1) = f[\mathbf{s}(t)]$$
$$= f\{y(t), y(t-T), y(t-2T), \cdots, y[t-(d-1)T]\}. \quad (15)$$
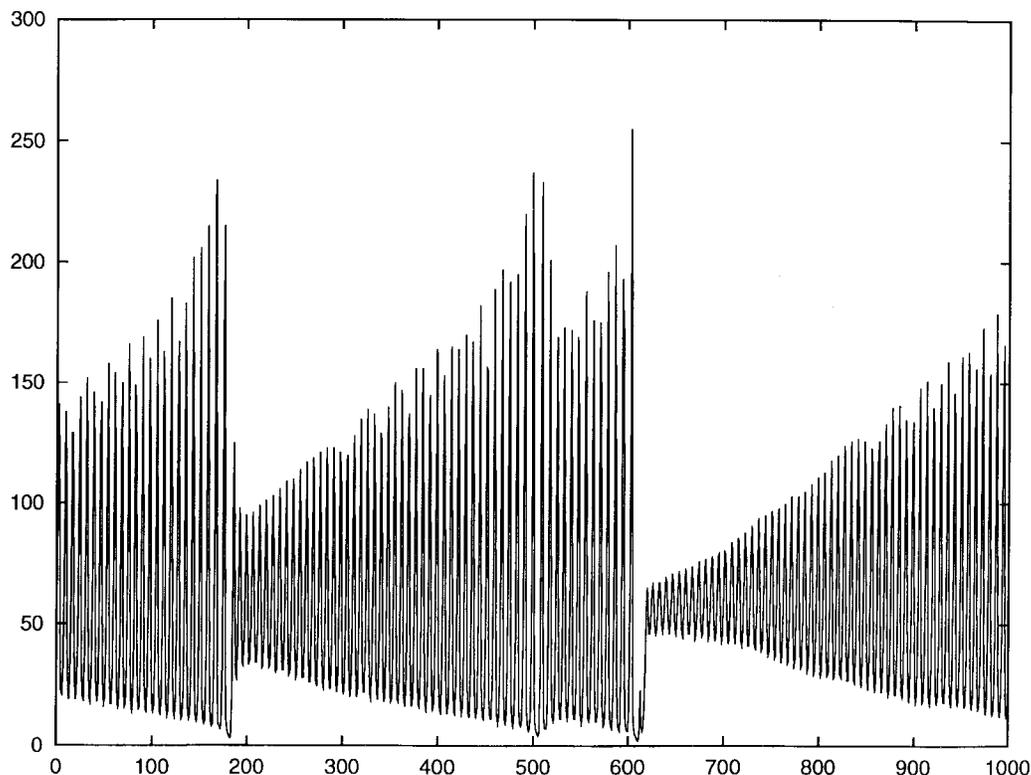
Fig. 7. Original 1000 laser data.

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], [28], [29]. 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 expressive 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 (NSAR's), 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.

C. Prediction of Sunspot Data

Sunspots are dark botches on the sun and yearly averages have been recorded since 1700. The series is shown in Fig. 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.* [64] trained a NSAR (or TDNN) network with an embedded dimension of 12 with eight hidden neurons and pruned the weights by adding a complexity term to the cost function. They were able to reduce a network with eight hidden nodes to three. The embedding dimension remained the same. Svarer *et al.* [59] 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.* [64] 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 predictor at the learning rate of 0.1 and weight decay of 0.001. After pruning the taps, the networks were retrained.
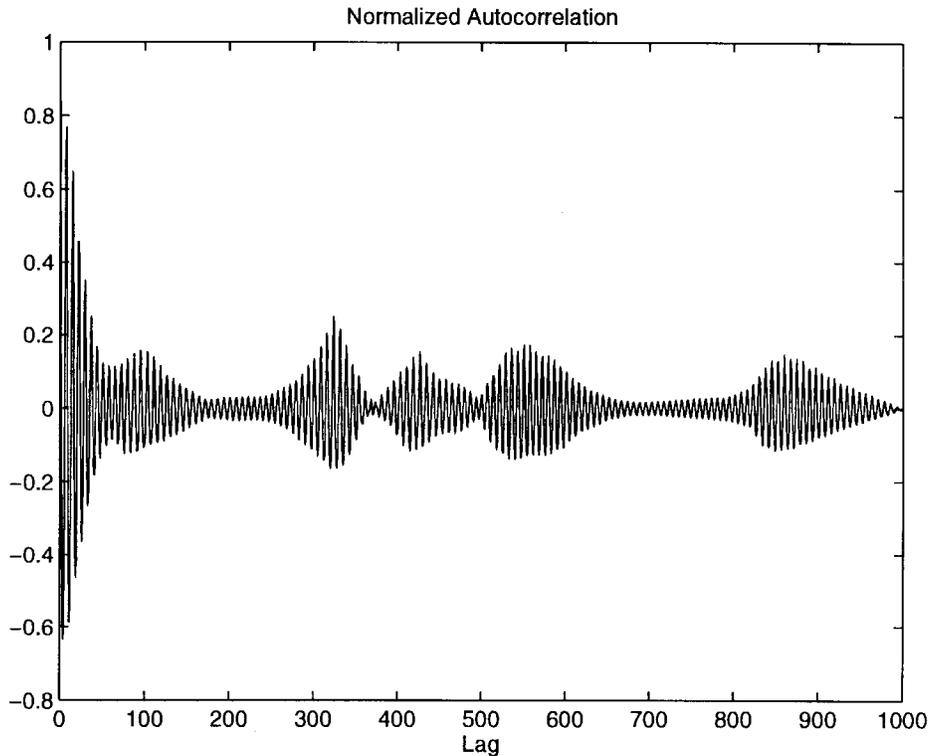
Fig. 8.   Normalized autocorrelation function of the 1000 laser data.

Our cost function was the normalized mean squared error defined as

$$\text{NMSE}(N) = \frac{\sum_{k \in S} (\text{target}_k - \text{prediction}_k)^2}{\sum_{k \in S} (\text{target}_k - \text{mean}_S)^2}$$

$$\cong \frac{1}{\sigma^2} \frac{1}{N} \sum_{k \in S} (x_k - \hat{x}_k)^2 \qquad (16)$$

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 [64]. Training was stopped when the normalized mean squared error was less than 0.1.

Our two best results from the architectures, which are denoted as B and C, are reported in Table I. (Recently, comparable improvements from committee machines have been reported [70].) 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 I lists the one-step-ahead prediction performance 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 consist 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.* [64] as a performance measure of the long-term behavior of the models. The *average relative I-times iterated prediction variance* is defined as

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

where

x_t real data point at time t;
M difference between the period of the prediction and I;
$\hat{x}_{t,I}$ 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 Fig. 5 and went through from 1921 to 1979. From Fig. 5, it can be seen that the networks B and C that 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 Fig. 6. Again, the networks B and C perform better as I is increased.

D. 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 of the Santa Fe competition. The data set consists of laser intensity collected from a laboratory experiment [65]. Although deterministic, its behavior is chaotic as seen in Fig. 7. Fig. 8 shows the normalized autocorrelation function of the first 1000 training data points.
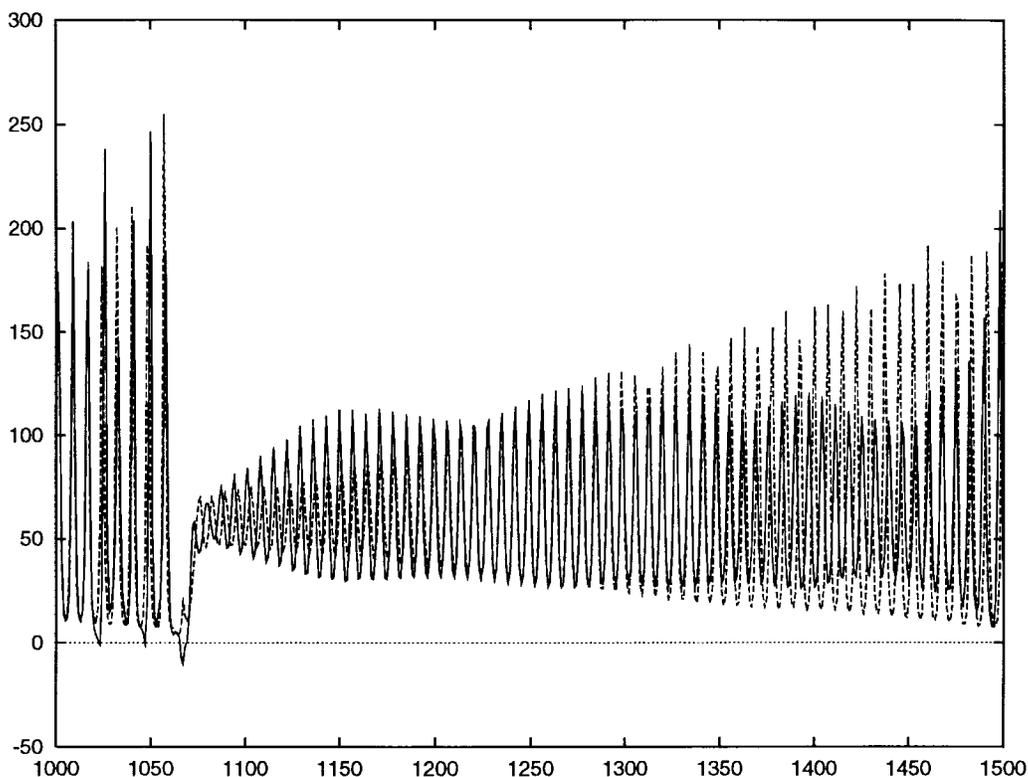
Fig. 9.   Solid line represents 500 data points predicted by the pruned network when using closed-loop iteration.

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 five neurons, and the second layer had three 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 13 subsets: three subsets from the past 100 data points (i.e., x_{t-99}, x_{t-100}, x_{t-101}), and ten 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 10% 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 the previous time step is restored, and the learning rate decreases by half until it reaches the minimum learning rate (10^{-8}); 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. By closed-loop iteration, we mean the network ran autonomously without any reference to the true data. Fig. 9 shows the 500-step iterated prediction from time $t = 1000$. Fig. 10 shows the 1000-step iterated prediction; note that the network was able to predict three collapses and that there were only two collapses in the training data set. We pruned five taps of the original network when trained on the first 1000 data points. The final architecture had only 127 weights. It is worth noting that our experiments with training the same network without pruning quickly led to predictions that were either noisy or a constant value.

V. 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 neu-
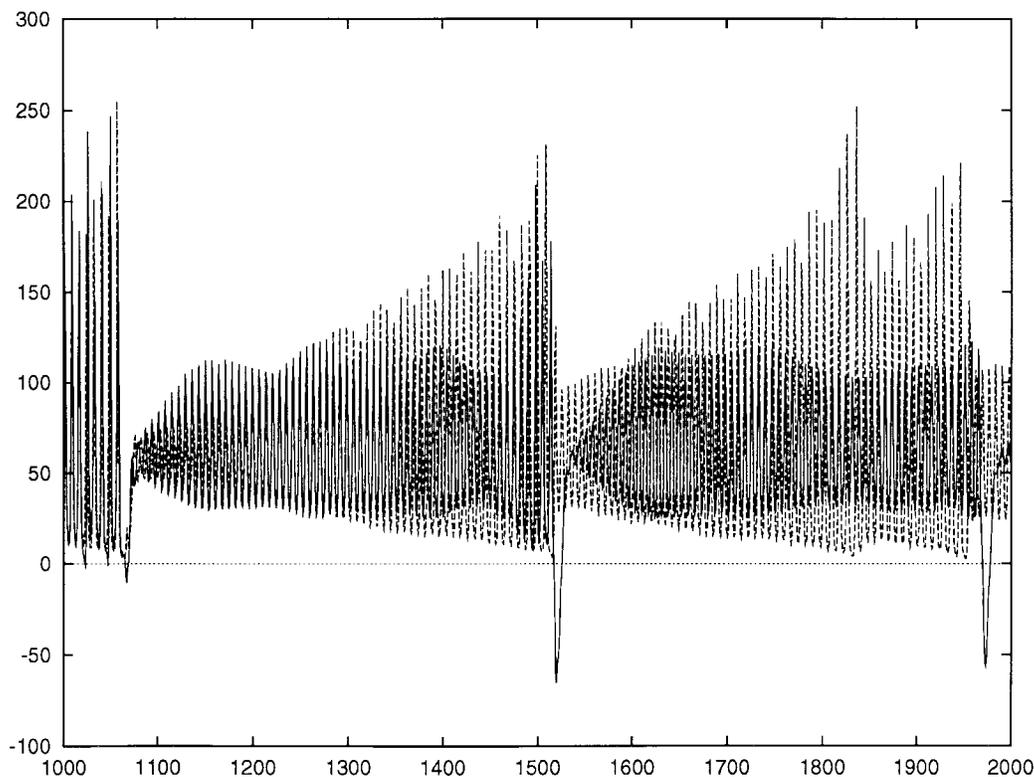
Fig. 10.   Solid line represents 1000 data points predicted by the pruned network when using closed-loop iteration.

ral 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 that are able to model the global features of the underlying system quite efficiently. It is an open question as to whether our pruning model will capture all important memory structure. Recent experiments on artificial problems imply that failure to capture important memory structure can lead to very poor generalization performance.

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], [62]. 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.

REFERENCES

[1] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Contr.*, vol. AC-19, pp. 716–723, 1974.
[2] W. Atmar, "Notes on the simulation of evolution," *IEEE Trans. Neural Networks*, vol. 5, pp. 130–148, 1994.
[3] A. D. Back and A. C. Tsoi, "A comparison of discrete-time operator models for nonlinear system identification," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds.   Cambridge, MA: MIT Press, 1995, pp. 883–890.
[4] U. Bodenhausen and A. Waibel, "The tempo 2 algorithm: Adjusting time-delays by supervised learning," in *Advances in Neural Information Processing Systems 3*.   San Mateo, CA: Morgan Kaufmann, 1991, pp. 155–161.
[5] Y. Cauvin, "A back-propagation algorithm with optimal use of hidden units," in *Advances in Neural Information Processing Systems 1*, D. Touretzky, Ed., 1989, pp. 519–526.
[6] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *Int. J. Contr.*, vol. 51, no. 6, pp. 1191–1214, 1990.
[7] Y.-F. Cheng and D. M. Etter, "Analysis of an adaptive technique for modeling sparse systems," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 254–264, Feb. 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 Trans. Neural Networks*, vol. 8, pp. 1065–1070, 1997.
[9] J. Connor, L. E. Atlas, and D. R. Martin, "Recurrent networks and NARMA modeling," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., 1992, pp. 301–308.
[10] Y. Le Cun, J. S. Denker, and S. A. Solla, "Handwritten digit recognition with a backpropagation network," *Advances in Neural Information Processing Systems*, 1990, vol. 2, pp. 396–404.

[11] ———, "Optimal brain damage," in *Advances in Neural Information Processing Systems,* D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, vol. 2, pp. 598–605.

[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, no. 4, pp. 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," Syst. Res. Center, Univ. Maryland, College Park, Tech. Rep. SRC-TR-92-59, 1992.

[14] S. P. Day and M. R. Davenport, "Continuous-time temporal back-propagation with adaptable time delays," *IEEE Trans. Neural Networks*, vol. 4, pp. 348–354, 1993.

[15] D. M. Etter and S. D. Stearns, "Adaptive estimation of time delays in sampled data systems," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 582–587, 1981.

[16] D. B. Fogel, "An information criterion for optimal neural network selection," *IEEE Trans. Neural Networks,* vol. 2, pp. 490–497, 1991.

[17] ———, "Using evolutionary programming for modeling: An ocean acoustic example," *IEEE J. Ocean. Eng.*, vol. 17, pp. 333–340, 1992.

[18] ———, "Evolutionary programming: An introduction and some current directions," *Statist. Comput.*, vol. 4, pp. 113–129, 1994.

[19] K. Funahashi, " On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, no. 3, pp. 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," Inst. Adv. Comput. Studies, Univ. Maryland, College Park, Tech. Rep. UMIACS-TR-94-94 and CS-TR-3328, 1994.

[21] ———, "Learning a class of large finite state machines with a recurrent neural network," *Neural Networks*, vol. 8, no. 9, pp. 1359–1365, 1995.

[22] C. L. Giles, T.-N. Lin, and B. G. Horne, "Remembering the past: The role of embedded memory in recurrent neural network architectures," in *Proc. Neural Networks Signal Processing VII 1997 IEEE Workshop*, J. Principe, L. Giles, N. Morgan, and E. Wilson, Eds. Piscataway, NJ: IEEE, 1997, pp. 34–43.

[23] C. L. Giles and C. W. Omlin, "Pruning recurrent neural networks for improved generalization performance," *IEEE Trans. Neural Networks,* vol. 5, pp. 848–851, 1994.

[24] F. Girosi and T. Poggio, "Networks and the best approximation property," *Biolog. Cybern.*, vol. 63, pp. 169–176, 1990.

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

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

[27] B. G. Horne and C. L. Giles, "An experimental comparison of recurrent neural networks," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 697–704.

[28] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[29] B. Irie and S. Miyake, "Capabilities of three-layered perceptrons," in *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, 1988, vol. 1, pp. 641–648.

[30] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 239–242, 1990.

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

[32] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, vol. 3, no. 1, pp. 23–44, 1990.

[33] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and signal modeling," Los Alamos Nat. Labs., Los Alamos, NM, Tech. Rep. LA-UR-87-2662, 1987.

[34] I. J. Leontaritis and S. A. Billings, "Input–output parametric models for nonlinear systems: Part I: Deterministic nonlinear systems," *Int. J. Contr.*, vol. 41, no. 2, pp. 303–328, 1985.

[35] D. T. Lin, J. E. Dayhoff, and P. A. Ligomenides, "Trajectory production with the adaptive time-delay neural network," *Neural Networks*, vol. 8, no. 3, pp. 447–461, 1995.

[36] T.-N. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in narx recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 7, pp. 1329–1338, 1996.

[37] T.-N. Lin, B. G. Horne, and C. L. Giles, "How memory orders effect the performance of narx networks," Inst. Adv. Comput. Studies, Univ.

[38] T. 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.* Cambridge, MA: MIT Press, 1996.

[39] L. Ljung, *System Identification: Theory for the User.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[40] G. A. N. Mbamalu and M. E. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Trans. Power Syst.*, vol. 8, pp. 343–348, 1993.

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

[42] J. E. Moody, "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems," in *Advances in Neural Information Processing Systems 4,* J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 847–854.

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

[44] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[45] G. Nave and A. Cohen, "Ecg compression using long-term prediction," *IEEE Trans. Biomed. Eng.*, vol. 40, pp. 877–885, 1993.

[46] R. Mané, "On the dimension of the compact invariant sets of certain nonlinear maps," in *Lecture Notes in Math. no. 898.* New York: Springer-Verlag, 1981, p. 230.

[47] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Comput.*, vol. 4, no. 4, pp. 473–493, 1992.

[48] E. Ott, T. Sauer, and J. Yorke, *Coping with Chaos.* New York: Wiley, 1994.

[49] N. Packard, J. Crutchfield, D. Farmer, and R. Shaw, "Geometry from a time series," *Phys. Rev. Lett.*, vol. 45, pp. 712–715, 1980.

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

[51] S.-Z. Qin, H.-T. Su, and T. J. McAvoy, "Comparison of four neural net learning methods for dynamic system identification," *IEEE Trans. Neural Networks*, vol. 3, pp. 122–130, 1992.

[52] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, 1993.

[53] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 629–636, 1984.

[54] T. Sauer, J. Yorke, and M. Casdagli, "Embedology," *J. Statist. Phys.*, vol. 65, pp. 579–616, 1991.

[55] R. Shibata, "Various model selection techniques in time series analysis," in *Handbook of Statistics.* Amsterdam, The Netherlands: Elsevier, 1985, vol. 5, pp. 179–187.

[56] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent narx neural networks," *IEEE Trans. Syst., Man Cybern., pt. B*, vol. 27, p. 208, Apr. 1997.

[57] H.-T. Su and T. J. McAvoy, "Identification of chemical processes using recurrent networks," in *Proc. Amer. Contr. Conf.*, 1991, vol. 3, pp. 2314–2319.

[58] H.-T. Su, T. J. McAvoy, and P. Werbos, "Long-term predictions of chemical processes using recurrent neural networks: A parallel training approach," *Ind. Eng. Chem. Res.*, vol. 31, pp. 1338–1352, 1992.

[59] C. Svarer, L. K. Hansen, and J. Larsen, "On design and evaluation of tapped-delay neural architectures," in *Proc. IEEE Int. Conf. Neural Networks*, 1992, pp. 46–51.

[60] F. Takens, "Detecting strange attractors in turbulence," in *Lecture Notes Math. no. 898.* New York: Springer-Verlag, 1981, pp. 366–381.

[61] S. T. Venkataraman, "On encoding nonlinear oscillations in neural networks for locomotion," in *Proc. Eighth Yale Workshop Adaptive Learning Syst.*, 1994, pp. 14–20.

[62] B. de Vries and J. Principe, "The gamma model—A new neural network for temporal processing," *Neural Networks*, vol. 5, no. 4, pp. 565–576, 1992.

[63] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 328–339, 1989.

[64] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Prediction the future: A connectionist approach," *Int. J. Neural Syst.*, vol. 1, no. 3, pp. 193–209, 1990.

Maryland, College Park, Tech. Rep. UMIACS-TR-96-76 and CS-TR-3706, 1996.

[65] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past.* Reading, MA: Addison-Wesley, 1994.

[66] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting sunspots and exchange rates with connectionist networks," in *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity,* M. Casdagli and S. Eubank, Eds. Reading, MA: Addison-Wesley, 1991, vol. 12.

[67] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting," in *Advances in Neural Information Processing Systems 3,* R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 875–882.

[68] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in *Back-Propagation: Theory, Architectures and Applications,* Y. Chauvin and D. E. Rumelhart, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1995, ch. 13, pp. 433–486.

[69] G.-H. Yu and Y.-C. Lin, "A methodology for selecting subset autoregressive time series models," *J. Time Series Anal.*, vol. 12, no. 4, pp. 363–373, 1989.

[70] E. A. Wan, "Combining fossil and sunspot data: Committee predictions," in *Proc. Int. Conf. Neural Networks*, 1997, pp. 2176–2180.

Bill G. Horne (M'93) received the B.S.E.E. degree from the University of Delaware, Newark, in 1986 and the M.S.E.E. and Ph.D. degrees in electrical engineering from the University of New Mexico, Albuquerque, in 1988 and 1993, respectively. The topic of his Ph.D. dissertation was the circuit complexity of neural networks.

Previously, he was a scientist at NEC Research Institute, Princeton, NJ. Currently, he is Chief Technical Officer, MakeWaves, Inc., Watchung, NJ, and is a consultant to NEC Research Institute. Current research interests involve neural networks, the World-Wide Web, image watermarking, and theoretical computer science.

Sun-Yuan Kung (F'88), for photograph and biography, see p. 2708 of this issue of this TRANSACTIONS.

Tsung-Nan Lin (M'96) received the B.S.E.E. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1989 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1993 and 1996, respectively.

From 1993 to 1996, he was member of NEC Research Institute, Princeton, NJ. Currently, he is a Senior Member of Technical Staff at Epson Palo Alto Laboratory, Palo Alto, CA. His primary research interests include neural networks, pattern recognition, time series prediction, color matching/management, and digital photography.

C. Lee Giles (F'97) received the B.S. and B.A. degrees from the University of Tennessee, Knoxville, and Rhodes College, Memphis, TN, respectively. He received the M.S. degree from the University of Michigan, Ann Arbor, and the Ph.D. degree in optical sciences from the University of Arizona, Tucson.

He is a Senior Research Scientist in Computer Science at NEC Research Institute, Princeton, NJ, and an adjunct faculty member at the Institute for Advanced Computer Studies, University of Maryland, College Park. His research interests include neural networks, machine learning and AI; hybrid systems and integrating neural networks with intelligent systems and intelligent agents; AI and machine learning applications in finance, communications, World-Wide Web and computer systems and software; and optics in computing and processing. Previously, he was a Program Manager at the Air Force Office of Scientific Research, Washington, DC, where he initiated and managed research programs in neural networks and in optics in computing and processing. Before that, he was a research scientist at the Naval Research Laboratory, Washington, DC, and an Assistant Professor of Electrical and Computer Engineering at Clarkson University, Potsdam, NY. During part of his graduate education, he was a research engineer at Ford Motor Scientific Research Laboratory.

Dr. Giles has served or is currently serving on the editorial boards of IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON NEURAL NETWORKS, *Journal of Parallel and Distributed Computing, Neural Networks, Neural Computation, Optical Computing and Processing, Applied Optics*, and Academic Press. In 1994, he co-edited a special issue for IEEE TRANSACTIONS ON NEURAL NETWORKS on "Dynamic Recurrent Neural Networks." He is a member of AAAI, ACM, INNS, and the OSA.