Using Neural Networks to Dynamically Model Chaotic Events Such as Sea Clutter

Making Sense of a Complex World

Complexity is an idea that we all encounter on a daily basis in highly diverse contexts throughout our lives [1]. It arises primarily from massive interactions among many different parts of a nonlinear system or in nonlinear physical phenomena that are intrinsically complex. Examples of complex systems include biological systems, weather-related phenomena, fluid turbulence, radar backscatter from the sea surface, and multipath in a mobile communications system. Complexity is an inseparable part of the world of nonlinear dynamical systems. As with linear systems, modeling plays a prominent role in the study of dynamical systems.

Dynamic modeling [2] is defined as identification of the mapping \( F: \mathbb{R}^d \rightarrow \mathbb{R}^n \) that describes the behavior of an unknown dynamical system of dimensionality \( d \) (Figure 1). The relation between
the model system and the unknown system must be stated in terms that are different from those we are accustomed to in linear systems, since the nonlinear model cannot be described by a transfer function.

Our interest in this article is the dynamic modeling of a time series produced by a physical system that is known to be chaotic. The time series may be real-valued, representing a single component of the system, or complex-valued, representing two independent components of the system. In this article, we confine our attention to the real-valued case.

The time series of interest may be, for example, the radar backscatter from the sea surface (i.e., sea clutter) produced by a noncoherent radar with its antenna dwelling onto a fixed patch of the sea surface. The time series of Fig. 2 shows a typical waveform of sea clutter. In an exhaustive study recently completed on real-life sea clutter data, it has been demonstrated in the most convincing way possible that the underlying dynamics of sea clutter are indeed chaotic [3]. Unfortunately, we do not know the system of nonlinear differential equations responsible for the generation of sea clutter, and the best we can do at present is to build a nonlinear model. It is largely due to this lack of knowledge that, for over half a century, sea clutter has been viewed as a stochastic process with a variety of probability distributions (mostly ad-hoc) proposed for its statistical characterization [4-6].

Given the time series of Fig. 2, how can we build a model that captures the underlying nonlinear dynamics responsible for its generation? The answer lies in the nonlinear theory of dynamic reconstruction discussed in the next section.

Dynamic-Reconstruction Theory

The primary reason for using dynamic reconstruction is to make “physical sense” from a given chaotic time series, while bypassing the need for a detailed mathematical knowledge of the underlying dynamics. A fundamental result in dynamic reconstruction theory is a geometric theorem called the delay-embedding theorem due to Takens [7] and Mañé [8]. Takens’ paper is difficult to read by non-mathematicians, and Mañé’s paper is written in even more abstract terms. Takens considered a noise-free situation, focusing on delay-coordinate maps or predictive models that are constructed from a time series representing an observable from a dynamical system. In particular, Takens [7] showed that if the dynamical system and the observable are generic, then the delay-coordinate map from a $d$-dimensional smooth compact manifold to $R^{2d}$ is a diffeomorphism on that manifold, where $d$ is the dimension of the phase (state) space of the dynamical system.

For an interpretation of Takens’ theorem in signal-processing terms, first consider an unknown dynamical system
whose evolution in discrete time is described by the nonlinear difference equation

$$x(n + 1) = F(x(n))$$ (1)

where $x(n)$ is the $d$-dimensional state vector of the system at time $n$, and $F(\cdot)$ is a vector-valued function. It is assumed here that the sampling period is normalized to unity. Let the time series, $\{y(n)\}$, observable at the output of the system be defined in terms of the state vector $x(n)$ as follows:

$$y(n) = h(x(n)) + w(n)$$ (2)

where $h(\cdot)$ is a scalar-valued function, and $w(n)$ denotes additive noise. The noise, $w(n)$, accounts for the combined effects of imperfections and imprecisions in the observable $y(n)$. Equations (1) and (2) describe the state-space behavior of the dynamical system. According to Takens' theorem, the geometric structure of the multi-variable dynamics of the system can be unfolded from the observable $y(n)$ with $w(n) = 0$ in a $D$-dimensional space constructed from the new vector

$$y_R(n) = [y(n), y(n - \tau), \ldots, y(n - (D - 1)\tau)]^T,$$ (3)

where $\tau$ is a positive integer called the normalized embedding delay. That is, given the observable $y(n)$ for varying discrete-time $n$, which pertains to a single component of an unknown dynamical system, dynamic reconstruction is possible using the $D$-dimensional vector $y_R(n)$ provided that $D \geq 2d + 1$, where $d$ is the dimension of the phase (state) space of the system. Hereafter, we refer to this statement as the delay-embedding theorem. The condition $D \geq 2d + 1$ is a sufficient but not necessary condition for dynamic reconstruction. The procedure for finding a suitable $D$ is called embedding, and the minimum integer $D$ that achieves dynamic reconstruction is called the embedding dimension; it is denoted by $D_e$.

The delay-embedding theorem has a powerful implication: evolution of the points $y_R(n) \rightarrow y_R(n + 1)$ in the reconstruction space follows that of the unknown dynamics $x(n) \rightarrow x(n + 1)$ in the original phase space. That is, many important properties of the unobservable state vector $x(n)$ are reproduced without ambiguity in the reconstruction space defined by $y_R(n)$. However, for this important result to be attainable, we need reliable estimates of the embedding dimension, $D_e$, and the normalized embedding delay, $\tau$, as summarized here [11]:

- The sufficient condition $D \geq 2d + 1$ makes it possible to undo the intersections of an orbit of the attractor with itself, which arise from projection of that orbit to lower dimensions. The embedding dimension, $D_e$, can be less than $2d + 1$. The recommended procedure is to estimate $D_e$ directly from the observable data. A reliable method for estimating $D_e$ is the method of false nearest neighbors [11, 12].

- Unfortunately, the delay-embedding theorem has nothing to say on the choice of the normalized embedding delay $\tau$. In fact, it permits the use of any $\tau$ so long as the available time series is infinitely long. In practice, however, we always have to work with observable data of finite length. The proper prescription for choosing $\tau$ is to recognize that the normalized embedding delay, $\tau$, should be large enough for $y(n)$ and $y(n - \tau)$ to be essentially independent of each other so as to serve as coordinates of the reconstruction space, but not so independent as to have no correlation with each other. This requirement is best satisfied by using the particular $\tau$ for which the mutual information between $y(n)$ and $y(n - \tau)$ attains its first minimum [11, 13].

Another limitation of the delay-embedding theorem as stated above is that it presumes a noise-free situation, that is, $w(n) = 0$ in Eq. (2). Sauer et al. [9] have extended the delay-embedding theorem to deal with noisy data by building on the previous results of Whitney [10] and Takens [7]. In particular, the general case of dynamic reconstruction using moving averages of delay coordinates is considered; discussion of the paper by Sauer et al. is pursued further later in this article.
The dynamic invariants measure global properties of the attractor, so they should gauge the success of dynamic modeling.

**Recursive Prediction**

From the discussion presented above, the dynamic-reconstruction problem may be interpreted as one of representing the signal dynamics properly (the embedding step), as well as the identification of a predictive mapping [14-16]. Thus, in practical terms, we have the following network topology for dynamic modeling:

1. **A short-term memory structure** to perform the embedding, whereby the reconstruction vector, \( y_r(n) \), is defined in terms of the observable \( y(n) \) and its delayed versions; see Eq. (3).

2. **A multiple input, single output adaptive nonlinear system trained as a one-step predictor** (e.g., neural network) to identify the unknown mapping \( f: \mathbb{R}^D \rightarrow \mathbb{R} \), which is defined by

\[
y(n + 1) = f(y_r(n))
\]

The predictive mapping described in Eq. (4) is the center piece of dynamic modeling: Once it is determined, the evolution \( y_r(n) \rightarrow y_r(n + 1) \) becomes known, which, in turn, determines the unknown evolution \( x(n) \rightarrow x(n + 1) \).

Presently, we do not have a rigorous theory to help us decide if the nonlinear predictor has successfully identified the mapping, \( f \). In linear prediction, minimizing the mean-square value of the prediction error leads to an accurate model. However, a chaotic time series is different. Two trajectories in the same attractor are vastly different on a sample-by-sample basis, so minimizing the mean-square value of the prediction error is a necessary but not a sufficient condition for a successful mapping.

The dynamic invariants (namely, correlation dimension, and Lyapunov exponents) measure global properties of the attractor, so they should gauge the success of dynamic modeling. Hence, a pragmatic approach for testing the dynamic model is to feed it with a point on the attractor, and feed the output back to its input as an autonomous system as illustrated in Fig. 3. Such an operation is called *iterated prediction* or *recursive prediction*. Once the initialization is completed, the output of the autonomous system is a realization of the dynamic reconstruction process. This, of course, presumes that the predictor has been designed properly in the first place.

We say that dynamic reconstruction performed by means of the autonomous system described in Fig. 3 is successful if the following two conditions are satisfied:

1. **Short-term behavior.** Once the initialization is completed, the reconstructed time series \( \{\hat{y}(n)\} \) in Fig. 3 closely follows the original time series \( \{y(n)\} \) for a period of time, on the average, equal to the horizon of predictability determined from the Lyapunov spectrum of the process.

2. **Long-term behavior.** The dynamic invariants computed from the reconstructed time series \( \{\hat{y}(n)\} \) closely match the corresponding ones computed from the original time series \( \{y(n)\} \).

To gauge the long-term behavior of the reconstructed dynamics, we need to estimate (1) the correlation dimension as a measure of attractor complexity, and (2) the Lyapunov spectrum as a framework for assessing sensitivity to initial conditions and estimating the Kaplan-Yorke dimension. The Kaplan-Yorke dimension should have a value close to that of the correlation dimension. These issues and those relating to the estimation of \( D_k \) and \( \tau \) are discussed in the companion paper by Abarbanel et al. [17].

**Two Possible Formulations for Recursive Prediction**

The reconstruction vector, \( y_r(n) \), defined in Eq. (3) is of dimension \( D_k \), assuming that the dimension \( D \) is set equal to the embedding dimension, \( D_e \). The size of the delay-line required to perform the embedding is \( tD_e \). But the delay line is required to provide only \( D_e \) outputs (the dimension of the reconstruction space); that is, we use \( \tau \) equally spaced taps, representing sparse connections.

Alternatively, we may define the reconstruction vector, \( y_r(n) \), as a full \( m \)-dimensional vector as follows:

\[
y_r(n) = [y(n), y(n - 1), \ldots, y(n - m + 1)]
\]

where \( m \) is an integer defined by

\[
m \geq D_k \tau
\]

This second formulation of the reconstruction vector, \( y_r(n) \), supplies more information to the predictive model than that provided by Eq. (3) and may therefore yield a more accurate dynamic reconstruction. Note, however,
that both formulations share a common feature: their compositions are uniquely defined by knowledge of the embedding dimension, $D_e$. In any event, it is wise to use the minimum permissible value of $D_e$, namely $D_e$, so as to minimize the effect of additive noise $w(n)$ on the quality of dynamic reconstruction.

**Dynamic Reconstruction is an Ill-posed Filtering Problem**

Two problems are said to be inverse to each other if the formulation of either one of them requires full or partial knowledge of the other [18]. According to this definition, it is arbitrary which of the two problems is referred to as the direct problem and which one is referred to as the inverse problem. In practice, however, it is customary to find that one of the two problems has been studied earlier, and perhaps in more detail. On this basis, we may refer to this earlier problem as the direct problem, in which case the other problem is referred to as the inverse problem.

For example, in the case of sea clutter, a great deal of work has been done on the study of its physics and statistical characteristics [5, 6]. It would therefore seem logical to refer to the dynamical process (i.e., the physical interaction between an incident electromagnetic wave and the sea surface) as the direct problem. The product of this dynamical process is a time series that represents the observable, on the basis of which we try to solve the inverse problem. That is, given a radar time series of sufficient length and sufficiently high signal-to-noise ratio, we wish to construct a nonlinear network that models the underlying dynamics responsible for generating the observable time series, which is a result of sea clutter itself. Figure 4 illustrates the relationship between direct and inverse problems in the case of sea clutter.

There is another, and perhaps even more important difference between direct and inverse problems. In particular, Hadamard introduced the concept of a well-posed problem. We say a problem is a well-posed one in the sense of Hadamard if it satisfies three conditions:

1. **Existence.** For every input vector $x \in X$, there exists an output vector $y = F(x)$, where $y \in Y$.

2. **Uniqueness.** For any pair of input vectors $x, z \in X$, we have $F(x) = F(z)$ if and only if $x = z$.

3. **Continuity.** The mapping is continuous, that is, for any $\epsilon > 0$ there exists $\delta = \delta(\epsilon)$ such that the condition $\delta \leq |x - z|$ implies that $|F(x) - F(z)| < \epsilon$, where $\delta(\cdot)$ is the symbol for distance between the two arguments in their respective spaces.

If any of these three conditions is not satisfied, the problem is said to be ill-posed.

The dynamic-reconstruction problem is, in reality, an ill-posed inverse problem for one or more of the following reasons. First, for some unknown reason the existence condition may be violated. Second, there may not be sufficient information in the observable time series to reconstruct the nonlinear dynamics uniquely; hence, the uniqueness criterion is violated. Third, the unavoidable presence of additive noise or some form of imprecision in the observable time series adds uncertainty to the dynamic reconstruction. In particular, if the noise level is too high, it is possible for the continuity criterion to be violated.

How then do we make the dynamic reconstruction problem to be well-posed? The answer lies in the inclusion of some form of prior knowledge about the input-output mapping as an essential requirement. In other words, some form of constraints would have to be imposed on the predictive model designed for solving the dynamic-reconstruction problem. One effective way in which this requirement can be satisfied is to invoke Tikhonov's regularization theory [19].

Another issue that needs to be considered is the ability of the predictive model to solve the inverse problem with sufficient accuracy. In this context, the use of a neural network to build the predictive model is appropriate. In particular, the universal approximation property of a multilayer perceptron [20] or that of a radial-basis function (RBF) network [21] means that we can take care of the issue of reconstruction accuracy by using one or the other of these neural networks. In addition, however, we need the solution to be regularized for the reasons explained above. In theory, both multilayer perceptrons and radial-basis function networks lend themselves to the use of regularization; in practice, it is in radial-basis function networks that we find regularization theory included in a mathematically tractable manner as an integral part of their design [22, 23]. Accordingly, in the next section we focus on regularized RBF networks as the basis for solving the dynamic-reconstruction problem.

**Regularized RBF Network for Solving the Dynamic Reconstruction Problem**

In the context of the network choice that we have made, the dynamic-reconstruction problem may be stated as fol-
Radial-basis function network.

5. Given a time series of total length \( N \), design a regularized RBF network (model) with an input layer of size \( n_z \) defined in accordance with Eq. (6), such that the mean-square error between the actual sample \( y(n + 1) \) and the one-step prediction \( \hat{y}(n + 1) \) produced in response to the input \( \{ y(n), y(n - 1), \ldots, y(n - n_z + 1) \} \) is minimized; the mean-square error is computed over the total length, \( N \), of the time series.

In keeping with the overall desire to weaken the predictive model assumptions so that the regularized solution is mathematically tractable, we choose to employ the method of regularized Gaussian RBF networks [22, 23]. Figure 5 shows the structure of an RBF network. We assume that the underlying dynamics responsible for the generation of this time series are sufficiently “smooth” (in some well-defined sense) so that the input-output mapping \( f() \) of the prediction model may be estimated from the training data (i.e., observable time series)

\[ T_N = \{ y(1), y(2), \ldots, y(N) \} \]

by finding the solution to the following optimization problem:

\[
\hat{f} = \arg \min_{f, \xi} \left\{ \sum_{n=m}^{N-1} (y(n + 1) - \hat{y}(n + 1))^2 + \lambda \| Pf \|^2 \right\} 
\]

where

\[
\hat{y}(n + 1) = \hat{f}(y(n))
\]

with

\[ y(n) = [y(n), y(n - 1), \ldots, y(n - m + 1)]^T \]

To simplify the notation, we have omitted the subscript \( R \) from the reconstruction vector \( y_R(n) \).

The space, \( S \), of dimensionality \( m \) is defined as a space of rapidly decreasing, infinitely continuous differentiable functions. The \( P \) (operating on \( f \)) is a differentiable operator. Prior knowledge about the form of the solution (i.e., the mapping \( f \)) is embedded in the operator, \( P \), which, naturally, makes the selection of \( P \) problem dependent. We refer to \( P \) as a stabilizer in the sense that it stabilizes the solution \( f \), making it smooth and, therefore, continuous. Note, however, that smoothness implies continuity, but the reverse is not necessarily true. The \( \lambda \) is a positive real number called the regularization parameter. In a sense, we may view \( \lambda \) as an indicator of the sufficiency of the given training data, \( T_N \), that specify the mapping, \( \hat{f}(\cdot) \). In particular, the limiting case \( \lambda \to 0 \) implies that the problem is unconstrained, with the solution of \( \hat{f}(\cdot) \) being completely determined from the training data. On the other hand, the limiting case \( \lambda \to \infty \) implies that the prior smoothness constraint is, by itself, sufficient to specify the mapping, \( \hat{f}(\cdot) \), which is another way of saying that the training data are unreliable. In practice, the regularization parameter, \( \lambda \), is assigned a value somewhere between these two limiting conditions, so that both the training data, \( T_N \), and the prior information contribute to the solution, \( \hat{f}(\cdot) \). Thus, the regularizing term \( \lambda \| Pf \|^2 \) represents a model complexity-penalty function, whose influence on the final solution is controlled by the regularization parameter, \( \lambda \).

With an appropriate choice of the differential operator, \( P \), we obtain the solution [22, 23]

\[
\hat{f}(y) = w^T g(y)
\]

where

\[
g(y) = [g_1(y), g_2(y), \ldots, g_{N-1}(y)]^T
\]

\[
g_j(y) = \exp\left(-\frac{1}{2} \| y - y_j \|_U \right), \quad j = 1, 2, \ldots, N - 1
\]

\[
w = [w_1, w_2, \ldots, w_{N-1}]^T
\]

The Euclidean norm, \( \| \cdot \| \) in \( \mathbb{R}^n \), is weighted by a symmetric positive definite matrix, \( U \). The linear weight vector, \( w \), is determined by the solution to the equation:

\[
(G + \lambda I)w = y
\]

where \( I \) is the \((N - 1)\)-by-\((N - 1)\) identity matrix, and

\[
G = \begin{bmatrix}
g_1^T(y_1) \\
g_2^T(y_2) \\
\vdots \\
g_{N-1}^T(y_{N-1})
\end{bmatrix}
\]

The vectors \( x_1, x_2, \ldots, x_{N-1} \) define the centers of the hidden units; their concatenation constitutes the entire training data \( T_N \).

As an illustrative example, consider the Lorenz attractor whose dynamics are described by a coupled system of three nonlinear differential equations [24]:

\[
\text{(MAY 1998 IEEE SIGNAL PROCESSING MAGAZINE 71)}
\]
The solution to the dynamic reconstruction problem presented in Fig. 6(a) is unacceptable as it fails to approximate the true trajectory of the Lorenz attractor in the long term; the unregularized system is just a predictor. On the other hand, the solution to the dynamic reconstruction problem presented in Fig. 6(b) using a regularized form of the RBF network has learned the dynamics, in the sense that the output of the network under iterated prediction closely approximates the actual trajectory of the Lorenz attractor. This is borne out by the results presented in Table 1. Here we have a summary of Lorenz data for three cases:

(a) Noise-free Lorenz system.
(b) Noisy Lorenz system with signal-to-noise ratio = 25 dB.

\begin{table}
\centering
\begin{tabular}{|l|}
\hline
(a) Noise-free Lorenz System
\hline
Number of samples used: 35,000
\hline
1. Normalized embedding delay, \( \tau = 4 \)
\hline
2. Embedding dimension, \( d_e = 3 \)
\hline
3. Lyapunov exponents:
\hline
\( \lambda_1 = 1.5697 \)
\hline
\( \lambda_2 = -0.0314 \)
\hline
\( \lambda_3 = -22.3054 \)
\hline
4. Horizon of predictability: 100 samples
\hline
(b) Noisy Lorenz system: 25 dB SNR
\hline
Number of samples used: 35,000
\hline
1. Normalized embedding delay, \( \tau = 4 \)
\hline
2. Embedding dimension, \( d_e = 5 \)
\hline
3. Lyapunov exponents:
\hline
\( \lambda_1 = 13.2689 \)
\hline
\( \lambda_2 = 5.8562 \)
\hline
\( \lambda_3 = -3.1447 \)
\hline
\( \lambda_4 = -18.0082 \)
\hline
\( \lambda_5 = -47.0872 \)
\hline
4. Horizon of predictability: 12 samples
\hline
(c) Reconstructed system using the noisy Lorenz data under (b)
\hline
Number of samples generated (reursively): 35,000
\hline
1. Normalized embedding delay, \( \tau = 4 \)
\hline
2. Embedding dimension, \( d_e = 3 \)
\hline
3. Lyapunov exponents:
\hline
\( \lambda_1 = 2.5655 \)
\hline
\( \lambda_2 = -0.6275 \)
\hline
\( \lambda_3 = -15.0342 \)
\hline
4. Horizon of predictability: 61 samples
\hline
\end{tabular}
\end{table}

Notes: All the Lyapunov exponents are expressed in units per second. Also, in case (b), the effect of noise is to increase both the size of the Lyapunov spectrum and the number of positive Lyapunov exponents.

Figure 6 shows the results of iterated prediction performed on two RBF networks with 400 centers using a "noisy" time series based on the component \( x(t) \) of the Lorenz attractor. The signal-to-noise ratio was +25 dB.

In Fig. 6(a), the design of the network is unregularized. In Fig. 6(b), the design of the network is regularized. These two parts of Figure 6 clearly demonstrate the practical importance of regularization. Without regularization, the solution to the dynamic reconstruction problem presented in Fig. 6(a) is unacceptable as it fails to approximate the true trajectory of the Lorenz attractor in the long term; the unregularized system is just a predictor.
(a) Unregularized iterated prediction \((N+\infty, m=20)\) on Lorenz data at +25 dB SNR. (b) Regularized iterated prediction \((N^{400}, m=20)\) on Lorenz data at +25 dB SNR. These results are obtained using a different initialization from that of Fig. 6 (solid line = reference, dashed line = predicted).

(c) Reconstructed data, using the noisy Lorenz time series described under Case (b).

The chaotic parameters of the reconstructed data using noisy data are seen to be close to the corresponding ones pertaining to the noise-free Lorenz data [25]. The deviations in absolute values are due to the residual effect of noise embedded in the reconstructed attractor and inaccuracies in the estimation procedure.

Figure 7 shows the results obtained with the unregularized and regularized RBF networks when the iterated prediction is initialized by a different point on the trajectory of the Lorenz attractor. The signal-to-noise ratio of the training data was +25 dB as in Fig. 6. Here again we see that the unregularized network described in Fig. 7(a) fails to provide a satisfactory solution to the dynamic-reconstruction problem, whereas the regularized network described in Fig. 7(b) does perform well.

Figures 6 and 7 clearly show that there is more to dynamic modeling than just prediction. Moreover, Figs. 6(b) and 7(b), and many others not included here, demonstrate the "robustness" of the regularized RBF solution with respect to the point on the attractor that is used to initialize the iterated prediction process.

The following two observations from Figures 6(b) and 7(b), pertaining to the use of regularization, are particularly noteworthy:

1. The short-term predictability of the reconstructed time series in Fig. 6(b) is about 80 samples. The short-term predictability of the reconstructed time series in Fig. 7(b) is about 60 samples. The theoretical horizon of predictability computed from the Lyapunov spectrum of the noiseless Lorenz attractor is 100 samples. The experimental deviations from the horizon of predictability of the noise-free Lorenz attractor are merely manifestations of the presence of noise in the actual data used to perform the dynamic reconstruction. Indeed, the theoretical horizon of predictability computed from the reconstructed data was 61 (Table 1), which is much closer to the experimentally observed values of short-term predictability.

2. Once the period of short-term predictability is over, the reconstructed time series in both Figs. 6(b) and 7(b) begin to deviate from the respective forms of the noiseless realizations of the actual Lorenz attractor. This is basically a manifestation of chaotic dynamics; namely, sensitivity to initial conditions. As mentioned previously, sensitivity to initial conditions is a hallmark of chaos.

**Choice of \(m\) and \(\lambda\)**

The size of the input layer, \(m\), is determined in accordance with Eq. (6). As explained previously, the recommended method is to use the smallest permissible value of \(m\) in accordance with the equality sign so to minimize the effect of noise on dynamic reconstruction.

The estimated value of the normalized embedding delay, \(\tau\), is essentially independent of the presence of noise for moderate to high signal-to-noise ratios. In contrast, the presence of noise has a profound impact on the estimated value of the embedding dimension, \(D_E\), which is intuitively satisfying. For example, for the noise-free Lorenz attractor, the correlation dimension is 2.01. We may therefore choose the embedding dimension \(D_E = 3\), which is confirmed by the method of false nearest neighbors. The normalized embedding delay is \(\tau = 4\). Thus, using Eq. (6) with the equality sign yields \(m = 12\) for dynamic reconstruction. However, for a noisy Lorenz attractor with \(\text{SNR} = +25\text{ dB}\), the use of the method of false nearest neighbors yields \(D_E = 5\), and the use of the method of mutual information yields \(\tau = 4\). Substituting these estimated values in Eq. (6) with the equality sign, we get \(m = 20\) for the noisy dynamic reconstructions reported in Figs. 6 and 7. Table 1 also includes the normalized embedding delay, \(\tau\), and embedding dimension, \(D_E\), for the three cases summarized therein.

As for the regularization parameter, \(\lambda\), used in Figs. 6(b) and 7(b), it was determined from the training data using the generalized cross-validation (GCV) due to Wahba [26]. The value of \(\lambda\) that was used in Figs. 6(b)
and 7(b), calculated on the basis of GCV, varied between a minimum value of $10^{-14}$ to a maximum value of $10^{-2}$ in accordance with the data.

**Multistep-Predictive Modeling (Trajectory Learning)**

In this section we discuss the idea of multistep-predictive modeling as an alternative to Tikhonov regularization. In dynamic modeling, a deterministic (albeit complex) model for the signal generation is assumed, which provides a temporal structure to the time series. So the weights can be further constrained during training if learning through time (also called trajectory learning) is implemented. In particular, we seek to develop a method of training that is equivalent to the way the time series is generated from an autonomous model; that is, using the present output to generate the next output (iterated prediction).

From our previous discussion (Eq. (4)), the dynamic model, $f$, can be obtained by a one-step prediction. This has been the conventional way to handle dynamic modeling. The predictor is adapted by minimizing the cost function

$$E = \sum_{i=1}^{N} \text{dist}(y(i + 1) - \hat{f}(y(i)))$$

where $N$ is the length of the time series, $y(i)$ is the $i^{th}$ observable sample, $\hat{f}$ is the map developed by the predictor and dist( ) is a distance measure (normally the L2 norm). The problem with this approach can be observed when we iterate the predictor to obtain the time series samples

$$y(i + 1) = \hat{f}(y(i)) + \delta_1$$
$$y(i + 2) = \hat{f}(y(i + 1)) + \delta_2$$

where $\delta_1$ and $\delta_2$ are the instantaneous prediction errors. Notice that the training to obtain the mapping is independent from sample to sample. If the goal is to predict two samples in the future from sample $i$ (as it will be the case of an autonomous system), the only way to obtain a good prediction for $i + 2$ is to use the original sample $i + 1$, which is not available. Errors will then accumulate rapidly. This training paradigm, which always requires the previous input, is more associated with extrapolation than with dynamic modeling. In the control-theory literature this scheme of training the autonomous neural model of Fig. 3 is equivalent to the equation error formulation [27]. We may therefore say that one-step prediction does not constrain the iterates of the model, which is a prerequisite to obtain an accurate dynamic model. In the previous section we overcame this problem through the use of regularization.

We propose multistep prediction or trajectory learning as the way to constrain the iterates of the mapping developed by the predictor. Define

$$E = \sum_{i=1}^{k} \text{dist}(y(i + 1) - \hat{y}(i + 1))$$

where $k$ is the number of prediction steps (length of the trajectory), and $\hat{y}(i + 1)$ may be considered as an estimate of the map:

$$\hat{y}(i + 1) = \hat{f}(\hat{y}(i - (D_E - 1)), \ldots, \hat{y}(i))$$

with

$$\hat{y}(i) = \begin{cases} y(i), & 1 \leq i \leq D_E \\ \hat{f}(y(i - D_E - 2), \ldots, y(i - 1)), & i > D_E \end{cases}$$

Expressing samples as a function of the mapping, we may write

$$y(i + 1) = \hat{f}(y(i)) + \delta_1 = \hat{y}(i + 1) + \delta_1$$
$$y(i + 2) = \hat{f}(y(i - D_E), \ldots, y(i), \hat{y}(i + 1)) + \delta_2 = (\hat{f}^{(2)}(\hat{y}(i))) + \delta_2$$

where the superscript 2 in $\hat{f}^{(2)}$ means two iterations. Minimizing $\delta_1$ and $\delta_2$ together not only constrains an optimal predictor but is also constraining the iterative map. The number of constraints that are imposed is associated with $k$, the number of prediction steps, which corresponds to the number of iterations of the map. The more iterations, the less likely the adaptation will end up with a sub-optimal solution. But the training time is increased in the process. Relating again the training process to control theory, we can say that the neural topology is trained with the output error formulation, which is known to be more accurate than the equation error method [27]. In a chaotic time series, there is still an important consideration that must be brought into the picture, the divergence of nearby trajectories, as discussed next.

**How Can Multistep Prediction Be Implemented with Neural Networks?**

Figure 8 shows the topology proposed in [32] to identify the nonlinear mapping. Note that the proposed topology is a recurrent neural network with a global feedback loop. This topology was selected to allow the training of the predictor in the same way as it will be used in testing; that is, using the previous network outputs to predict the next point. This recurrent architecture should be trained with
a mechanism that will constrain the iterates of the map as follows.

We seed the dynamic net with a set of input samples, disconnect the input and feed back the predicted sample to the input for $k$ steps. The mean-square error between the predicted and true sample at each step is used as the cost function.

As the network has a recurrent topology, a learning paradigm such as backpropagation through time (BPTT) [28] or real-time recurrent learning (RTRL) [29] must be utilized. The use of these training methods should not come as a surprise since we are in fact fitting a trajectory over time, so the gradients are time varying.

This learning method is sometimes called "trajectory learning" in the recurrent learning literature [30]. A criterion to select the length of the trajectory, $k$, will be presented below.

The procedure described above must be repeated for several different segments of the time series. For each new training segment, $D_t$ samples of the original time series are used to seed the predictor. To ease the training, we suggest that successive training sequences of length $k$ overlap by $q$ samples, (where $q$ can be any number less than $k$). For a chaotic time series, we also suggest that the error be weighted according to the largest Lyapunov exponent. Hence the cost function becomes

$$E = \sum_{j=0}^{r} \sum_{i=0}^{k} b(i) \cdot \operatorname{dist}(y(i+j+1) - \hat{y}(i+j+1))$$

where $r$ is the number of training sequences, and

$$b(i) = (\exp(\lambda_{\text{max}} T_s ))^{(i-D_t)}$$

In this equation, $\lambda_{\text{max}}$ is the largest Lyapunov exponent and $T_s$ is the actual sampling interval. With this weighting, the errors for later iteration are given less credit, as should be since (due to the divergence of trajectories) a small error is magnified proportionally to the largest Lyapunov exponent. This is merely restating the sensitivity of a strange attractor to initial conditions.

**Finding the Length of the Trajectory**

From the point of view of dynamic modeling, each training sequence should preferably contain enough information to model the attractor. This means that each sequence should be no shorter than the orbital length around the attractor. We propose to estimate the orbital length as the reciprocal of the median frequency of the spectrum of the time series [32]. Basically, this quantity is the average time required for a point to return to the same neighborhood in the attractor.

Consider then the case of a chaotic signal with small positive $\lambda_{\text{max}}$. The length of the trajectory is equivalent to the number of constraints we impose on the iterative map describing the dynamical model. This length should be large enough to capture the information about the attractor. However, in a chaotic time series there is another fundamental limitation imposed on the trajectory length: the natural divergence of trajectories, which is controlled by $\lambda_{\text{max}}$, the largest Lyapunov exponent. If the trajectory is too long, then instabilities in the training can be expected. A full discussion of this topic is beyond the scope of this article and is presented elsewhere [31]. It suffices here to say that when $\lambda_{\text{max}}$ is positive there is a conical "uncertainty region" around each predicted point that is a function of the number of prediction steps (due to the accumulation of errors). If the trajectory length is too long, the uncertainty regions from two neighboring trajectories will overlap, creating conflicting requirements for training (Fig. 9).
Hence, to apply this method, three quantities must be estimated: the largest Lyapunov exponent, the initial separation (using one-step predictors), and \( i_s \), by averaging local divergence. The computation time required to estimate these quantities is usually much less than setting by trial and error the length of the trajectory.

**Dynamic Modeling Results**

The methodology discussed above was applied to model the Mackey-Glass system described by the following delay-differential equation [33]

\[
\frac{dy(t)}{dt} = \frac{0.2 y(t-T_d)}{1 + y^{10}(t-T_d)} - 0.1 y(t)
\]  

(22)

with \( T_d = 30 \), sampled at 1/6 Hz; henceforth, this model is referred to as MG30. A signal segment of 500 samples was obtained by fourth-order Runge-Kutta integration and normalized between -1,1. The largest Lyapunov exponent for this signal is 0.0071 nats/sec. We selected a time-delay neural network (TDNN) with a delay line with 8 taps, 14 hidden processing elements (PEs), and one output PE. The output unit is linear, and the hidden layer has sigmoid nonlinearities.

To compare results, we trained a one-step predictor and the multistep predictor with the methodology described above. The one-step predictor was trained with backpropagation with no momentum and step size of 0.001. Training was stopped after 500 iterations. The final MSE was 0.000288. After training, the predictor was seeded with the first 8 points of the time series and iterated for 3,000 times. Figure 10(a) shows the corresponding output. Notice that the waveform produced by the model resembles the Mackey-Glass signal but it is much more regular than the original signal, showing that some fine detail of the attractor has not been captured.

Next we trained the same TDNN with a global feedback loop as shown in Fig. 8 (TDNNGF). The average \( i_s \) is 14, and it is taken as the length of the trajectory. We displaced each training sequence by 3 samples (\( q = 3 \) in Eq. (19)). BPTT was used to train the TDNNGF for 500 iterations. The final MSE was 0.000648, higher than for the TDNN case. We could think that the resulting predictor was worse since the final MSE is higher. The TDNNGF predictor was initialized with the same 8 samples of the time series and iterated for 3,000 times. Figure 10(b) shows the resulting waveform. It “looks” much closer to the original Mackey-Glass time series. We computed the average prediction error as a function of forward iterations for both predictors and also the theoretical rate of divergence of trajectories assuming an initial error, \( \epsilon_0 \) (Casdagli conjecture, which is the square of Eq. (22) [2]). As can be seen in Fig. 11, the TDNNGF is much closer to the theoretical limit, which means a much better model. We also computed the correlation dimension and the Lyapunov exponent estimated from the generated time series with both predictors (Table 2), and the values obtained from TDNNGF are again closer to the original time series.

We thus concluded that the training of the neural network as a recurrent system using trajectory learning produces a better dynamic model than a conventional one-step predictor. The difference in training is essentially using the structure of the time series as added constraints to find the values of the neural network free parameters.

This example reinforces what was shown with the RBF network: In order to obtain the dynamic model, the training process must be constrained, either through the use of regularization or trajectory learning.
The Embedding Theorem and Short-Term Memory Structures

In the modeling procedure described in the previous two sections, we used a conventional tapped-delay-line to implement the first step in dynamic modeling. Sauer et al. [9] recently proposed an extension to the delay-embedding theorem, which can be implemented with filtered delay coordinates. The reason for seeking alternative delay-embedding methodologies is to minimize the effects of noise contaminating the observed time series. Sauer et al. showed that constructing a vector of measurements

\[ z(n) = M[y(n), y(n-\tau), \ldots, y(n-(D-1)\tau)]^T, \]  

(23)

where \( M \) is a weighting matrix of rank equal to the size of the reconstruction space, which guarantees virtually a one-to-one correspondence between the state of the original system and that of the reconstruction space. Equation (23) is interpreted as a linear operation performed by a finite-duration impulse response (FIR) filter on the observed time series. Equation (3) is indeed a special case of Equation (23) with \( M \) equal to the identity matrix. Sauer et al. suggest how to choose the elements of the matrix \( M \) to implement low-pass filters and thereby improve the quality of dynamic reconstruction in the presence of noise. The discussion of this issue in Sauer et al. is restricted to the case of FIR filters since it is known that infinite-duration impulse response (IIR) filters may alter the dimension of the attractor.

In this section, we describe a signal-processing extension for the filter-embedding technique by defining a generalized feedforward memory structure that consists of a cascade of nonideal delay operators. Some members of this class are IIR filters. However, by careful design, experience has shown that IIR filters are able to preserve the underlying dynamics of the attractor, while attenuating the effect of additive noise in the observed time series. This methodology was originally developed to design short-term memory structures for dynamic neural networks [34].

An interesting feature of the delay-embedding theorem due to Takens is that it is physically implemented by delay lines. For example, a delay line with two ideal delays (three taps) provides \( x(n), x(n-1) \) and \( x(n-2) \), which are exactly the coordinates needed to build the points of the trajectory in a three-dimensional reconstruction space with \( \tau = 1 \), as illustrated in Fig. 12.

The cascade of ideal delay operator, \( z^{-1} \), which is at the core of linear digital signal processing, is instrumental for dynamical reconstruction. Although we normally do not think of it this way, the delay-line (tap) outputs in a FIR filter of order \( N \) are effectively reconstructing the signal dynamics in a space of dimension, \( N \), according to Takens' theorem (see Fig. 12). The basis vectors for the reconstruction space become the input signal and its delayed versions. The output of the filter is a linear projection of the reconstructed input signal trajectory on a hyperplane defined by the filter weights.

This observation dissects the linear FIR filter as a single-input multiple-output structure, which we call a memory structure (the delay line) followed by a linear regressor. In this perspective, there is no reason to restrict ourselves to the ideal delay operator as in the tapped-delay line, which automatically determines the axes of the reconstruction space (the signal and its delayed versions). There are two main reasons to pursue other delay operators in the context of dynamic modeling. First, real-world signals are noisy, so alternative delay operators may attenuate noise. Second, alternative delays may lead to a

![12. Delay embedding and its implementation with delay lines.](image-url)
more principled way to choose the normalized embedding delay.

We have been studying generalized delay operators in the context of memory structures for neural networks [34]. In order to extract information from a time signal, a neural network needs to represent the signal’s time structure in order to develop the desired input-output map. So, in the neural-network setting the representational problem is the same as the one described for dynamic modeling. One way to proceed is to create a reconstruction space now called the memory space, because effectively the memory structures are storing traces of the signal’s past. The basic idea is to create a delay line that is not necessarily made of ideal delays, as in Fig. 13.

We define a memory structure as a single-input multiple-output linear system whose generating kernel, $g(n)$, is causal, that is, $g(n) = 0$ for $n > 0$; and it is normalized, which means that

$$\sum_{n=0} |g(n)| = 1 \quad (24)$$

We define memory depth, $D$, as the modified center of mass (first moment in time) of $g_k(n)$ pertaining to the last memory tap, $K$; see Fig. 13. We thus have

$$D = \sum_{n=0}^{\infty} n g_k(n) \quad (25)$$

We define the memory resolution, $R$, as the number of taps by unit time. A generalized feedforward memory structure is defined as

$$g_k(n) = g(n) \otimes g_{k-1}(n) \quad k \geq 1 \quad (26)$$

where $\otimes$ represents the convolution operation, and $g_0(n) = \delta(n)$. Equation (26) means that the next tap signal is constructed from the previous tap signal by convolution with the same function $g(n)$, yet unspecified. Therefore, generalized feedforward memory structures are recursively computable and generalize our concept of delay line. Different choices of $g(n)$ provide different choices for the memory space axes.

The choice of $g(n) = \delta(n - 1)$ corresponds to the direct use of the past samples of the input signal. This is the most common choice in digital signal processing and provides the best resolution but a depth that is strictly limited by the size of the taps. An alternative choice was proposed by deVries and Principe [34] and called the gamma delay operator, where

$$g(n) = \mu(1 - \mu)^n \quad n \geq 0 \quad (27)$$

Figure 14 shows the gamma memory structure and its characteristics, and it also shows the family of impulse response from the input to tap $k$ in the gamma memory given by

$$x_k(n) = \binom{n-1}{k-1} \mu^k (1-\mu)^{n-k} \quad \text{for } n \geq k \quad (28)$$

These functions are discrete versions of the integrands of the gamma function [34]. They represent, with an error as small as desired, any function with finite energy (i.e., they constitute a complete set in L2) provided the order of the memory is large enough. An interesting property of this family is that the time axis is scaled by the parameter $\mu$, which means that there is a change in time scale from the input to the memory traces (uniform time

In the neural-network setting the representational problem is the same as the one described for dynamic modelling.
that is hardly recognized as the Lorentz attractor as Fig. 15 shows.

The difficulty in applying the gamma-delay operator for embedding is how to estimate a proper value for the recursive parameter, \( m \), that controls the location of the pole of the generalized delay operator, so as to filter the noise but preserve the dynamics. We have found a way to automatically set, \( \mu \), from the input data [37] by analyzing its effect on both the crosscorrelation among consecutive taps and the autocorrelation of the tap signals defined, respectively, as follows:

\[
\rho_{\text{cc},i} = \frac{\langle x_i(n)x_{i+1}(n) \rangle}{\sqrt{\langle x_i^2(n) \rangle \langle x_{i+1}^2(n) \rangle}}
\]

\[
\rho_{\text{ac},i} = \frac{\langle x_i(n)x_{i+1}(n+k) \rangle}{\langle x_i^2(n) \rangle}
\]

where \( i \) is the tap index, \( x(n) \) is the input signal and \( \langle . . \rangle \) denotes time averaging. Our work suggests that \( i = 1 \) should be used to estimate the value of \( \mu \) for the best reconstruction. Figure 16 shows that \( \mu \) affects these quantities in different ways, as one could expect by reasoning in terms of low-pass filtering. The best value of \( \mu \) for the reconstruction was selected as the point where the two curves cross in the range \( 0 < \mu < 1 \), which for this case was \( \mu = 0.3 \).

Mutual information (a better measure for statistical independence) provides basically the same value [36].

Figure 16 also shows the change of the largest Lyapunov exponent (LLExp), averaged over 10 different signal segments, as a function of \( \mu \) for two different signal-to-noise ratios. The dashed line is the accepted value of \( \text{LLExp}=2.16 \text{bits/sec} \) for the Lorenz attractor. As we can observe, the value of \( \mu = 0.3 \) estimated by our method provides a LLExp very close to the theoretical one. The LLExp estimated from ordinary delay embedding is much higher: 2.76 bits/sec (1.91 nats/sec).

![Figure 15](image1.png)

(a) The Lorenz attractor reconstructed with \( \tau = 3 \) in a 3-D space when white Gaussian noise was added (SNR = 30 dB). (b) Reconstruction with the gamma memory for the best value of \( \mu \) as described in the text (\( \mu = 0.3 \)).

![Figure 16](image2.png)

(a) Crosscorrelation among the first and second tap (solid) and the autocorrelation of the second tap (dashed). (b) Change of the Lyapunov exponent with \( \mu \) for several SNR. Note: the LLExp in (a) is measured in bits/sec.
The correlation dimension at $\mu = 0.3$ estimated with the Grassberger-Proccacia algorithm yielded the values of 2.11 for the case of $\text{SNR} = 30 \text{ dB}$ and 2.13 for $\text{SNR} = 25 \text{ dB}$, which are close to the actual value of 2.01 for the Lorenz attractor. Using the delay-embedding technique, the correlation dimension could not be assessed with the same algorithm due to the lack of a clear saturating region. This example (and others reported in [36]) show that the gamma delay operator is an effective method to perform the embedding of a time series corrupted by noise.

Summary

This article has addressed the identification of nonlinear systems from its output time series, which we have called dynamic modeling. We started by providing the mathematical basis for dynamic modeling and showed that it is equivalent to a multivariate nonlinear prediction problem in the reconstructed space. We addressed the importance of dynamic reconstruction for dynamic modeling. Recognizing that dynamic reconstruction is an ill-defined inverse problem, we described a regularized RBF network for solving the dynamic reconstruction problem. Prior knowledge in the form of smoothness of the mapping is imposed on the solution via regularization.

We also showed that, in time-series analysis, some form of regularization can be accomplished by using the structure of the time series instead of imposing a smoothness constraint on the cost function. We developed a methodology based on iterated prediction to train the network weights with an error derived through trajectory learning. This method provides a robust performance because during learning the weights are constrained to follow a trajectory. The dynamic invariants estimated from the generated time series are similar to the ones estimated from the original time series, which means that the properties of the attractor have been captured by the neural network.

We finally raised the question that generalized delay operators may have advantages in dynamic reconstruction, primarily in cases where the time series is corrupted by noise. We have shown how to set the recursive parameter of the gamma operator to attenuate noise and preserve the dynamics.

In this article we have emphasized the use of memory structures external to a neural network as the basis for dynamic reconstruction. In this context it is noteworthy that Sandberg and Xu [37] discuss the approximation of nonlinear input-output mapping. In particular, they show that any single-variable, shift-invariant, causal uniformly fading memory map can be approximated uniformly well using a focused gamma network.

Acknowledgments

Dr. Simon Haykin wishes to thank his former Ph.D. students Dr. B. Li, Dr. Paul Yee, and research colleague Dr. Sadasivan Puthusserypady for their contributions to dynamic reconstruction of chaotic time series using neural networks. This part of the work was supported by the Natural Sciences and Engineering Research council of Canada.

Dr. Jose Principe wishes to thank his former students Dr. Jyh-Ming Kuo, Dr. Bert deVries, Dr. Same1 Celebi, and Mr. Alok Rathie whose work in dynamic modeling and the gamma operator served as the basis of his contribution to this article. He would also like to thank Mr. Konstantino Pappas for the gamma embedding results. This work was partially supported by NSF grant #ECS-9510715, ONR #1494-94-1-0858, and NIH R01NS31451.

Simon Haykin is a University Professor at McMaster University's Communications Research Laboratory in Hamilton, Ontario, Canada. Jose Principe is a Professor at the Department of Electrical and Computer Engineering of the University of Florida in Gainesville, Florida, U.S.A.

References