Fast Forecasting of Time Series Data

Deepayan Chakrabarti DEEPAY@CS.CMU.EDU
Center for Automated Learning and Discovery

Abstract

"Delay Coordinate Embedding" is a promising approach to time-series prediction, but no completely automated approaches exist for setting the parameters in this model. We provide an algorithm based on the concept of fractal dimensions, which solves this problem without having to resort to thresholding/bucketization. Parameters are chosen efficiently so that, given a time series, we can automatically generate a prediction system optimised for that particular dataset. We show the effectiveness of this method by using it on two well-known datasets.

1. Introduction

Scientific research on physical systems is often based on the measurement of observables over time. Prediction of future behaviour of a system is an important technique for characterizing and understanding it. In many cases, the underlying dynamical equations are unknown, so regularities in the past have to be used as guides to forecasting the future. Given a time series from the past, the aim of a prediction algorithm is to forecast the expected future of the time series by drawing analogies from previous behaviour, and thus, indirectly forecast the future behaviour of the system. This paper describes techniques which help in quickly building forecasting systems for a large range of datasets.

Time series show up in a variety of areas(A.S.Weigend & N.A.Gershenfeld, 1993; Fessant et al., 1995):

- *Physics:* Measurements of laser fluctuations.
- *Biology:* Physiological data, such as heart rate, blood oxygen concentration, and EEG state, of patients with sleep apnea.
- *Economics:* High-frequency currency exchange rate data, and daily stock quote data.
- *Astrophysics:* Intensity variations of white dwarf stars, prediction of solar activity.
- *Geophysics:* Measurements of magnetic storm data.
- *Transportation:* Highway usage-over-time data.

Thus, this is a field with wide-ranging applications.

Formally, a dynamical system is a physical system which evolves over time. The system state is the underlying state of the dynamical system, and is usually an unknown quantity. An observable is a function of the system state which can be measured directly, or can be calculated from other measurements. A time series is a list of values taken by the observable over time. The fundamental property of such a system is that the state carries all information about the system. If the system returns to the same state later, the following time series will also be the same.

Given time series data over some period, the problem is to predict the future evolution of the system. Several methods have been proposed, but many are slow of suffer from the problem of too many free parameters. We utilize an approach called "Delay Coordinate Embedding", which avoids these problems. Still, this approach requires some parameter setting, and currently, there are no automated methods for setting these. We develop an algorithm to solve this, which chooses parameters for an efficient implementation of Delay Coordinate Embedding. This gives researchers a black-box into which they can feed in any time-series and get a tuned prediction system as output.

The layout of this paper is as follows: in Section 2, we discuss our approach, and give the required background information. Section 3 highlights our results. Section 4 describes related prior work, and its relation to our work. The limitations of our approach are described in Section 5, and we finally conclude in Section 6.

2. Approach

2.1 Background Tutorial

2.1.1 Delay Coordinate Embedding

Some of the following discussion is derived from (Sauer, 1993). The *state* of a deterministic system carries all the information necessary for predicting the behaviour of the system. Under minimal requirements on the smoothness of the underlying equations, the existence and uniqueness of the *state* holds. Thus, at any moment, the system has a state, and if a state is ever repeated later, then the future behaviour of the system will also be the same, within limits placed by noise. Thus, the internal system state can provide information for prediction. The Delay Coordinate Embedding approach uses this principle by trying to reconstruct the state space from the given time series, and using this reconstructed space for future predictions.

Let the observation sequence be represented by the series x_t, which gives the value of the time series at time t. Let us define the following vector:

$$\mathbf{b} = [\mathbf{x_t}, \mathbf{x_{t-\tau}}, \mathbf{x_{t-2\tau}}, \ldots, \mathbf{x_{t-(m-1)\tau}}] \qquad (1)$$

Here, τ (called the *time delay*) is some real number greater than zero, and m (called the *lag length*) is any integer greater than zero. The vector $\mathbf{b}$ is called a *delay coordinate vector* because its terms are the time-delayed data values from the time series. The technique of delay coordinate embedding is based on certain results (Takens, 1981; Sauer et al., 1991) which state that any structure that exists in the state space is carried over faithfully into the delay-coordinate-vector space, and that it is a one-to-one transformation. Thus, properties of the state space, such as equilibrium, periodicity, Lyapunov exponents (for measuring the degree of chaos) can all be observed in delay-coordinate space. Of course, characterizing every possible state would require infinite data, so we typically can observe only a portion of the entire state space. Still, the delay coordinate vector can be a useful representation of the internal state of the dynamical system. If, at two instances along the time series, the delay coordinate vectors are same (or very similar), then the data values following the two instances should be the same (or very similar). There is predictive power in finding this one-to-one map.

Hence, the problem of time series prediction is broken into two problems:

1. Finding the right values for the time delay and lag length
2. Prediction of future values given the current delay coordinate vector

The embedding theorem (Takens, 1981) does not specify any value for the time delay τ, except certain values, as long as there is infinite amount of data. But since this is never true, we need to find some workable value for the time delay. (Abarbanel, 1995) suggests certain principles for choosing this value:

- It must be some multiple of the sampling rate, so that the values of the observable at those time instants are available directly from our set of measurements. If this was not true, we would have to interpolate from observed values at different time instants, which may be a noisy process.
- If the time delay is too short, the system would not have explored enough of its state space to produce, in a numerical sense, new information about that state space. We would expect, from a physical standpoint, that the measurements x_t and $x_{t-\tau}$ to be essentially the same measurement. The system's dynamics would not have been explored.
- If the time delay is too large, the chaos in the system would effectively make x_t and $x_{t-\tau}$ look random. Even a small measurement error, which is inevitable, will blow up exponentially over time.

(Fraser & Swinney, 1986; Fraser, 1989) suggested the use of Mutual Information to set the time delay. Mutual information in this setting is defined as

$$I(T) = \sum_{x_t, x_{t-T}} P(x_t, x_{t-T}) \log_2[\frac{P(x_t, x_{t-T})}{P(x_t)P(x_{t-T})}] \qquad (2)$$

It represents the amount of information in bits which the value of x_t provides about x_{t-T}. The prescription is to take the time delay τ to be the first minimum in the plot of $I(T)$ versus T. Thus, we make sure that x_t and $x_{t-\tau}$ are not totally correlated, and yet not so far apart as to be random.

Finding the lag length remains an art, and the following sections describe the methods we developed to automatically set its value. For solving the second problem, the k-Nearest Neighbours $\mathbf{b}_1, \ldots, \mathbf{b}_k$ of the current state vector $\mathbf{b}$ are found, and the time-series values immediately succeeding these k instances (lets call them $p_1, \ldots, p_k$) are used to forecast the value after $\mathbf{b}$. One method is to just average over $p_1, \ldots, p_k$; another is to weight the p_i values by the exponential of the negative of the squared distance between $\mathbf{b}_i$

and **b**. One particular implementation using Singular Value Decomposition is given in (Sauer, 1993). Again, the determination of k is ad hoc.

2.1.2 False Nearest Neighbours

(Abarbanel, 1995) suggests a method called *False Nearest Neighbours* to determine the required length of the delay coordinate vector. Suppose we have made a state-space reconstruction in dimension d. Consider a vector $\mathbf{v}$ in this space. We find its nearest neighbour, and call it $\mathbf{v}^{\mathbf{NN}}$. If $\mathbf{v}^{\mathbf{NN}}$ is a true neighbour of $\mathbf{v}$, then it arrived at the neighbourhood of $\mathbf{v}$ through dynamical origins. If it is a *false* neighbour, then it arrived by projection from a higher dimension, because the current dimension does not fully unfold the attractor. Incrementing the dimension to $d+1$ should remove this false nearest neighbour from the neighbourhood of $\mathbf{v}$.

This suggests a technique for finding the "right" length of the delay-coordinate vector. Starting with a low dimension, we calculate the total number of false nearest neighbours for all vectors formed by the time series. As dimension increases, the number of false nearest neighbours should fall off till it reaches some constant value (which might not be zero because of noise). The minimum dimension at which this number stops falling significantly is a good estimate for the length of the delay-coordinate vector.

While this approach does provide a useful estimate of the proper lag length, it uses thresholding with the threshold being set manually. We found that the value of this threshold often affects the final result obtained. Thus, the results obtained by this method are somewhat unstable.

2.2 Our Approach

The primary advantages of Delay Coordinate Embedding(DCE) are its firm mathematical basis and the relaticely few free parameters. But, in current implementations, these parameters need to be set manually for each new dataset. Manual setting also leads to inconsistencies in results obtained by different researchers. The False Nearest Neighbours technique can solve this, but it itself uses thresholding, and we found that the result may depend considerably on the threshold used. Our aim is to develop a black-box which, given a time series, would generate a DCE-based prediction system with automatic parameter setting without using thresholding/bucketization. For this we used the concept of Fractal Dimension.

Fractal Dimension(FD) is a measure of the intrinsic dimensionality of data. For example, if we consider points along the diagonal of a cube, the intrinsic dimensionality of these points is one (because they actually lie along a straight line), but the extrinsic dimensionality is three (because the individual points are expressed in 3D-coordinates). If the time series is self-similar, that is, the series looks similar whether we look at a short portion of the series or a long portion, the fractal dimension of the time series can be used for prediction. Several time series, such as those for disk-accesses and network topology (Faloutsos et al., 1999), have been found to obey this assumption.

The technique we used was to plot FD of the vector space vs. lag length of that vector space. Beyond a certain point, increasing the lag length adds no new information regarding the state space (by Takens Theorem (Takens, 1981)), so the fractal dimension should flatten out. The lag at which it settles down could be used as the optimal lag length. Also, if the fractal dimension increases "significantly" as the lag length is increased from d to $d+1$, we can infer that the $d+1$th component of the delay vector is important. We could then only retain the important components and discard the others, leading to shorter lag vectors and thus, reducing dimensionality problems. Current algorithms for finding fractal dimension scale almost linearly with number of datapoints, so using this method can lead to increased speed.

Another approach for feature selection is to use a greedy approach. Consider the set $S = 1, \ldots, M$. We first find a $c_1 (c_1 \in S)$ such that the vector space $\{x_t, x_{t-c_1\tau}\}$ has the maximum fractal dimension for all $c \in S$. This is the most important component. Now we find a second component $c_2 (c_2 \in S, c_2 \neq c_1)$, such that $\{x_t, x_{t-c_1\tau}, x_{t-c_2\tau}\}$ has the maximum FD for all possible c_2. We continue in this fashion till the FD of the final vector space is within some range (say 5%) of the FD of $\{x_t, \ldots, x_{t-M\tau}\}$ or no appreciable increase in FD occurs on adding a new component. We were unable to fully test this approach because of instability of our fractal-dimension-finding code.

The result of all this is to provide a "good" feature vector. The corresponding vector space can be stored in a Spatial Access Method for retrieval during similarity searches. We use an R-Tree (Guttman, 1984) for this. The delay vectors of length d are considered as (infinitesimal) volumes in d-dimensional space, and are recursively grouped together based on their proximity. A height-balanced tree is created where each non-leaf node is associated with a specific volume, and its subtrees may contain points only from that volume. R-Trees allow us to do fast searches over the delay vector space. One problem with this method is the dimensionality curse: the larger the dimension of the delay coordinate vector, the more acute the problem. The dimensionality curse is tackled by the use of certain tricks, as detailed in (Faloutsos et al., 1994).

Another parameter used in DCE is the number of nearest neighbours used in a similarity search (refer Sec 2.1.1). We use leave-one-out crossvalidation for this. We start with $k = k_{min} = 1$. For each point in the vector space, we retrieve its nearest neighbours (excluding itself) and use them to predict the next value, noting the prediction error. We keep incrementing k till $k = k_{max}$ or the average prediction accuracy begins to decrease.

3. Results

We compared out algorithm to manual parameter setting based on computation time required and prediction accuracy. Two well-known datasets were used: the LORENZ dataset is a synthetic dataset modeling convection in fluid dynamics; and the LASER dataset measures fluctuations in a laser (Santa Fe competition (A.S.Weigend & N.A.Gershenfeld, 1993)). For prediction accuracy, we use a measure called *Normalized Mean Squared Error* (A.S.Weigend & N.A.Gershenfeld, 1993).

$$NMSE = \frac{1}{\sigma^2 N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3}$$

where x_i is the true value of the ith point in the series of length N, $\hat{x}_i$ is the predicted value, and σ is the standard deviation of the true time series during the prediction interval. The lower this value, the better the algorithm is compared to a random walk. An NMSE of less than one represents "better-than-random" predictions.

3.1 LORENZ dataset

The LORENZ dataset is a synthetic dataset which models convection currents. Results from the LORENZ dataset are shown in Figure 3.1. Fractal dimension was plotted against lag length, and the plot is seen to peak at a lag length of five. NMSE was also computed for each lag length, and we see that the plot has its minimum at lag length of four. This is the plot we would get if we used Cross-Validation to determine lag length. Cross-Validation would suggest using a lag length of four, whereas our approach gives a lag length of five, and the NMSE in both cases is very similar. Thus, Cross-Validation does not give much gain in prediction accuracy. But in crossvalidation, for each lag length, the corresponding vector space needs to be stored in an R-Tree, and predictions have to be carried out on the holdout set. This is a very time consuming operation, which makes Cross-Validation very slow. Our approach is more scalable and can be extended to very large datasets.

Table 1. Computation times taken when using R-Trees and sequential search for finding the k-nearest neighbours. All computations are on the LORENZ dataset. R-Tree times include time taken to build the R-Tree.

Training set size	Total predictions	R-Tree time(ms)	Sequential time(ms)
500	1000	8873	6100
500	2000	15182	12303
2000	100	913	2312
2000	700	6405	16306
2000	1000	8876	22591
2000	5000	45781	111959
5000	1000	6958	54658
5000	2000	16412	108861

3.2 LASER dataset

This is a dataset of fluctuations measured during the operation of a Laser. It is more widely known as "Data Set A" from the Santa Fe competition (A.S.Weigend & N.A.Gershenfeld, 1993). My results on a particular stretch of the dataset are shown in Figure 3.2. Here we notice from the fractal dimension vs. lag plot that the fractal dimension keeps increasing with increasing lag length, and never flattens out. This implies that the prediction accuracy should continue increasing with increasing lag lengths, which is exactly what we notice in the NMSE vs lag plot.

For comparision with (Sauer, 1993), we tried our method for prediction over the range given in the Santa Fe competition. The fractal dimension vs lag plot suggested a lag length of 6, with which we got an NMSE of 1.339. Using a lag length of 32 (used in (Sauer, 1993)) gave an NMSE of 1.392, so our approach gave comparable results to manual parameter setting. Here we must mention that the author of the referred paper had achieved an accuracy of 0.079, and we could not duplicate this result.

We also compared computation times using R-Trees as against using naive sequential search. Table 1 shows the results. We see that R-Trees outperform sequential search in all cases. The difference in computation time increases dramatically as the size of the training set increases. Thus, using R-Trees leads to a lot of improvement, and also makes the system scalable to

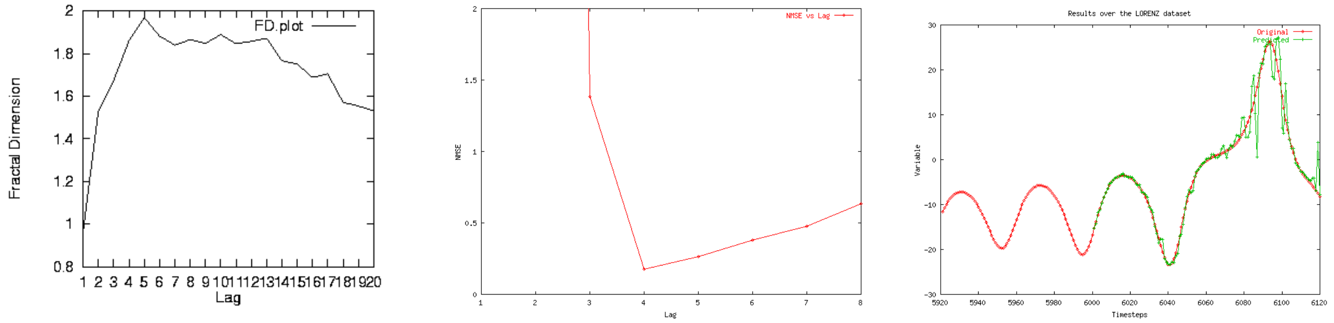

Figure 1. The LORENZ dataset: (a)Fractal Dimension vs Lag plot, which suggests a lag length of 5; (b)Error vs Lag plot, which reaches its minimum at lag of 4, and (c)sample prediction sequence.

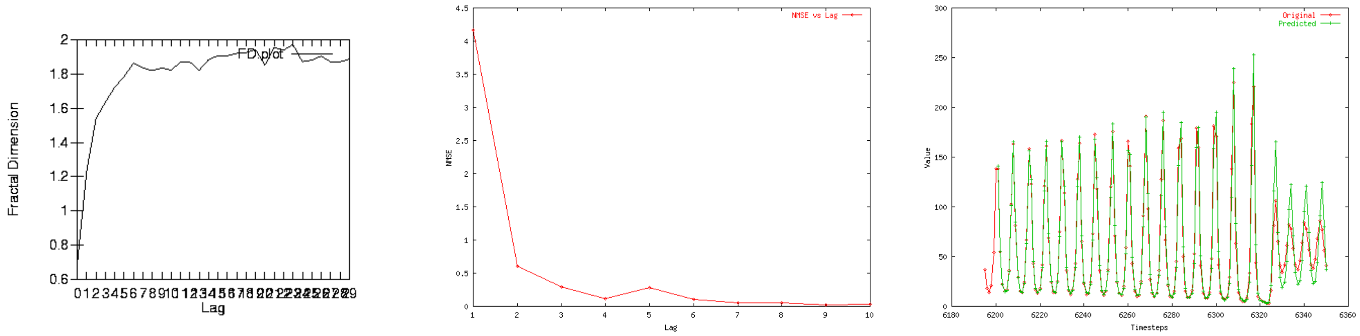

Figure 2. The LASER dataset: (a)Fractal Dimension vs Lag plot, which suggests a lag length of 6; (b)Error vs Lag plot, with the NMSE being low at lag 6 and continuing to decrease for higher lags, and (C)sample prediction sequence.

huge datasets.

4. Related Work

Time-series prediction is a well-studied problem. Prior to the 1920s, forecasting was carried out by extrapolating the series through a global fit in the time domain. Then came the linear time series models, (such as the ARMA models), which attempted to describe the series as some linear function of internal and external variables. These models assume linearity and time invariance(ie., unchanging model coefficients over time) of the system. Reference texts for this are (Chatfield, 1989; Priestley, 1981).

One class of linear models is the Moving Average(MA) Models. These models assume that the system receives some external impulses which change the state of the system. Let the external input series be e_t and the output series be x_t.

$$x_t = \sum_{n=0}^{N} b_n e_{t-n} \tag{4}$$

The output is guaranteed to go to zero at N time steps after the input becomes zero. There is no feedback in the system. For the system to possess internal dynamics, some memory-property is needed. This is implemented by the following equation:

$$x_t = \sum_{m=1}^{M} a_m x_{t-m} + e_t \tag{5}$$

These are called the Autoregressive(AR) Models. The autocorrelation coefficient doesn't disappear after M steps, as opposed to MA models. Also, without any input, the output will finally either decay to zero, diverge, or oscillate periodically. Combining the previous two models gives the ARMA(M,N) model. Here, the equation for x_t is given by:

$$x_t = \sum_{m=1}^{M} a_m x_{t-m} + \sum_{n=0}^{N} b_n e_{t-n} \tag{6}$$

It has been widely used in all areas of time series analysis and discrete-time signal processing. They had some success, but failed to model non-linearities which often arise in real-world data.

Then, several non-linear approaches were developed.

One such method is called the *delay coordinate embedding* method, which will be described in Section 2.1.1. Another approach has been through the use of Artificial Neural Networks (Werbos, 1974; Werbos, 1988; Lapedes & Farber, 1987; Weigend et al., 1990; Wan, 1993). Here, ANNs are modified to incorporate past values of time series. (Wan, 1993) describes a method where each network weight and neuron input is a vector encoding past time series values. A vector generalization of the backpropagation algorithm is used for learning in such networks. Another network structure for time series prediction is called the *Time-Delay Neural Network* (Lang & Hilton, 1988; Waibel, 1989). In these too, embedded time delays are incorporated into the network structure. A taxonomy of neural network structures for time-series processing is provided in (Mozer, 1993). But all these methods suffer from the traditional problems associated with neural networks: long training times and large number of parameters.

Hidden Markov Models(HMMs) have also been used (Fraser & Dimitriadis, 1993). There is a similarity between these models and the Delay-Coordinate Embedding technique. The notion of a hidden state in the former is similar to that of a *reconstructed state space* used in the latter technique (refer Section 2.1.1). Discrete HMMs are not suitable for problems involving continuous data, so, a modified class of models called mixed-state HMMs are used. But, either these models do have non-linearities or they make somewhat strong assumptions. In either case, the Delay Coordinate Embedding approach seems to lead to a more natural problem description.

Another method is called the "Method of Analogues" (Kostelich & Lathrop, 1993). This method takes a vector of the last p values of the time series (p being determined manually), and finds, say, the three best matches to this series from the entire time series history. These are the three nearest neighbours in p-dimensional space. Then, Singular Value Decomposition is used to fit a linear function through these nearest neighbours, and this function is used for prediction purposes. This approach is simple and has few free parameters, but works only for low-dimensional chaotic attractors, and only for predictions over a short period of time.

We must also mention in passing that all these methods were applied by various researchers in the Santa Fe competition for predicting certain given time sequences. Delay Coordinate Embedding and Neural Networks gave almost similar prediction accuracy, and outperformed all the other methods. We believe that the firm mathematical grounding of the Delay Coordinate Embedding approach and its relatively few parameters make it the ideal method for time series forecasting. As shown above, we found a fast automated method to set the lag-length parameter in Delay Coordinate Embedding, which should make it an even more preferable approach.

5. Limitations

Our approach makes use of the concept of fractal dimension, which is useful only if the self-similarity assumption holds true. Several real-world datasets, such as disk accesses and network traffic, have been found to adhere to it, yet prediction accuracy may be affected if this is not true. Also, to find the fractal dimension with a degree of confidence, the dataset size should not be too small. Typically there should be at least 1000 points in the delay vector space.

6. Conclusions and Future Work

Time series prediction is the problem of forecasting the future behaviour of a system given past behaviour data. Delay Coordinate Embedding is a promising technique for this, but its parameters had to be set on an ad-hoc basis. We developed a technique using the concept of fractal dimensions, which automatically sets these parameters without any thresholding/adhocness. We showed that the parameters chosen by this algorithm give good prediction accuracy and require low computation time. Also, we used R-Trees to store the data required by Delay Coordinate Embedding, and we showed that this leads to an improvement in computation time. R-Trees also allow the system to be scaled up for use with very large datasets.

One open research issue is whether "Lyapunov exponents" can be used under such circumstances for performing range queries instead of k-nearest neighbour queries. Another issue is whether the number of nearest neighbours bears any relation to the fractal dimension of the delay vector space. Also, cross-validation for finding the number of nearest neighbours did not give good results in our case, and we ended up using a fixed value for the number of nearest neighbours. Coming up with a fast approach to setting this value is an important area of future work, and we are actively pursuing it.

References

Abarbanel, H. D. I. (1995). *Analysis of observed chaotic data*. Springer.

A.S.Weigend, & N.A.Gershenfeld (Eds.). (1993). *Time series prediction: Forecasting the future and understanding the past*. Addison Wesley.

Chatfield, C. (1989). *The analysis of time series*. London: Chapman and Hall.

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *SIGMOD*.

Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. *SIGCOMM 99*.

Fessant, F., Bengio, S., & Collobert, D. (1995). On the prediction of solar activity using different neural network models. *Annales Geophysicae*.

Fraser, A. M. (1989). *Information theory and strange attractors*. Doctoral dissertation, University of Texas, Austin.

Fraser, A. M., & Dimitriadis, A. (1993). *Forecasting probability densities by using hidden markov models with mixed states*.

Fraser, A. M., & Swinney, H. L. (1986). Independent coordinates for strange attractors with mutual information. *Phys. Rev. A*, *33*, 1134–1140.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD*, 47–57.

Kostelich, E. J., & Lathrop, D. P. (1993). *Time series prediction by using the method of analogues*.

Lang, K., & Hilton, G. (1988). *A time-delay neural network architecture for speech recognition* (Technical Report CMU-CS-88-152). Carnegie Mellon University, Pittsburgh, PA.

Lapedes, A., & Farber, R. (1987). *Nonlinear signal processing using neural networks* (Technical Report LA-UR-87-2662). Los Alamos National Laboratory, Los Alamos, NM.

Mozer, M. C. (1993). *Neural network architectures for temporal sequence processing*, 243–264. Addison Wesley.

Priestley, M. (1981). *Spectral analysis and time series*. London: Academic Press.

Sauer, T. (1993). *Time series prediction by using delay coordinate embedding(data set a)*, 175–193. Addison Wesley.

Sauer, T., Yorke, J. A., & Casdagli, M. (1991). Embedology. *J. Stat. Phys.*, *65(3/4)*, 579–616.

Takens, F. (1981). Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence; Lecture Notes in Mathematics* (pp. 336–381). Berlin: Springer-Verlag.

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neur. Comp.*, *1(1)*, 39–46.

Wan, E. (1993). *Time series prediction by using a connectionist network with internal delay line*, 195–217. Addison Wesley.

Weigend, A. S., Huberman, B. A., & Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems*, *1*, 193–209.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioural sciences*. Doctoral dissertation, Harvard University, Cambridge, MA.

Werbos, P. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neur. Net.*, *1*, 339–356.