Winning Entry of the K. U. Leuven Time Series Prediction Competition

J. McNames[1] , J.A.K. Suykens[2] and J. Vandewalle[2]

[1]Information Systems Laboratory

Stanford University, Stanford, CA 94305-9510

Tel: 503/239-4374

Email: mcnames@simoon.stanford.edu

[2]Katholieke Universiteit Leuven, Dept. of Electr. Eng., ESAT-SISTA

Kardinaal Mercierlaan 94, B-3001 Leuven (Heverlee), Belgium

Tel: 32/16/32 11 11 Fax: 32/16/32 19 70

Email: johan.suykens,joos.vandewalle@esat.kuleuven.ac.be

Abstract

In this paper we describe the winning entry of the time series prediction competition which was part of the International Workshop on *Advanced Black-Box Techniques for Nonlinear Modeling* , held at K.U. Leuven, Belgium on July 8–10, 1998. We also describe the source of the data set, a nonlinear transform of a 5-scroll generalized Chua's circuit. Participants were given 2000 data points and were asked to predict the next 200 points in the series.

The winning entry exploited symmetry that was discovered during exploratory data analysis and a method of local modeling designed specifically for the prediction of chaotic time series. This method includes an exponentially weighted metric, a nearest trajectory algorithm, integrated local averaging, and a novel multi-step ahead cross-validation estimation of model error for the purpose of parameter optimization.

1 Introduction

The prediction of chaotic time-series is a challenging problem for nonlinear modeling techniques. Although local instability on attractors prohibits accurate long-term predictions, short term predictions can be made with varying degrees of accuracy depending upon the technique that is used.

A time-series prediction competition sponsored by the Santa Fe institute was held in 1991. The purpose of the competition was twofold: first, to bring together researchers from diverse disciplines that study time series (physics, biology, economics, etc) and quantitatively compare different approaches within a common domain of problems. Second, the competition was held to incite collective discussion and research on common problems within the field of time-series analysis.

The K.U. Leuven competition was held to take a current assessment of where this discussion has led and to give a fresh quantitative evaluation of leading time series prediction techniques.

The competition data set consisted of 2000 points[1] generated by a nonlinear transform of a 5-scroll generalized Chua's circuit. The data set is shown in Figure 2. The data set was made available in November 1997 and the dealine for entries was in April 1998. No information was given about the source of the data. Participants were asked to predict the next 200 points in the series. 17 entries were submitted.

The winning entry in the competition was generated by a local modeling method that is described in this paper. The method was inspired by the success of the second place entry in the Santa Fe competition [Sauer, 1994] and the success of local models as reported by other researchers [Casdagli et al., 1992; Farmer and Sidorowich, 1988; Kugiumtzis et al., 1998]. Other entries in the K.U. Leuven competition that incorporated local models also performed very well [Suykens and Vandewalle, 1998b; Suykens and Vandewalle, 1998a].

This paper is organized as follows. Section 2 describes the origin of the competition data set. Section 3 describes a general approach to time series

[1] The data set was available at http://www.esat.kuleuven.ac.be/sista/workshop/.

prediction including assumptions about the source of the time series, issues that arise when applying a local model to this type of problem, and some innovations that can improve the accuracy of local models for time series prediction. Section 4 describes the exploratory data analysis, user specified decisions, and the parameter optimization that generated the winning entry.

2 Origin of Data

Consider the following generalized Chua's circuit

$$\begin{aligned} \dot{x}_1 &= \alpha\,[x_2 - h(x_1)] \\ \dot{x}_2 &= x_1 - x_2 + x_3 \\ \dot{x}_3 &= -\beta x_2 \end{aligned} \tag{1}$$

with piecewise linear characteristic

$$h(x_1) = m_{2q-1}x_1 + \frac{1}{2}\sum_{i=1}^{2q-1}(m_{i-1} - m_i)\,(|x_1 + c_i| - |x_1 - c_i|) \tag{2}$$

where q denotes a natural number. A complete family of n-scroll attractors, allowing an even and odd number of scrolls, can be obtained from this circuit (see Suykens, Huang, and Chua [1997]). For the parameters $\alpha = 9$, $\beta = 14.286$ and the vectors $m = [m_0\ m_1\ ...\ m_{2q-1}]$, $c = [c_1\ c_2\ ...\ c_{2q-1}]$ one obtains

- $q = 1$: double scroll (2-scroll) [Chua et al., 1986; Chua, 1994; Madan, 1993]

$$\begin{aligned} m &= [\,-1/7 \;\; +2/7\,] \\ c &= 1 \end{aligned} \tag{3}$$

- $q = 3$: 5-scroll [Suykens et al., 1997]

$$\begin{aligned} m &= [\; 0.9/7 \quad -3/7 \quad 3.5/7 \quad -2.7/7 \quad 4/7 \quad -2.4/7 \;] \\ c &= [\; 1 \quad 2.15 \quad 3.6 \quad 6.2 \quad 9 \;]. \end{aligned} \tag{4}$$

In order to generate the competition data we have selected the 5-scroll attractor (see Figure 1).

The generalized Chua's circuit was simulated with the initial state vector $[\; 0.1 \quad -0.2 \quad 0.3 \;]^{\mathrm{T}}$ using a Runge-Kutta integration rule (ode23 in Matlab). The time series was generated by taking a nonlinear combination of the three state variables,

$$y = W \tanh(Vx), \tag{5}$$

where x is the 3-dimensional state vector and the nonlinearity is a multilayer perceptron with three hidden units, interconnection matrices

$$W = \begin{bmatrix} -0.0124 & 0.3267 & 1.2288 \end{bmatrix}, V = \begin{bmatrix} -0.1004 & -0.1102 & -0.2784 \\ 0.0009 & 0.5792 & 0.6892 \\ 0.1063 & -0.0042 & 0.0943 \end{bmatrix}$$

and a zero bias vector. This multilayer perceptron hides the underlying structure of the attractor.

The resulting time-series $y(t)$ is shown in Figure 2. 2000 data points were given (data before the vertical line) and the aim was to predict the next 200 data points (between the 2 vertical lines) so as to minimize the mean squared error (MSE),

$$\text{MSE} \triangleq \frac{1}{200} \sum_{i=2001}^{2200} (y_i - \hat{y}_i)^2 \tag{6}$$

In Figure 1 we have marked with a star the data points that were to be predicted. One can observe that the 2000 data points cover the whole attractor, i.e. the 5 scrolls. Within the 200 data points to be predicted the time series makes a transition between scrolls that is difficult to predict.

A physical implementation of a related family of n-double scroll attractors has been given in [Arena, Baglio, Fortuna, and Manganaro, 1996a; Arena, Baglio, Fortuna, and Manganaro, 1996b]. Further extensions to hyperchaotic n-double scroll hypercubes in cellular neural networks (one-dimensional array) have been proposed in [Suykens and Chua, 1997]. The n-scroll attractors have been used for secure communications applications exploiting chaos, such as the method of nonlinear H_∞ synchronization of chaotic Lur'e systems in [Suykens, Vandewalle, and Chua, 1997; Suykens, Curran, Vandewalle, and Chua, 1997]. Steps towards unmasking chaotic communications schemes have been taken in [Short, 1994; Short, 1996; Yang, 1995], which are related to nonlinear system identification. The latter is a motivation for understanding the limits of performance of time-series prediction methods.

3 Local Modeling

Local models generate predictions by finding segments of the time series that closely resemble the segment of the points immediately preceding the point to be predicted. The prediction is an estimate of the average change that occurred immediately after these similar segments of points.

Previous studies have shown that forecasting methods based on local models produce predictions that are better than or comparable to competing models and they have a number of favorable properties not shared by other methods [Casdagli et al., 1992; Farmer and Sidorowich, 1988; Farmer and Sidorowich, 1987; Liu et al., 1997; Sauer, 1994].

To use local models for time series prediction there are many decisions that one must make. For example, how should *local* be defined mathematically, how should the input vector be constructed, what should the model be designed to predict, what type of local model should be used, and how values for model parameters should be chosen. This section describes how these decisions were made for the method of local modeling that generated the winning entry.

3.1 Assumptions

The local modeling method described here was designed to predict time series generated by nonlinear dynamic systems that can be described by a set of nonlinear ordinary differential equations:

$$\dot{w}_t = f_c(w_t) \tag{7}$$

$$y_t = g_c(w_t) \tag{8}$$

where $w_t \in \mathbb{R}^{n_s}$ is state of the system, n_s is the order of the system, and $y_t \in \mathbb{R}^1$ is the time series to be predicted. An equivalent discrete-time system also exists,

$$w_t = f_d(w_{t-1}) \tag{9}$$

$$y_t = g_d(w_t) \tag{10}$$

where we have assumed that the sampling period is one for ease of presentation. This system can be thought of as a nonlinear oscillator as illustrated in Figure 3.

We assume that $f_d(w_t)$ and $g_d(w_t)$ are smooth in the sense that they have continuous and bounded derivatives. We also assume that the system is time invariant, that $f_d(w_t)$ has a finite number of equilibria, and that the order of the system is small ($n_s < 10$).

Many equivalent mathematical representations exist for any given set of ordinary differential equations. For the type of nonlinear dynamic system discussed here Takens has shown that an equivalent representation exists where the equivalent state is defined as a finite window of past values of the time series [Takens, 1981]. This is illustrated in Figure 4. This window is called a time delay embedding and is defined as

$$x_t \triangleq [\, y_t, y_{t-\delta}, y_{t-2\delta}, \ldots, y_{t-(m-1)\delta} \,] \tag{11}$$

where m is the embedding dimension and δ is the embedding delay, both of which are natural numbers. Takens' work was later generalized and shown to apply to a broader class of systems by Sauer et al., [1991].

This is theorem is important because it implies that an exact one-step ahead prediction is possible given a finite window of previous values:

$$\begin{aligned} y_{t+1} &= g(x_t) && (12) \\ &= g(y_t, y_{t-\delta}, y_{t-2\delta}, \ldots, y_{t-(m-1)\delta}) && (13) \end{aligned}$$

for some unknown function $g(x_t)$. This effectively reduces the one-step ahead prediction problem to a standard nonlinear modeling problem.

Multi-step ahead predictions can be generated by making further iterations. For example, if the embedding delay is 1 then the following equations illustrate how iterative prediction can be used to predict more than one-step ahead.

$$\begin{aligned} \hat{y}_{t+1} &= \hat{g}(y_t, y_{t-1}, y_{t-2}, \cdots, y_{t+1-m}) \\ \hat{y}_{t+2} &= \hat{g}(\hat{y}_{t+1}, y_t, y_{t-1}, \cdots, y_{t+2-m}) \\ \hat{y}_{t+3} &= \hat{g}(\hat{y}_{t+2}, \hat{y}_{t+1}, y_t, \cdots, y_{t+3-m}) \\ &\vdots \end{aligned} \tag{14}$$

The generalization to other values of δ is straightforward.

Any of the standard nonlinear models, such as neural networks, radial basis functions, local models, or support vector machines could be used to create an approximation for $g(x_t)$. However, a more accurate model can be built if the aforementioned assumptions are incorporated into the model structure. Several techniques that tailor local models specifically for time series prediction are described in the following section.

3.2 Introduction to Local Modeling

Generating a prediction with a local model can generally be divided into two steps: first, the k nearest neighbors in the data set of a given input query vector x_τ (Equation 11) are found and, second, a simple model is constructed using only the k neighboring points to generate the prediction $\hat{y}_{\tau+1}$.

3.2.1 Local Model Types

The second step, building a model given k data points, is an unusual function approximation problem because the dimension of the input vector (m) is often large compared to the number of data points (k) that are used to construct the model. Most nonlinear modeling methods are not viable under these conditions because they require that the number of data points be larger than the dimension of the input vector. Models that are computationally expensive to construct are also not viable since a different model must be constructed for every query. Instead a relatively simple model must be used that can be cheaply constructed and evaluated with just a few data points.

For this reason, the two most common types of local models are local averaging models and local linear models. Local linear models generally produce more accurate predictions but are very sensitive to the method of regularization [Kugiumtzis et al., 1998] and generally require more neighbors than local averaging models. Local averaging models can be used with smaller neighborhoods, are more stable, and are often more accurate for very short data sets. Since only 2000 points were available in the competition data set and because of its simplicity a local averaging model was used to generate the winning entry.

3.2.2 Discontinuities

One commonly mentioned disadvantage of local models is that they generate discontinuous predictions [Casdagli et al., 1992; Lillekjendlie and Christophersen, 1994; Liu et al., 1997]. For example, consider a local averaging model:

$$\hat{y}_\tau = \frac{1}{k}\sum_{i=1}^{k} y_{c_i} \tag{15}$$

where c_i is the data set index of the ith nearest neighbor. In some regions of the input space where two data points are nearly equidistant to the input vector, an arbitrarily small perturbation can result in a different kth nearest neighbor. This can cause a discontinuous change in the summation above, which in turn causes a discontinuous change in $\hat{y}_\tau$.

This problem can be remedied by decreasing the influence of neighbors that are relatively far away. For example, consider the weighted average

$$\hat{y}_\tau = \frac{\sum_{i=1}^{k} y_{c_i} w_i}{\sum_{i=1}^{k} w_i}. \tag{16}$$

where w_i are non-negative weights that are a monotonically decreasing function of the distance to the ith nearest neighbor.

It generally accepted that model accuracy is insensitive to the type of weighting function used [Atkeson et al., 1997]. A good choice is the biweight function,

$$w_i = \left(1 - \frac{d_i^2}{d_{k+1}^2}\right)^2 \tag{17}$$

where d_i is the distance to the ith nearest neighbor. Since the biweight function is smooth in the sense that it has a continuous first derivative, the weighted average produces an estimate that is also a smooth function of the input vector.

3.2.3 Fast Nearest Neighbor Algorithms

Another common criticism of local models is that they are computationally expensive. For most local modeling techniques the majority of the computation is used finding the k nearest neighbors in the data set. For time series prediction problems the data sets can be very large and the computational cost of the naïve brute force approach can be prohibitive.

To solve this problem a fast nearest neighbor algorithm can be used. Many of these algorithms fall into two categories: axis-partitioning algorithms and triangle inequality-based algorithms. Axis-partitioning algorithms divide the m-dimensional input space into hypercubes and establish a lower bound on the distance from the query point to all points contained in each hypercube. If the lower bound is greater than the distance to the kth nearest neighbor found so far, all of the points contained by the hypercube can be eliminated without explicitly calculating the distance to each point [Friedman et al., 1977; Kim and Park, 1986; Nene and Nayar, 1997; Niemann and Goppert, 1988; Tai et al., 1996].

Triangle inequality-based algorithms use the triangle inequality to find lower bounds on the distance to points. If the lower bounds is greater than the distance to the kth nearest neighbor found so far, the point can be eliminated from consideration without explicitly calculating the distance to that point. Subsets of points can also be eliminated by finding a lower bound on the distance to an enclosing hypersphere [Fukunaga and Narendra, 1975; Kalantari and McDonald, 1983; Kamgar-Parsi and Kanal, 1985; McNames, 1998; Micó et al., 1996; Vidal, 1986].

If the data set variables are independently distributed and there are relatively few of them (i.e. a low-dimensional space) then both types of nearest neighbor algorithms perform drastically better than the brute force approach. In low dimensions most of these algorithms achieve $O(\log n_c)$ search time and require only $O(n_c \log n_c)$ preprocessing time and storage, where n_c is the number of points in the data set.

In high-dimensional spaces, say $m > 15$, the bounding techniques of both types of algorithms are ineffectual and the performance is much worse. In fact, the computational cost of these algorithms can be significantly higher than the brute force approach due to overhead imposed for ordering the search of the input space and for calculating the lower bounds. Generally, the search time increases exponentially with m up to a limit where the distance is calculated for nearly all of the points in the data set.

If all of the points in the data set lie in a low-dimensional subspace of $\mathbb{R}^m$, the triangle inequality algorithms have an advantage over axis-partitioning algorithms—they can achieve search times comparable to low-dimensional data sets. If the dimension of the subspace is held constant, the search time grows roughly linearly with m rather than exponentially. This is especially relevant when the data points are taken from a time series generated by a nonlinear dynamic system because the points will often lie on a manifold of low dimension that is independent of m [Gershenfeld and Weigend, 1994; Sauer et al., 1991; Takens, 1981].

Figure 5 illustrates the advantage of using a fast triangle-inequality based algorithm when the points lie on a low-dimensional manifold, which is always the case for time series generated by the type of nonlinear dynamic systems discussed here (Section 3.1). This figure shows the average query time for four different nearest neighbor algorithms applied to the competition data set. The trace labeled *Brute* represents the brute force algorithm. The trace labeled *KD Tree* represents an axis-partitioning algorithm proposed by Friedman et al., [1977]. The traces labeled *FN* and *FNM* represent triangle inequality algorithms proposed by Fukunaga and Narendra, [1975] and

McNames, [1998] respectively.

3.3 Nearest Neighbor Metrics

Local models are constructed using only the nearest points on the k nearest neighboring trajectory segments, as described in the next section. Choosing an appropriate measure of nearness, or metric, is an important decision that can strongly affect accuracy [García et al., 1996; Murray, 1993; Tanaka et al., 1995]. The most common metric is the square of the Euclidean distance between the query vector x_τ and a vector taken from the data set, x_t (Equation 11).

$$\mathrm{D}_E(x_\tau, x_t) \triangleq \sum_{i=0}^{m-1} (y_{\tau-i\delta} - y_{t-i\delta})^2 \tag{18}$$

In this case the only parameters are the embedding dimension (m) and the embedding delay (δ). If the time series is generated from a continuous-time dynamic system then the Euclidean distance is an estimate of the integrated squared error,

$$\begin{aligned} \mathrm{ISE}(x_\tau, x_t) &\triangleq \frac{1}{\delta T} \int_0^{m\delta T} (y_{\tau T - v} - y_{tT-v})^2 \; dv &(19)\\ &\approx \sum_{i=0}^{m-1} (y_{\tau-i\delta} - y_{t-i\delta})^2 &(20)\\ &= \mathrm{D}_E(x_\tau, x_t) &(21)\end{aligned}$$

where T is the sampling period. If the window length, $m\delta T$, is fixed then smaller values of δ produce more accurate estimates of the ISE. However, smaller values of δ also require more computation so δ is treated here as a user-selected parameter that should be chosen as small as possible within the limits of available computational resources.

Choosing a more general measure of nearness with more parameters can greatly increase model accuracy. For example, a diagonally weighted Euclidean distance,

$$\mathrm{D}_{\mathrm{WE}}(x_\tau, x_t) \triangleq \sum_{i=0}^{m-1} \lambda_i \left(y_{\tau-i\delta} - y_{t-i\delta}\right)^2 \tag{22}$$

could be used where $\lambda_i \geq 0 \, \forall i$. However, short time series are often too short to accurately estimate the best m parameters (λ_i) and optimizing over so many parameters is computationally impractical for longer time series.

Murray has proposed using an exponentially weighted diagonal metric similar to the one described by the following equation,

$$\mathrm{D}_\lambda(x_\tau, x_t) = \sum_{i=0}^{m-1} \lambda^{i-1} \left(y_{\tau-i\delta} - y_{t-i\delta}\right)^2 \tag{23}$$

where $0 < \lambda \leq 1$ [Murray, 1993]. Although this metric is not optimal [Tanaka et al., 1995] it is intuitively appealing because the variables closest in time to the prediction are given exponentially more weight. It is especially appropriate for chaotic systems when m is large because neighboring states are known to diverge exponentially with time.

The metric D_λ is defined by the parameters m, δ and λ. Assuming δ is fixed, if m is sufficiently large and λ is sufficiently small then the influence of the later elements of the sum multiplied by λ^{i-1} (for large i) will have a negligible influence. Thus, if a model is constructed with m sufficiently large, only λ will affect model accuracy. This can be implemented practically by defining a cutoff value, $\lambda_{min} \triangleq \lambda^{m-1}$, for which larger values of i in Equation 23 will have a negligible effect on D_λ:

$$\lambda = \lambda_{\min}^{\frac{1}{(m-1)}} \tag{24}$$

$$D_\lambda(x_\tau, x_t) \triangleq \sum_{i=1}^{m} \lambda^{i-1} \left(y_{\tau-i\delta} - y_{t-i\delta}\right)^2 \tag{25}$$

$$\approx \sum_{i=1}^{\infty} \lambda^{i-1} \left(y_{\tau-i\delta} - y_{t-i\delta}\right)^2 . \tag{26}$$

This modified metric has one free parameter (m) and two user-selected parameters (δ and $\lambda_{\min}$). Figure 6 shows the weight λ^{i-1} of each element in the sum for different values of m. Since the model accuracy is sensitive to the value of m it is usually worth the computational cost to find the value that maximizes model accuracy.

3.4 Nearest Trajectories

One of the differences between a regression problem and a time series prediction problem is that it is possible to increase the number of data points by upsampling, or interpolating, the time series; more data can be generated without adding any new information. This is problematic because a signal sampled at a high enough rate will have all k of its nearest neighbors adjacent to each other in the time series. For example, in Figure 7 many of the points on the nearest trajectory segment are closer than the points on the third nearest trajectory segment.

This problem can be solved by finding the *nearest trajectory segments* instead of the nearest neighbors [Farmer and Sidorowich, 1988]. Fortunately, it is possible to modify existing nearest neighbor algorithms to find the nearest trajectory segments. This is advantageous because, as described previously, much research has gone into the development of efficient nearest neighbor algorithms.

The nearest trajectory modification described by McNames, [1998] is repeated here. Suppose a nearest neighbor algorithm has found a point x_i that is closer than the k nearest points found by the algorithm so far.

1. Calculate the distance to the points preceding x_i, $\{x_{i-1}, x_{i-2}, \dots\}$, until the nearest local minimum is found. Repeat this procedure for the points succeeding x_i. The local minimum found by this procedure, $x_{\min}$, is the closest point in the trajectory segment.

2. Calculate the distance to the points preceding $x_{\min}$ until either a maximum is found or the distance becomes greater than the distance to the

kth nearest neighbor found so far. Call this point $x_{\max}$. Eliminate all of the points between $x_{\min}$ and $x_{\max}$ from consideration by the nearest neighbor algorithm.

3. Repeat the previous step for the points succeeding $x_{\min}$.

4. Replace the kth nearest neighbor found so far with $x_{\min}$ and continue with the nearest neighbor algorithm.

3.5 Local Averaging Models

Although local averaging models are reasonably accurate at interpolation they are generally poor at extrapolation. Since the local model described previously is a weighted average, the output will never be greater than or less than any of the points that make up the average. In high-dimensional spaces this can be a significant disadvantage since nearly every point is an outlier [Friedman, 1991]. For example, Figure 8 shows a prediction generated by averaging the next point of the three nearest trajectories. Because two of the trajectories are above the query point the predicted next point is biased upward.

One way to fix this problem is to build a model that predicts the change, or residual, $y_{t+1} - y_t$, rather than y_{t+1} directly. The final prediction can then be found by integrating from the current state. The equations for direct local averaging ($\hat{y}_{D,\tau}$) and integrated local averaging ($\hat{y}_{I,\tau}$) are given below:

$$\hat{y}_{D,\tau} \triangleq \frac{\sum_{i=1}^{k} w_i y_{c_i}}{\sum_{i=1}^{k} w_i} \tag{27}$$

$$\hat{y}_{I,\tau} \triangleq \hat{y}_{I,\tau-1} + \frac{\sum_{i=1}^{k} w_i \left(y_{c_i} - y_{c_i-1}\right)}{\sum_{i=1}^{k} w_i} \tag{28}$$

where w_i are the weights defined in Equation 17 and c_i is the data set index of the ith nearest neighbor.

Figure 8 illustrates why integrated local averaging may be better than direct local averaging. In this case the integrated local average is much closer to the actual y_τ than the direct local average.

For long term predictions these methods have very similar performance. However, if the time series is a smoothly varying function of time then integrated local averaging is usually more accurate for very short term predictions.

3.6 Assessing Model Accuracy

The type of local averaging discussed so far has four parameters: the embedding dimension (m), the embedding delay (δ), the minimum metric weight ($\lambda_{\min}$), and the number of neighbors (k). Reasonable values of $\lambda_{\min}$ and δ can be chosen by the user but there is no obvious means of picking the best values of m and k, which can strongly affect the model accuracy [Casdagli et al., 1992; Casdagli and Weigend, 1994; Gershenfeld and Weigend, 1994;

Smith, 1994]. However, these two parameters can be optimized globally if an accurate estimate of the model accuracy can be calculated cheaply.

One advantage that local models have over global models is that it is possible to calculate the leave-one-out cross-validation error very cheaply. This method begins by constructing a model to predict one step ahead with all but one of the n_c points in the data set. The prediction error for the omitted point is calculated and the process is repeated for all n_c points. The average leave-one-out error is used as a measure of the model accuracy.

Any of a variety of measures of error could be used. Absolute error, squared error, and the coefficient of correlation are common choices. For example, if squared error is used the cross-validation error is defined as

$$
\begin{aligned}
\mathrm{MSE}_1 &\triangleq \frac{1}{n_c}\sum_{i=1}^{n_c}\left(y_{i+1} - f_{i+1}^{-(i+1)}(x_i)\right)^2 && (29)\\
&\approx \mathrm{E}\left[\left(y_{t+1} - f_{t+1}^{-(t+1)}(x_t)\right)^2\right] && (30)\\
&\approx \mathrm{E}\left[(y_{t+1} - f_{t+1}(x_t))^2\right] && (31)
\end{aligned}
$$

where $f_{i+1}^{-(i+1)}(x_i)$ is the model constructed to predict one step ahead with the $(i+1)$th point left out of the data set and $\mathrm{E}[\,]$ denotes expectation.

A disadvantage of using one-step ahead cross-validation error (OSCV) for parameter optimization is that it does not take into account the model sensitivity to errors in the input vector that occur with iterated prediction (Equation 14). Hence, the parameter values that minimize the OSCV error are generally not the same values that minimize the p-steps ahead cross-validation error.

It is better to choose an error measure that represents the cost of making poor predictions in the application at hand. In many cases an average (possibly weighted) model accuracy over p-steps ahead is appropriate,

$$\mathrm{MSE}_{1,p} \triangleq \frac{1}{p\,n_p} \sum_{j=1}^{p} \sum_{i=1}^{n_p} \left(y_{i+j} - f_{i+j}^{-(i+1,i+p)}(x_i) \right)^2 \tag{32}$$

where $n_p = n_c - p + 1$ and $f_{i+j}^{-(i+1,i+p)}(x_i)$ is the model output j-steps ahead constructed with the points $(i+1)$ to $(i+p)$ left out. Kantz and Jaeger, [1997] have argued that multi-step ahead cross-validation (MSCV) also reduces bias.

To reduce the computation required to calculate $\mathrm{MSE}_{1,p}$, fewer than n_p terms could be used to approximate the expected error. For example, windows of p-points could be used that are spaced s points apart.

$$\mathrm{MSE}_{1,p} \approx \frac{1}{p\,n_v} \sum_{j=1}^{p} \sum_{i=1}^{n_v} \left(y_{v_i+j} - f_{v_i+j}^{-(v_i+1,v_i+p)}(x_{v_i}) \right)^2 \tag{33}$$

where $v_1 = 1$, $v_2 = s$, $v_3 = 2s$, and so on. Here n_v is the number of windows, or validation sets.

For large values of p and short time series, removing p values from the data set may significantly affect the model that is constructed. In this case, the cross-validation error will not be representative of the expected error using all n_c points. One approach to reduce this effect is to omit only the trajectory segment that surrounds the point being predicted, $y_{v_i+\rho}$, instead of all p points.

4 The Winning Entry

The winning competition entry was generated using the method of local modeling described in the previous section. The exploratory data analysis, selection of user-specified parameters, and the model analysis that was performed prior to the competition deadline is described in this section.

4.1 Measure of Accuracy

Entrants were told that the predictions would be compared using mean squared error (Equation 6). Hence the cross-validation MSE of Equation 33 was chosen for the purpose of parameter optimization and accuracy estimation. This is equivalent to minimizing the average normalized mean squared error, which is defined as

$$\mathrm{NMSE}_\rho \triangleq \frac{\frac{1}{n_v}\sum_{i=1}^{n_v}\left(y_{v_i+\rho} - f_{v_i+\rho}^{-(v_i+1,v_i+200)}(x_{v_i})\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} \tag{34}$$

where $y_{v_i+\rho}$ is the $(v_i + \rho)$th point in the time-series, v_i is the index of the first point in the ith validation set, and $\bar{y}$ is the average value of y_t. $f_{v_i+\rho}^{-(v_i+1,v_I+200)}(x_{v_i})$ is the prediction at time $v_i + \rho$ of a model that has been iterated ρ times with the points $y_{v_i+1}, y_{v_i+2}, \ldots, y_{v_i+200}$ left out of the data set.

Similarly, the normalized root mean squared error is defined as

$$\mathrm{NRMSE}_\rho \triangleq \sqrt{\mathrm{NMSE}_\rho}. \tag{35}$$

NRMSE_ρ is a convenient measure of error because it is independent of the scale of the time series and $\mathrm{NRMSE}_\rho = 1$ can be interpreted as meaning the model prediction error is no better than predicting the sample mean $\bar{y}$, on average.

4.2 Exploratory Data Analysis

By visual inspection, the data set appeared to be noise-free (Figure 2). A plot of the power spectrum (Figure 9) indicated that the series had been sampled at approximately five times the Nyquist frequency and, therefore, little could be gained by upsampling the time series. This is also confirmed by considering that more than 99.3% of the estimated signal power is at frequencies less than one-tenth the Nyquist frequency.

Further inspection indicated that the series was probably chaotic with three unstable equilibria at approximately -0.25, 0, and 0.25, as shown in Figure 10. The period and growth of the oscillations about the upper and lower equilibria visually appeared to be the same. This manifested the possibility that the source of the time series might be symmetrical; that is, it may just as likely have generated the time series reflected about the horizontal axis (i.e. multiplied by -1), as shown in Figure 11.

If the system was symmetrical a more accurate model could be built using both the original and reflected time series. To determine if this was reasonable for the purpose of prediction, the cross-validation error *on the original time series* was compared for two models: one built with the original time series and the other built with both time series. Figure 12 illustrates that using both time series results in significantly less cross-validation error than using only the original time series. Consequently, both time series were used to build the model that generated the winning entry. It has since been shown that the actual system that generated the time series is noise-free and symmetric (see Section 2).

Figure 12 also illustrates that the average prediction error becomes worse than predicting the sample mean ($\bar{y}$) after approximately 80 steps ahead. However, this is only an *average* measure of error and does not represent the expected error for this particular prediction interval and data set. The accuracy of any single prediction depends upon many factors such as how densely the attractor is covered by the data set in the prediction region and the value of the largest Lyapunov exponent in the prediction region.

It is now known that the prediction region contained a single transition between scrolls (Section 2). Near the transition local trajectories diverge very rapidly which makes prediction after this point especially difficult.

4.3 Parameter Selection and Optimization

The user-selected parameters that had to be chosen are the minimum metric weight ($\lambda_{\min}$), the number of points used to calculate the MSE (n_v), and the embedding delay (δ). All three of these parameters were constrained only by limited computational resources. The values that were selected are given in Table 1. The choice of these parameters was fortuitous; slight changes to these parameters result in worse predictions after the transition point, though the predictions up to that point are much less sensitive.

Both the embedding dimension (m) and the number of neighbors (k) were chosen so as to minimize MSE[2]. Figure 13 shows the MSE_ρ for different

[2]An equivalent approach was actually used to generate the winning entry where the embedding dimension (m) was held constant at 50 and λ was optimized (Equation 23). However, the method described here produces the same prediction and has the advantages

values of m and ρ. This figure illustrates the importance of minimizing the average multi-step ahead cross-validation (MSCV) error rather than the one-step ahead cross-validation (OSCV) error. If OSCV had been used for parameter optimization a much smaller value of m would have been chosen ($m = 22$) which would have had a larger MSCV and generated a worse prediction.

Similarly, Figure 14 shows the NRMSE_ρ for different values of k and ρ. The same problem with OSCV is illustrated by this figure; the value of k that minimized the average OSCV ($k = 5$) would not have minimized the MSCV and would have generated a worse prediction.

The values of m and k that minimized the cross-validation error are given in Table 1. The winning prediction generated by the model using these values is shown in Figure 15. A summary of the method that generated the winning entry is also given in the table.

of being slightly cheaper computationally and the user-selected parameter is more intuitive (λ_{min} instead of m). See McNames, [1998] for more details.

5 Conclusions

Takens' theorem provides a sound theoretic basis for using general nonlinear modeling methods for time series prediction. However, if the time series is generated by a nonlinear dynamic system and exhibits chaotic behavior, a nonlinear model that incorporates this knowledge will generally perform better.

Several such modifications to local modeling have been described in this paper including the use of integrated local averaging, an exponentially weighted Euclidean metric, a fast triangle-inequality based nearest neighbor algorithm, a *nearest trajectory* algorithm, and a novel multi-step ahead cross-validation method of accuracy estimation for parameter optimization.

The prediction interval contained a single transition between scrolls. All five of the competition entries that employed local modeling generated accurate predictions up to this transition (about 80 steps ahead) [Suykens and Vandewalle, 1998b; Suykens and Vandewalle, 1998a]. A synergy of exploiting the data set symmetry, a fortuitous choice of user-selected parameters, and a method of local modeling tailored specifically to chaotic time series prediction produced the winning entry.

References

Arena, P., S. Baglio, L. Fortuna, and G. Manganaro (1996a). Generation of n-double scrolls via cellular neural networks. *Int. J. Circuit Theory and Applications 24*, 241–252.

Arena, P., S. Baglio, L. Fortuna, and G. Manganaro (1996b). State controlled cnn: A new strategy for generating high complex dynamics. *IEICE Trans. Fundamentals E79-A*, 1647–1657.

Atkeson, C. G., A. W. Moore, and S. Schaal (1997). Locally weighted learning. *Artificial Intelligence Review 11*(1–5), 11–73.

Bakamidis, S. G. (1993). An exact fast nearest neighbor identification technique. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume 5, pp. 658–661.

Casdagli, M., D. Des Jardins, S. Eubank, J. D. Farmer, J. Gibson, and J. Theiler (1992). Nonlinear modeling of chaotic time series: Theory and applications. In J. H. Kim and J. Stringer (Eds.), *Applied Chaos*, pp. 335–380. John Wiley & Sons, Inc.

Casdagli, M. C. and A. S. Weigend (1994). Exploring the continuum between deterministic and stochastic modeling. In A. S. Weigend and N. A. Gershenfeld (Eds.), *Time Series Prediction*, Santa Fe Institute

Studies in the Sciences of Complexity, pp. 347–366. Addison-Wesley.

Chua, L. (1994). Chua's circuit: an overview ten years later. *Journal of Circuits, Systems, and Computers 4*(2), 117–159.

Chua, L., M. Komuro, and T. Matsumoto (1986). The double scroll family. *IEEE Trans. Circuits and Systems-I 33*(11), 1072–1118.

Farmer, J. D. and J. J. Sidorowich (1987). Predicting chaotic time series. *Physical Review Letters 59*(8), 845–848.

Farmer, J. D. and J. J. Sidorowich (1988). Exploiting chaos to predict the future and reduce noise. In Y. C. Lee (Ed.), *Evolution, Learning and Cognition*, pp. 277–330. World Scientific.

Friedman, J. H. (1991). An overview of predictive learning and function approximation. In V. Cherkassky, J. H. Friedman, and H. Wechsler (Eds.), *From Statistics to Neural Networks*, Volume 136 of *Computer and Systems Sciences*, pp. 1–61. Springer-Verlag.

Friedman, J. H., J. L. Bentley, and R. A. Finkel (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software 3*(3), 209–226.

Fukunaga, K. and P. M. Narendra (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers C-24*, 750–753.

García, P., J. Jiménez, A. Marcano, and F. Moleiro (1996). Local optimal metrics and nonlinear modeling of chaotic time series. *Physical Review Letters 76*(9), 1449–1452.

Gershenfeld, N. A. and A. S. Weigend (1994). The future of time series: Learning and understanding. In A. S. Weigend and N. A. Gershenfeld (Eds.), *Time Series Prediction*, Santa Fe Institue Studies in the Sciences of Complexity, pp. 1–70. Addison-Wesley.

Kalantari, I. and G. McDonald (1983). A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering SE-9*(5), 631–634.

Kamgar-Parsi, B. and L. N. Kanal (1985). An improved branch and bound algorithm for computing k-nearest neighbors. *Pattern Recognition Letters 3*, 7–12.

Kantz, H. and L. Jaeger (1997). Improved cost functions for modelling of noisy chaotic time series. *Physica D 109*, 59–69.

Kim, B. S. and S. B. Park (1986). A fast k nearest neighbor finding algorithm based on the ordered partition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*(6), 761–766.

Kugiumtzis, D., N. Lingjærde, and N. Christophersen (1998). Regularized local linear prediction of chaotic time series. *Physica D 112*, 344–360.

Lillekjendlie, D. K. and N. Christophersen (1994). Chaotic time series. part ii. system identification and prediction. *Modeling, Identification and Control 15*(4), 225–243.

Liu, Z., X. Ren, and Z. Zhu (1997). Equivalence between different local prediction methods of chaotic time series. *Physics Letters A 227*, 37–40.

Madan, R. (1993). *Chua's Circuit: A Paradigm for Chaos*. Signapore: World Scientific Publishing Co. Pte. Ltd.

McNames, J. (1998). A nearest trajectory strategy for time series prediction. In *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*, Katholieke Universiteit Leuven, Belgium, pp. 112–128.

Micó, L., J. Oncina, and R. C. Carrasco (1996). A fast branch & bound nearest neighbor classifier in metric spaces. *Pattern Recognition Letters 17*, 731–739.

Murray, D. B. (1993). Forecasting a chaotic time series using an improved metric for embedding space. *Physica D 68*, 318–325.

Nene, S. A. and S. K. Nayar (1997). A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*(9), 989–1003.

Niemann, H. and R. Goppert (1988). An efficient branch-and-bound nearest neighbor classifier. *Pattern Recognition Letters 7*, 67–72.

Sauer, T. (1994). Time series prediction by using delay coordinate embedding. In A. S. Weigend and N. A. Gershenfeld (Eds.), *Time Series Prediction*, Santa Fe Institue Studies in the Sciences of Complexity, pp. 175–193. Addison-Wesley.

Sauer, T., J. A. Yorke, and M. Casdagli (1991). Embedology. *Journal of Statistical Physics 65*(3), 579–616.

Short, K. (1994). Steps toward unmasking secure communications. *Int. J. of Bifurcation and Chaos 4*(4), 957–977.

Short, K. (1996). Unmasking a modulated chaotic communications scheme. *Int. J. of Bifurcation and Chaos 6*(2), 367–375.

Smith, L. A. (1994). Local optimal prediction: exploiting strangeness and the variation of sensitivity to initial condition. In *Philosophical Transactions of the Royal Society*, Volume 348 of *A*, pp. 371–381.

Suykens, J. and L. Chua (1997). n-double scroll hypercubes in 1d-cnns. *International Journal Bifurcation and Chaos 7*(8), 1873–1885.

Suykens, J., P. Curran, J. Vandewalle, and L. Chua (1997). Robust nonlinear h_∞ synchronization of chaotic lur'e systems. *IEEE Transactions on Circuits and Systems-I, (special issue on Chaos Synchronization, Control and Applications) 44*(10), 891–904.

Suykens, J., A. Huang, and L. Chua (1997). A family of n-scroll attractors from a generalized chua's circuit. *Archiv fur Elektronik und Ubertragungstechnik (International Journal of Electronics and Communications) 51*(3), 131–138.

Suykens, J. and J. Vandewalle (Eds.) (1998a). *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modelings*, Katholieke Universiteit Leuven, Belgium.

Suykens, J., J. Vandewalle, and L. Chua (1997). Nonlinear h_∞ synchronization of chaotic lur'e systems. *International Journal of Bifurcation and Chaos 7*(6), 1323–1335.

Suykens, J. A. K. and J. Vandewalle (1998b). *Nonlinear Modeling Advanced Black-Box Techniques*. Kluwer Academic Publishers.

Tai, S. C., C. C. Lai, and Y. C. Lin (1996). Two fast nearest neighbor searching algorithms for image vector quantization. *IEEE Transactions on Communications 44*(12), 1623–1628.

Takens, F. (1981). Detecting strange attractors in turbulence. In D. A. Rand and L. S. Young (Eds.), *Dynamical Systems and Turbulence*, Volume 898 of *Lecture Notes in Mathematics*, pp. 336–381. Springer-Verlag.

Tanaka, N., H. Okamoto, and M. Naito (1995). An optimal metric for predicting chaotic time series. *Japanese Journal of Applied Physics 34* (1), 388–394.

Vidal, E. (1986). An algorithm for finding nearest neighbors in (approximately) constant average time. *Pattern Recognition Letters 4*, 145–157.

Yang, T. (1995). Recovery of digital signal from chaotic switching. *Int. J. of Circuit Theory and Applications 33* (6), 611–615.

Captions of Figures

Figure 1: The 5-scroll attractor [Suykens, Huang, and Chua, 1997] from which the competition data have been generated. The marked part of the trajectory corresponds to the data which have to be predicted.

Figure 2: Competition data. The first 2000 points (data before the red vertical line) were given to participants. The next 200 points (after the red vertical line) were to be predicted and judged according to the mean squared error (MSE).

Figure 3: Nonlinear oscillator. Chaotic time series generated by nonlinear dynamic systems can be thought of as the output of a nonlinear oscillator as illustrated by this diagram.

Figure 4: Equivalent oscillator. Takens has shown that the nonlinear oscillator shown in Figure 3 has an equivalent representation shown here.

Figure 5: Average query time to find 5 nearest neighbors in the Leuven time series with 1900 points. The query vectors were taken from the time series continuation that was released after the competition deadline. *Brute* is the brute force algorithm. *KD Tree* is an axis partitioning algorithm. *FN* is

a triangle inequality algorithm proposed by Fukunaga and Narendra, [1975] and *FNM* is a modified version of the same algorithm.

Figure 6: The weight, λ^{i-1}, of each element in the metric summation given in Equation 23 for various values of the embedding dimension (m).

Figure 7: Trajectory segments from the Lorenz time series in a two-dimensional embedding space. The query point, x_τ, is shown by a diamond. The three nearest points on each of the three nearest trajectory segments are shown by red squares. The inner green circle shows the distance to the third nearest neighbor and the outer green circle shows the distance to the nearest point on the third nearest trajectory segment. Note that the three nearest neighbors all lie on the nearest trajectory segment.

Figure 8: Trajectory segments from the Lorenz time series in a two-dimensional embedding space. The query point, x_τ, is shown by ♦. The prediction generated by direct averaging is shown by ▼. The prediction generated by integrated averaging is shown by ▲.

Figure 9: The estimated power spectrum of the workshop time series. The spectrum was estimated using Bartlett's method of periodogram averaging with four nonoverlapping windows. The x-axis has been scaled so that 1

represents half the Nyquist frequency.

Figure 10: The competition time series. The three red horizontal lines show the approximate location of the three unstable equilibria in the series.

Figure 11: The competition time series reflected about the horizontal axis. The three red horizontal lines show the approximate location of the three unstable equilibria in the series.

Figure 12: Plot of the NRMSE_ρ as a function of ρ, the number of steps predicted ahead. The upper red trace shows the error using only the original time series. The lower blue trace shows the error using both the original time series and the reflected time series.

Figure 13: NRMSE_ρ as a function of the number of steps predicted ahead (ρ) and the embedding dimension (m). $m = 4$ minimized the one-step ahead cross-validation error and $m = 16$ minimized the multi-step ahead cross-validation error. This plot was generated using 200 cross-validation points.

Figure 14: NRMSE_ρ as a function of the number of steps predicted ahead (ρ) and the number of neighbors (k). $k = 5$ minimized the one-step ahead cross-validation error and $k = 1$ minimized the multi-step ahead cross-validation

error. This plot was generated using 200 cross-validation points.

Figure 15: The blue trace shows the winning competition entry and the dashed red trace shows the true time series. The method used to generate the winning entry is summarized in Table 1.

Figure 16: A continuation of the predicted data for the winning entry. An accurate prediction is obtained over a time horizon of about 300 points. The dashed red trace shows the true continuation and the full blue trace shows the 1000 step-ahead prediction.

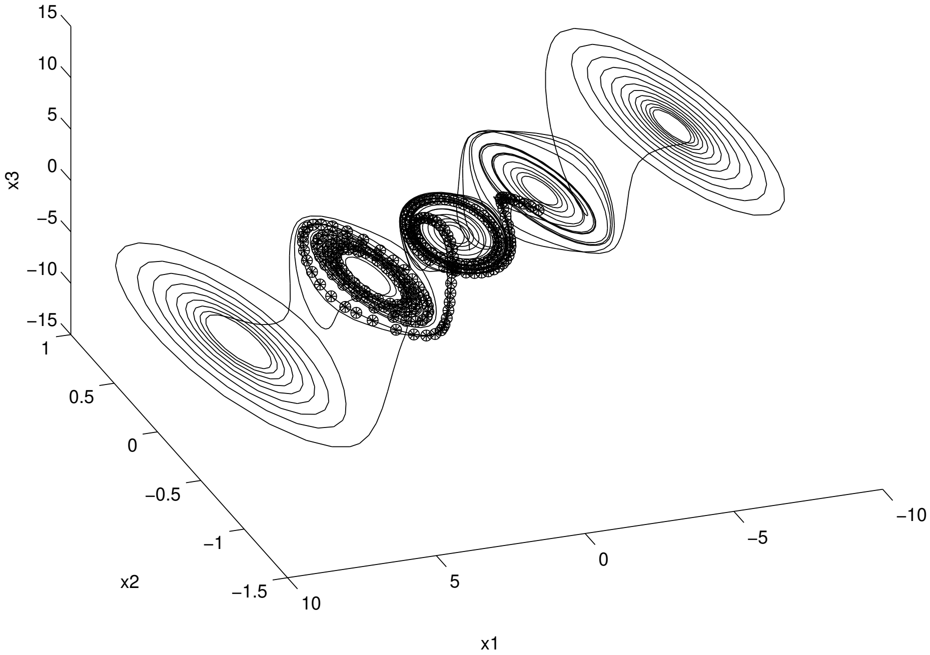

Figure 1:

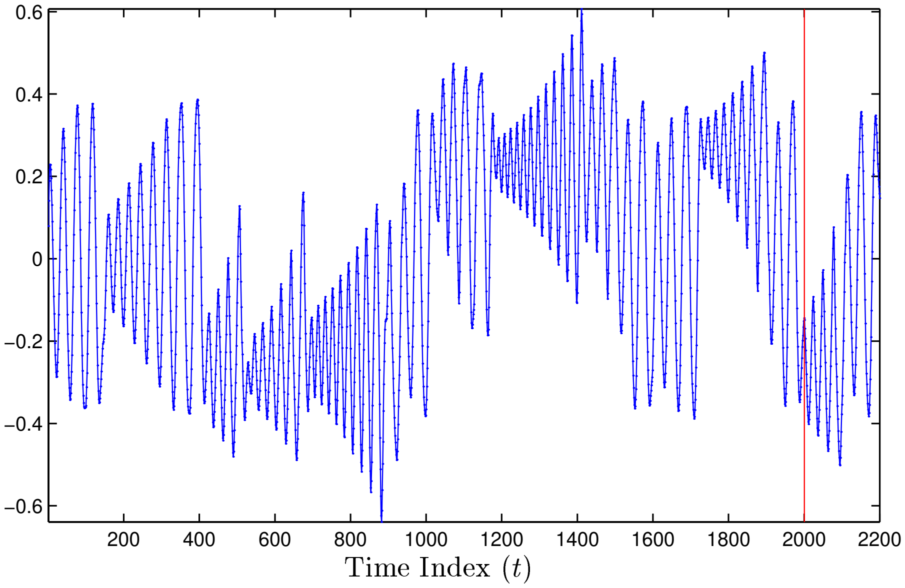

Figure 2:

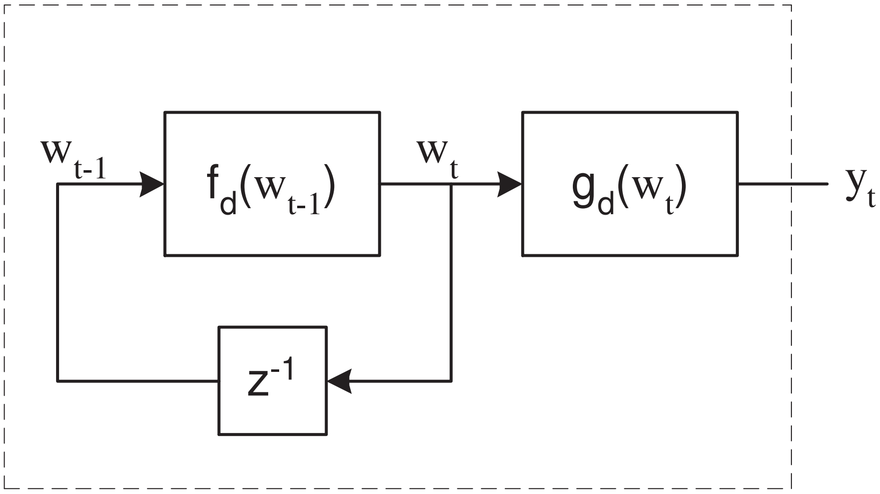

Figure 3:

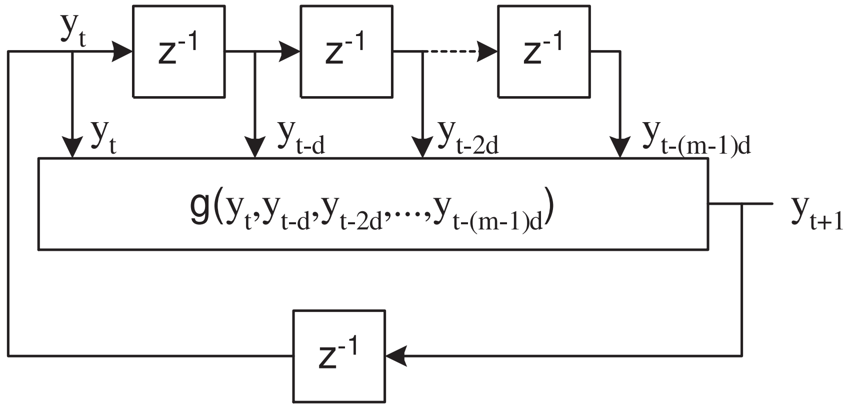

Figure 4:

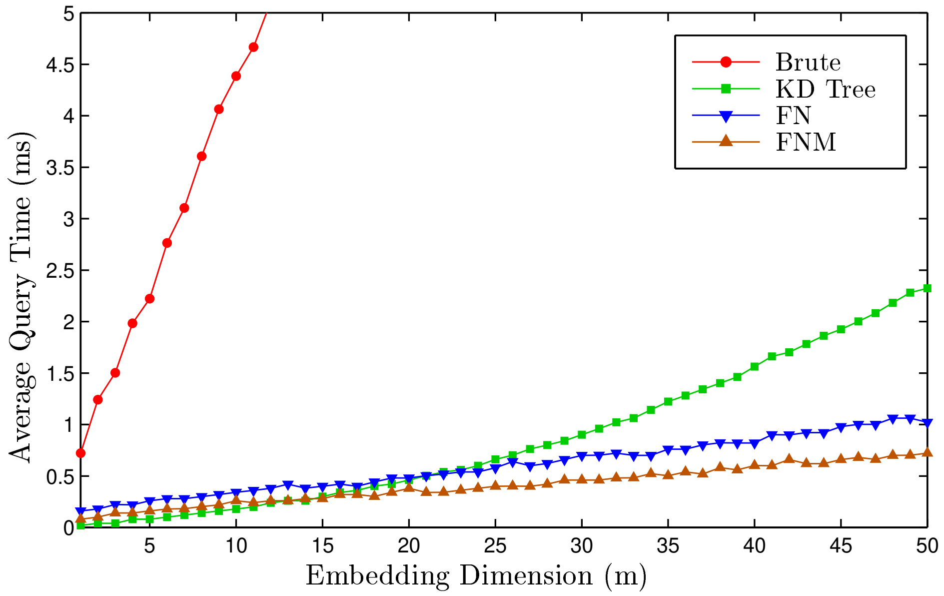

Figure 5:

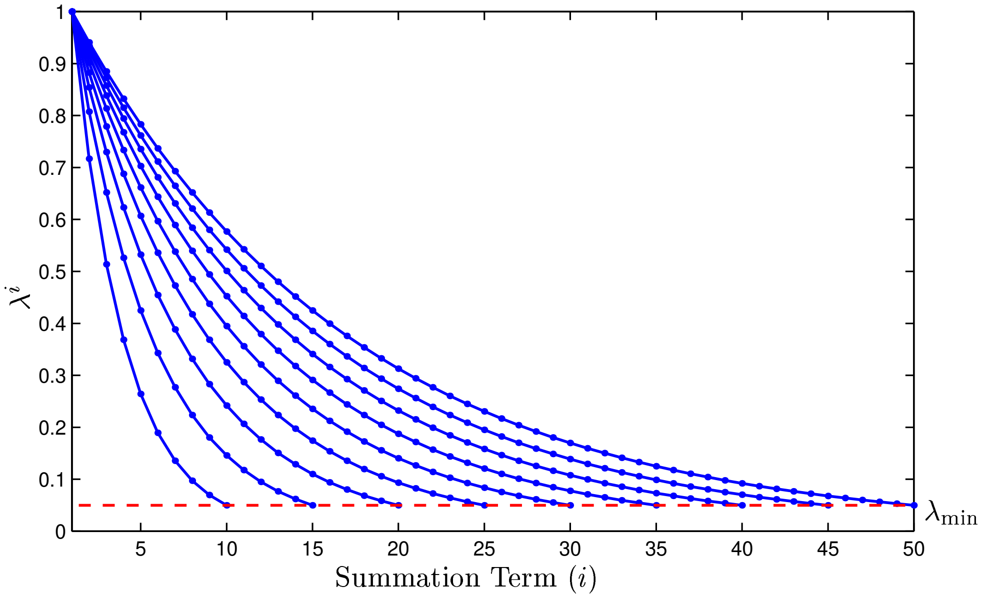

Figure 6:

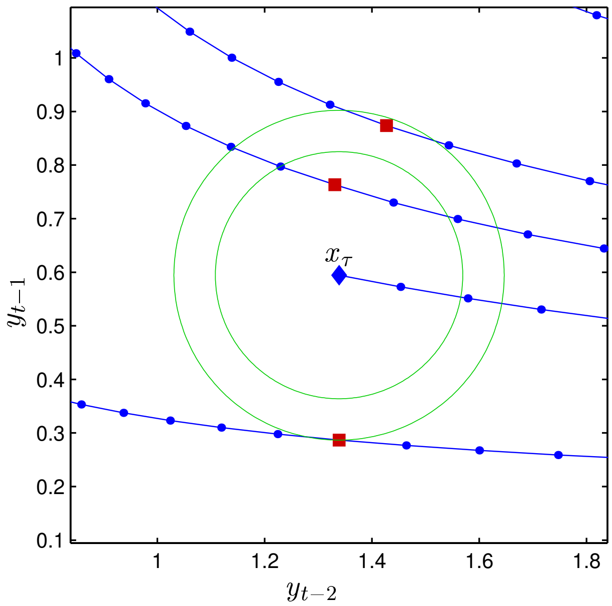

Figure 7:

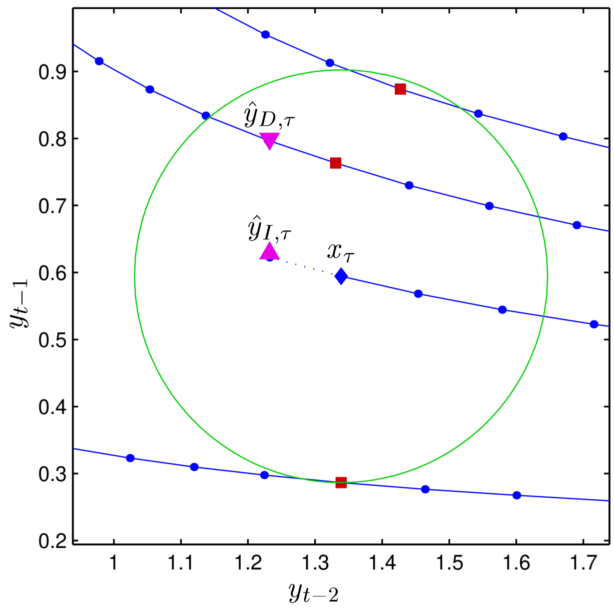

Figure 8:

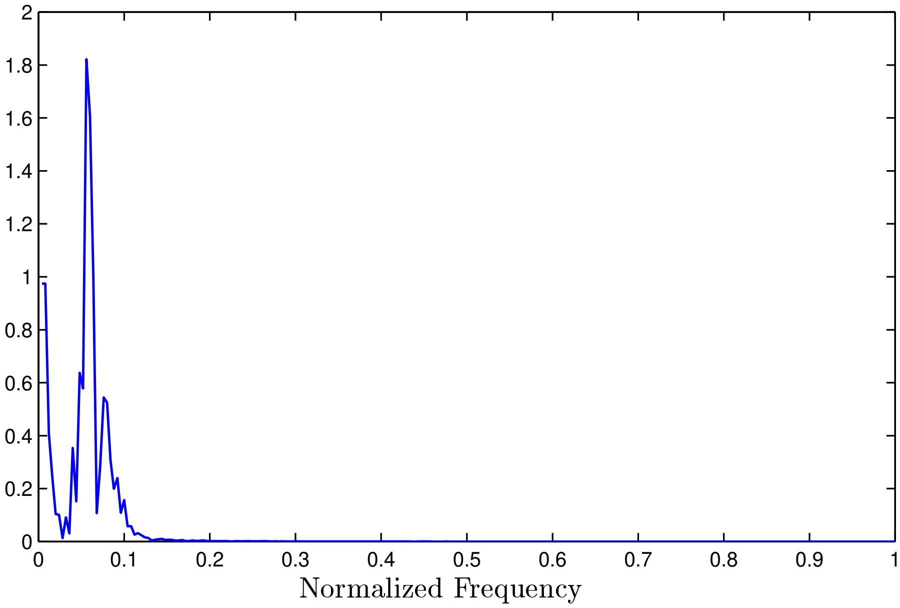

Figure 9:

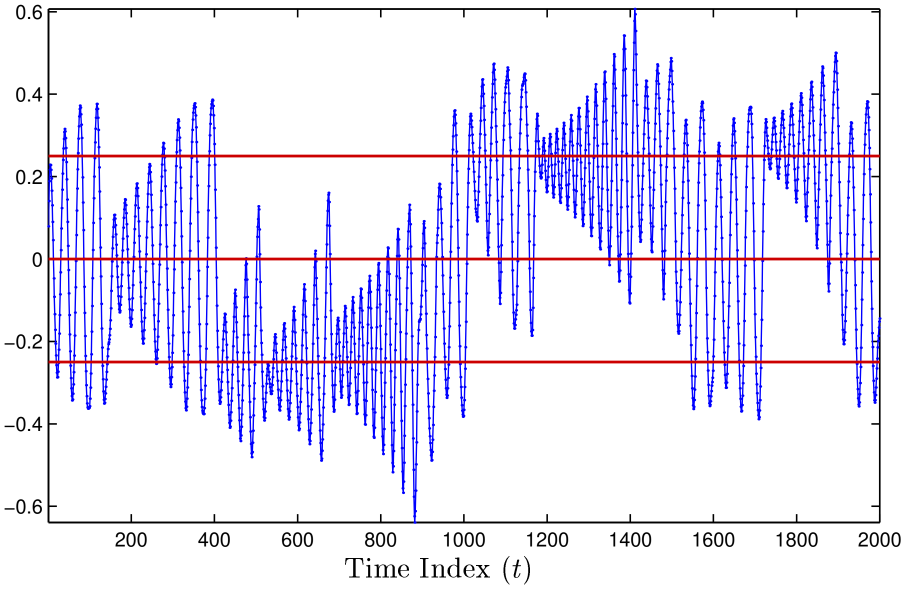

Figure 10:

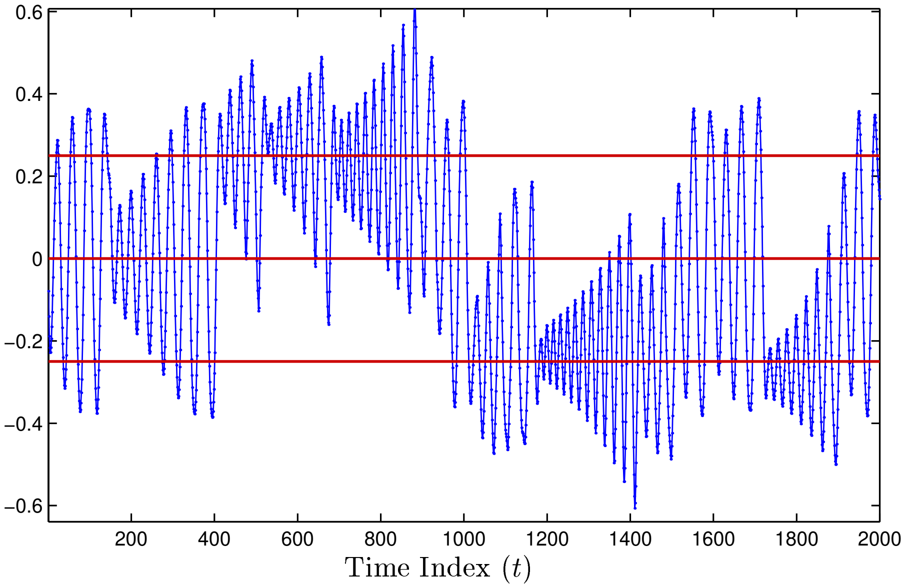

Figure 11:

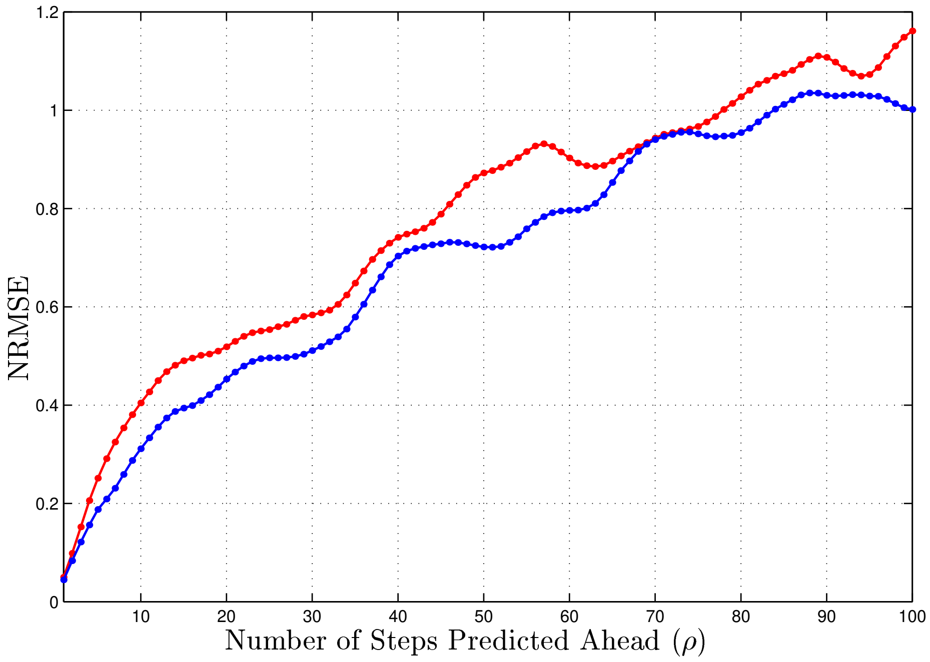

Figure 12:

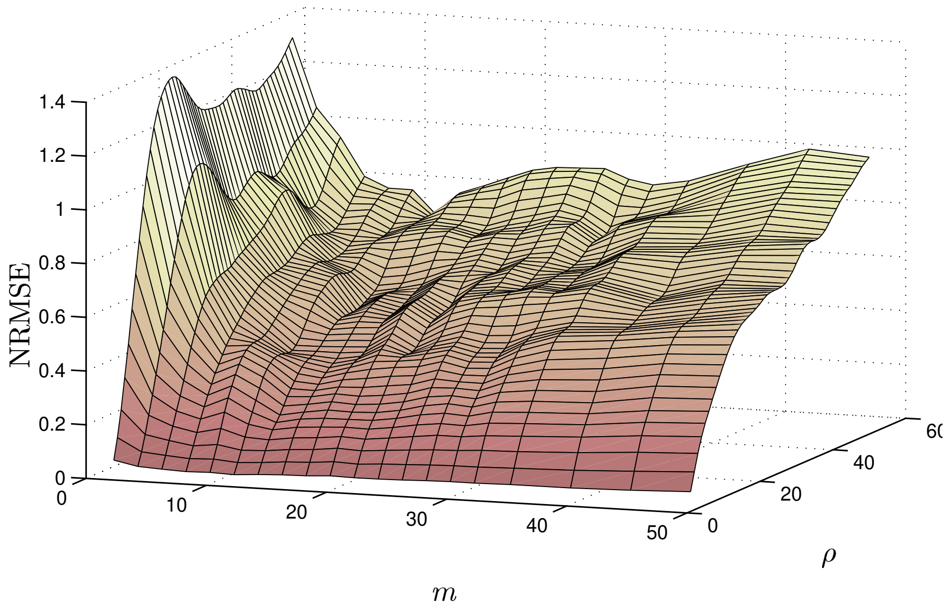

Figure 13:

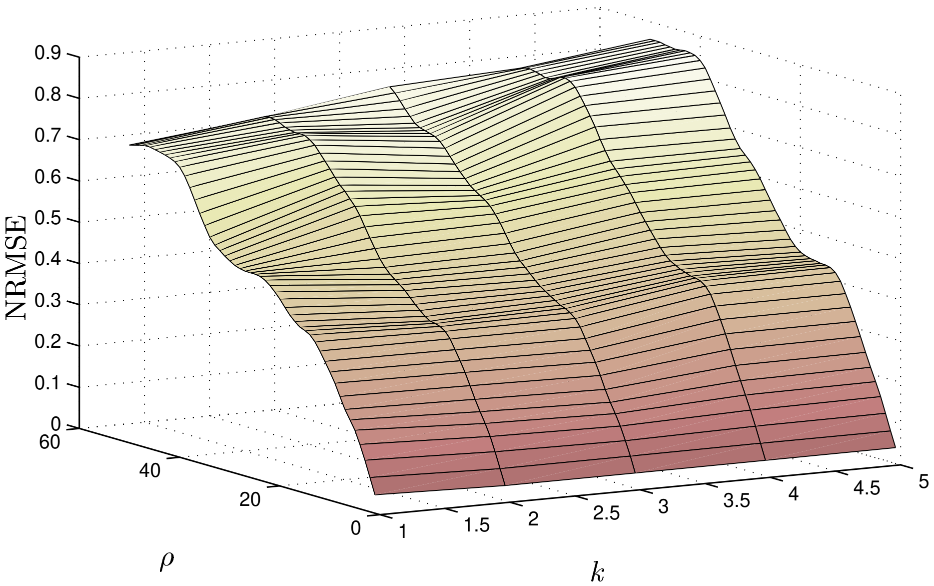

Figure 14:

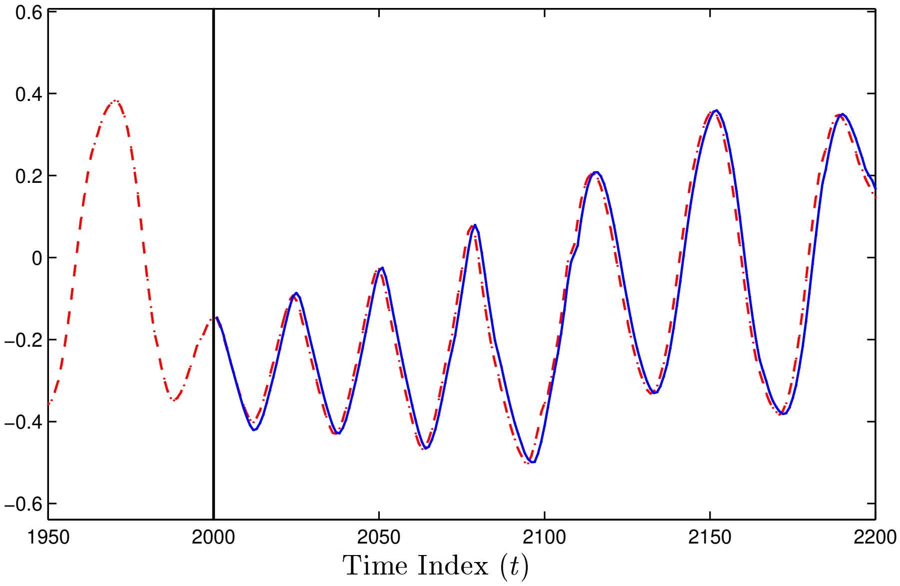

Figure 15:

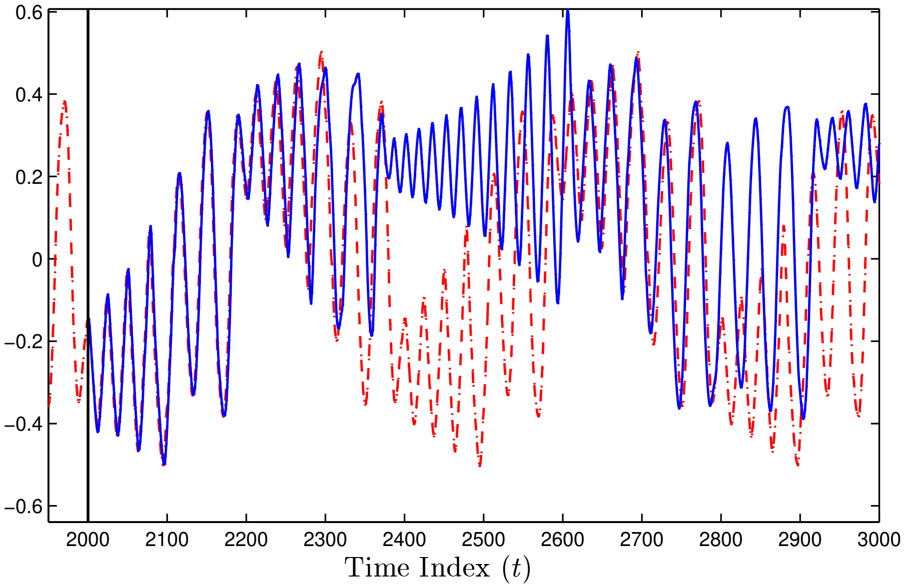
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
2000
2100
2200
2300
2400
2500
2600
2700
2800
2900
3000
Time Index (t)

Figure 16:

Preprocessing:	Both the original data set and the reflected data set were used to construct the local model
Nearest Neighbor Algorithm:	ANNA as described by McNames, [1998]. However, any nearest neighbor algorithm that incorporates the nearest trajectory modification described in Section 3.4 could have been used.
Nearest Neighbor Metric:	Exponentially weighted Euclidean (Equation 25) with user-selected parameters $\lambda_{\text{min}} = 0.05$ and $\delta = 2$
Local Model Type:	Integrated local averaging (Equation 28)
Local Weighting:	Biweight function (Equation 17)
Parameter Optimization:	Global minimization of multi-step ahead cross-validation mean squared error (Equation 33 with $n_v = 33$)
Optimized Parameters:	Embedding dimension: $m = 16$. Number of nearest neighbors: $k = 1$.

Table 1: Summary of local modeling method that generated the winning entry.