NONLINEAR MODELING AND ITS PREDICTION RANGE

Ahmet Ademoglu Michael E. Brandt and Tamer Demiralp

Institute of Biomedical Engineering Bogazici University, Istanbul
Deptartment of Psychiatry, University of Texas, Houston
Department of Physiology, Istanbul University, Istanbul

1. INTRODUCTION

The linear theory of prediction is capable of performing long term forecasting when the observed time series is linear. For the chaotic signals, where linear modeling techniques are insufficient, nonlinear approximation theory can be used effectively for predicting. Long-term prediction however, is always difficult to achieve for such signals because of the amplification of errors in each prediction step.

Prediction of chaotic time series dates back to the work by Farmer and Sidorowich [1] who employed the local linear polynomial approximation based on K nearest neighbors after time delay embedding of the signal in state space. Other studies such as higher order polynomial functions [1, 2], radial basis functions [2, 3], and neural networks [4, 5, 6, 7] have limited success in achieving long-term prediction of chaotic signals especially in the presence of noise. Since nonlinear signals are very sensitive to noise, it often becomes a problem to apply nonlinear techniques to estimate parameters such as correlation dimension, Lyapunov exponents and prediction error.

2. METHODS

Given a time series s_i, a state space trajectory is reconstructed using m-dimensional state vectors

$$\mathbf{x}_i = \left[s_i, s_{i+\tau}, ..., s_{i+(m-1)\tau}\right], \tag{1}$$

where τ is the time delay and m is the embedding dimension [8]. For τ-step ahead prediction of s_i, a nonlinear map f can be defined on this trajectory as

$$f(\mathbf{x}_i) = s_{i+m\tau}\,, \tag{2}$$

where f can be approximated using local or global polynomial kernels [1, 2, 9], neural networks [5], radial basis functions [2, 3] or wavelets [10].

2.1. Local Linear Polynomial Approximation

The simplest approach to local polynomial representation is the K nearest neighbor local linear (KNNLL) approximation. A local nonlinear approximation to f fits a hypersurface to the K nearest neighbors of $\mathbf{x}_i$. In the KNNLL approach, hyperplanes are used to approximate f. The time series s_i is divided into training and prediction data sets. For a vector $\mathbf{x}_p$ in the training set, the corresponding K nearest neighbor vectors $\mathbf{x}_{p_1} ... \mathbf{x}_{p_K}$ in the training set are determined using a suitable norm (*e.g.* Euclidean), and a least squares optimization is performed based on their τ-step ahead predictions

$$\begin{aligned} f(\mathbf{x}_{p_1}) &= s_{p_1+m\tau} \\ f(\mathbf{x}_{p_2}) &= s_{p_2+m\tau} \\ \vdots \;&\; \vdots \;\; \vdots \\ f(\mathbf{x}_{p_K}) &= s_{p_K+m\tau}. \end{aligned} \tag{3}$$

If f is to be approximated by a hyperplane with parameters $\mathbf{L}_j$ in the vicinity of the vector $\mathbf{x}_p$, then Eq. (3) becomes

$$\begin{bmatrix} 1 & s_{p_1} & s_{p_1+\tau} & \cdots & s_{p_1+(m-1)\tau} \\ 1 & s_{p_2} & s_{p_2+\tau} & \cdots & s_{p_2+(m-1)\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s_{p_K} & s_{p_K+\tau} & \cdots & s_{p_K+(m-1)\tau} \end{bmatrix} \begin{bmatrix} L_0 \\ L_1 \\ L_2 \\ \vdots \\ L_m \end{bmatrix}$$

$$= \begin{bmatrix} s_{p_1+m\tau} \\ s_{p_2+m\tau} \\ \vdots \\ s_{p_K+m\tau} \end{bmatrix}, \tag{4}$$

where $\mathbf{L}_j = [L_0 L_1 L_2 \cdots L_m]$. Least-squares optimization of Eq. (4) yields a local linear approximation of f around the vector $\mathbf{x}_p$. The hyperplane parameter vector $\mathbf{L}_j$ can be used for τ-step ahead prediction as

$$\hat{s}_{p+m\tau} \;=\; [1\ \mathbf{x}_p]\,\mathbf{L}_j{}'. \tag{5}$$

Using all the training vectors (as neighbors) for optimization corresponds to autoregressive AR modeling which can be expressed in terms of the data points s_i as

$$s_i = \sum_{k=1}^{m} a_k s_{i-k} + \epsilon_i\,, \tag{6}$$

where the a_k's are the AR coefficients, m is the model order and ϵ_i represents the unpredictable portion of the signal.

2.2. Radial Basis Function (RBF) Approximation

The radial basis function approximation is a single hidden layer network with one of the following radially symmetric functions in each node such as thin plate spline function

$$\phi(z) = z^2 log(z)\,, \tag{7}$$

the multiquadric function,

$$\phi(z,\rho) = (z^2+\rho^2)^{1/2}\,, \tag{8}$$

or the gaussian function,

$$\phi(z,\rho) = \exp(-z^2/\rho^2)\,, \tag{9}$$

ρ being a positive scalar. The overall input-output relation can be expressed as

$$f(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \phi(\|\mathbf{x}_i - \mathbf{c}_j\|)\,, \tag{10}$$

where $\mathbf{c}_j$ are the optimal centers and n is the number of nodes.

There are several methods of determining the optimal α_i and $\mathbf{c}_i$ in Eq. (10). One efficient technique called the orthogonal least squares (OLS) algorithm is described in [11]. Assuming that the nonlinear RBF function is chosen, the regression equations for the network can be expressed as

$$\mathbf{x}_i = \sum_{j=1}^{n} \theta_j \phi(\|\mathbf{x}_i - \mathbf{c}_j\|) + \epsilon_i, 1 \le i \le N\,, \tag{11}$$

where ϵ is the approximation error and N is the number of training inputs. Eq. (11) can be expanded in matrix form as

$$\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nn} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}, \tag{12}$$

where ϕ_{ij} is the output of the jth node to the ith input vector and the θ_i are the weight parameters to be estimated. More concisely, Eq. (12) can be expressed as

$$\mathbf{Y} = \mathbf{\Phi\Theta} + \mathbf{E}\,. \tag{13}$$

The orthogonal least squares method consists of transforming the columns of Φ into a set of orthogonal basis vectors where Φ can be expressed as

$$\mathbf{\Phi} = \mathbf{WB}\,, \tag{14}$$

where the columns of $\mathbf{W}$ are orthogonal and

$$\mathbf{B} = \begin{bmatrix} 1 & \beta_{12} & \cdots & \beta_{1n} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & \beta_{n-1n} \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{15}$$

Eq. (13) can be rewritten as

$$\mathbf{Y} = \mathbf{W\Gamma} + \mathbf{E}\,, \tag{16}$$

where $\mathbf{\Gamma}$ satisfies the triangular system

$$\mathbf{\Gamma} = \mathbf{B\Theta}\,. \tag{17}$$

The OLS solution for Γ is

$$\hat{\mathbf{\Gamma}} = (\mathbf{W}_T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Y} \tag{18}$$

2.2.1. The OLS Algorithm

Step 1; for $1 \le i \le n$, compute

$$u_i = \Phi_i \tag{19}$$
$$\gamma(i) = {u_i}^T\mathbf{Y}/({u_i}^T u_i) \tag{20}$$
$$e(i) = \gamma(i)^2 {u_i}^T u_i/(\mathbf{Y}^T\mathbf{Y}) \tag{21}$$

Find $\hat{e}_1 = \max\{e(i)\}$, $1 \le i \le n$, and select

$$\gamma(1) = \gamma_{i_k} \tag{22}$$
$$w_1 = \Phi_{i_1} \tag{23}$$

At the kth step, $k > 1$ for $i \ne i_1, ..., i_{k-1}$, compute

$$\beta_{ij} = {u_j}^T\Phi_i/({u_j}^T u_j), 1 \le j < k \tag{24}$$
$$w_i = \Phi_i - \sum_{j=1}^{k-1} \beta_{ij} u_j, \tag{25}$$
$$\gamma(i) = w_i^T\mathbf{Y}/({w_i}^T w_i), \tag{26}$$
$$e(i) = \gamma(i)^2 {w_i}^T w_i/(\mathbf{Y}^T\mathbf{Y}) \tag{27}$$

Find $\hat{e}_k = \max\{e(i)\}$, $1 \le i \le n$, $i \ne i_1, ..., i_{k-1}$, and select

$$\beta_{jk} = \beta_{j,i_k}, 1 \le j < k \tag{28}$$
$$\gamma(k) = \gamma(i_k) \tag{29}$$
$$u_k = \Phi_{i_k} - \sum_{j=1}^{k-1} \beta_{ji_k} u_j \tag{30}$$

Once the $\mathbf{B}$ and $\mathbf{\Gamma}$ are found $\mathbf{\Theta}$ can be determined by solving the upper triangular system given in Eq. (17).

The procedure is terminated when the sum of maximum errors in each step reaches to a certain tolerance value, i.e.

$$\sum_{j=1}^{M} \hat{e}_j < 1 - \zeta\,. \tag{31}$$

2.3. Neural Network Approximation

The multilayer perceptron (MLP) is a network topology which allows for an appoximation of any continuous function within an arbitary accuracy. The input/output relation for the kth layer is given as

$${z_i}^{(k)} = \sum_{j=1}^{n_k-1} \eta_{ij}^{(k-1)} x_j^{(k-1)} + \mu_i^{(k)} \tag{32}$$
$$x_i^{(k)} = \sigma(z_i^{(k)}) \tag{33}$$

where ${\eta_{ij}}^{(k)}$ and ${\mu_j}^{(k)}$ are the node connection weights, n_k is the number of hidden nodes in the kth layer and σ is the node activation function which is typically chosen as

$$\sigma(z) = \frac{1}{1+\exp(-z)}\,. \tag{34}$$

The output node usually does not have a threshold parameter and yields a linear combination of the outputs from the previous layer

$$\hat{y} = {x_i}^{(L)} = \sum_{j=1}^{n_{L-1}} \eta_{ij}^{(L)} x_j^{(L-1)} , \tag{35}$$

where L is the number of layers in the network. Nonlinear optimization techniques are used for the estimation of the parameters in these models. One of several algorithms that falls in this category described in [11] is the prediction error algorithm with batch processing.

2.3.1. Prediction Error Learning Algorithm for a Single Hidden Layer

The following definitions will be made for a single layer network:

- m: dimension of the input vectors
- r: number of hidden nodes
- $x_i(t)$: ith entry of the m dimensional input vector
- μ_i: threshold of the ith hidden node
- w_{ij}: connection weight from jth entry of the input vector to the ith hidden node
- $a_i(t)$: output of the ith hidden node
- v_i: connection weight from ith hidden node to the output node
- $\hat{y}(t, \Theta)$: output of the output node.

The input/output relationship can be expressed as

$$\hat{y}(t,\Theta) = \sum_{i=1}^{r} v_i a_i(t) = \sum_{i=1}^{r} v_i \sigma(\sum_{j=1}^{m} w_{ij} x_j(t) + \mu_i) , \tag{36}$$

All the parameters i.e the weights and thresholds of the network having m dimensional inputs and n hiddden nodes can be arranged into a P dimensional parameter vector

$$\Theta = [\theta_1 \cdots \theta_P]^T , \tag{37}$$

where $P = r(m+1) + r$. The difference between the system output and the network output

$$\epsilon(t,\Theta) = y(t) - \hat{y}(t,\Theta) , \tag{38}$$

is called the prediction error. The gradient of $\hat{y}(t,\Theta)$ with respect to Θ is a P dimensional vector

$$\Psi_i(t,\Theta) = \frac{d\hat{y}(t,\Theta}{d\Theta}$$

$$= \begin{cases} o_i(t) & \text{if} \quad \Theta_i = v_i, \quad 1 \le i \le r \\ v_i o_i(t)(1-o_i(t)) & \text{if} \quad \Theta_i = \mu_k, \quad 1 \le i \le r \\ v_i o_i(t)(1-o_i(t)) x_j(t) & \text{if} \quad \Theta_i = w_{kj}, \quad 1 \le k \le r, \quad 1 \le j \le m \\ 0 & \text{otherwise} \end{cases} , \tag{39}$$

The optimization criterion is chosen as

$$J_N(\Theta) = \frac{1}{2N} \sum_{t=1}^{N} \epsilon^T(t,\Theta)\epsilon(t,\Theta) , \tag{40}$$

where N is the length of the training data. The recursive learning algorithm is

$$\Theta(k) = \Theta(k-1) - \delta \Xi(\Theta(k-1)) , \tag{41}$$

where δ is a positive constant which guarantees convergence and $\Xi(\Theta)$ is the gradient search direction

$$\Xi(\Theta) = M(\Theta)(-\nabla J_N(\Theta)) , \tag{42}$$

where

$$\nabla J_N(\Theta) = -\frac{1}{N} \sum_{t=1}^{N} \Psi(t,\Theta)\epsilon(t,\Theta) , \tag{43}$$

is the gradient of $J_N(\Theta)$ with respect to Θ. $\mathbf{M}(\Theta)$ is chosen as the inverse of the approximate Hessian of $J_N(\Theta)$

$$\mathbf{M}^{-1}(\Theta) = \mathbf{H}(\Theta) = \frac{1}{N} \sum_{t=1}^{N} \Psi(t,\Theta)\Psi^T(t,\Theta) , \tag{44}$$

2.4. Nonlinear Prediction

The nonlinear approximation methods described previously can be used for predicting the future values of a nonlinear time series. Generally, the data is divided into a training and a prediction set. Then the nonlinear approximation of the trajectory is performed using the vectors in the training set. The approximated map is used for predicting the future values in the prediction set. More specificially, given a time series with N_1 points for training and N_2 points for prediction, the number of embedding vectors are $N_1 + N_2 - (m-1)\tau$. The first $N1-(m-1)\tau-1$ vectors are used for approximation of the map. Starting from the $N1-(m-1)\tau$th vector, the next $N_T = N_2 - M\tau + 1$ vectors are input to the map to perform N_T τ-step ahead predictions where M is the maximum number of iterative prediction steps. For $p\tau$-step prediction, the prediction routine is iterated p times for each vector chosen from the prediction set. The relative mean squared error (RMSE) for $p\tau$-step iterative prediction can be calculated as:

$$\epsilon(p) = \frac{\sum_{k=1}^{N_T} (s_{k+q} - \hat{s}_{k+q})^2}{N_T \sigma^2}, q = N_1 + (p-1)\tau , \tag{45}$$

where $\{\hat{s}_i\}$ and $\{s_i\}$ are the predicted and real sequences of data points respectively and σ is the standard deviation of $\{s_i\}$.

3. APPLICATION

The nonlinear prediction methods described above were tested on simulated data consisting of the discrete Henon map. The Henon map is given by

$$x_{n+1} = 1 - a{x_n}^2 + bx_{n-1} , \tag{46}$$

where the parameters are chosen as $a = 1.4$ and $b = 0.3$ with the initial condition

4. RESULTS

The KNNLL, the RBF and the MLP approximations were applied to the time series. 500 points were used for the training. The prediction error (RMSE) was calculated over 500 points in the testing set. The mean squared fitting error (MSFE) was also calculated for

each method. The optimal value of m and τ were searched for different values to obtain the best prediction performance. The number of neighbors for KNNLL, the number of centers for RBF, and the number of hidden nodes for MLP were optimized to achieve the lowest possible MSFE. The iterative prediction step after which the RMSE exceeded 10% was called *maximum range of prediction* and denoted by δ. The prediction performance for KNNLL, RBF and MLP approximation on Henon data are given in Table 1. For the KNNLL and RBF methods the fitting error cannot be reduced indefinitely. A maximum prediction range is obtained for a minimum fitting error. Increasing the number of neighbors in KNNLL or nunmber of centers in RBF does not reduce the fitting error and increase the prediction range. On the other hand, for the MLP, as the number of hidden nodes are increased the fitting error reduces and the prediction range increases.

Table 1: Prediction Range for KNNLL, RBF and MLP approximations. MSFE is the logarithm of the mean squared fitting error. n is the number of neighbors in KNNLL, p is the number of centers in RBF, and q is the number of hidden nodes in MLP.

KNNLL			RBF			MLP		
n	MSFE	δ	p	MSFE	δ	q	MSFE	δ
8	-4.4	7	12	-3.0	3	8	-2.8	3
12	-3.3	6	16	-3.1	3	10	-4.5	6
16	-3.3	6	20	-4.1	6	12	-4.3	7
20	-3.7	6	24	-4.3	6	14	-4.3	7
24	-3.5	5	28	-4.4	7	16	-6.7	12
28	-3.3	5	32	-4.6	8	18	-6.3	12
32	-3.1	5	36	-4.2	8	20	-6.8	13
36	-3.0	5	40	-3.6	6	22	-9.1	17
40	-2.9	4	44	-3.2	5	24	-9.7	18

5. DISCUSSIONS AND CONCLUSIONS

There appears to be a relation between the MSFE and the δ. Reducing the MSFE yields a higher δ hence a better prediction performance. The MLP seems to have the capability of reducing the MSFE to an arbitrary precision provided that the number of hidden nodes and the number of iterations for recursive paramater estimation are high enough. KNNLL and RBF methods do not seem to have such an accuracy when it comes to approximate the map, given a limited amount of data. There is an optimum value, for the number of neighbors in KNNLL and for the number of centers in RBF, that yields the lowest possible MSFE. Increasing or decreasing this value only deteriorates the fitting and the prediction performance.

6. REFERENCES

[1] J D Farmer and J J Sidorowich, "Exploiting chaos to predict the future and reduce noise," preprint, Los Alamos, LA-UR-88-901, 1988.

[2] M Casdagli, "Nonlinear prediction of chaotic time series," *Physica D*, vol. 35, pp. 335–356, 1989.

[3] D. Lowe D and A. R. Webb, "Time series prediction by adaptive networks: a dynamical systems perspective," *IEE Proc.-F*, vol. 138, pp. 17–24, 1992.

[4] A. S. Lapedes and R. Farber, "Nonlinear signal processing using neural networks:prediction and system modeling," preprint, Los Alamos, LA-UR-87-2662, 1987.

[5] A S Weigend, B A Huberman, and D E Rumelhart, "Predicting sunspots and exchange rates with connectionist networks," in *Nonlinear Modeling and Forecasting*, M Casdagli and S Eubank, Eds., vol. 12, chapter 395–432. Addison-Wesley, Massachusettes, 1992.

[6] A. M. Albano A. Passamante T. Hediger and M. E. Farrell, "Using neural nets to look for chaos," *Physica D*, vol. 58, pp. 1–9, 1992.

[7] J. C. Principe A. Ratie and J. M. Kuo, "Prediction of chaotic time series with neural networks and the issue of dynamic modeling," *Int J Bifur and Chaos*, vol. 2, pp. 989–996, 1992.

[8] F Takens, *Detecting strange attractors in turbulence*, vol. 898 of *Lecture Notes in Mathematics*, pp. 366–381, Springer, Berlin, 1981.

[9] K Stokbro and D K Umberger, "Forecasting with weighted maps," in *Nonlinear Modeling and Forecasting*, M Casdagli and S Eubank, Eds., vol. 12 of *SFI Studies in the Sciences of Complexity*, pp. 73–93. Addison-Wesley, Massachusettes, 1992.

[10] L Cao, Y Hong, H Fang, and G He, "Predicting chaotic time series with wavelet networks," *Physica D*, vol. 85, pp. 225–238, 1995.

[11] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *Int J of Control*, vol. 56, pp. 319–346, 1992.

[12] M Casdagli, "Chaos and deterministic versus stochastic nonlinear modeling," *Journal of Royal Statistical Society B*, vol. 54, pp. 303–328, 1992.