Chaotic Time Series Prediction Using Data Fusion

Mohamed Aly
Department of Electrical and Computer Engineering
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
mohamedaa@hotmail.com

Henry Leung
Department of Electrical and Computer Engineering
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
leungh@enel.ucalgary.ca

Abstract - One of the main problems in chaotic time series prediction is that the underlying nonlinear dynamics is usually unknown. Using a nonlinear predictor to predict a chaotic time series usually puts a limit on the accuracy since the nonlinear predictor is basically an approximation of the unknown nonlinear mapping. In this paper, we propose using fusion of predictors as a method to improve the performance of chaotic time series prediction. Different nonlinear predictors with distinct characteristics including the multi-layer perceptron neural network, radial basis function (RBF) neural network, fuzzy inference system, recurrent neural network, Volterra filter, and local linear predictor are used to predict a chaotic time series. Their predictions are then combined to produce a more accurate prediction by using the linearly constrained least square (LCLS) fusion method. The proposed prediction fusion method is evaluated using simulated chaotic time series based on the Mackey-Glass equation and Ikeda system. Results show that the fused predictor consistently outperforms all the individual predictors.

Keywords: Sensor fusion, time series prediction, neural networks, fuzzy logic.

1 Introduction

Accurate forecasting of time series is a problem that has diverse range of applications including economic and business planning, stock market index, inventory and production control, signal processing, electrical power consumption, temperature and weather related forecasting, radar clutter suppression, speech coding and many others. Since a wide range of signal processes such as acoustic and radar clutter have been shown to be chaotic, prediction of chaotic time series becomes a problem of great interest to many researchers. Different kinds of nonlinear and intelligent techniques such as Volterra filter, neural networks, fuzzy logic and other techniques have been applied to predict simulated and real chaotic data sequence.

One of the main problems in using these nonlinear predictors to predict a chaotic time series is that these nonlinear predictors use some nonlinear functions with universal approximation capability to approximate the unknown nonlinear relationship underlying the chaotic sequence. It therefore puts a limit on the accuracy since these nonlinear predictors must have some errors in the approximation.

In this paper, we propose using predictors with different functional forms to predict a chaotic time series. The predictions of these nonlinear predictors are then combined using some data fusion technique. In particular, the linearly constrained least square (LCLS) fusion method is proposed for prediction fusion. The LCLS method combines the different predictors in a linear fashion. The linear weightings are then determined using the least square method with a constraint that the sum of these weights are equal to one. Because the fused prediction is combined optimally in the least square sense, the final prediction result is better than all the individual predictors in terms of prediction error variance.

The paper is organized as follows. Section II briefly discusses various predictors that are widely used for chaotic time series prediction. The LCLS predictor fusion method is presented in Section III. Performances of the nonlinear predictors, along with the prediction fusion method are evaluated in Section IV.

2 Nonlinear prediction of chaotic time series

In chaotic time-series prediction, past values of the time series up to the point in time, say \( t \), are used to predict the value at some point in the future, say \( t+p \). The standard method for this type of prediction is to create a mapping from \( d \) sample data points, \( x(t), x(t-1), \ldots, x(t-(d-1)) \) which can be represented in vector form as \( x(t) = (x(t), x(t-1), \ldots, x(t-(d-1))) \) to a predicted future value \( x(t+p) \). In this paper we only study the case of one step ahead prediction where \( p=1 \), that is predicting \( x(t+1) \).

Based on the Takens theorem [1,2], we know that if the embedding dimension \( d \) is greater than or equal to \( 2d_a + 1 \) where \( d_a \) is the attractor dimension of the chaotic system, the prediction can reconstruct the dynamic from its time series measurements, at least in theory. To determine a suitable embedding dimension, we can first use the correlation
The next step is to find out the functional relationship between the current state \(x(t)\) and the future sample \(x(t+1)\).

\[
x(t+1) = f(x(t))
\]

(1)

Since \(f\) is usually unknown, we need to find another function \(g\) to approximate \(f\). By choosing approximate parameters in \(g\), we can try to make \(g\) close to \(f\). To obtain these appropriate parameters of \(g\), we usually use part of the time series as a training set. During training, the model parameters are adjusted based on the known prediction results to minimize the prediction error. It should be noted that the number of training data points should be several times larger than the number parameters being estimated.

Multiresolution learning [4,5,6] is employed in our training of predictors. It creates a more general model, therefore improving the effectiveness of these predictors and helps the predictor to have a better generalization rather than a better fitting on the training data. It is based on the multi-resolution analysis in wavelet theory, where the signal is decomposed and approximated on different levels of details. Approximation sequences by representations from coarsest to finest are introduced in the training process.

Assume that the training sequence of \(x(t)\) is a time series called \(s^m(t)\) at the finest resolution \(2^m\). Using wavelet analysis it can be represented as the addition of a coarser reconstructed approximation, and a reconstructed detail signal,

\[
s^m(t) = s^{m-1}(t) + d^{m-1}(t)
\]

(2)

where \(s^{m-1}(t)\) is a coarser representation of the training sequence \(s^m(t)\) at resolution \(2^{m-1}\), and \(d^{m-1}(t)\) is the reconstructed detail. This process can be repeated for more approximation levels, say \(L\) levels. Then the sequence of training time series \(s^{m-1}(t), s^{m-2}(t), \ldots, s^0(t)\) can be used in this order to train a nonlinear predictor in a sequence of training activities. The first level of learning is basically a simplified version of the original signal. The finer resolution is then introduced so that the predictors only need to learn the small incremental details to refine the overall learning behavior. We now briefly present the nonlinear predictors that are used in our prediction fusion study.

### 2.1 Multi-Layer Perceptron

Feed-forward networks [7,8,9] often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. We used a three layer network, the first layer is input layer consisting of \(d\) input neurons representing the \(d\)-element embedded vector \(x(t)\), then a hidden layer of \(n_h\) hidden neuron, and last layer is the output layer that has only one output neuron representing the output \(x(t+1)\). The hidden layer has a sigmoid transfer function, and the output layer has a linear transfer function. This sigmoid/linear network can be used as a general function approximator. The predicted value \(x(t+1)\) can be calculated from the following equation,

\[
x(t+1) = \sum_{i=1}^{n_h} w_i^o (\frac{2}{1 + e^{(\sum_{j=1}^{m-1} w_{ij}^o x(t-(j-1))+b_i^o)}}) + b_o^o
\]

(3)

where \(w_{ij}^o\) is the weight from neuron \(i\) of the hidden layer to the output neuron, \(b_i^o\) is the bias of the output layer, \(b_o^o\) is the bias of the hidden layer, and \(w_{ij}^o\) is the weight from input neuron \(j\) to hidden neuron \(i\). These weights and biases are determined through the training process. The number of hidden nodes is determined by cross training the network.

A network of this structure can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer. Sigmoid transfer functions by nature cover the whole input spectrum with more focus on local details. To achieve better network generalization and avoid network overfitting, the early stopping method was used in addition to the multiresolution learning mechanism.

### 2.2 Radial Basis Function Neural Network

Radial Basis functions are used for strict interpolation in a multi-dimensional space. Radial Basis networks [7,9] consist of an input layer representing the \(d\)-element vector \(x(t)\), a hidden layer consisting of \(n_h\) radial basis neurons, and an output layer of one linear neuron. Hidden layer neurons have a radial basis bell shaped transfer functions, while output neurons have linear transfer functions. In this case the Gaussian function is used as the radial basis transfer function. The predicted signal is calculated from the following equation,

\[
x(t+1) = \sum_{j=1}^{n_h} w_j^o e^{-\|x(t)-w_j^o\|^2} + b_o^o
\]

(4)

where \(w_{ij}^o\) is the weight from the \(j^{th}\) neuron of the hidden layer to the output neuron, \(b_i^o\) is the bias of the output layer, \(w_j^o\) is the \(d\)-element weight vector of the \(j^{th}\) hidden neuron., and \(b_o^o\) is bias of the hidden layer, which controls the spread (variance) of the Gaussian transfer function. The network is trained to determine the weights and biases. The number of hidden layer neurons can be determined through iteratively adding neurons until the network error falls under a certain training error goal. Radial basis transfer functions by nature tend to cover a local portion of the input spectrum thus having better focus on local details.

### 2.3 Fuzzy Neural Network

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. In this paper we used the first order Sugeno (or Takagi-Sugeno-Kang) method of fuzzy inference [10,11]. Inputs to the network are assigned membership functions representing...
ranges of the input spectrum (for example, \( Hi \) and \( Lo \)). A set of rules are developed as shown in Equation 4, to generate the mapping between variable memberships and linear output,

\[
\text{if } x(t) \text{ is } Hi \text{ and } x(t-l) \text{ is } Lo \text{ then }
\]

\[
z = p^*x(t) + q^*x(t-l) + r
\]

(5)

where \( Hi \) and \( Lo \) are fuzzy membership functions of the antecedent, with each input is fuzzified, while \( p, q, \) and \( r \) are all constants. The second layer of the network will have a linear combination of the fuzzy rules outputs. During training the membership functions are reshaped, and set the rules that minimize the training errors are determined.

Because of the linear dependence of the rule on the system’s input variables, the Sugeno system is suited for modeling nonlinear systems by using a piece wise linear systems.

### 2.4 Recurrent Neural Network

The Elman recurrent network is employed here for time series prediction, it is similar to the multi-layer perceptron back-propagation networks, with the addition of a feedback connection from the output of the hidden layer to its input. Its model equation will be the same equation as in the multi layer perceptron neural network, with the exception that the output of the hidden layer is delayed and fed back into its inputs.

\[
x(t+1) = \sum_{i=1}^{m_h} \left( \frac{2}{1+e^{-(\sum_j w_{ij} + \sum_k a(k) + b)}} \right) + b'
\]

(6)

where \( w_{ij} \) is the weight from neuron \( i \) of the hidden layer to the output neuron, \( b' \) is the bias of the output layer, \( b \) is the bias of the hidden layer, and \( w_{hj} \) is the weight from input neuron \( j \) to hidden neuron \( k \). The delayed outputs of the hidden layer (where \( a(k) \) is the delayed output of for neuron \( k \)) are fed back to this layer, with the weight \( w_{i,k} \) representing the feedback weight from the \( k \)th neuron to the \( i \)th neuron of the hidden layer.

\[
a(k) = \left( \frac{2}{1+e^{-(\sum_j w_{kj,l} + \sum_k a(k) + b)}} \right)
\]

(7)

These weights and biases are determined through the training process. The number of hidden nodes is determined by cross training the network.

### 2.5 Volterra Filter

Volterra filters are considered an extension of linear least squares filter. We used the second order Volterra filter [7,13] of the form,

\[
x(t) = \sum_{i=1}^{d} h_i x(t-(i-1)) + \\
\sum_{j=1}^{d} \sum_{k=1}^{d} h_{jk} x(t-(j-1))x(t-(k-1))
\]

(8)

where \( h_i \)'s are the linear coefficients, \( h_{jk} \)'s are the nonlinear (second order) model coefficients. This is basically the same least squares model of linear predictor, but applied on the expanded input vector \( u(t) \), where

\[
u(t) = (x(t), x(t-l),...x(t-(d-1)), \\
x(t)^2, x(t)x(t-l),...x(t-(d-1))^2)
\]

(9)

### 2.6 Local Predictor

Local predictors use the concept of fitting local models using spatial neighbor states. They build globally non-linear models, while fitting few parameters of local models. The closest spatial neighbors of a the current input vector \( x(t) \) are selected based on a cutoff distance \( \varepsilon \),

\[
\left\| x(t) - x(t_j) \right\| < \varepsilon
\]

(10)

All vectors \( \{ x(t_j) \}, j=1,2,... \) satisfying this equation are chosen to build a linear model by either using the average of their outputs, or a weighted average based on their distances. One improvement that we have implemented is taking also temporal neighbors [13] to the current vector into account. Local predictors have the advantage of overcoming change in system parameters by utilizing only part of the history. However, they need a plenty of training data to give accurate results.

### 3 Fusion of Predictors using the Linearly Constrained Least Square (LCLS) technique

The objective of prediction fusion is to have a linear combination of predictor outputs, by assigning larger weights to predictors with less error and smaller weights to those of larger prediction errors [14,15], thus reducing the uncertainty by utilizing redundant information.

The Linearly Constrained Least Squares (LCLS) is a modified version of the optimal method Linearly Constrained Minimum Variance (LCMV) [16,17,19]. It does not require a priori knowledge of signal statistics, and it uses the least square estimates instead.
When combining $M$ predictors, where the $m$th predictor is $\hat{x}_m(t)$, using the training sequence $s(t)$, a vector $\hat{x}(t) = \sum_{i=1}^{M} w_i \hat{x}_m(t)$ is constructed based on the $m$ predictions with the constraint $\sum_{i=1}^{M} w_i = 1$. The optimum weights $\hat{w}(t) = (w_1, w_2, ..., w_M)$ can be calculated from the following equation,

$$\hat{w} = \hat{R}_a^{-1} \hat{p}_{ss} + \hat{R}_a^{-1} a^T \hat{R}_a^{-1} a^{-1} (1 - a^{-1} \hat{R}_a^{-1} \hat{p}_{ss})$$  \hspace{0.5cm} (11)$$

where $\hat{R}_a$ is the sample covariance matrix of the predictors, $\hat{p}_{ss}$ is the cross covariance vector, and $a$ is a vector of ones.

$$\hat{R}_a = \sum_{i=1}^{N} \hat{x}(t) \hat{x}(t)^T, \hat{p}_{ss} = \sum_{i=1}^{N} \hat{x}(t) s(t)^T$$  \hspace{0.5cm} (12)$$

Colinearity and correlation [18] may appear in fusion of predictors when taking the inverse of covariance matrix in consideration, causing instability due to linear dependence (multi-colinearity) in the rows and columns of the error covariance matrix. This problem can be avoided by using principal component preprocessing.

## 4 Performance Evaluation

The results of two chaotic time series simulation data (Ikeda chaotic series, and Mackey Glass series) are presented, with the improved performance of the fusion compared to the best and worst individual predictors.

### 4.1 Ikeda chaotic system

Here we consider predicting the well-known Ikeda chaotic system. It is described by the following system of coupled difference equations

$$x(t) = 1 + \mu [x(t - 1) \cos(v) - y(t - a) \sin(v)]$$ \hspace{0.5cm} (13)$$

$$y(t) = \mu [x(t - 1) \cos(v) - y(t - a) \cos(v)]$$ \hspace{0.5cm} (14)$$

$$v = 0.4 - \frac{6}{1 + x(t - 1)^2 + y(t - 1)^2}$$ \hspace{0.5cm} (15)$$

We generated 1400 data points for our test, used 600 of them for training the predictors and fusion algorithm, and the remaining 800 for testing. All the predictors discussed above were applied, and their performance was evaluated.

Multiresolution learning was used to train the predictors. It had improved the performance of the MLP neural network predictors with about 38% reducing the overall prediction error from 0.0031 to 0.0019. The three predictors that had the best performance were the MLP neural network, the RB neural network, and the fuzzy inference system. These three predictors were used for linear fusion, as shown in Table 1.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>MLP</th>
<th>RB</th>
<th>ANFIS</th>
<th>LCLS Fused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set error</td>
<td>0.0021</td>
<td>0.0022</td>
<td>0.0011</td>
<td>0.0007</td>
</tr>
<tr>
<td>Entire data error</td>
<td>0.0019</td>
<td>0.0028</td>
<td>0.0019</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

The LCLS fusion algorithm is used, as in Equation (11). It is clear that the improvement of using LCLS predictor...
fusion is 9.6% error reduction. Fig. 2 shows in details the predicted signal and prediction error for the best, worst and fused predictors.

![Figure 2. Mackey-Glass chaotic system prediction and prediction errors](image)

We tested the case where there is not enough data for training, by using 300 samples for training, all predictors gave poor results, but the improvement of fusion was 22%, which shows that linear fusion of predictors is more robust for insufficient training data than individual predictors.

5 Conclusions

The previous section clearly demonstrated the effectiveness of the proposed fusion of predictors using the LCLS method. The fused prediction outperforms all the individual predictors in the sense of prediction accuracy. In addition, fusion of predictors using the LCLS method has shown to be robust for insufficient training data, where almost all individual predictors will give poor results, the fusion of predictors will have more performance improvement.

References