Modeling and prediction with a class of time delay dynamic neural networks

Yasar Becerikli a,*, Yusuf Oysal b

a Department of Computer Engineering, Kocaeli University, Izmit, Turkey
b Anadolu University, Computer Engineering Department, Eskisehir, Turkey

Available online 28 February 2006

Abstract

In this paper, we propose a time delay dynamic neural network (TDDNN) to track and predict a chaotic time series systems. The application of artificial neural networks to dynamical systems has been constrained by the non-dynamical nature of popular network architectures. Many of the drawbacks caused by the algebraic structures can be overcome with TDDNNs. TDDNNs have time delay elements in their states. This approach provides the natural properties of physical systems. The minimization of a quadratic performance index is considered for trajectory tracking applications. Gradient computations are presented based on adjoint sensitivity analysis. The computational complexity is significantly less than direct method, but it requires a backward integration capability. We used Levenberg–Marquardt parameter updating method.

Keywords: Dynamic neural networks; Time delay; Attractor; Chaos; Tracking trajectory; Prediction; Adjoint theory

1. Introduction

Chaotic time series are considered as the outputs of nonlinear dynamic systems. If one cannot specify the initial condition with infinite precision, the long time future behavior of these time series is unpredictable. But, the short time behavior can be exactly encapsulated. Many types of time delays are observed such as axonal propagation delays and synaptic transmission delays in biological neural networks. There are various types of neural networks with time delays. For solving time-sequence recognition problem, the delayed synaptic connections were used in some neural networks [1,2]. Such time-delay neural networks (TDNNs) have been widely used in some practical engineering problem such as nonlinear predictions and recognition. Hebbian-type neural network was based as delayed feedback connections [3,4]. Stability analysis of TDNNs has been extensively analyzed in many studies [5–9].

Our focus in this work is to model and predict a nonlinear system with time delay by using time delay dynamic neural networks (TDDNNs) with fast supervised training algorithms such as Levenberg–Marquardt algorithm [10,11]. The TDDNN stands for a continuous-time recurrent neural network that has time-delayed feedbacks. The gradient algorithm is based on adjoint theory [12–17] that is faster than the forward method.

In Section 2, we present a model for time-delay dynamic neural networks (TDDNNs) and describe the class of applications we have considered—trajectory tracking and prediction. Some illustrative examples are given in Section 3. Given a desired trajectory, a nonlinear optimization problem must be solved to determine appropriate values for network parameters, and we have employed gradient-based approaches, discussed in Section 4. Chaotic time series prediction experimental results are presented in Section 5.

2. The time delay dynamical neural network model structure

The dynamical system considered describes an electronic circuit of $n$ saturable amplifiers (called neurons) coupled by a resistive interconnection matrix (that is weights). This can be contrasted three simpler network architectures: (i) In which connectivity is constrained to be feedforward and there are no dynamics in the processing units. Feedforward/algebraic networks are the workhorses of neural network applications. (ii) In which feedback connections are allowed but dynamics in processing units are not. Networks of this type have been used for several symbolic processing tasks [18–20]. (iii) In which the connectivity is constrained to be feedforward but dynamical and delay elements are included. For example, the output of the unit...
nonlinearity can be passed through a linear time-delayed filter before serving as input the other, downstream, units [4–9,21].

The computational model for time-delay dynamic neural networks (TDDNNs) with two-neuron two-input/outputs that we have used is shown in Fig. 1. In general, there are \( L \) input signals and \( n \) neural network units. The units have dynamics associated with them, as indicated, and they receive inputs from themselves and delayed-themselves and all other units. The output of a unit \( y_i \) is a general sigmoidal function \( h(x_i, \gamma, \beta_i) \) of a state variable \( x_i \) associated with the unit. The output of the network is a linear weighted sum of the unit outputs. Weights \( p_{ij} \) are associated with the connections from input signals \( j \) to units \( i \), \( w_{ij} \) with inter-unit connections from \( j \) to \( i \), \( w_{dij} \) with delay-inter-unit connections from \( j \) to \( i \) and \( q_{ij} \) is the output connection weights from \( j \)th neuron to \( i \)th output. \( T_i \) is the dynamic constant of \( i \)th neuron and \( b_i \) is the bias (or polarization) term of \( i \)th neuron. \( \gamma_i \) and \( \beta_i \) are sigmoidal nonlinearity parameters. And the lastly, \( \tau_i \) is the time-delay of a neuron.

The general computational model is summarized in the following equations:

\[
\begin{align*}
\dot{x}_i &= f_i(x_i, p) = \frac{1}{T_i} \left[ -x_i + \sum_{j=1}^{n} w_{dij} x_j (t - \tau_j) + \sum_{j=1}^{n} w_{ij} y_j + \sum_{j=1}^{L} p_{ij} u_j + b_i \right], \\
x_i(0) &= x_{i0}, \quad i = 1, \ldots, n \\
z_i &= \sum_{j=1}^{n} q_{ij} y_j, \quad i = 1, 2, \ldots, M \\
y_i &= h_i(x_i, \gamma, \beta_i) = \frac{1}{1 + e^{-\gamma_i x_i + \beta_i}}, \quad i = 1, 2, \ldots, n
\end{align*}
\]

The initial conditions on the state variables \( x_i(0) \) must be specified. This model is similar to those in the literature [4–9,22].

3. Illustrative modeling examples of some dynamical behaviors of TDDNN

This model (TDDNN) approximates some most known nonlinear behaviors of nonlinear dynamic systems with delays. In this section, some examples are given in which TDDNN converges to an attractor or limit cycle, oscillator and chaos. Problem of training trajectories by means of continuous recurrent neural networks with delay whose feedforward parts are as multilayer perceptron was currently studied.

Given a set of parameters, initial conditions, and input trajectories, the set of Eqs. (1)–(3) can be numerically integrated from \( t = 0 \) to final time \( t_f \). This will produce trajectories overtime for the state variables \( x_i \). We have used Runge–Kutta with five-degree method [23,24]. The integration step size has to be commensurate with the temporal scale of dynamics, determined by the time constants \( T_i \). In our work, we have specified a lower bound on \( T_i \) and have used a fixed integration time step of some fraction (e.g., 1/10) of this bound. Fig. 1 shows an example of state diagram of two-unit TDDNN with two-input/two-outputs.

3.1. TDDNN as point attractor example

Consider a point attractor system. The interconnection and some weight parameters of TDDNN obtained from a training algorithm whose details will be given in next section are as follows:

\[
\begin{align*}
w &= \begin{bmatrix} 0 & 1 \\ -10 & 0 \end{bmatrix}, & \quad w_{d} &= \begin{bmatrix} 0.75 & 0 \\ 0 & -0.75 \end{bmatrix}, & \quad b &= \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \\
T &= \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, & \quad \gamma &= \begin{bmatrix} 1 \end{bmatrix}, & \quad \beta &= \begin{bmatrix} .5 \\ .5 \end{bmatrix}, & \quad x(0) &= \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \\
\tau &= 19, & \quad u(t) &= 0.0
\end{align*}
\]
3.2. TDDNN as periodic attractor example

In this application, an oscillator system is modeled with a two-neuron-TDDNN. The interconnection parameters and some neuron parameters in this case are

\[ w = \begin{bmatrix} 0 & 1.9 \\ 1.9 & 0 \end{bmatrix}, \quad \text{wd} = \begin{bmatrix} -0.95 & 0 \\ 0 & -0.95 \end{bmatrix}, \quad b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \]

\[ T = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \beta = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \quad \tau = 19, \quad u(t) = 0.0 \]

TDDNN converges to a limit cycle (periodic attractor) situation for given initial conditions \( (x_i(0), i = 1, 2) \). Fig. 3 shows an example of state space trajectories for two-unit networks with time delay (19 s) and zero input. Trajectory tracking performance is excellent in this application.

3.3. TDDNN as three strange attractor examples

The interconnection parameters of the TDDNN by experiencing were found to be as follows, respectively:

\[ w = \begin{bmatrix} 10 & 1.9 \\ 1.9 & 10 \end{bmatrix}, \quad \text{wd} = \begin{bmatrix} -0.95 & 0.1 \\ 0 & -0.1 \end{bmatrix}, \quad b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \]

\[ T = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \beta = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \quad \tau = 19, \quad u(t) = 0.0 \quad \text{for strange attractor 1} \]

\[ w = \begin{bmatrix} 10 & 11.9 \\ 1.9 & 10 \end{bmatrix}, \quad \text{wd} = \begin{bmatrix} -0.95 & 0.1 \\ 0 & -0.1 \end{bmatrix}, \quad b = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \]

\[ T = \begin{bmatrix} 0.5 & 0.2 \\ 0 & 0.5 \end{bmatrix}, \quad \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \beta = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad x(0) = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \quad \tau = 19, \quad u(t) = 0.0 \quad \text{for strange attractor 3} \]

TDDNN converges to a very interesting and strange situations for given initial conditions \( (x_i(0), i = 1, 2) \). Fig. 4(a)–(c) shows state space trajectories for two-unit networks with time delay (19 s) and zero input. These simulations show that this type of TDNNN has very interesting and challenging capabilities.

4. Parameter identification based on adjoint sensitivity analysis for TDDNN training

The TDDN training is done to encapsulate a given set of trajectories by adjusting network parameters. In this section, adjusting parameters of TDDNN is aimed for trajectory tracking. This is done by minimizing the cost functional (error functional). The gradient-based algorithms have been used for this problem. Cost gradients with respect to the network parameters are required for this algorithm. Our focus in this paper has been on adjoint sensitivity analysis for calculating the cost gradients with respect to all TDDNN parameters. The total network parameters are \( w, p, q, b, T, \tau, \gamma, \beta \).

A performance index or cost structure is selected in the simple quadratic form as follows:

\[ J = \frac{1}{2} \int_0^T [z(t) - z^d(t)]^T [z(t) - z^d(t)] \, dt \quad (4) \]

where \( e(t) = z(t) - z^d(t) \) is error function, \( z(t) \) the TDDNN model response (output) and \( z^d(t) \) is the desired (target) system response. It is necessary to compute the cost sensitivities with respect to various parameters:

\[ \frac{\partial E}{\partial w}, \frac{\partial E}{\partial p}, \frac{\partial E}{\partial q}, \frac{\partial E}{\partial T}, \frac{\partial E}{\partial b}, \frac{\partial E}{\partial \tau}, \frac{\partial E}{\partial \gamma}, \frac{\partial E}{\partial \beta} \quad (5) \]
Gradients with respect to the output weights can be easily obtained by differentiating (4) and (1).

One approach for solving constrained dynamic optimization problem is based on the use of calculus of variations which is called “adjoint” method for sensitivity computation [14, 15, 25–27]. Differential equation number to be solved only depends on neuron number, does not depend on TDDNN parameters. A new dynamical system defined with adjoint state variables \( \lambda_i \) is obtained as

\[
-\dot{\lambda}_i = -\frac{\lambda_i}{T_i} + \frac{1}{T_i} \sum_j w_{ij} x'_j (t - \tau_j) \lambda_j - \frac{1}{T_i} \sum_j w_{ij} y'_j \lambda_j + e_i(t) \sum_j q_{ij} y'_j, \\
\lambda_i(t_f) = 0
\]  

(6)

The size of adjoint vector is \( n \) and is independent of network parameters. There are \( n \) quadratures for computing the sensitivities. The integration of the differential equations must be performed backwards in time, from \( t_f \) to 0. Let \( p \) be a vector containing all network parameters. Then, the cost gradients with respect to model parameters are given by the following quadratures:

\[
\frac{\partial J}{\partial p} = \int_{t_f}^{t_0} \left( \frac{\partial f}{\partial p} \right)^T \lambda \, dt
\]  

(8)

The cost gradients as in [14–16] can be easily computed. We assume that at each iterations gradients of the performance index with respect to all TDDNN parameters, \( g = \partial J / \partial p \), is computed. Here an algorithm was described for updating parameter values based on this gradient information.

### 4.1. Levenberg–Marquardt algorithm

Levenberg–Marquardt (LM) algorithm has been used extensively for parameter estimation in linear continuous-time systems (a nonlinear parameter estimation problem).

The parameter update is computed as

\[
p^{k+1} = p^k + \Delta p^k, \quad \Delta p^k = -[\hat{H} + \mu I]^{-1} \nabla_p J
\]  

(9)

\( I \) is the \( m \times m \) identity matrix, and \( \hat{H} \) is an approximation to the Hessian matrix of the response (the matrix of second derivatives of \( z \) with respect to \( \bar{p} \)). The exact Hessian is
computationally prohibitive. The Levenberg–Marquardt scheme approximates it as

$$\frac{\partial^2 z}{\partial p_i \partial p_j} \approx \left[ \frac{\partial z}{\partial p_i} \right]^T \left[ \frac{\partial z}{\partial p_j} \right]$$  \hspace{1cm} (10)$$

$\mu$ in (9) is a parameter that controls the parameter step size and the extent to which second-order information influences the update. For large $\mu$, (9) essentially implements a gradient-descent rule. $\mu$ is updated automatically by the algorithm based on the error history. Thus, the Levenberg–Marquardt algorithm implements a continuously varying hybrid of gradient descent and an approximate second order optimization.

The adjoint way of computing performance index sensitivities is efficient in the number of differential equations that need to be solved, but the intermediate computations within the time interval do not produce information that is meaningful in the original TDDNN. Whereas the forward sensitivity method produces trajectories of state and response sensitivities, the adjoint method produces trajectories of adjoint variables.

For algorithms requiring the exact Hessian, a computationally efficient approach is available using both the adjoint and forward response sensitivities [27]. Thus, by performing both the forward and adjoint sensitivity analysis, an exact Newton method in function space can be implemented at substantially lower cost than that involved in the “forward” computation of exact second order sensitivities.

5. Illustrative prediction examples of chaotic dynamical systems with TDDNN

In this section, chaotic time series system has been used for testing of TDDNN prediction performance. There are many studies for time series modeling, forecasting and prediction [28–31]. For this aim in this study, the MacKey–Glass chaotic time series is produced by the following differential equation [28]:

$$\frac{dx}{dt} = \frac{\alpha x(t - \tau)}{1 + x^{10}(t - \tau)} + \beta x(t)$$  \hspace{1cm} (11)$$

where $\beta$ is a different parameter found in TDDNNs. $x(t)$ is chaotic and semi-periodical for $\alpha = 0.2$, $\beta = -0.1$, $\tau = 17$. The fifth-order Runge–Kutta [23,24] method was used to obtain simulation data with the initial conditions $x(0) = 1.2$ and $x(t - \tau) = 0$ for $0 \leq t < \tau$. The simulation step size was taken as 1.

We have used one dynamic-neuron with delay. Time delay of TDDNN was taken the fixed value as the same as the MacKey–Glass, $\tau = 17$. The other weights are adjusted as presented previously. The first 100 data points were used to train the TDDNNs. The prediction performance of the TDDNN was tested after 200th data points.

Fig. 5 shows the result of the test with 200 training points. The prediction capability is excellent. The neural network prediction error is shown in Fig. 6.

6. Conclusions

The TDDNN presented in this paper was successfully applied to time series prediction and modeling with delay-system. Simulations show that the TDDNN structure can grow more accurate neural-approximators. The significance of this work is that efficient computational algorithms have been developed for parameter identification in fully nonlinear dynamical systems with delay and for training. The gradients were computed by adjoint sensitivity analysis methods. Also, time series prediction with TDDNN has developed by adjoint sensitivity approach.

The other class of time-delay approach for neural networks can be investigated to improve the prediction capability.

References

[1] J.J. Hopfield, D.W. Tank, Neural architecture and biophysics for sequence recognition, in: J.H. Byrne, W.O. Berry (Eds.), Neural Architecture and


