TECHNICAL RESEARCH REPORT

A Learning Algorithm for Adaptive Time-Delays in a Temporal Neural Network

by D-T. Lin, J.E. Dayhoff, and P.A. Ligomenides

T.R. 92-59

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

A Learning Algorithm for Adaptive Time-Delays in a Temporal Neural Network[1,2]

Daw-Tung Lin[†], Judith E. Dayhoff[‡] and Panos A. Ligomenides[†]

Systems Research Center[‡], and Department of Electrical Engineering[†]

University of Maryland

College Park, MD 20742.

[2]This work was supported in part by the Systems Research Center at the University of Maryland, under Grant CDR-88-03012 from the NSF.

A Learning Algorithm for Adaptive Time-Delays in a Temporal Neural Network

Daw-Tung Lin[†], Judith E. Dayhoff[‡] and Panos A. Ligomenides[†]

Systems Research Center[‡], and Department of Electrical Engineering[†]

University of Maryland

College Park, MD 20742.

Abstract

The time delay neural network (*TDNN*) is an effective tool for speech recognition and spatiotemporal classification. This network learns by example, adapts its weights according to gradient descent, and incorporates a time delay on each interconnection. In the *TDNN*, time delays are fixed throughout training, and strong weights evolve for interconnections whose delay values are important to the pattern classification task. Here we present an adaptive time delay neural network (*ATNN*) that adapts its time delay values during training, to better accommodate to the pattern classification task. Connection strengths are adapted as well in the *ATNN*. We demonstrate the effectiveness of the *TDNN* on chaotic series prediction.

1 Introduction

The time-delay neural network (*TDNN*) proposed by Waibel et al [17] employs time-delays on connections and has been successfully applied to phoneme recognition [18, 19]. The time-delay neural network also classifies spatiotemporal patterns and provides robustness to noise and graceful degradation [10]. In the *TDNN* architecture, each neuron takes into account not only the current information from all the neurons of the previous layer, but also a certain amount of past information from those neurons due to delays on interconnections. Typically the time delays are evenly spaced over a time interval called the frame window, although arbitrary time delays may be used. Training is on spatiotemporal patterns, and inputs are classified at each time step by the output layer. After training, weights are strengthened along interconnections whose time delays are important to recognition.

A limitation of the *TDNN* as originally posed [17] is its inability to learn or adapt the values of the time delays. Time delays are fixed initially and remained the same throughout training. In this paper we present the adaptive time delay neural network (*ATNN*), which adapts time delays as well as weights during training. The result is a dynamic learning technique for spatiotemporal classification.

Biological studies have shown that time delays do occur along axons due to different conduction times and differing lengths of axonal fibers, and that, in addition, temporal properties such as temporal decay and integration occur at synapses. Thus biological systems inspire the *TDNN* and *ATNN* architectures.

Previous work has lacked adaptive capabilities for time delays in neural networks. Studies on the temporal behavior of neural network models have examined signal transmission

along neurons [13, 14] and identification of temporal events [1, 4, 8, 15, 16, 20]. Work by Bodenhausen [2, 3] utilizes time-delay adaptation, but applies only to a particular recirculation network. Related techniques such as backpropagation through time [21] have applied to temporal pattern recognition but not with adaptive time delays.

In Section 2 of this paper we demonstrate the effects of time delays on system performance through modeling chaotic series prediction with a *TDNN*. We describe the ATNN architecture in Section 3 and derive its learning rule in Section 4.

2 Effects of Time-Delays on Chaotic Series Prediction

The prediction of future variations from past and current system measurements is an essential task in dynamic systems analysis; this task can be addressed by a *TDNN*. To show the effects of changing time-delays on system performance, we have applied a time-delay neural network with fixed time-delays to chaotic time series prediction of the Mackey-Glass delay differential equation. The delay differential equation of the form

$$\frac{dx(t)}{dt} = F(x(t), x(t-\tau)) \tag{1}$$

describes systems in which a stimulus has a delayed response [7]. This equation fits practical examples from physiological control systems, economics, and other complex systems. In this study, we used the example proposed by Mackey and Glass which describes the production of blood [11] as the following

$$\dot{x} = \frac{ax(t)}{1 + x^c(t-\tau)} - bx \tag{2}$$

This differential equation possesses many dynamic properties such as nonlinearity, limit cycle oscillations, aperiodic wave forms and other dynamic behaviors [6] and it can be used as a

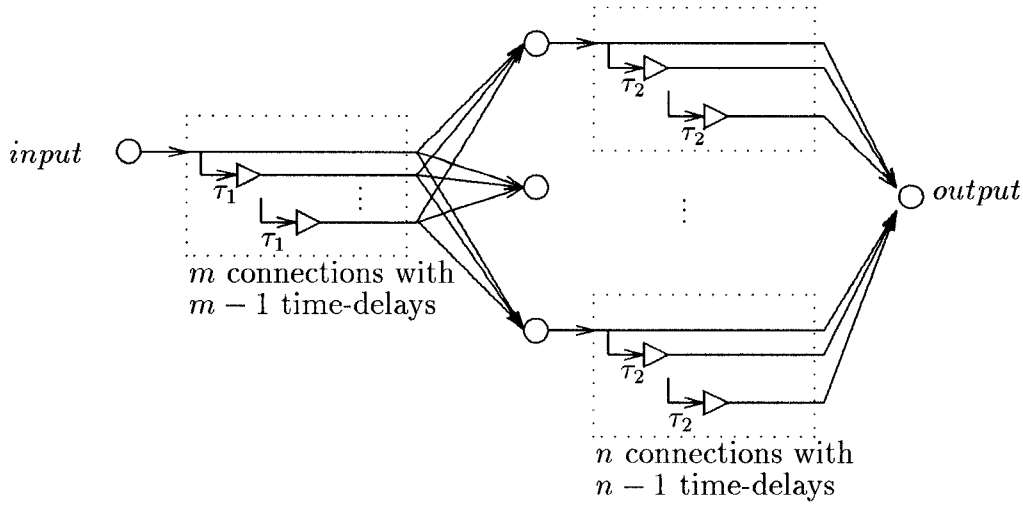

Figure 1: *TDNN* configuration for prediction

benchmark for temporal learning.

With the Mackey-Glass differential equation, the state of the system is determined by the function $f(.)$ on the interval $[t-\tau, t]$. This function can be approximated by N samples taken at $\delta t = \frac{\tau}{(N-1)}$. These N samples can be considered as N variables from an N dimensional vector space

$$(x_1, x_2, ..., x_N) = (x(t-(N-1)\delta t), ..., x(t-2\delta t), x(t-\delta t), x(t))$$

Conventionally, the prediction is accomplished by mapping this N dimensional vector space to a real value. The mapping $f : R^n \longrightarrow R$ maps the n most recent samples of the time series to the value at a future moment $x(t+T)$. Thus, the prediction task is accomplished by finding a function such that $x(t+T) = f(x_1, x_2, ..., x_n)$. The function $f(.)$ can be approximated by many approaches, e.g., Euler integration, Runge-Kutta algorithm, or other methods.

The *TDNN* accomplishes this prediction in a novel manner. The schematic architecture of the *TDNN* used in this study is illustrated in Figure 1. Output from each neuron in the

network is given by a single line, which is then subject to delays of various lengths. These various delayed signals are then connected as inputs to the neurons of the second layer and again, each unit in the second layer is connected in the same manner to the next layer. The synapse weights are updated by an error backpropagation algorithm until the *RMSE* goes below a threshold criteria [5]. The detailed description of the activation rule and learning rule for the *TDNN* can be found in [10].

In our simulation, we used one input unit, three hidden units, and one output unit. Experiments were conducted by changing the number of delays on each layer (m and n in Figure 1) and the values of the delays (τ_1 and τ_2, indicated in Figure 1). For simplicity, we kept all delays in the same layer consistent (e.g., each neuron's output signals were subject to delays of $0, \tau_i, 2\tau_i, ..., (n-1)\tau_i$). The error measure *NMSE* (normalized mean square error) is defined as the following:

$$NMSE = \frac{E[(x(t) - x(\hat{t}))^2]}{E[x^2(t)]}$$

where $x(t)$ is the original signal value and $\hat{x}(t)$ is the network prediction value.

In the experiment, the observation of prediction performance and convergence speed is made with different values of time-delay τ_1 under three different network configurations (different values of m and n); different performance levels are observed for different configurations. Two of the simulation results are shown in Figures 2 and 3, which give performance for differing values of τ_1.

The *TDNN* performance was compared to the original function. The network output of two different trials was plotted to overlap with the system output in Figure 2 and 3, where solid lines and dashed lines stand for network output and the original function value

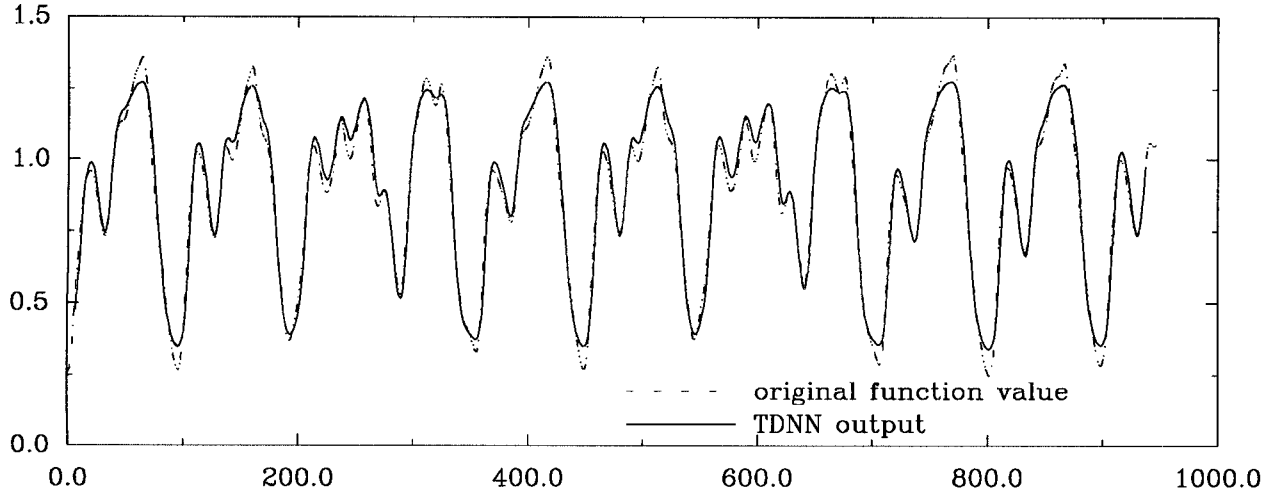

Figure 2: Simulation results,$\tau_1 = 1$,$\tau_2 = 1$,$m = 4$,$n = 8$

respectively. In Figure 2, the network predictions match the function value relatively well, following the original function waveform closely except for some bumping peaks. In Figure 3, which utilizes different time delay values, the match is not very good although the overall trends are preserved.

3 Time-Delay Learning Adaptation

The schematic architecture of the connections from one neuron to another neuron of the *ATNN* is illustrated in Figure 4. Node i of layer $h-1$ is connected to node j of the next layer h, with the connection having time-delay $\tau_{jik,h-1}$ and synaptic weight $w_{jik,h-1}$. The multilayered network is not necessarily fully connected layer by layer, but may be sparsely connected, and connections can skip layers. It is not necessary to have the same number of delays from different units in the same layer or the same delay values from different units, since the computation is local for each interconnection. Each connection can have an

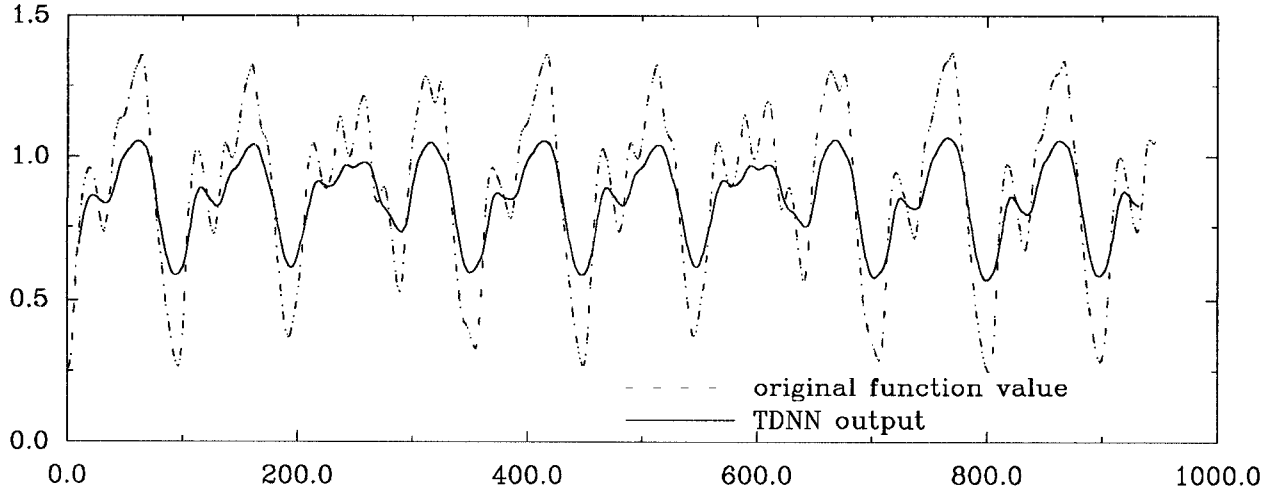

Figure 3: Simulation results,$\tau_1 = 4$,$\tau_2 = 1$,$m = 4$,$n = 8$

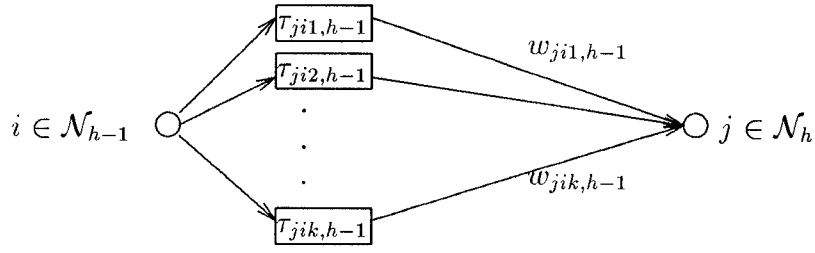

Figure 4: Basic time-delay connections between two neurons (node i of layer $h-1$ and node j of layer h) in *ATNN*

arbitrary delay value. For the sake of simplicity, we assume the network is layered and fully connected layer by layer.

We assume in this discussion that network parameters are updated in a discrete manner and the input signals are differentiable and sampled at a Nyquist rate, i.e. the sampling timestep $r = \frac{1}{2f_{max}}$, where f_{max} is the maximum frequency of all input channels. We define the following notation:

L $\equiv$ the number of layers in the network.

$\mathcal{N}_h$ $\equiv$ the set of nodes $\{1, 2, ..., |\mathcal{N}_h|\}$ of layer h.

$\tau_{jik,h-1}$ $\equiv$ the time-delay of the kth connection to node j of layer h from node i of layer $h-1$

$K_{ji,h-1}$ $\equiv$ the total number of connections to node j (layer h) from node i of layer $h-1$

$\mathcal{T}_{ji,h-1}$ $\equiv$ the set of delays on connections to node j (layer h) from node i of layer $h-1$, i.e. $\mathcal{T}_{ji,h-1} = \{\tau_{ji1,h-1}, \tau_{ji2,h-1}, ..., \tau_{jim,h-1}\}$, where $m = K_{ji,h-1}$

$a^{\mu}_{i,0}(t)$ $\equiv$ the ith channel of the input training pattern μ at time t

t_n $\equiv$ the nth sampling time, where r is a single time step (e.g., $t_n = nr$)

$w_{jik,h-1}$ $\equiv$ the synapse weight of the kth connection to node j from node i of layer $h-1$, and $k = 1, 2, ..., K_{ji,h-1}$

The activation value of node j of layer h when input pattern μ is present at time t_n is

defined as:

$$a_{j,h}^{\mu}(t_n) = \begin{cases} f_j(S_{j,h}^{\mu}(t_n)) & \text{if } h \geq 2 \\ a_{j,0}^{\mu}(t_n) & \text{if } h = 1 \end{cases}$$

where

$$S_{j,h}^{\mu}(t_n) = \sum_{i \in \mathcal{N}_{h-1}} \sum_{k=1}^{K_{ji,h-1}} w_{jik,h-1} \cdot a_{i,h-1}^{\mu}(t_n - \tau_{jik,h-1}) \tag{3}$$

$$f_j(x) = \frac{\beta_j}{1 + e^{-\alpha_j x}} - \gamma_j \tag{4}$$

where α_j, β_j and γ_j are real numbers which define the symmetry point $(0, \frac{\beta_j}{2} - \gamma_j)$ of $f_j(x)$ if x is symmetric to 0, the range $[-\gamma_j, \beta_j - \gamma_j]$ of $f_j(x)$ and the steepness of $f_j(x)$ (e.g. $f_j'(0) = \frac{\alpha_j \beta_j}{4}$). For simplicity, we use the same sigmoid function for each node in this paper, although in practice f_j may differ for each node.

The adaptation of the delays and weights are derived based on the gradient descent method to minimize the error measure E during training. The training set consists of a set of spatiotemporal patterns and target outputs over time. Thus, for spatiotemporal inputs, training may proceed in any of three ways: *batch mode*, *pattern mode*, and *incremental mode*. Theoretically, the true gradient descent should be based on the total error measure over all spatiotemporal training patterns over all times when patterns are presented. This is called *batch mode* learning: the parameters are updated after all patterns have been presented. It requires additional local storage for each connection. Instead of measuring all patterns, *pattern mode* takes into account a single complete spatiotemporal pattern and the network parameters are updated after each spatiotemporal pattern is presented.

The third alternative is *incremental mode*. This incremental approach updates all parameters after every time step of each spatiotemporal pattern is considered. For a static back-error propagation network [5], pattern mode learning is the same as incremental mode

learning, and batch mode has been shown equivalent to pattern mode for small learning rates [12]. For any of these training modes, learning will become inefficient and parameters may overshoot if the learning rate is too large. The relative effectiveness of these differing approaches depends on the problem, but the latter one is likely to be superior for very regular or redundant training sets [9].

We will follow the incremental updating method (on-line learning) in this discussion. An instantaneous error measure is defined as MSE:

$$E(t_n) = \frac{1}{2} \sum_{j \in \mathcal{N}_L} (d_j(t_n) - a_j(t_n))^2 \tag{5}$$

where L denotes the output layer and $d_j(t_n)$ indicates the desired (target) value of output node j at time t_n.

The time-delay is modified step by step proportional to the opposite direction of the error gradient with respect to this delay. The updating rule is therefore:

$$\Delta \tau_{jik,h-1} \equiv -\eta_1 \frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} \tag{6}$$

where η_1 is the learning rate.

4 Learning Algorithm Derivation

By the chain rule

$$\frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} = \frac{\partial E(t_n)}{\partial S_{j,h}} \frac{\partial S_{j,h}(t_n)}{\partial \tau_{jik,h-1}} \tag{7}$$

The second factor of Equation (7) can be expressed as

$$\begin{aligned} \frac{\partial S_{j,h}(t_n)}{\partial \tau_{jik,h-1}} &= \frac{\partial}{\partial \tau_{jik,h-1}} \sum_{p \in \mathcal{N}_{h-1}} \sum_{q=1}^{K_{ji,h-1}} w_{jpq,h-1} a_{p,h-1}(t_n - \tau_{jpq,h-1}) \\ &= -w_{jik,h-1} a'_{j,h-1}(t_n - \tau_{jik,h-1}) \end{aligned} \tag{8}$$

We define

$$\rho_{j,h}(t_n) = \frac{\partial E(t_n)}{\partial S_{j,h}} \tag{9}$$

Substitute Equation (8) and (9) into Equation (7), we obtain

$$\frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} = -\rho_{j,h}(t_n) w_{jik,h-1} a'_{j,h-1}(t_n - \tau_{jik,h-1}) \tag{10}$$

$$\stackrel{(6)}{\Longrightarrow} \quad \Delta\tau_{jik,h-1} = \eta_1 \rho_{j,h}(t_n) w_{jik,h-1} a'_{j,h-1}(t_n - \tau_{jik,h-1}) \tag{11}$$

To derive $\rho_{j,h}(t_n)$, we need to apply chain rule and consider two cases:

$$\begin{aligned} \rho_{j,h}(t_n) &= \frac{\partial E(t_n)}{\partial S_{j,h}} \\ &= \frac{\partial E(t_n)}{\partial a_{j,h}} \frac{\partial a_{j,h}(t_n)}{\partial S_{j,h}} \\ &= \frac{\partial E(t_n)}{\partial a_{j,h}} f'(S_{j,h}(t_n)) \end{aligned} \tag{12}$$

To find $\frac{\partial E(t_n)}{\partial a_{j,h}}$, we consider the following two cases:

1. If j is an output unit:

$$\frac{\partial E(t_n)}{\partial a_{j,h}} = -(d_j(t_n) - a_{j,h}(t_n)) \tag{13}$$

$$\stackrel{(12)}{\Longrightarrow} \quad \rho_{j,h}(t_n) = -(d_j(t_n) - a_{j,h}(t_n)) f'(S_{j,h}(t_n)) \tag{14}$$

2. If j is a hidden unit:

$$\begin{aligned} \frac{\partial E(t_n)}{\partial a_{j,h}} &= \sum_{p \in \mathcal{N}_{h+1}} \frac{\partial E(t_n)}{\partial S_{p,h+1}} \frac{\partial S_{p,h+1}(t_n)}{\partial a_{j,h}} \\ &= \sum_{p \in \mathcal{N}_{h+1}} \frac{\partial E(t_n)}{\partial S_{p,h+1}} \frac{\partial}{\partial a_{j,h}} \Big(\sum_{i \in \mathcal{N}_h} \sum_{q=1}^{K_{pi,h}} w_{piq,h} a_{i,h}(t_n - \tau_{piq}) \Big) \\ &= -\sum_{p \in \mathcal{N}_{h+1}} \rho_{p,h+1}(t_n) \Big(\sum_{q=1}^{K_{pi,h}} w_{pjq,h}(t_n) \Big) \end{aligned} \tag{15}$$

$$\stackrel{(12)}{\Longrightarrow} \quad \rho_{j,h}(t_n) = -\Big[\sum_{p \in \mathcal{N}_{h+1}} \sum_{q=1}^{K_{pi,h}} \rho_{p,h+1}(t_n) w_{pjq,h}(t_n)\Big] f'(S_{j,h}(t_n)) \tag{16}$$

We have now found ρ.

From Equation (11) it remains to find $a'_{j,h-1}(t_n - \tau_{jik,h-1})$. The value of $a'_{j,h-1}(t_n - \tau_{jik,h-1})$ can be approximated as following: From the elementary calculus we know that if the function $f(x)$ is differentiable, then

$$f(x_0 + h) - f(x_0) = h\frac{df}{dx} \mid_{x=x_*} \tag{17}$$

for some point x_* such that $x_0 \leq x_* \leq x_0 + h$. The derivative is evaluated at a point x_* (between x_0 and $x_0 + h$), and by the Mean Value theorem of differential calculus, x_* always exists. Accordingly, we get

$$a'_{j,h-1}(t_n - \tau_{jik,h-1}) \approx \begin{cases} \frac{a(t_n)-a(t_{n-1})}{r} & \text{if } \tau_{jik,h-1} = 0 \\ \frac{a(t_{k+1})-a(t_{k-1})}{2r} & \text{if } t_n - \tau_{jik,h-1} = t_k,\ \tau_{jik,h-1} \neq 0 \end{cases}$$

where $r = t_k - t_{k-1}, k \in \{0, 1, ..., n\}$.

Theoretically, $\tau_{jik,h-1}$ can be zero or any positive real number, and the value $a_{j,h-1}(t_n - \tau_{jik,h-1})$ can be easily again approximated by the numerical approximation method. But, in this way, the network will be trained by a simulated pattern or noisy signal due to the approximation. Therefore, more consideration of this effect is necessary and this issue will be addressed elsewhere. In this paper, we will consider all time-delays as integer times of timestep, i.e. $\tau_{jik,h-1} = nr, n \in \mathcal{N}$, and $\Delta\tau_{jik,h-1}$ should be rounded before updated or save the non-rounded value and use the rounded value for updating.

The time-delay learning rule is summarized as follows:

$$\Delta\tau_{jik,h-1} = \eta_1 \rho_{j,h}(t_n) w_{jik,h-1} a'_{j,h-1}(t_n - \tau_{jik,h-1}) \tag{18}$$

where

$$\rho_{j,h}(t_n) = \begin{cases} -(d_j(t_n) - a_{j,h}(t_n))f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ -[\sum_{p\in\mathcal{N}_{h+1}} \sum_{q\in\mathcal{T}_{pj,h}} \rho_{p,h+1}(t_n) w_{pjq,h}(t_n)] f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases}$$

Similarly, the learning rule for weights can also be obtained in the same manner and is summarized as following:

$$\Delta w_{jik,h-1} = \eta \delta_{j,h}(t_n) a_{j,h-1}(t_n - \tau_{jik,h-1}) \tag{19}$$

where

$$\delta_{j,h}(t_n) = \begin{cases} (d_j(t_n) - a_{j,h}(t_n))f'(S_{j,h}(t_n)) & \text{,if } j \text{ is an output unit} \\ (\sum_{p\in\mathcal{N}_{h+1}} \sum_{q\in\mathcal{T}_{pj,h}} \delta_{p,h+1}(t_n) w_{pjq,h}(t_n)) f'(S_{j,h}(t_n)) & \text{,if } j \text{ is a hidden unit} \end{cases}$$

5 Conclusion

We have introduced an extended algorithm of time-delay adaptation which can be viewed as a generalization of the *TDNN* and the error back-propagation learning paradigm. Unlike previous studies of *TDNN*s whose time-delay values have to be predetermined and remain fixed throughout training, in our approach the time-delays as well as the weights are adjusted with a learning procedure to achieve the desired system criterion. This work provides more dynamics and flexibility for the network itself to approach an efficient performance level and to optimize its configuration. Our results show that the time-delays and connection strengths in an *ATNN* can be optimized automatically. The learning paradigm proposed here is expected to improve performance of convergence speed and generalization significantly compared to the utilization of fixed delays in the conventional *TDNN*.

References

[1] A. D. Back and A. C. Tsoi. FIR and IIR synapse, a new neural architecture for time series modeling. *Neural Computation*, 3:375–385, 1991.

[2] U. Bodenhausen. Learning internal representations of pattern sequences in a neural network with adaptive time-delays. In *International Joint Conference on Neural Networks*, volume 3, pages 113–118, San Diego, 1990. IEEE, New York.

[3] U. Bodenhausen. The tempo-algorithm: Learning in a neural network with variable time-delays. In *International Joint Conference on Neural Networks*, volume 1, pages 597–600, Washington, D.C., 1990. IEEE, New York.

[4] N. E. Cotter and O. N. Mian. A pulsed neural network capable of universal approximation. *IEEE Trans. on Neural Networks*, 3:308–314, March 1992.

[5] J. E. Dayhoff. *Neural Network Architectures : An Introduction.* Van Nostrand Reinhold, New York, 1990.

[6] D. Farmer and J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59:845–848, 1987.

[7] J.D. Farmer. Chaotic attractors of an infinite-dimentional dynamical system. *Physica D*, 4:366–393, 1982.

[8] E. Gelenbe. Temporal behavior of neural networks. In M. Caudill and C. Butler, editors, *IEEE International Conference on Neural Networks*, volume 3, pages 267–276, San Diego, 1987. IEEE, New York.

[9] J. Hertz, A. Krough, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, Reading, 1991.

[10] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Trajectory recognition with a time-delay neural network. Accepted by IJCNN'92 Baltimore, 1992.

[11] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287, 1977.

[12] S.-Z. Qin, Su H.-T., and J. McAvoy. Comparison of four neural net learning methods for dynamic system identification. *IEEE Trans. on Neural Networks*, 3(1):122–130, January 1992.

[13] M. Reiss and J. G. Taylor. Storing temporal sequences. *Neural Networks*, 4:773–787, 1991.

[14] D. C. Tam. Temporal-spatial coding transformation: Conversion of frequency-code to place-code via a time-delayed neural network. In *International Joint Conference on Neural Networks*, volume 1, pages 130–133, Washington, 1990. Lawrence Erlbaum, Hillsdale.

[15] D. C. Tam and D. H. Perkel. A model for temporal correlation of biological neuronal spite trains. In *International Joint Conference on Neural Networks*, volume 1, pages 781–785, Washington, 1989. IEEE, New York.

[16] A. Waibel. Consonant recognition by modular construction of large phonemic time-delay neural networks. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 215–223, Denver 1988, 1989. Morgan Kaufmann, San Mateo.

[17] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition: Neural networks versus hidden markov models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 107–110, April 1988.

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoust., Speech, Signal Processing*, 37:328–339, 1989.

[19] A. Waibel, K. J. Lang, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[20] E. A. Wan. Temporal backpropagation for FIR neural networks. In *International Joint Conference on Neural Networks*, volume 1, pages 575–580, San Diego, 1990. IEEE, New York.

[21] P. J. Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560, Octtober 1990.