

NONLINEAR ADAPTIVE FILTERING WITH FIR SYNAPSES AND ADAPTIVE ACTIVATION FUNCTIONS

A. Neil Birkett and Rafik A. Goubran

Department of Systems and Computer Engineering
 Carleton University, 1125 Colonel By Drive
 Ottawa, Canada, K1S 5B6
 birkett@sce.carleton.ca goubran@sce.carleton.ca

ABSTRACT

This paper focuses on multilayer perceptron neural networks where the activation functions are adaptive and where each neuron synapse is modelled by a finite impulse response (FIR) filter. A simplified architecture consisting of a variable activation (VA) function which is sandwiched between two FIR synapses is studied. The VA function consists of a mixed linear-tanh sigmoid with a parameter which controls the linear region. The VA parameters and FIR synaptic weights are updated using a modified form of the instantaneous-cost (IC) temporal backpropagation algorithm [1]. Simulations for identifying cascaded nonlinear transfer functions with internal memory and arbitrary activation functions illustrate the improved modelling performance over models with non-adaptive activation functions.

1. INTRODUCTION

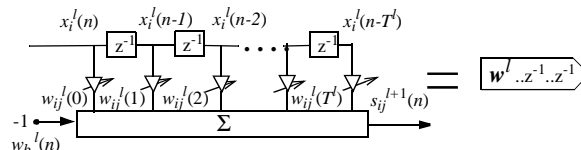
Multilayer perceptrons (MLPs) with synapses described by filters have been recently proposed as alternative architectures for modelling nonlinear time-dependent signals [2]. The approach is to replace the traditional synaptic weights with finite impulse response (FIR) or infinite impulse response (IIR) filters [3] to capture the underlying dynamics of time-dependent input information. However, these techniques are based on the use of a fixed activation function and may not be optimum architectures for the identification of networks that contain arbitrary activation functions.

In this paper we propose an adaptive FIR MLP where an arbitrary nonlinearity is sandwiched between the FIR synapses as shown in Figure 1. This architecture is similar to the linear-nonlinear cascaded Volterra system described in [4], however, in this case the fixed nonlinear Volterra kernels are replaced by an adaptive activation function with a variable linear region. This architecture offers an additional degree of freedom, however, derivation of the update parameter for the activation function is more complicated than in a conventional MLP since the error signal must be propagated backwards through the FIR sections.

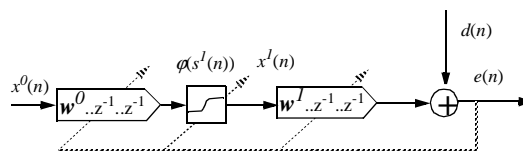
Teshnehlab and Watanabe [5] recently proposed a flexible sigmoid function for application in a conventional MLP but their network did not have temporal FIR synapses as described here.

The idea of using an adaptive activation function was also proposed in [6], however, the activation function was placed at the output only, and was trained with all other weights being held constant. In this paper, we introduce an architecture where both the FIR synapse weights and the activation function can be placed arbitrarily and adapted on-line.

An algorithm for training FIR synapses was first published by Wan [2] and later extended by Back and Tsoi [1] into four distinct variants based on using either an instantaneous cost or total cost function. Since we are primarily concerned with on-line training applications, the algorithm presented below is based on the instantaneous cost function, with an accumulated gradient vector of length n_w . If $n_w=1$, the algorithm is similar to the instantaneous cost instantaneous gradient (IC1) algorithm described in [1] whereas if $n_w=T$, where T is the length of the FIR synapse, it is similar to the instantaneous cost accumulated gradient (IC2) algorithm. The choice of n_w however is not restricted to 1 or T , and may assume any value in between.



(a) FIR synapse model



(b) Proposed structure using adaptive activation function

FIGURE 1. Synapse model MLP. (a) A single weight is replaced by an FIR synapse. (b) Proposed architecture consists of a variable activation function sandwiched between FIR synapses.

2. DESCRIPTION OF THE LEARNING ALGORITHM

A general FIR MLP is defined by;

$$\mathbf{w}_{ij}^l = [w_{ij}^l(0), w_{ij}^l(1), \dots, w_{ij}^l(T^l)]^T \quad (1)$$

where w_{ij}^l is the weight connecting the output of neuron i to the input of neuron j in the l -th layer. Similarly,

$$\mathbf{x}_i^l = [x_i^l(0), x_i^l(1), \dots, x_i^l(T^l)]^T \quad (2)$$

$$s_{ij}^{l+1}(n) = \mathbf{w}_{ij}^{l+1} \bullet \mathbf{x}_i^l(n) \quad (3)$$

$$s_j^l(n) = \sum_{i=1}^{N_l} s_{ij}^l(n) \quad (4)$$

$$x_j^l(n) = \varphi(s_j^l(n)) \quad (5)$$

where " \bullet " represents the vector dot product and n refers to the discrete time index.

Define the instantaneous cost function as the squared Euclidean distance between the network output and the desired output $d(n)$.

$$E = e^2(n) = \sum_{i=1}^{N_L} (d(n) - x^l(n))^2 \quad (6)$$

The weights are updated using stochastic gradient method which minimizes the cost function E ;

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \frac{\partial E}{\partial \mathbf{w}} \quad (7)$$

Details of the derivation of the basic algorithm are given in [2] however in this paper the weight update equations for the FIR tap weights have been modified slightly to allow for a selectable gradient accumulation of size n_w . The updates are given as;

$$\mathbf{w}_{ij}^l(n+1) = \mathbf{w}_{ij}^l(n) - \eta \delta_j^{l+1}(n) \mathbf{x}_i^l(n) \quad (8)$$

$$\delta_j^l(n) = \begin{cases} -2e(n) \varphi'_s(s_j^l(n), a_j^l(n)) & \text{for } \dots l = L \\ \varphi'_s(s_j^l(n), a_j^l(n)) \cdot \sum_{k=1}^{N_{l+1}} [\tilde{\Delta}_k^{l+1}(n) \bullet \mathbf{w}_{jk}^l(n)] & \text{for } \dots 1 \leq l \leq L-1 \end{cases} \quad (9)$$

where the quantities $\tilde{\Delta}_k^{l+1}(n)$ and \mathbf{w}_{jk}^l define the *accumulated* weight and delta gradient components of length n_w ;

$$\mathbf{w}_{ij}^l = [w_{ij}^l(0), w_{ij}^l(1), \dots, w_{ij}^l(n_w)]^T \quad (10)$$

$$\tilde{\Delta}_k^l(n) = [\delta_k^l(n), \delta_k^l(n-1), \dots, \delta_k^l(n-n_w)]^T \quad (11)$$

and $\varphi'_s(s_j^l(n), a_j^l(n))$ represents the derivative of the activation function with respect to the input s .

3. ADAPTIVE ACTIVATION FUNCTION

The activation function used here is defined by

$$\varphi(s, a) = \begin{cases} s & \text{for } |s| \leq a \\ \text{sign}(s) \left[(1-a) \cdot \tanh\left(\frac{|s|-a}{1-a}\right) + a \right] & \text{for } |s| > a \end{cases} \quad (12)$$

where s is the input and a defines the linear region which is adaptive. The adaptation of a is done according to the stochastic gradient update;

$$a(n+1) = a(n) + \mu \left(\frac{-\partial E}{\partial a} \right) \quad (13)$$

$$= a(n) - \mu \xi_j^l(n)$$

$$\xi_j^l(n) =$$

$$\begin{cases} -2e(n) \varphi'_p(s_j^l(n), a_j^l(n)) & \text{for } \dots l = L \\ \varphi'_p(s_j^l(n), a_j^l(n)) \sum_{k=1}^{N_{l+1}} [\tilde{\Delta}_k^{l+1}(n) \bullet \mathbf{w}_{jk}^l(n)] & \text{for } \dots 1 \leq l \leq L-1 \end{cases} \quad (14)$$

Essentially, the derivative of the activation function is computed with respect to a and then it is multiplied by the filtered *delta* vector, which is the quantity in square brackets. It should also be noted that the value of a is clamped between 0 and 1.

Figure 2 shows the activation function of equation (12) with values of a equal to 0, 0.5 and 0.9, along with the associated $\frac{\partial \varphi(s_j^l(n))}{\partial a}$ and $\varphi'(s_j^l(n))$ values. There is no restriction on the type of nonlinearity used. For example, we may define an alternate hyperbolic tangent function such as;

$$\varphi(s, a) = \frac{1}{a} \tanh(sa) \quad (15)$$

Yamada *et al.* [7] have used a similar unit function to construct a direct neural network controller for robot manipulators, but with a different definition from that described above and without giving the learning algorithm of the parameter a .

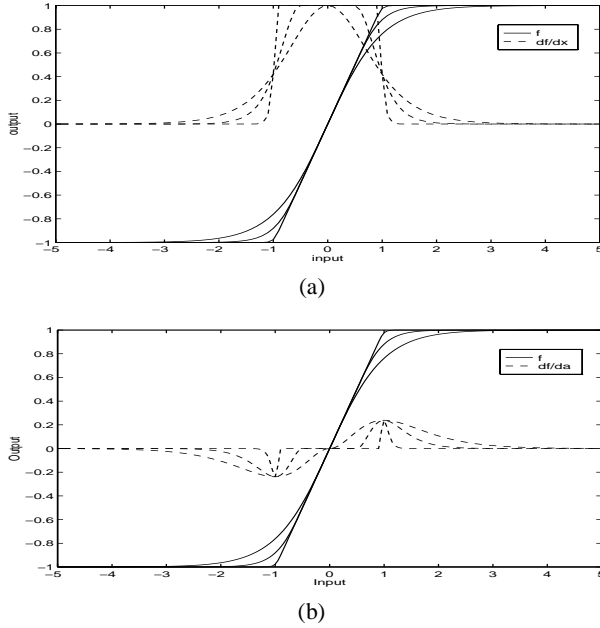


FIGURE 2. Adaptive activation function and derivatives. (a) With respect to the input value s . (b) With respect to the activation parameter a .

4. SIMULATION

In this section we apply the proposed structure to the identification of a nonlinear system comprised of a structure as shown in Figure 3. The plant activation function parameter a was set to

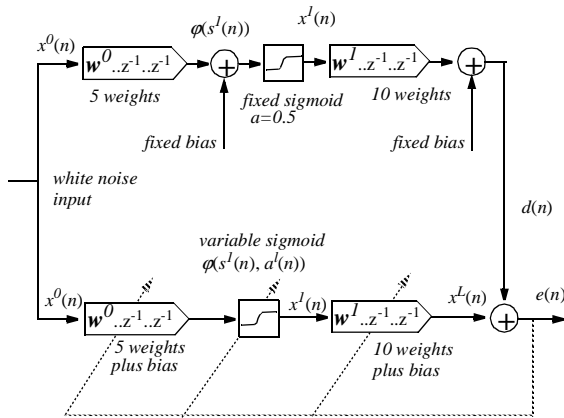


FIGURE 3. System identification using the proposed model.

0.5, and the number of taps in the 2nd FIR section is 10. Both of the FIR sections have the weights and biases randomly assigned.

Three structures were tested. The first structure (called ‘FIR’) is a conventional FIR structure consisting of 15 taps with the inclusion of a bias weight to compensate for output bias. It is trained with the Normalized Least Mean Square (NLMS) algorithm using a normalized step size α equal to 0.5.

The second structure tested (called ‘IC’) consists of two FIR sections with a *fixed* sigmoidal activation function between them, essentially equation (12) with $a=0$. The number of taps in the FIR sections is set to 5 and 10 respectively, and the gradient accumulation n_w is set to 1. This is equivalent to the instantaneous-cost instantaneous-gradient algorithm first published in [1]. However, we have made one more minor modification in that the fixed step sizes μ and ξ have been replaced by a normalized step size with $\alpha=0.5$, in much the same way as in the NLMS algorithm.

The third structure is the same as the IC structure except we allow adaptation of the parameter a according to the proposed training algorithm. We call this structure ‘VA’. The activation parameter a is initialized to 0 at the beginning of training.

The training sequence consists of 8000 randomly generated data points. For all the algorithms, the normalized mean square error (NMSE) is plotted according to the formula;

$$NMSE(n) = 10 \log \left(\frac{\sum_{r=0}^{500} [e_r(k)]^2}{\sum_{r=0}^{500} [d_r(k)]^2} \right) dB \quad (16)$$

where $e_r(k)$ and $d_r(k)$ represent the averaged error and desired signals and r represents the window values over which these averages are then smoothed, in this case equal to 500. The convergence results are shown in Figure 4.

Subsequent simulations also show that as the number of weights in the first FIR section increases, so does the training time required to accurately model the unknown plant for the ‘IC’ algorithm, but not so for the ‘VA’ algorithm. For example, a second simulation was performed using a plant with 50 taps in the first FIR and 10 in the second. 48,000 training data were used in this simulation and $n_w=1$ for both the IC and the VA architecture. Simulation results are shown in Figure 4.

5. DISCUSSION OF RESULTS

In Figure 4 the FIR structure trained with the NLMS algorithm is clearly unable to identify the unknown system accurately, and obtains an average NMSE of only -11 dB. The IC structure performs considerably better, however, since it has a fixed sigmoidal nonlinearity it has trouble dealing with the linear portion of the sigmoid in the unknown system and slows down after achieving approximately -13 dB NMSE in the first portion of

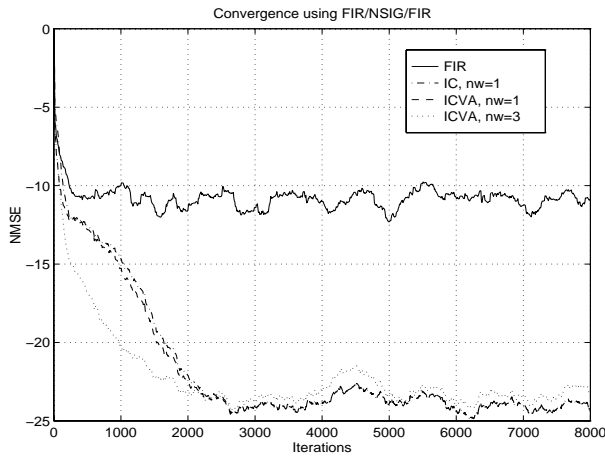


FIGURE 4. Comparison of convergence curves using the proposed architectures and algorithms.

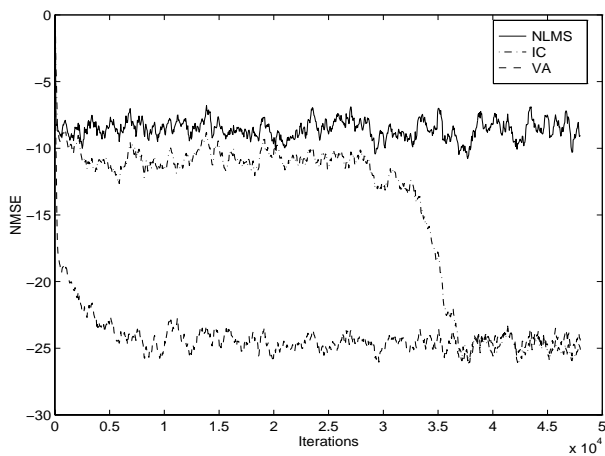


FIGURE 5. Convergence curves using 50 taps in the first FIR section.

training. Eventually, it converges to -20 dB NMSE after 1700 iterations. The proposed structure VA with $n_w=1$ is able to modify the activation function and achieve slightly better convergence as compared with the IC structure in the 0-2000 iteration range when using a gradient window of $n_w=1$. If n_w is increased to 3, the VA structure converges to -20 dB NMSE in 1000 iterations, and eventually converges to -25 dB.

The results in Figure 4 illustrate the convergence performance that is obtained by utilizing an adaptive activation function when the number of weights in the first section is increased to 50. The IC structure has some trouble in modelling the linear portion of the sigmoid in the unknown plant and achieves -12 dB NMSE in the first portion of training. Eventually it converges, but only after 35,000 iterations. The proposed structure VA is able to

modify the activation function immediately to obtain an NMSE of -25 dB in far fewer iterations.

6. CONCLUSIONS

This paper has presented a new architecture consisting of an FIR synaptic neural network with adaptive activation function. The simplified architecture studied consisted of a single adaptive activation function sandwiched between two FIR synapses. A temporal training algorithm is used to train the weights and adaptive sigmoids using a gradient accumulation method. Simulation results show that the gradient accumulation method offers some improvement in initial convergence rates, however the most striking improvements in convergence are obtained when the order of the first FIR section is large. In this case, the standard FIR synaptic MLP has a slower convergence than the variable activation architecture for the cases studied. The proposed variable activation FIR synaptic MLP can be considered as an interesting alternative architecture to conventional MLPs which utilize fixed sigmoidal activation functions only. Simulation results indicate that improved modelling performance can be obtained over models with non-adaptive activation functions.

7. REFERENCES

- [1] A. Back, E. Wan, S. Lawrence, A.C. Tsoi, "A Unifying view of some training algorithms for multilayer perceptrons with FIR filter synapses", *Neural Networks for Signal Processing IV*, Proceedings of the 1994 IEEE Workshop, New York, N.Y., pp.146-154.
- [2] E. Wan, "Temporal Backpropagation for FIR neural Networks", *Proc. IJCNN*, Vol. I, June 1990, pp. 575-580.
- [3] A. D. Back, A. C. Tsoi, "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling", *Neural Computation*, No. 3, 1991, pp. 375-385.
- [4] C. F. N. Cowan, P. F. Adams, "Nonlinear System Modeling: Concept and Application", *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1984, pp. 45.6.1-45.6.4.
- [5] M. Teshnehlab, K. Watanabe, "Neural network controller with flexible structure based on feedback error learning approach", *J. Intelligent and Robotic Sys.*, Vol 15, pp. 367-387, 1996.
- [6] J. Zhan, F. Li, "A new algorithm for realizing arbitrary nonlinear filters- adaptive neural filters", *Proceedings ICASSP*, 1994, Vol3, pp. 89-92.
- [7] T. Yamada, T. Yabuta, K. Takahashi, "Remarks on an adaptive type self tuning controller using neural networks", *Proceedings of IECON*, 1991, pp. 1389-1394

Acknowledgement

The authors wish to acknowledge the financial support of Nortel, TRIO, and NSERC.