Accepted to *Proceedings of the IEEE International Joint Conference on Neural Networks*
Anchorage, Alaska, USA, May 4–9, 1998

# Exact Hessian Calculation in Feedforward FIR Neural Networks

Tomasz J. Cholewo and Jacek M. Zurada

Department of Electrical Engineering, University of Louisville
Louisville, KY 40292, USA
e-mail: t.cholewo@ieee.org

## Abstract

*FIR neural networks are feedforward neural networks with regular scalar synapses replaced by linear finite impulse response filters. This paper introduces the Second Order Temporal Backpropagation algorithm which enables the exact calculation of the second order error derivatives for a FIR neural network. This method is based on the error gradient calculation method first proposed by Wan and referred to as Temporal Backpropagation. A reduced FIR synapse model obtained by ignoring unnecessary time lags is proposed to reduce the number of network parameters.*

## 1. Introduction

Dynamic neural networks are feedforward neural networks with commonly used scalar synapses replaced by linear filters. This provides them with the capability of performing dynamic mappings which depend on past input values, making them suitable for time series prediction, nonlinear system identification, and signal processing applications. Their most popular type are Finite Impulse Response (FIR) neural networks which are obtained by replacing synapses with finite impulse response filters. Due to their guaranteed stability characteristic and easy to minimize error surface they have been used with great success in many time series processing tasks. For example, the Santa-Fe Institute Time Series Prediction Competition [1] was won by a FIR neural network [2].

As of yet no algorithms for calculating the Hessian in FIR networks have been published. In this paper the Second Order Temporal Backpropagation algorithm which enables the exact calculation of second order error derivatives for a FIR neural network is introduced. The method is based on error gradient calculation method developed by Wan [2] referred to as Temporal Backpropagation.

A general network architecture and the notation used in the paper are introduced in Section 2. In Section 3 the Temporal Backpropagation algorithm is described as a precursor to development of the Hessian calculation. Section 4 presents the new Second Order Temporal Backpropagation algorithm. Conclusions and future work directions are discussed in Section 5.

## 2. Network Architecture

We use a general feedforward neural network (FNN) as our network architecture. Neurons in such a network form an oriented acyclic graph. The general network structure is static (memoryless) since there are no global recursive connections. As a result, neurons can be arranged so that each neuron's inputs originate only at the outputs of previous neurons. This generalizes a multilayer feedforward neural network (MFNN) which has neurons arranged in layers with connections only between neurons in consecutive layers [3].

Each neuron performs two calculations: first, it sums the output values $y_{ji}(t)$ of the incoming synapses to calculate the activation value ("net" value)

$$a_j(t) = \sum_{i \in \mathrm{IN}_j} y_{ji}(t). \tag{1}$$

of a neuron. $\mathrm{IN}_j$ denotes a set of neurons connected to the input of neuron $j$. Second, a nonlinear function $f_j(\cdot)$ is applied to $a_j(t)$ yielding an output value:

$$z_j(t) = f_j(a_j(t)).$$

Unit bias is realized by means of a "bias synapse" which supplies a constant value to the summation node of a neuron:

$$y_{ji}(t) = b_{ji}. \tag{2}$$

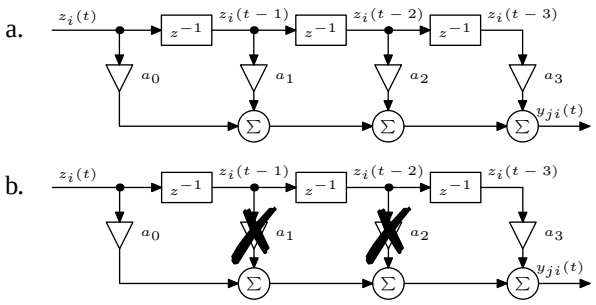


Figure 1. An example of a canonic realization of a third order FIR synapse: a. regular ($P_{ji} = \{a_{ji0}, \ldots, a_{ji3}\}$); b. reduced ($P_{ji} = \{a_{ji0}, a_{ji3}\}$).

The Finite Impulse Response (FIR) synapse is a FIR linear filter described by the linear difference equation

$$y_{ji}(t) = \sum_{l \in \mathrm{MA}_{ji}} a_{jil} z_i(t-l) \tag{3}$$

where $\mathrm{MA}_{ji}$ is a set of lags of moving average parameters $a_{jil}$. A well-known scalar synapse can be considered a special case of a FIR synapse with $\mathrm{MA}_{ji} = \{0\}$.

The original FIR synapse has weights for each time lag from 0 up to its order (Figure 1a). This leads to a large number of weights when many time lags are used to model the dependence of the current output on events from the past which is required in the case of low-pass frequency time series. Networks with too many weights typically display poor generalization capabilities unless additional complexity control techniques like early stopping [2] or pruning are used. We propose a modification of a FIR NN architecture in which weights are at desired time lags only (Figure 1b). In this manner only important time lags are left in the architecture while unnecessary time lags are omitted. Such a realization leads to methods for pruning unnecessary parameters in order to achieve better generalization performance.

Another NN structure commonly used for time series prediction is a multilayer perceptron with time lagged inputs. Its generalization produces a Time Delay Neural Network (TDNN) in which both inputs and hidden layer outputs are buffered several time steps and then input to following layers [4]. The FIR network is functionally equivalent to a TDNN but facilitates comparison to other dynamic networks and derivation of efficient training algorithms.

## 3. Temporal Backpropagation

Dynamic neural networks can approximate a wide class of nonlinear dynamic mappings. Adaptive numerical optimization (training) algorithms are used to find the appropriate values of network parameters by searching for a minimum on a multidimensional surface of some error function. The values of the derivatives of the error function with respect to each parameter (error gradient) are usually required in such approaches.

In the batch (also known as "off-line") approach the criterion to minimize is the error $E$ over all samples of a time series (total error):

$$E \triangleq \sum_t E(t)$$

where the instantaneous error $E(t)$ is usually defined as a sum of squares of all units' output errors:

$$E(t) \triangleq \frac{1}{2} \sum_i [e_i^o(t)]^2.$$

The unit's output error is defined as

$$e_i^o(t) \triangleq \begin{cases} d_i(t) - z_i(t) & \text{if } i \text{ is an output neuron,} \\ 0 & \text{otherwise.} \end{cases}$$

The proposed algorithm follows the Temporal Backpropagation method developed by Wan [2] for networks with FIR filter synapses. The derived formulas for the gradient of the total error are exact under the assumption that model parameters remain constant during the training epoch. This assumption is satisfied by the batch (off-line) training method which is appropriate for most time series prediction tasks but not for control problems or large data sets requiring online adaptation.

Wan [2] proposed a new way of applying the chain rule of derivation by considering derivatives of the total error with respect to units' activations at different moments of time. The derivative of the total error $E$ with respect to any parameter $p_{ji}$ from synapse connecting neuron $i$ to neuron $j$ can be obtained as:

$$\frac{\partial E}{\partial p_{ji}} = \sum_t \frac{\partial E}{\partial a_j(t)} \frac{\partial a_j(t)}{\partial p_{ji}} = \sum_t \delta_j(t) \frac{\partial y_{ji}(t)}{\partial p_{ji}}$$

where

$$\delta_j(t) \triangleq \frac{\partial E}{\partial a_j(t)} = \frac{\partial E}{\partial z_j(t)} \frac{\partial z_j(t)}{\partial a_j(t)} = e_j(t) f_j'(a_j(t)).$$

The quantity $f_j'(a_j(t))$ is a derivative of the activation function. The local error $e_i(t)$ is a generalization of the training error at the output neurons to all neurons. For output neurons this error is simply equal to the output error $e_i^o(t)$. For hidden neurons $e_i(t)$ is obtained using generalization of the error backpropagation formula introduced by Werbos [5]:

$$\begin{aligned} e_i(t) \triangleq \frac{\partial E}{\partial z_i(t)} &= \sum_{j \in \mathrm{OUT}_i} \sum_{t'} \frac{\partial E}{\partial a_j(t')} \frac{\partial a_j(t')}{\partial z_i(t)} \\ &= \sum_{j \in \mathrm{OUT}_i} \sum_{l \in \mathrm{MA}_{ji}} \delta_j(t+l) a_{jil} \end{aligned}$$

Summarizing:

$$e_i(t) = \begin{cases} e_i^o(t) & \text{if } i \text{ is an output,} \\ \sum_{j \in \text{OUT}_i} \sum_{l \in \text{MA}_{ji}} \delta_j(t+l) a_{jil} & \text{otherwise.} \end{cases} \tag{4}$$

Equation (4) can be interpreted as a back-filtering of error terms through dynamic synapses in the backward direction. The result is a noncausal dependence for $e_i(t)$ on terms $\delta_j(t')$ for $t' > t$. This problem can be resolved by delaying computation of $e_i(t)$ until all $\delta_j(t')$ are known. The quantity $\frac{\partial y_{ji}(t)}{\partial p_{ji}}$ depends on the synapse type. For a constant synapse from Equation (2) we obtain:

$$\frac{\partial y_{ji}(t)}{\partial b_{ji}} = \frac{\partial b_{ji}}{\partial b_{ji}} = 1 \tag{5}$$

and for a FIR synapse from Equation (3):

$$\frac{\partial y_{ji}(t)}{\partial a_{jil}} = \frac{\partial}{\partial a_{jil}} \sum_{l \in \text{MA}_{ji}} a_{jil} z_i(t-l) = z_i(t-l). \tag{6}$$

## 4. Second Order Temporal Backpropagation

A Hessian matrix is formed by the second derivatives $\frac{\partial^2 E}{\partial p_{ji} \partial p_{j'i'}}$ of an error function with respect to parameters in synapse $s_{ji}$ and $s_{j'i'}$. The first order derivatives (gradient) allow only for a local approximation of the error function surface in the form of a hyperplane in a $|P|$-dimensional parameter space where $|P|$ is the number of the network's parameters. When the Hessian matrix is employed, more accurate quadratic modeling becomes possible.

The first algorithm for an exact Hessian calculation in multilayer feedforward networks was published by Bishop [6] and later extended by the same author to general feedforward networks in [7]. A similar method was presented in a survey paper by Buntine and Weigend [8]. Piché [9] presented a Hessian algorithm for general recurrent networks.

The following algorithm is based on both Bishop's method and the Temporal Backpropagation which uses the Wan's chain rule twice: first for the partial derivative of $E$ with respect to $p_{ji}$ and secondly while substituting the formula for the error gradient $\frac{\partial E}{\partial p_{j'i'}}$ derived in Section 3.:

$$\begin{aligned} \frac{\partial^2 E}{\partial p_{ji} \partial p_{j'i'}} &= \sum_t \frac{\partial a_j(t)}{\partial p_{ji}} \frac{\partial}{\partial a_j(t)} \left( \frac{\partial E}{\partial p_{j'i'}} \right) \\ &= \sum_t \frac{\partial y_{ji}(t)}{\partial p_{ji}} \frac{\partial}{\partial a_j(t)} \left( \sum_{t'} \frac{\partial y_{j'i'}(t')}{\partial p_{j'i'}} \delta_{j'}(t') \right) \\ &= \sum_{t,\,t'} \frac{\partial y_{ji}(t)}{\partial p_{ji}} \left( \frac{\partial^2 y_{j'i'}(t')}{\partial a_j(t) \partial p_{j'i'}} \delta_{j'}(t') + \frac{\partial \delta_{j'}(t')}{\partial a_j(t)} \frac{\partial y_{j'i'}(t')}{\partial p_{j'i'}} \right) \end{aligned}$$

Here we note that this derivation is valid only when the synapse $s_{ji}$ does not occur on any forward propagation path connecting unit $j'$ to the outputs of the network, that is, $\frac{\partial E}{\partial p_{j'i'}}$ cannot be a function of $p_{ji}$. This condition is equivalent in a feedforward network to specifying that $j' \geq i$. It does not limit the generality of this algorithm since the Hessian matrix is symmetric and thus its elements for which this condition does not hold are obtained by calculating the equivalent diagonally opposite elements.

The quantities $\frac{\partial y_{ji}(t)}{\partial p_{ji}}$ have already been presented for different synapse types in the previous section. The second order error terms can be found as

$$\begin{aligned} \frac{\partial \delta_i(t')}{\partial a_k(t)} &= \frac{\partial \left[ f'(a_i(t')) e_i(t') \right]}{\partial a_k(t)} \\ &= \frac{\partial f'(a_i(t'))}{\partial a_k(t)} e_i(t') + f'(a_i(t')) \frac{\partial e_i(t')}{\partial a_k(t)} \end{aligned}$$

According to the Equation (4) the only source of error for output units is the teacher error $e_i^o(t')$:

$$\begin{aligned} \frac{\partial \delta_i(t')}{\partial a_k(t)} &= \frac{\partial f'(a_i(t'))}{\partial a_k(t)} e_i^o(t') + f'(a_i(t')) \frac{\partial e_i^o(t')}{\partial a_k(t)} \\ &= f''(a_i(t')) \frac{\partial a_i(t')}{\partial a_k(t)} e_i^o(t') + [f'(a_i(t'))]^2 \frac{\partial a_i(t')}{\partial a_k(t)} \\ &= \frac{\partial a_i(t')}{\partial a_k(t)} \left[ f''(a_i(t')) e_i^o(t') + [f'(a_i(t'))]^2 \right] \end{aligned}$$

For hidden units the local error is calculated by temporal backpropagation according to Equation (4). Taking a derivative of the $\delta_i$ terms results in a second-order backpropagation:

$$\begin{aligned} \frac{\partial \delta_i(t')}{\partial a_k(t)} &= \frac{\partial f'(a_i(t'))}{\partial a_k(t)} e_i^o(t') \\ &\quad + f'(a_i(t')) \frac{\partial}{\partial a_k(t)} \sum_{j \in \text{OUT}_i} \sum_{l \in \text{MA}_{ji}} \delta_j(t+l) a_{jil} \\ &= f''(a_i(t')) \frac{\partial a_i(t')}{\partial a_k(t)} e_i(t') \\ &\quad + f'(a_i(t')) \sum_{j \in \text{OUT}_i} \sum_{l \in \text{MA}_{ji}} \frac{\partial \delta_j(t+l)}{\partial a_k(t)} a_{jil} \end{aligned} \tag{7}$$

The $\frac{\partial \delta_j(t')}{\partial a_k(t)}$ term occuring in the second sum is calculated by repeated application of the above formula.

The derivative $\frac{\partial a_k(t')}{\partial a_j(t)}$ can be interpreted as a generalized nonlinear "impulse response" between the nodes $j$ and $k$. For $t' = t$ and $k = j$, $\frac{\partial a_k(t')}{\partial a_j(t)} = 1$. Under the assumption that all synapses are causal and the network is ordered we obtain that $\frac{\partial a_k(t')}{\partial a_j(t)} = 0$ when $t' < t$ or $j > k$ (unit $j$ follows unit $k$ in feedforward ordering). In the remaining cases

($t' \geq t$ and $j < k$) a differentiated feedforward propagation (Equation (1)) can be used:

$$\frac{\partial a_k(t')}{\partial a_j(t)} = \sum_{r \in \mathrm{IN}_k} \frac{\partial y_{kr}(t')}{\partial a_j(t)}$$

where

$$\begin{aligned} \frac{\partial y_{kr}(t')}{\partial a_j(t)} &= \sum_{t \leq t'' \leq t'} \frac{\partial a_k(t')}{\partial a_r(t'')} \frac{\partial a_r(t'')}{\partial a_j(t)} \\ &= \sum_{t \leq t'' \leq t'} \frac{\partial a_k(t')}{\partial z_r(t'')} \frac{\partial z_r(t'')}{\partial a_r(t'')} \frac{\partial a_r(t'')}{\partial a_j(t)} \\ &= \sum_{l \in \mathrm{MA}_{kr}} a_{krl} f'(a_r(t'-l)) \frac{\partial a_r(t'-l)}{\partial a_j(t)} \end{aligned} \tag{8}$$

The above three cases can be summarized as:

$$\frac{\partial a_k(t')}{\partial a_j(t)} = \begin{cases} \sum_{r \in \mathrm{IN}_k} \frac{\partial y_{kr}(t')}{\partial a_j(t)} & \text{if } k > j \text{ and } t \geq t' \\ 1 & \text{if } k = j \text{ and } t = t' \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The $\frac{\partial a_r(t'-l)}{\partial a_j(t)}$ term in Equation (8) is obtained by recursive application of Formula (9). The neural network processing unit nonlinearities $f_i(\cdot)$ make the above formula time variant because of the dependency on values $a_r(t')$ which set the "working point" of these nonlinearities.

The formulas for synapse-specific local derivatives $\frac{\partial^2 y_{j'i'}(t')}{\partial a_j(t) \partial p_{j'i'}}$ are derived below. For a constant synapse from Equation (5):

$$\frac{\partial^2 y_{j'i'}(t')}{\partial a_j(t) \partial w_{j'i'}} = \frac{\partial 1}{\partial a_j(t)} = 0$$

while for a FIR Synapse from Equation (6):

$$\begin{aligned} \frac{\partial^2 y_{j'i'}(t')}{\partial a_j(t) \partial a_{j'i'l'}} &= \frac{\partial z_{i'}(t'-l')}{\partial a_j(t)} \\ &= f'(a_{i'}(t'-l')) \frac{\partial a_{i'}(t'-l')}{\partial a_{j'}(t)} \end{aligned}$$

## 5. Conclusions

Th Hessian matrix can be used for a number of purposes including second order optimization methods, post-training pruning using, e.g., the Optimal Brain Surgeon algorithm [10], and post-training analysis, e.g., determination of the effective number of parameters. The algorithm presented above makes these tools accessible for a new class of neural networks.

Computational complexity can be reduced by limiting the number of time lags considered during backpropagation of the error terms in the Equations (4) and (7). This way one can determine a trade-off between the desired accuracy and computational cost.

The authors plan to extend the presented algorithm to dynamic neural networks containing synapses of other types, e.g., IIR, lattice, and gamma linear filters. As a further application we intend to apply the method to the reduction of the number of network parameters.

## References


[1] Andreas S. Weigend and Neil A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past. Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis*, Santa Fe, N.M., May 14–17 1993. Addison-Wesley. 1.

[2] Eric A. Wan. *Finite Impulse Response Neural Networks with Applications in Time Series Prediction*. PhD thesis, Stanford University, 1993. 1., 1., 2., 3., 3.

[3] Jacek M. Zurada. *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul, Minn., 1992. 2.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, March 1989. 2.

[5] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, Mass., 1974. 3.

[6] Christopher M. Bishop. Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation*, 4:494–501, 1992. 4.

[7] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. 4.

[8] Wray Y. Buntine and Andreas S. Weigend. Computing second derivatives in feed-forward networks: A review. *IEEE Transactions on Neural Networks*, 5(3):480–488, May 1994. 4.

[9] Stephen W. Piché. The second derivative of a recurrent network. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 245–250, Orlando, Fla., USA, 27 June–2 July 1994. 4.

[10] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 293–299, San Francisco, Calif., March 28–April 1 1993. 5.