

# A Modified Hidden Weight Optimization Algorithm for Feed-forward Neural Networks

Changhua Yu and Michael T. Manry  
Department of Electrical Engineering  
Arlington, TX 76011  
Telephone:(817) 272-3469  
FAX: (817) 272-2253

## Abstract

The OWO-HWO feedforward network training algorithm alternately solves linear equations for output weights and reduces a separate hidden layer error function with respect to hidden layer weights. Here, a new hidden layer error function is proposed which de-emphasizes net function errors that correspond to saturated activation function values. In addition, an adaptive learning rate based on the local shape of the error surface is used in hidden layer training. Faster learning convergence is experimentally verified.

## 1. Introduction

The output weight optimization-hidden weight optimization (OWO-HWO) algorithm [1] is widely used in the training of feed-forward neural networks such as the multilayer perceptron (MLP). It works well for many approximation and pattern recognition problems [1, 3, 4]. In OWO-HWO, we alternately modify output weights and hidden unit weights to reduce the training error. It modifies the hidden weights based on the minimization of the *Mean Square Error* (MSE) between the desired and the actual net function, which was originally proposed in [2]. Although, OWO-HWO often converges quickly, it didn't use some effective techniques, for examples, premature saturation prevention, learning rate adaptation.

In this paper, we develop improvements to OWO-HWO. In section 2, we review the OWO-HWO algorithm. In section 3, we introduce a new hidden layer error function and an adaptation law for the hidden layer learning rate. The convergence of this method is shown. In section 4, we compare this modified HWO with the original OWO-HWO by running on several different training data sets. Finally,

section 5 concludes this paper.

## 2. Review of OWO-HWO

In this section, we first describe the notation and error functions of feed-forward neural network. Then the strategies of output weight changes and hidden weight changes are introduced.

### 2.1 Notation in HWO

We are given a set of  $N_p$  training patterns  $\{(\mathbf{x}_p, \mathbf{t}_p)\}$  where the  $p$ th input vector  $\mathbf{x}_p$  and the  $p$ th desired output vector  $\mathbf{t}_p$  have dimension  $N$  and  $M$ , respectively.

Here we use three layer, fully connected MLP networks. For the  $j$ th hidden unit, its net input  $net_p(j)$  and the output activation  $O_p(j)$  for the  $p$ th training pattern are

$$\begin{aligned} net_p(j) &= \sum_{i=1}^{N+1} w(j, i) \cdot x_p(i) \\ O_p(j) &= f(net_p(j)) \end{aligned} \quad (2.1)$$

for  $j$  between 1 and  $N_h$  where  $x_p(i)$  denotes the  $i$ th element of  $\mathbf{x}_p$  and  $w(j, i)$  denotes the weight connecting the  $i$ th unit to the  $j$ th unit.  $N_h$  is the number of hidden units and net function thresholds are handled by letting  $x_p(N+1)=1$ . The activation function  $f$  is sigmoidal so

$$f(net_p(j)) = \frac{1}{1 + e^{-net_p(j)}} \quad (2.2)$$

For the  $p$ th pattern, the  $k$ th output is

$$y_p(k) = \sum_{i=1}^{N+1} w_{oi}(k, i) \cdot x_p(i) + \sum_{j=1}^{N_h} w_{oh}(k, j) \cdot O_p(j) \quad (2.3)$$

$k=1, \dots, M$ , where  $w_{oi}(k, i)$  denotes the weight connecting the  $i$ th input node to the  $k$ th output unit,  $w_{oh}(k, j)$  denotes the weight connecting the  $j$ th hidden unit to the  $k$ th output unit.

In batch mode training, the overall performance of a feed-forward network, measured as MSE can be written as

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M [t_p(k) - y_p(k)]^2 \quad (2.4)$$

## 2.2 OWO-HWO algorithm

As the output units have linear activation functions, the OWO procedure can be realized by solving linear equations [1], which result when gradients of  $E$  with respect to the output weights are set to zero.

In HWO, the hidden weights are updated by minimizing a separate error function for each hidden unit. These error functions use differences between the desired and the actual net function. For  $j$ th hidden unit and  $p$ th pattern, the desired net function  $net_{pd}(j)$  is constructed as [1]:

$$net_{pd}(j) \cong net_p(j) + Z \cdot \delta_p(j) \quad (2.5)$$

$Z$  is the learning rate and  $\delta_p(j)$  is the delta function defined as [1]:

$$\delta_p(j) = f'(net_p(j)) \sum_n \delta_{po}(n) w_{oh}(n, j) \quad (2.6)$$

where  $n$  is the index of units in the output layers which are connected to the  $j$ th hidden layer and  $\delta_{po}(n)$  is the delta function of the output layer:

$$\delta_{po}(n) = t_p(n) - y_p(n) \quad (2.7)$$

The hidden weights are updated as

$$w(j, i) \leftarrow w(j, i) + Z \cdot e(j, i) \quad (2.8)$$

where  $e(j, i)$  is the hidden weight change. Starting with equation (2.8) and

$$net_{pd}(j) \cong \sum_{i=1}^{N+1} [w(j, i) + Z \cdot e(j, i)] \cdot x_p(i) \quad (2.9)$$

we obtain

$$\delta_p(j) \cong \sum_{i=1}^{N+1} e(j, i) \cdot x_p(i) \quad (2.10)$$

Using (2.10), the error function for the  $j$ th hidden unit is written as

$$E_\delta(j) = \sum_{p=1}^{N_v} \left[ \delta_p(j) - \sum_i e(j, i) x_p(i) \right]^2 \quad (2.11)$$

which is minimized with respect to  $e(j, i)$  using the conjugate gradient method.

## 3. Modified HWO algorithm

Use of  $E_\delta(j)$  in (2.11) ignores the effects of activation function saturation when  $|net_p(j)|$  is large. In this section, we derive a new hidden layer error function, which takes

saturation into account.

### 3.1 New hidden layer error function

If we substitute equation (2.3) into (2.4), we can rewrite the total MSE as

$$E = \sum_p \sum_{m=1}^M \left[ t_p(m) - \sum_{i=1}^{N_h} w_{oh}(m, i) f_p(i) - \sum_{j=1}^{N+1} w_{oi}(m, j) x_p(j) \right]^2 \quad (3.1)$$

The desired net function change in OWO-HWO is:

$$\Delta net_{pd}(i) = Z \cdot \left( - \frac{\partial E}{\partial net_p(i)} \right) = Z \cdot \delta_p(i) \quad (3.2)$$

and the corresponding desired activation function of the  $i$ th hidden unit is:

$$\begin{aligned} f_{pd}(i) &= f(net_p(i) + \Delta net_{pd}(i)) \\ &\approx f(net_p(i)) + f'(net_p(i)) \Delta net_{pd}(i) \end{aligned} \quad (3.3)$$

Similarly, the actual activation function of  $i$ th hidden unit, after being changed, is:

$$\begin{aligned} f_p(i) &= f(net_p(i) + \Delta net_p(i)) \\ &\approx f(net_p(i)) + f'(net_p(i)) \Delta net_p(i) \end{aligned} \quad (3.4)$$

with

$$\Delta net_p(i) = Z \cdot \sum_{k=1}^{N+1} e(i, k) \cdot x_p(k) \quad (3.5)$$

The  $m$ th output caused by the inputs and the desired hidden outputs is:

$$\bar{T}_p(m) = \sum_{j=1}^{N+1} w_{oi}(m, j) \cdot x_p(j) + \sum_{i=1}^{N_h} w_{oh}(m, i) \cdot f_{pd}(i) \quad (3.6)$$

now the total error can be rewritten as:

$$\begin{aligned} E &= \sum_p \sum_{m=1}^M \left[ t_p(m) - \bar{T}_p(m) + \bar{T}_p(m) - \sum_{j=1}^{N+1} w_{oi}(m, j) x_p(j) \right. \\ &\quad \left. - \sum_{i=1}^{N_h} w_{oh}(m, i) f_p(i) \right]^2 \\ &= \sum_p \sum_{m=1}^M \left\{ [t_p(m) - \bar{T}_p(m)] + \sum_{i=1}^{N_h} w_{oh}(m, i) [f_{pd}(i) - f_p(i)] \right\}^2 \end{aligned} \quad (3.7)$$

From equation (3.2) to (3.5), equation (3.7) becomes:

$$\begin{aligned} E &= \sum_p \sum_{m=1}^M [t_p(m) - \bar{T}_p(m)]^2 \\ &\quad + \left\{ \sum_{i=1}^{N_h} w_{oh}(m, i) f_p'(i) Z \left[ \delta_p(i) - \sum_{k=1}^{N+1} e(i, k) x_p(k) \right] \right\}^2 \end{aligned} \quad (3.8)$$

if we assume that  $[t_p(m) - \bar{T}_p(m)]$  is uncorrelated with  $[\delta_p(i) - \sum_k e(i,k)x_p(k)]$  and where  $f'_p(i)$  is shorthand for the activation derivative  $f'(net_p(i))$  in (3.3). Then we get:

$$E = \sum_p \sum_{m=1}^M [t_p(m) - \bar{T}_p(m)]^2 + \sum_p \sum_{m=1}^M \left\{ \sum_{i=1}^{N_h} w_{oh}(m,i) f'_p(i) Z \left[ \delta_p(i) - \sum_{k=1}^{N+1} e(i,k)x_p(k) \right] \right\}^2 \quad (3.9)$$

Evaluating (3.9) further we have:

$$E \approx \sum_p \sum_{m=1}^M [t_p(m) - \bar{T}_p(m)]^2 + \sum_{i=1}^{N_h} w_{oh}^2(m,i) Z^2 (f'_p(i))^2 \left[ \delta_p(i) - \sum_{k=1}^{N+1} e(i,k)x_p(k) \right]^2 \quad (3.10)$$

Because  $[t_p(m) - \bar{T}_p(m)]$  and  $w_{oh}(m,i)$  are constant during the HWO procedure, if we want to minimize  $E$ , we have to minimize:

$$E_\delta(i) = \sum_p \left( f'_p(i) \right)^2 \left[ \delta_p(i) - \sum_{k=1}^{N+1} e(i,k)x_p(k) \right]^2 \quad (3.11)$$

This hidden layer error function successfully de-emphasizes error in saturated hidden units. However, the de-emphasis has experimentally been found to be too severe. Accordingly, we actually use

$$E_\delta(j) = \sum_{p=1}^{N_v} \left[ \delta_p(j) - \sum_i e(j,i)x_p(i) \right]^2 \cdot f'(net_p(j)) \quad (3.12)$$

### 3.2 Adaptive hidden layer learning rate

In OWO-HWO, a bold drive (BD) technique [5] is used to adapt the hidden layer learning rate. It increases the learning rate  $Z$ , when the error decreases at a given step, as

$$Z \leftarrow Z \cdot 1.2 \quad (3.13)$$

Otherwise the learning rate is decreased as

$$Z \leftarrow Z \cdot 0.5 \quad (3.14)$$

As the error function has broad flat regions adjoining narrow steep ones [6], the learning rate should be adapted according to the local shape of the error surface. That is, when in flat regions, larger  $Z$  can be used, while in a steep

region, the learning rate should be kept small.

Combining this idea with the BD technique, we developed a new adaptive learning rate for the hidden layer. When the error increases at a given step, we still use (3.14). When the error decreases at a given step, we use

$$Z \leftarrow Z \cdot G \quad (3.15)$$

where the gain  $G$  is

$$G = 1 + \frac{\alpha}{1 + e^R} \quad (3.16)$$

The parameter  $\alpha$  is used to limit the maximum value of  $G$ . In our simulations,  $\alpha$  is set to 0.49. The ratio  $R$  is calculated using the last epoch's parameters, as

$$R = \frac{|\Delta E|}{\|\Delta \mathbf{W}\|} = \frac{|E_1 - E_2|}{Z \|\mathbf{e}\|} \quad (3.17)$$

The hidden weight change vector  $\mathbf{e}$  in (3.17) is available from solving (3.12) in the previous HWO iteration.  $E_1$  and  $E_2$  are the total output MSEs of last two iterations respectively.

In fact, if we assume the hidden weight changes are small,  $R$  approximates the current gradient magnitude of the hidden unit error surface. When the gradient magnitude is small, the local shape of error function is flat; otherwise it is steep [6]. So from (3.15) to (3.17), the learning rate  $Z$  is modulated by an adaptive factor  $G$ , which varies from 1 to  $(1+\alpha)$  with respect to the local shape of the error function.

### 3.3 Convergence of the modified HWO algorithm

From equation (3.12), we can see that when  $E_\delta(j)$  approaches zero,

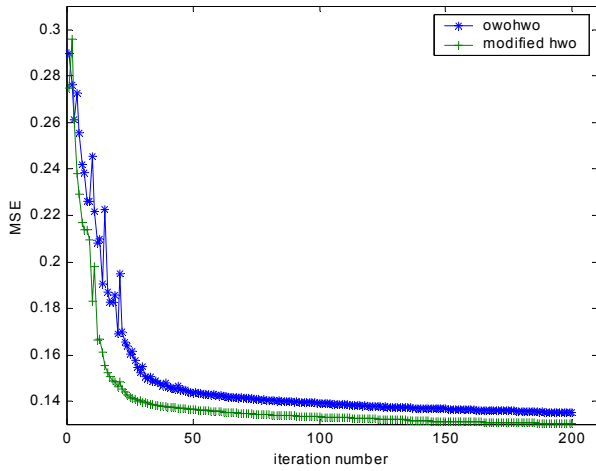
$$\sum_i e(j,i) \cdot x_p(i) \approx \delta_p(j) \quad (3.18)$$

In a given iteration, the change in  $E$  caused by the  $j$ th hidden unit is approximated as

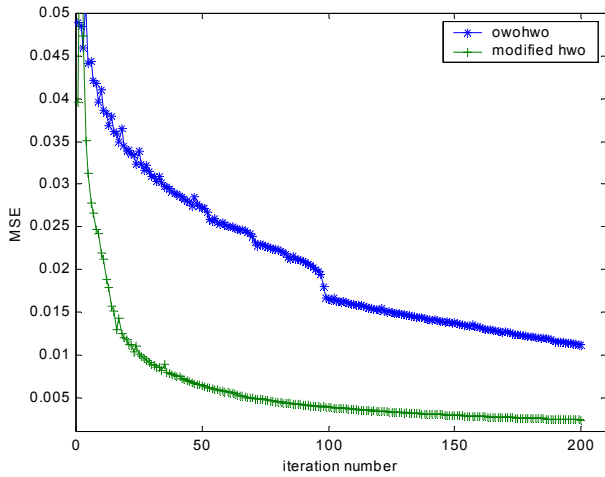
$$\begin{aligned} \Delta E(j) &\cong \sum_i \frac{\partial E}{\partial w(j,i)} \cdot \Delta w(j,i) \\ &= Z \cdot \sum_i \frac{\partial E}{\partial w(j,i)} \cdot e(j,i) \\ &= Z \cdot \sum_i \frac{1}{N_v} \sum_{p=1}^{N_v} \frac{\partial E_p}{\partial w(j,i)} \cdot e(j,i) \\ &= Z \cdot \sum_i \frac{1}{N_v} \sum_{p=1}^{N_v} (-\delta_p(j) \cdot x_p(i) \cdot e(j,i)) \\ &= -Z \frac{1}{N_v} \sum_{p=1}^{N_v} \delta_p(j) \sum_i x_p(i) \cdot e(j,i) \\ &\cong -Z \cdot \frac{1}{N_v} \sum_{p=1}^{N_v} \delta_p^2(j) \end{aligned} \quad (3.19)$$

The total change of the error function  $E$ , due to changes

in all hidden weights, becomes



**Fig. 1 Simulation results for example 1 Data: Twod.tra, Structure: 8-10-7**



**Fig. 2 Simulation results for example 2 Data: fmtrain.dat, Structure: 5-10-1**

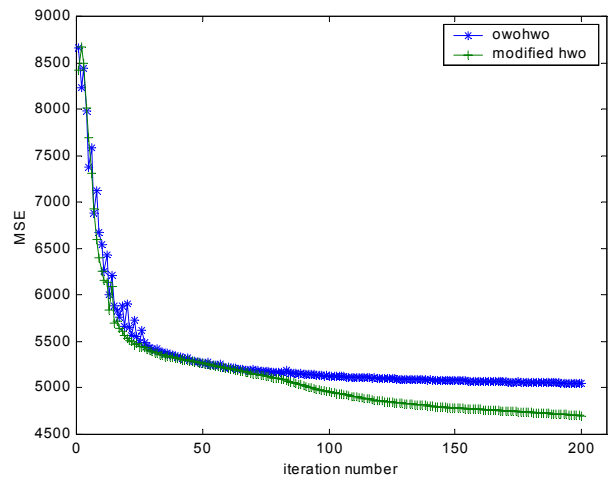
$$\Delta E \cong -Z \frac{1}{N_v} \sum_j \sum_{p=1}^{N_v} \delta_p^2(j) \quad (3.20)$$

so the changes in  $E$  will be a nonincreasing sequence and the total MSE  $E$  converges.

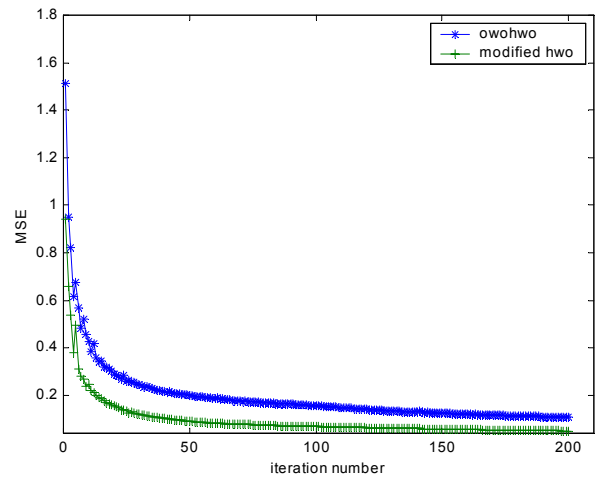
#### 4. Simulation Results

The proposed ideas were verified using several training data sets. Training data set *Twod.tra* contains simulated data based on models from backscattering measurements. It has 8 inputs and 7 outputs, 1768 patterns. The simulation result is shown in figure 1.

Training data set *Fmtrain.dat* is used to train a neural



**Fig. 3 Simulation results for example 2 Data: Power12.tra, Structure: 12-10-1**



**Fig. 4 Simulation results for example 3 Data: single2.tra, Structure: 16-20-3**

network to perform demodulation of an FM (frequency modulation) signal containing a sinusoidal message. It has 5 inputs and 1 output, 1024 patterns. Figure 2 shows the result.

The third training data set, *Power12.tra*, was generated using data obtained from TU electric company in Texas. It has 12 inputs, 1 output and 1414 patterns. The training results are shown in figure 3. The last training example *Single2.tra* consists of 16 inputs, 3 outputs and 10000 patterns. It represents the training set for inversion of surface permittivity, the normalized surface rms roughness, and the surface correlation length found in the back scattering models from randomly rough dielectric surfaces.

The training results are given in figure 4.

From the simulation results, it is obvious that the modified HWO algorithm has faster convergence speed.

algorithm using learning adaptation Methods”. *Neural Computation*, Vol.11, 1999, pp. 1769-1796

## 5. Conclusion

In order to accelerate convergence of the original OWO-HWO algorithm, this paper proposed some improvements. The new hidden layer error function takes hidden unit saturation into account. The hidden layer learning rate now adapts according to the local shape of the error surface. Both techniques successfully increased the training speed.

When testing on several data sets, the modified HWO algorithm outperforms the original one in most cases. However, there are some issues that need further investigation. More theoretical analysis of the new hidden layer error function is needed. We also need to design an adaptation law for the learning rate without a heuristic parameter  $\alpha$ .

## Acknowledgement

This work was supported by the Advanced Technology Program of the state of Texas, under grant number 003656-0129-2001.

## Reference

1. Hung-Han Chen, Michael T. Manry, and Hema Chandrasekaran, “A neural network training algorithm utilizing multiple set of linear equations”. *Neurocomputing*, Vol. 25, No. 1-3, April 1999, pp. 55-72.
2. Robert S. Scalero and Nazif Tepedelenlioglu, “A fast new algorithm for training feedforward neural networks”. *IEEE Transactions on signal processing*, Vol. 40, No.1, January 1992, pp. 202-210.
3. M. T. Manry, et al, “Fast training of neural networks” for remote sensing”. *Remote sensing reviews*, Vol. 9, 1994, pp. 77-96.
4. Hahn-Ming Lee, Chih-Ming Chen, Tzong-Ching Huang, “Learning efficiency improvement of back-propagation algorithm by error saturation prevention method”. *Neurocomputing*, Vol.41, 2001, pp. 125-143.
5. Sang-Hoon Oh and Soo-Young Lee, “Optimal Learning rates for each pattern and neuron in gradient descent training of multilayer perceptrons”. *IJCNN '99. International Joint Conference on Neural Networks*, vol.3, pp. 1635-1638.
6. G. D. Magoulas, M. N. Vrahatis, G. S. Androulakis, “Improving the convergence of the backpropagation