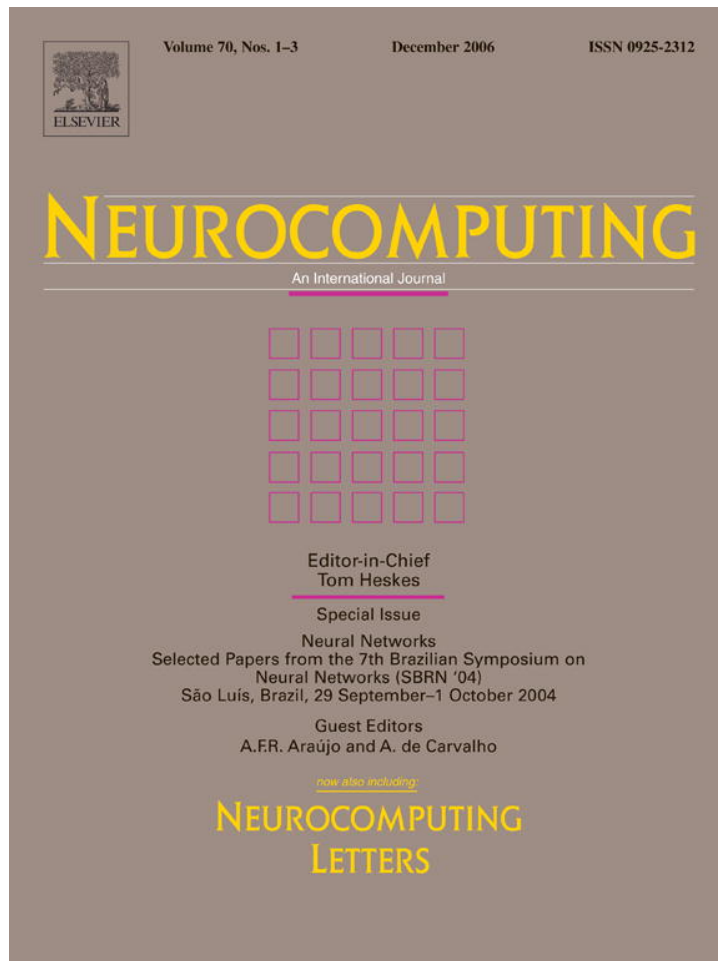


Provided for non-commercial research and educational use only.
Not for reproduction or distribution or commercial use.



This article was originally published in a journal published by Elsevier, and the attached copy is provided by Elsevier for the author's benefit and for the benefit of the author's institution, for non-commercial research and educational use including without limitation use in instruction at your institution, sending it to specific colleagues that you know, and providing a copy to your institution's administrator.

All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution's website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier's permissions site at:

<http://www.elsevier.com/locate/permissionusematerial>

An efficient hidden layer training method for the multilayer perceptron

Changhua Yu^{a,*}, Michael T. Manry^b, Jiang Li^c, Pramod Lakshmi Narasimha^b

^a*FastVDO LLC, Columbia, MD 21046, USA*

^b*Department of Electrical Engineering, University of Texas at Arlington, TX 76019*

^c*Department of Radiology, Clinical Center National Institutes of Health, Bethesda, MD 20892, USA*

Received 17 December 2003; received in revised form 22 November 2005; accepted 23 November 2005

Communicated by S.Y. Lee

Available online 18 May 2006

Abstract

The output-weight-optimization and hidden-weight-optimization (OWO–HWO) training algorithm for the multilayer perceptron alternately solves linear equations for output weights and reduces a separate hidden layer error function with respect to hidden layer weights. Here, three major improvements are made to OWO–HWO. First, a desired net function is derived. Second, using the classical mean square error, a weighted hidden layer error function is derived which de-emphasizes net function errors that correspond to saturated activation function values. Third, an adaptive learning factor based on the local shape of the error surface is used in hidden layer training. Faster learning convergence is experimentally verified, using three training data sets.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Hidden weight optimization (HWO); Convergence; Hidden layer error function; Saturation; Adaptive learning factor

1. Introduction

Back propagation (BP) [10,11,26,29] was the first effective training algorithm for the multilayer perceptron (MLP). Subsequently, the MLP has been widely applied in the fields of signal processing [16], remote sensing [17], and pattern recognition [3]. However, the inherent slow convergence of BP has delayed the adoption of the MLP by much of the signal processing community.

During the last two decades, many improvements to BP have been made. One reason for BP's slow convergence is the saturation occurring in some of the nonlinear units. When hidden units become saturated, the error function reaches a local minimum early in the training stage. In [14,30], proper weight initializations are used to avoid premature saturation. Lee et al. [14] derived the probability of incorrect saturation for the case of binary training patterns. They concluded that the saturation probability is a function of the maximum value of the initial weights, the number of units in each layer, and the slope of the sigmoid function. Yam and Chow [30]

proposed to initialize the weights to ensure that the outputs of the hidden units are in the active region of the sigmoid function. Various techniques, such as an error saturation prevention function in [13], the cross-entropy error function [23] and its extension [19], are proposed to prevent the units from premature saturation.

In [2], Battiti reviewed first and second order algorithms for learning in neural networks. First order methods are fast and effective for large-scale problems, while second order techniques have higher precision. The Levenberg–Marquardt (LM) [6,9,17] method is a well-known second order algorithm with better convergence properties [2] than conventional BP. Unfortunately, it requires $O(N_w^3)$ storage and calculations of order $O(N_w^3)$ where N_w is the total number of weights in an MLP [18]. Hence the LM method is impractical for all but small networks.

Many investigators have trained the different MLP layers separately. In layer-by-layer (LBL) training [20,28], the output weights are first updated. Then the corresponding desired outputs for the hidden units are found. Finally, the hidden weights are found by minimizing the difference between the actual net function and a desired net function. In [28], for the current training iteration, the optimal output and hidden weights are found by solving sets of

*Corresponding author. Tel.: +1 817 272 3469; fax: +1 817 272 3483.

E-mail addresses: ychemailuta@yahoo.com (C. Yu), manry@uta.edu (M.T. Manry).

linear equations using pseudo-inverse matrices. In [20] by contrast, the output and hidden weights are updated in the gradient directions with optimal learning factors. This strategy makes each step in [20] similar to steepest descent.

Some researchers have developed fast training techniques by solving sets of linear equations [1,3,16,25]. When output units have linear activation functions, linear equations can be solved for the output weights. For example, the output weight optimization (OWO) algorithm [1,17] has been successfully used to minimize the MSE by solving linear equations. In [17], the MLP is trained by the so-called output weight optimization-back propagation (OWO-BP) algorithm, where the output weights are found by solving linear equations and the hidden weights are updated by BP.

Scalero and Tepedelenlioglu [27] developed a non-batch training approach for feed-forward neural networks in which separate error functions are minimized for each hidden unit. This idea greatly improved training efficiency. However, they didn't use OWO to solve for the output weights. Using ideas from [27], Chen [3] constructed a batch mode training algorithm called output weight optimization-hidden weight optimization (OWO-HWO). In OWO-HWO, output weights and hidden unit weights are alternately modified to reduce training error. The algorithm modifies the hidden weights based on the minimization of the MSE between the desired and the actual net function, as originally proposed in [27]. Although, OWO-HWO increases training speed, it still has room for improvement [28] because it uses the delta function as the desired net function change. This makes HWO similar to steepest descent, except that net functions are varied instead of weights. In addition, HWO is equivalent to BP applied to the hidden weights under certain conditions [3].

In this paper, we develop some improvements to OWO-HWO. In Section 2, we describe the notation used in this paper and review the OWO-HWO algorithm. In Section 3, we introduce a new desired net function change, which updates the net function in the optimal gradient direction. We derive a weighted hidden layer error function, first used in [20], directly from the global MSE. After that, we construct an adaptation law for the hidden layer learning factor. The convergence of this method is shown. In Section 4, by running on several different training data sets, we compare the new HWO with the original OWO-HWO, and other training algorithms. Finally, Section 5 concludes this paper.

2. Review of OWO-HWO

In this section, we describe the structure and notation of a fully connected MLP and then review the OWO-HWO algorithm.

2.1. Structure and notation of a fully connected MLP

Without loss of generality, we restrict ourselves to a three layer fully connected MLP, which is commonly used. The structure of the MLP is shown in Fig. 1. Bypass

weights from input layer to output layer are used, but are not shown in the figure for clarity. These connections allow the network to easily model the linear component of the desired mapping, and reduce the number of required hidden units. The training data set consists of N_v training patterns $\{(x_p, t_p)\}$, where the p th input vector x_p and the p th desired output vector t_p have dimensions N and M , respectively. Thresholds in the hidden layer are handled by letting $x_{p, (N+1)} = 1$.

For the k th hidden unit, the net input net_{pk} and the output activation O_{pk} for the p th training pattern are

$$net_{pk} = \sum_{n=1}^{N+1} w_{hi}(k, n) \cdot x_{pn},$$

$$O_{pk} = f(net_{pk}), \quad (2.1)$$

where x_{pn} denotes the n th element of x_p and $w_{hi}(k, n)$ denotes the weight connecting the n th input unit to the k th hidden unit. N_h is the number of hidden units. The activation function f is sigmoidal

$$f(net_{pk}) = \frac{1}{1 + e^{-net_{pk}}}. \quad (2.2)$$

The i th output y_{pi} for the p th training pattern is

$$y_{pi} = \sum_{n=1}^{N+1} w_{oi}(i, n) \cdot x_{pn} + \sum_{k=1}^{N_h} w_{oh}(i, k) \cdot O_{pk}. \quad (2.3)$$

For convenience, in the OWO procedure we augment the input vector as

$$\tilde{x}_{pn} = \begin{cases} x_{pn}, & n = 1, 2, \dots, N, \\ 1, & n = N + 1, \\ O_p, (n - N - 1), & n = N + 2, \dots, N + N_h + 1. \end{cases} \quad (2.4)$$

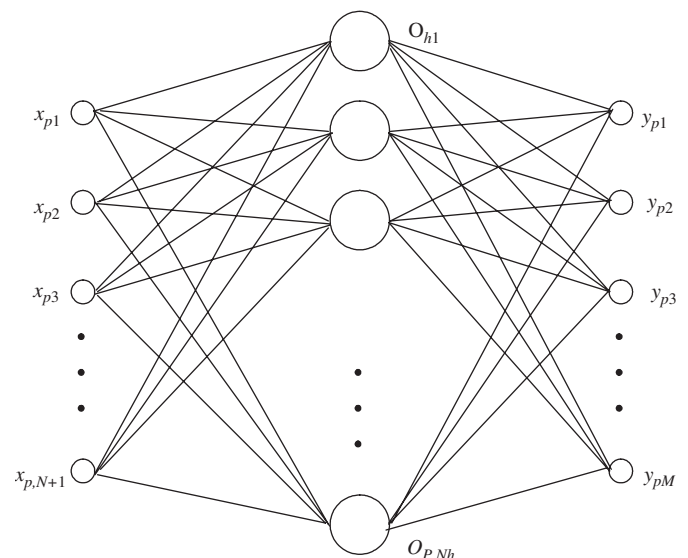


Fig. 1. MLP structure for one hidden layer.

Using the augmented input vector in (2.4), (2.3) is rewritten as

$$y_{pi} = \sum_{n=1}^L w_o(i, n) \tilde{x}_{pn}, \quad (2.5)$$

where the weights $w_o(i, n) = w_{oi}(i, n)$ for $1 \leq n \leq N + 1$ and $w_o(i, n) = w_{oh}(i, n - N - 1)$ for $N + 2 \leq n \leq N + N_h + 1 = L$.

For each training pattern, the training error is

$$E_p = \sum_{i=1}^M [t_{pi} - y_{pi}]^2, \quad (2.6)$$

where t_{pi} denotes the i th element of the p th desired output vector. In batch mode training, the overall performance of a feed-forward network, measured as mean square error (MSE) can be written as

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M [t_{pi} - y_{pi}]^2. \quad (2.7)$$

2.2. Output weight optimization

As the output units have linear activation functions, the OWO procedure, for finding weights $w_o(i, n)$ in (2.5), can be realized by solving linear equations [1,3,17], which result when gradients of E with respect to the output weights are set to zero. These equations are

$$\sum_{n=1}^L w_o(i, n) R(n, m) = C(i, m), \quad (2.8)$$

where C is the cross correlation matrix with elements:

$$C(i, m) = \frac{1}{N_v} \sum_{p=1}^{N_v} t_{pi} \cdot \tilde{x}_{pm}, \quad (2.9)$$

R is the auto correlation matrix defined as:

$$R(n, m) = \frac{1}{N_v} \sum_{p=1}^{N_v} \tilde{x}_{pn} \tilde{x}_{pm}. \quad (2.10)$$

Since the equations are often ill-conditioned, meaning that the determinant of R is close to 0, it is often unsafe to use Gauss–Jordan elimination. The *singular value decomposition* (SVD) [8], *LU decomposition* (LUD) [24], and *conjugate gradient* (CG) [6] approaches are better.

2.3. Hidden weight optimization

Hidden weight optimization is a full batch algorithm developed from the ideas in [27]. In HWO, the hidden weights are updated by minimizing separate error functions for each hidden unit. The error functions measure the difference between the desired and the actual net function. For the k th hidden unit and p th pattern, the desired net function is constructed as [3]

$$net_{pkd} \cong net_{pk} + Z \cdot \delta_{pk} \quad (2.11)$$

where net_{pkd} is the desired net function and net_{pk} is the actual one in (2.1). Z is the learning factor and δ_{opi} is the delta function [3,11,26] of the i th output:

$$\delta_{opi} = -\frac{\partial E_p}{\partial y_{pi}} = [t_{pi} - y_{pi}]. \quad (2.12)$$

The delta function for the k th hidden unit [2,3,7] is

$$\delta_{pk} = -\frac{\partial E_p}{\partial net_{pk}} = f'(net_{pk}) \sum_{i=1}^M w_{oh}(i, k) \delta_{opi}. \quad (2.13)$$

The hidden weights are to be updated as

$$w_{hi}(k, n) \leftarrow w_{hi}(k, n) + Z \cdot e(k, n), \quad (2.14)$$

where $e(k, n)$ is the weight change. The weight changes are derived using

$$net_{pk} + Z \cdot \delta_{pk} \cong \sum_{n=1}^{N+1} [w_{hi}(k, n) + Z \cdot e(k, n)] \cdot x_{pn}. \quad (2.15)$$

Therefore,

$$\delta_{pk} \cong \sum_{n=1}^{N+1} e(k, n) \cdot x_{pn}. \quad (2.16)$$

The error of (2.15) and (2.16) for the k th hidden unit is measured as

$$E_\delta(k) = \frac{1}{N_v} \sum_{p=1}^{N_v} \left[\delta_{pk} - \sum_{n=1}^{N+1} e(k, n) \cdot x_{pn} \right]^2. \quad (2.17)$$

Equating the gradient of $E_\delta(k)$ with respect to the hidden weight change to zero, we have

$$\sum_{n=1}^{N+1} e(k, n) R(n, m) = C_\delta(k, m), \quad (2.18)$$

where

$$C_\delta(k, m) = \frac{1}{N_v} \sum_{p=1}^{N_v} [\delta_{pk} \cdot x_{pm}] = \frac{-\partial E}{\partial w_{hi}(k, m)}. \quad (2.19)$$

The hidden weight change $e(k, n)$ can be found from (2.18) by using the conjugate gradient method. After finding the learning factor Z , the hidden weights are updated as in (2.14).

3. Enhancement of OWO-HWO

From (2.11) and (2.13), we can see that the net functions are updated in the gradient direction. It is well known that optimizing in the gradient direction, as in steepest descent, is very slow. In addition, using $E_\delta(k)$ in (2.17) ignores the effects of activation function saturation when $|net_{pk}|$ is large as was pointed out in [15] for the BP algorithm. In this section, we update the net functions along new directions that tend to minimize the total training error and derive the corresponding new hidden layer error functions, which takes saturation into account.

3.1. The optimal direction of desired net function change

As mentioned in (2.11), for the original HWO algorithm, the desired net function change is $Z \cdot \delta_{pj}$, where the delta function is the negative gradient of E with respect to the current net function net_{pj} . This strategy of updating net functions using their gradient directions results in slow training for the hidden units.

In this section, we introduce a new desired net function net_{pj}^* :

$$net_{pj}^* = net_{pj} + Z \cdot \Delta net_{pj}^*, \quad (3.1)$$

where Δnet_{pj}^* , written as

$$\Delta net_{pj}^* = net_{pj}^* - net_{pj}, \quad (3.2)$$

is the difference between current net function net_{pj} and an optimal value net_{pj}^* . Now the net function approaches the optimal value net_{pj}^* , instead of moving in the negative gradient direction. The current task for constructing the new algorithm HWO is to find Δnet_{pj}^* .

Using Taylor series and Eqs. (2.1), (3.2), the corresponding hidden layer output O_{pj}^* caused by net_{pj}^* can be written as

$$O_{pj}^* = f(net_{pj}^*) \approx O_{pj} + f'_{pj} \cdot Z \cdot \Delta net_{pj}^* \quad (3.3)$$

where f'_{pj} is shorthand for the activation derivative $f'(net_{pj})$. Replacing O_{pj} by its optimal value O_{pj}^* in (2.3), (2.7) becomes

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \left[t_{pi} - \sum_{n=1}^{N+1} w_{oi}(i,n)x_{pn} - \sum_{\substack{k=1 \\ k \neq j}}^{N_h} w_{oh}(i,k)O_{pk} - w_{oh}(i,j)O_{pj}^* \right]^2 \quad (3.4)$$

Now Δnet_{pj}^* can be derived based on

$$\left. \frac{\partial E}{\partial net_{pj}} \right|_{net_{pj}=net_{pj}^*} = 0. \quad (3.5)$$

Using (3.1) to (3.5) yields

Using (3.3) and (2.12), we replace $(O_{pj}^* - O_{pj})$ and $(t_{pj} - y_{pi})$ in (3.6) by $f'_{pj} \Delta net_{pj}^*$ and δ_{opi} , yielding

$$\begin{aligned} \frac{\partial E}{\partial net_{pj}^*} &= -\frac{2}{N_v} \sum_{i=1}^M \left\{ \left[t_{pi} - \sum_{n=1}^{N+1} w_{oi}(i,n)x_{pn} - \sum_{\substack{k=1 \\ k \neq j}}^{N_h} w_{oh}(i,k)O_{pk} - w_{oh}(i,j)O_{pj}^* \right] \cdot w_{oh}(i,j) \frac{\partial O_{pj}^*}{\partial net_{pj}^*} \right\} \\ &= -\frac{2}{N_v} \left\{ \sum_{i=1}^M \left[t_{pi} - y_{pi} - w_{oh}(i,j)(O_{pj}^* - O_{pj}) \right] \cdot w_{oh}(i,j) \right\} \cdot \frac{\partial O_{pj}^*}{\partial net_{pj}^*}. \end{aligned} \quad (3.6)$$

$$\begin{aligned} \frac{\partial E}{\partial net_{pj}^*} &\approx -\frac{2}{N_v} \cdot \frac{\partial O_{pj}^*}{\partial net_{pj}^*} \left\{ \sum_{i=1}^M \left[\delta_{opi} - w_{oh}(i,j)f'_{pj} \Delta net_{pj}^* \right] \cdot w_{oh}(i,j) \right\} \\ &= -\frac{2}{N_v} \cdot \frac{\partial O_{pj}^*}{\partial net_{pj}^*} \left[\sum_{i=1}^M \delta_{opi} w_{oh}(i,j) - f'_{pj} \Delta net_{pj}^* \sum_{i=1}^M w_{oh}^2(i,j) \right]. \end{aligned} \quad (3.7)$$

Setting the right hand side of (3.7) to zero, we can find Δnet_{pj}^* as

$$\Delta net_{pj}^* \approx \frac{\sum_{i=1}^M \delta_{opi} w_{oh}(i,j)}{f'_{pj} \sum_{i=1}^M w_{oh}^2(i,j)}. \quad (3.8)$$

Using (2.12) and (2.13), (3.8) becomes

$$\Delta net_{pj}^* \approx \frac{\delta_{pj}}{(f'_{pj})^2 \sum_{i=1}^M w_{oh}^2(i,j)}. \quad (3.9)$$

3.2. A weighted hidden error function

If we substitute Eq. (2.3) into (2.7), we can rewrite the total MSE as

$$E = \frac{1}{N_v} \sum_p \sum_{i=1}^M \left[t_{pi} - \sum_{k=1}^{N_h} w_{oh}(i,k)O_{pk} - \sum_{n=1}^{N+1} w_{oi}(i,n)x_{pn} \right]^2. \quad (3.10)$$

During the HWO procedure, if we define net_{pkd} as in (3.1), the corresponding hidden unit output O_{pkd} could be approximated by using Taylor series as

$$O_{pkd} = f(net_{pkd}) \approx O_{pk} + Z \cdot f'_{pk} \cdot \Delta net_{pkd}^*. \quad (3.11)$$

However, because the hidden weights are updated as in (2.14), the actual net function we find is

$$\begin{aligned} \bar{net}_{pk} &= \sum_{n=1}^{N+1} [w_{hi}(k,n) + Z \cdot e(k,n)]x_{pn} \\ &= net_{pk} + Z \cdot \sum_{n=1}^{N+1} e(k,n)x_{pn} \end{aligned} \quad (3.12)$$

The actual k th hidden unit output after HWO is approximated as

$$\bar{O}_{pk} = f(\bar{net}_{pk}) \approx O_{pk} + Z \cdot f'_{pk} \cdot \sum_{n=1}^{N+1} e(k,n)x_{pn}. \quad (3.13)$$

If we denote the i th output caused by the inputs and O_{pkd} as

$$T_{pi} = \sum_{n=1}^{N+1} w_{oi}(i,n) \cdot x_{pn} + \sum_{k=1}^{N_h} w_{oh}(i,k) \cdot O_{pkd} \quad (3.14)$$

then after HWO, the actual total error can be rewritten as

$$\begin{aligned} E &= \frac{1}{N_v} \sum_p \sum_{i=1}^M \left[t_{pi} - T_{pi} + T_{pi} - \sum_{n=1}^{N+1} w_{oi}(i,n)x_{pn} - \sum_{k=1}^{N_h} w_{oh}(i,k)\bar{O}_{pk} \right]^2 \\ &= \frac{1}{N_v} \sum_p \sum_{i=1}^M \left\{ [t_{pi} - T_{pi}] + \sum_{k=1}^{N_h} w_{oh}(i,k)[O_{pkd} - \bar{O}_{pk}] \right\} \end{aligned} \quad (3.15)$$

If we assume that $[t_{pi} - T_{pi}]$ is uncorrelated with

$$\left[\Delta net_{pk} - \sum_n e(k,n)x_{pn} \right],$$

and use Eq. (3.11) and (3.13), Eq. (3.15) becomes

$$E \approx \frac{1}{N_v} \sum_p \sum_{i=1}^M [t_{pi} - T_{pi}]^2 + \frac{1}{N_v} \sum_p \sum_{i=1}^M \left\{ \sum_{k=1}^{N_h} w_{oh}(i, k) f'_{pk} Z \left[\Delta net_{pk}^* - \sum_{n=1}^{N+1} e(k, n) x_{pn} \right] \right\}^2. \quad (3.16)$$

Here Z is the learning factor along the Δnet_{pj}^* direction. We can evaluate (3.16) further because it is reasonable to assume that the value

$$\left\{ w_{oh}(i, k) f'_{pk} \left[\Delta net_{pk}^* - \sum_n e(k, n) x_{pn} \right] \right\}$$

associated with different hidden units are uncorrelated with each other. This assumption yields

$$E \approx \frac{1}{N_v} \sum_p \sum_{i=1}^M [t_{pi} - T_{pi}]^2 + \frac{1}{N_v} \sum_p \sum_{i=1}^M \sum_{k=1}^{N_h} w_{oh}^2(i, k) Z^2 (f'_{pk})^2 \left[\Delta net_{pk}^* - \sum_{n=1}^{N+1} e(k, n) x_{pn} \right]^2. \quad (3.17)$$

Since Z , $[t_{pi} - T_{pi}]$ and $w_{oh}(i, k)$ are constant during the HWO procedure, minimizing E is equivalent to minimizing

$$E_\delta(k) = \frac{1}{N_v} \sum_p (f'_{pk})^2 \left[\Delta net_{pk}^* - \sum_{n=1}^{N+1} e(k, n) x_{pn} \right]^2. \quad (3.18)$$

This hidden layer error function, which has been introduced without derivation in [19], successfully de-emphasizes error in saturated hidden units using the square of the derivative of the activation function. Then the amount of the hidden weight's update depends upon whether the hidden net functions are in the linear or saturation regions of the sigmoid function. If the current net_{pk} is in a saturation region, the difference between desired and actual hidden output is small although the difference between net_{pk} and net_{pkd} is large. In this case, because there is no need to change the associated weights according to the large difference between net_{pk} and net_{pkd} , the small value of $(f'_{pk})^2$ term will decrease the weight change $e(k, n)$. When net_{pk} is in the linear region, the hidden weights will be updated according to the difference between net_{pk} and net_{pkd} .

That is, the term $(f'_{pk})^2$ de-emphasizes net function error corresponding to saturated activations. When net_{pk} is in the sigmoid's linear region, errors between net_{pk} and net_{pkd} receive large weight in (3.18).

The hidden weight change $e(k, m)$ in (3.18) can be found as in Section 2. First, the gradient of the new $E_\delta(k)$ with respect to the hidden weight change $e(k, m)$ is equated to zero, yielding Eq. (2.18). However, the autocorrelation matrix \mathbf{R} and cross correlation matrix \mathbf{C}_δ

are now found as:

$$R(n, m) = \frac{1}{N_v} \sum_{p=1}^{N_v} (x_{pn} x_{pm}) \cdot (f'_{pk})^2 \quad (3.19)$$

and

$$C_\delta(k, m) = \frac{1}{N_v} \sum_{p=1}^{N_v} [\Delta net_{pk}^* \cdot x_{pm}] \cdot (f'_{pk})^2. \quad (3.20)$$

Again, we have $(N + 1)$ equations in $(N + 1)$ unknowns for the k th hidden unit. After finding the learning factor Z , the hidden weights are updated as in (2.14).

3.3. Improved bold drive technique

In [12,20], the weights and the desired hidden outputs were updated using an optimal learning factor. However, those optimal learning factors are found just based on current gradients. They do not guarantee an optimal solution for the whole training procedure, and do not guarantee faster convergence than heuristic learning factors. As the error function has broad flat regions adjoining narrow steep ones [15], the learning factor 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 factor should be kept small.

Combining this idea with the BD technique [3,21], we developed a new adaptive learning factor for the hidden layer. If the learning factor Z for hidden weight updating is small, then the change in the output layer error function E due to the change in $w_{hi}(k, n)$ is approximately

$$\Delta E = \frac{\partial E}{\partial w_{hi}(k, n)} \cdot \Delta w_{hi}(k, n) = Z \cdot \frac{\partial E}{\partial w_{hi}(k, n)} \cdot e(k, n). \quad (3.21)$$

The total effect of hidden weight changes on the output error will be

$$\Delta E = Z \sum_{k=1}^{N_h} \sum_{n=1}^{N+1} \frac{\partial E}{\partial w_{hi}(k, n)} \cdot e(k, n). \quad (3.22)$$

Assume E is reduced by a small factor Z' between 0.0 and 0.1,

$$\Delta E = -Z' \cdot E. \quad (3.23)$$

Combining (3.22) and (3.23), the learning factor Z will be

$$Z = \frac{-Z' E}{\sum_k \sum_n (\partial E / \partial w_{hi}(k, n)) \cdot e(k, n)}. \quad (3.24)$$

Here, we develop an adaptive law for Z' . When the error increases at a given step, the learning factor is decreased by setting

$$Z' \leftarrow Z' \cdot 0.5 \quad (3.25)$$

and the best previous weights are reloaded. When the error decreases at a given step, we use

$$Z' \leftarrow Z' \cdot G, \quad (3.26)$$

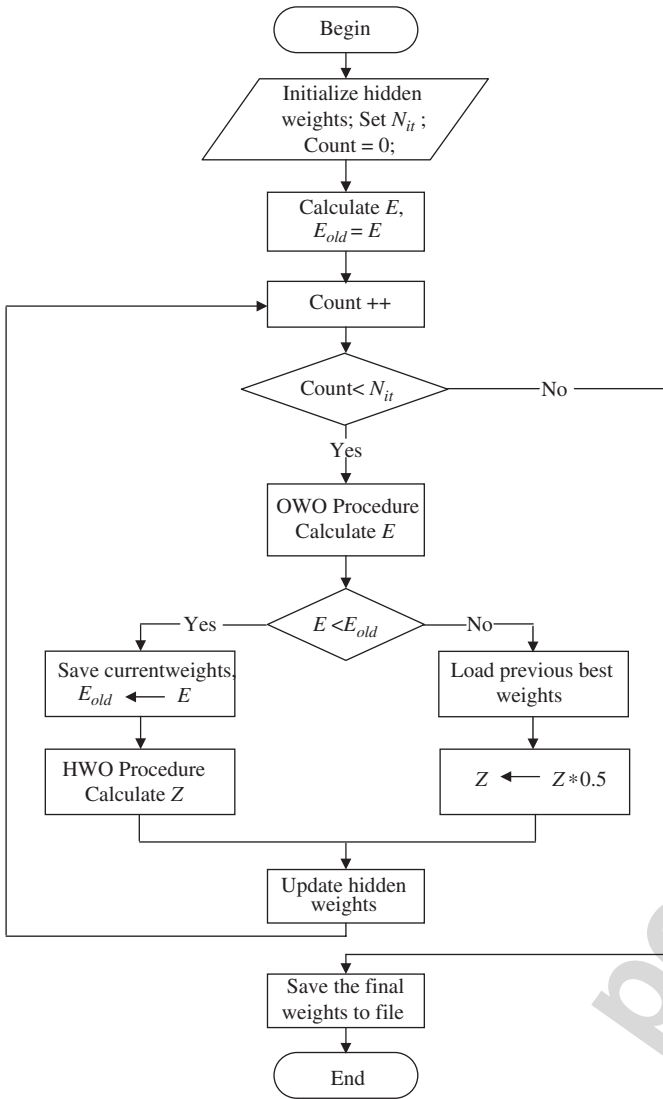


Fig. 2. OWO-HWO flowchart. (a) training MSE vs. iteration number for data set: *Twod.tra*; (b) training MSE v.s. time for data set: *Twod.tra*.

where the gain G is

$$G = 1 + \frac{\alpha}{1 + e^R}. \quad (3.27)$$

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

$$R = \frac{|\Delta E|}{\|\Delta W\|} = \frac{|E_1 - E_2|}{Z\|e\|}. \quad (3.28)$$

The hidden weight change vector e in (3.28) is available from solving (3.18) in the previous HWO iteration. E_1 and E_2 are the total output MSEs of last two iterations, respectively. The learning factor Z then is calculated by using (3.24).

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 [25]. So from (3.26) to (3.28), the learning factor 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. In the remainder of this paper, we refer to the new OWO-HWO algorithm as new HWO, for brevity. A flow chart for the original OWO-HWO and new HWO algorithms is shown in Fig. 2.

3.4. Convergence of the modified algorithm

In the following, we show that the hidden weight changes from HWO make the global error E decrease until a local minimum or a given stopping criteria is reached.

Denoting \mathbf{W}^* as the optimal weights, then we have $E(\mathbf{W}) \geq E(\mathbf{W}^*)$. And E is convex in the neighborhood of \mathbf{W}^* : $N(\mathbf{W}) = \{\mathbf{W} \mid \|\mathbf{W} - \mathbf{W}^*\| < \varepsilon, \varepsilon > 0\}$. Then the change in E caused by the hidden weight changes can be approximated as

$$\Delta E = E(\mathbf{W} + \Delta \mathbf{W}) - E(\mathbf{W}) \approx \left(\frac{\partial E}{\partial \mathbf{W}} \right)^T \Delta \mathbf{W}. \quad (3.29)$$

In addition, ΔE is the total effects of the N_h hidden units

$$\Delta E = \sum_{k=1}^{N_h} \Delta E(k), \quad (3.30)$$

where $\Delta E(k)$ is the change in E caused by the k th hidden unit

$$\begin{aligned} \Delta E(k) &\approx \sum_n \frac{\partial E}{\partial w_{hi}(k, n)} \cdot \Delta w_{hi}(k, n) = Z \cdot \sum_n \frac{\partial E}{\partial w_{hi}(k, n)} \cdot e(k, n) \\ &= Z \cdot \frac{1}{N_v} \sum_n \sum_{p=1}^{N_v} \frac{\partial E_p}{\partial w_{hi}(k, n)} \cdot e(k, n) \\ &= Z \cdot \frac{1}{N_v} \sum_n \sum_{p=1}^{N_v} (-\delta_{pk} \cdot x_{pn} \cdot e(k, n)) \\ &= -Z \cdot \frac{1}{N_v} \sum_{p=1}^{N_v} \delta_{pk} \sum_n x_{pn} \cdot e(k, n) \end{aligned} \quad (3.31)$$

From (3.18), we can say that, in the neighborhood of \mathbf{W}^* :

$$\sum_n e(k, n) \cdot x_{pn} \approx \Delta n e_{pk}^* = \frac{\delta_{pk}}{(f'_{pk}) \sum_{i=1}^M w_{oh}^2(i, k)}. \quad (3.32)$$

The second step in above equation is from (3.9). The total change of the error function E , due to changes in all hidden weights, becomes

$$\Delta E = \sum_k \Delta E(k) \approx -Z \frac{1}{N_v} \sum_k \sum_{p=1}^{N_v} \frac{\delta_{pk}^2}{(f'_{pk})^2 \sum_{i=1}^M w_{oh}^2(i, k)}. \quad (3.33)$$

As every term in the summation is non-negative, we have $\Delta E \leq 0$. So the global error E will keep decreasing until reaching a local minimum.

4. Simulation and discussion

The proposed ideas were verified using several training data sets. The performance of the new HWO was compared to the original OWO–HWO, the Levenberg–Marquardt (LM) algorithm, and the new HWO with an optimal learning factor [12]. The optimal learning factor is derived in the Appendix. We also compare the new HWO with the LBL method [20] for one data set. Our simulations were carried out on a 2.8 GHz Pentium IV, Windows NT workstation using the Visual C++ 6.0 compiler.

4.1. Simulations

Training data set 1: Twod contains simulated data based on models from backscattering measurements. This training file is used in the task of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half space, where both interfaces are randomly rough. The parameters to be inverted are the effective permittivity of the surface, the normalized rms height, the normalized surface correlation length, the optical depth, and single scattering albedo of an inhomogeneous irregular layer above a homogeneous half space from back scattering measurements.

The training data file contains 1768 patterns. The inputs consist of eight theoretical values of back scattering coefficient parameters at V and H polarization and four incident angles. The outputs were the corresponding values of permittivity, upper surface height, lower surface height, normalized upper surface correlation length, normalized lower surface correlation length, optical depth and single scattering albedo which had a joint uniform pdf [5].

The three-layer MLP has 8 inputs, 10 hidden units and 7 outputs. All the algorithms are trained for 200 iterations. From the simulation results of Fig. 3, the new HWO has the fastest convergence speed in terms of both iteration number and time. The LM algorithm costs much more time due to its heavy computational load. As in any location of quadratic error surface, optimal learning rates will be superior to heuristic ones, we also tried optimal Z in our experiments. However, as the global error function actually is not a quadratic function of Z , we found that using optimal learning factors [12] in every iteration does not result in better performance. In the following, we compare the new HWO with the LBL algorithm [20]. In addition, as the initial MSE in LBL is too large, we use OWO to initialize the output weights and term it as OWO + LBL. In Fig. 4, we present the simulation result for data *Twod*. From Fig. 4, we can see that the LBL has larger MSE than the new HWO. If LBL initialized by OWO, the LBL algorithm won't decrease the MSE any further. From the figure we also see that LBL takes more time per iteration than OWO–HWO. This occurs, in part, because LBL requires three passes through the data per iteration, but OWO–HWO requires two passes. LBL also requires

additional calculations for the learning factors in each pattern.

Training data set 2: OH7.TRA. This remote sensing training data set [22] contains VV and HH polarizations at L 30, 40 deg, C 10, 30, 40, 50, 60 degree, and X 30, 40, 50 deg along with the corresponding unknowns $\Theta = \{\sigma, l, m_v\}^T$, where σ is the rms surface height; l is the surface correlation length; m_v is the volumetric soil moisture content in g/cm^3 .

There are 20 inputs, 3 outputs, and 7320 training patterns. We used a 20–20–3 network and trained the network for 200 iterations for all the algorithms. The simulation results of Fig. 5 show that the advantage of the new algorithm over the other algorithms is overwhelming.

Training data set 3: F17. This prognostics data set contains 3335 training patterns for onboard *flight load synthesis* (FLS) in helicopters [3]. In FLS, we estimate

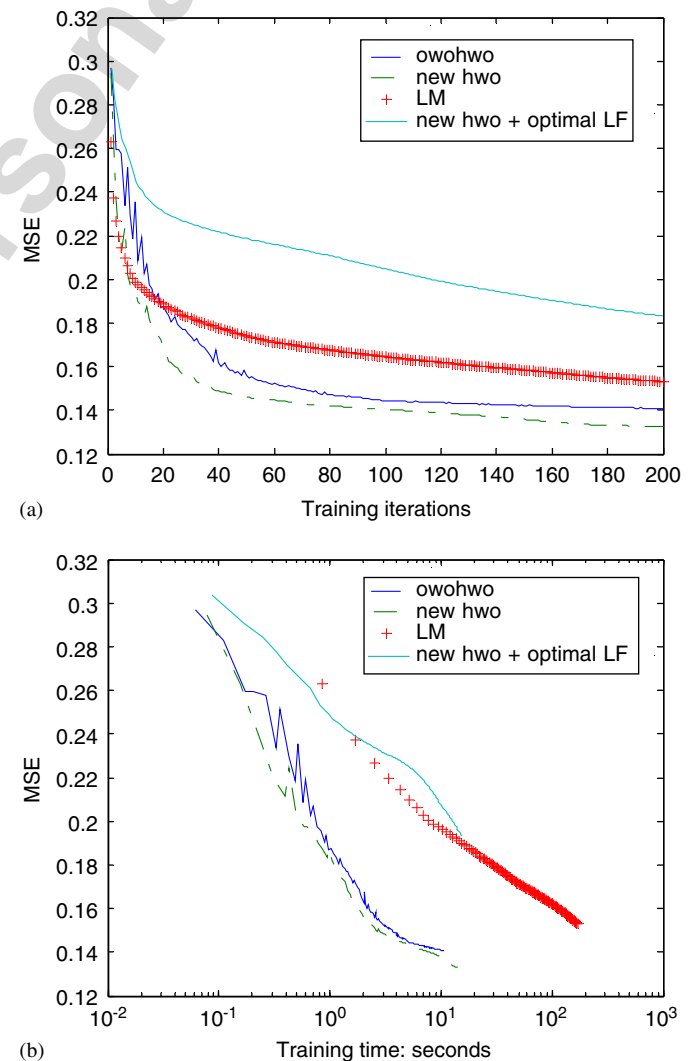


Fig. 3. Simulation results for example 1 Data: *Twod.tra*, structure: 8–10–7: (a) training MSE vs. iteration number for data set: *Twod.tra*; (b) training MSE vs. time for data set: *Twod.tra*.

mechanical loads on critical parts, using measurements available in the cockpit. The accumulated loads can then be used to determine component retirement times. There are 17 inputs and 9 outputs for each pattern. In this approach, signals available on an aircraft, such as airspeed, control attitudes, accelerations, altitude, and rates of pitch, roll, and yaw, are processed into desired output loads such as fore/aft cyclic boost tube oscillatory axial load (OAL), lateral cyclic boost tube OAL, collective boost tube OAL, main rotor pitch link OAL, etc. This data was obtained from the M430 flight load level survey conducted in Mirabel Canada in early 1995 by Bell Helicopter Textron of Fort Worth.

We chose the MLP structure 17-10-9 and trained the network for 200 iterations with all the algorithms. The simulation results of Fig. 6 show that the new algorithm outperforms the other algorithms again.

We have shown that the new HWO performs better than the original OWO–HWO and LM methods. The adaptive

learning factor also gave better results than the optimal learning factors.

5. Discussion

Evaluation of the performance of an algorithm should not just be based on the training errors. We also need to consider the testing error [4,11]. In the following, we evaluate the training algorithms using testing data sets disjoint from the training sets. The numbers of testing patterns are 1000 for twod, 3133 for oh7, and 1410 for f17. For each data set, we present both the training MSE and testing MSE for every training algorithm in Table 1. From the numerical results in Table 1, it is obvious that the new HWO algorithm has the fastest convergence speed and the best generalization ability. This good generalization is due to the facts that (1) the networks are small enough to prevent memorization, (2) the training and testing sets, in each experiment, are generated by the same random

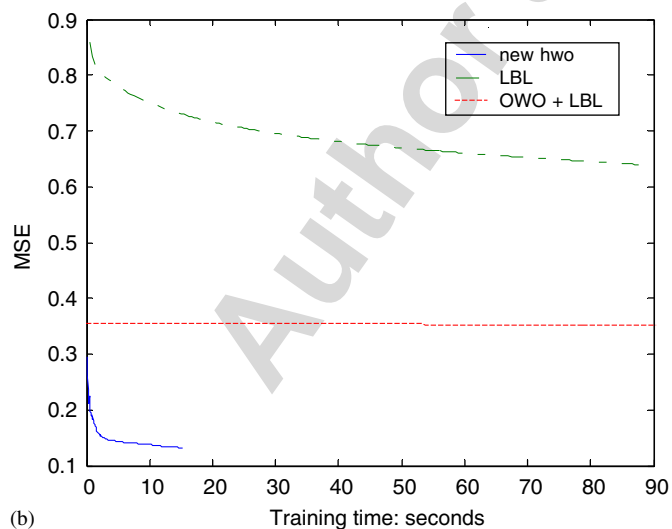
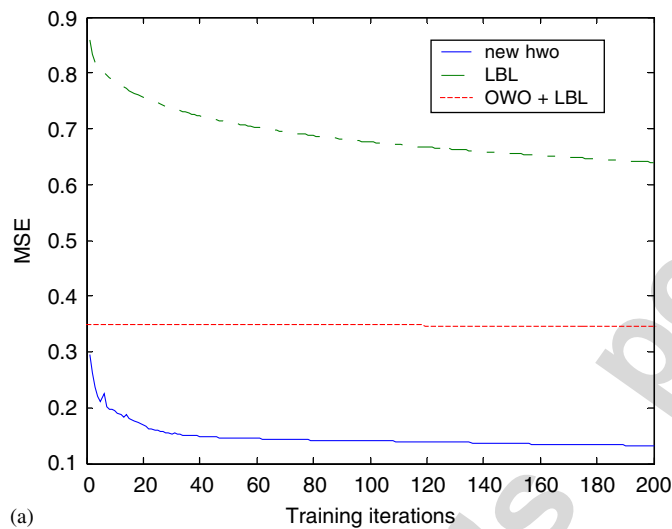


Fig. 4. Comparison of the algorithms for: *Twod.tra*, structure: 8-10-7: (a) training MSE vs. iteration number for data set: *oh7.tra*; (b) training MSE v.s. time for data set: *oh7.tra*.

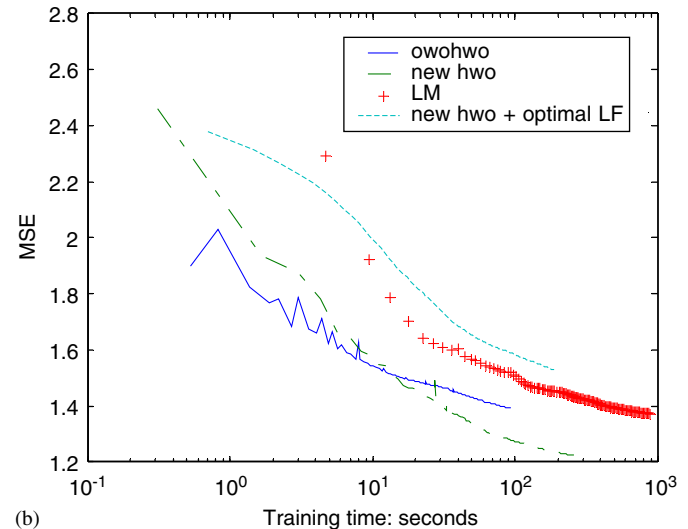
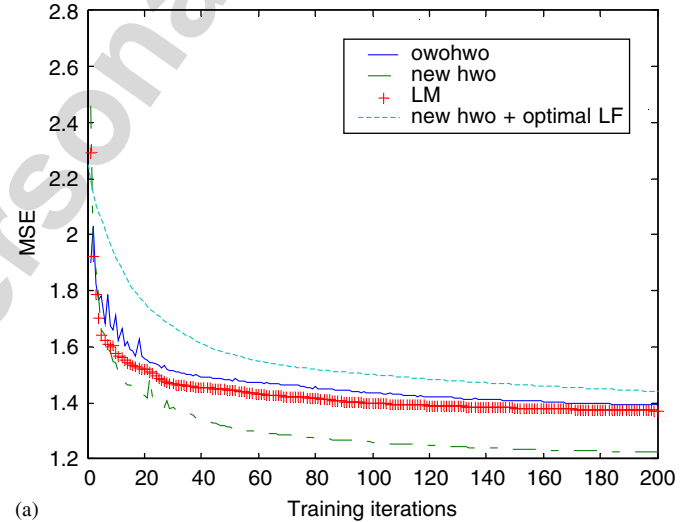


Fig. 5. Simulation results for example 2 Data: *oh7.tra*, structure: 20-20-3: (a) training MSE vs. iteration number for data set: *F17.dat*; (b) training MSE vs. time for data set: *F17.dat*.

process, and (3) the new algorithm is very effective at reducing the training MSE. The increased speed of convergence for the new algorithm is due to the facts that (1) linear equations are solved for output weights and hidden unit weight changes, (2) very few passes through the data are required per iteration, and (3) the error function places little emphasis on weights feeding into saturated hidden units.

6. Conclusions

In order to accelerate convergence of the OWO–HWO algorithm and to reduce the number of heuristics, this paper proposes some improvements. The net functions evolve in a new more optimal direction. The weighted hidden layer error function was derived directly from the global MSE. The hidden layer learning factor now adapts according to the local shape of the error surface. All these techniques successfully increase the training speed.

In addition, when testing on several data sets, the new HWO algorithm always outperforms the other algorithms. However, there are some issues that need further investigation. It is unclear whether it is necessary to update the net function in the optimal direction in the first few iterations. We also need to design an adaptation law for the learning factor, without the heuristic parameter α . And another issue we didn't attack here is the proper choice of initial hidden weights.

Acknowledgments

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

Appendix

In the following, we present the derivation of the optimal learning factor [12] for the hidden weight optimization.

To calculate Z , the error function E is rewritten as

$$E = \frac{1}{N_v} \sum_p \sum_{i=1}^M \left[t_{pi} - \sum_{k=1}^{N_h} w_{oh}(i, k) f \left(\sum_{n=1}^{N+1} (w_{hi}(k, n) + Z \cdot e(k, n)) \cdot x_{pn} \right) - \sum_{n=1}^{N+1} w_{oi}(i, n) x_{pn} \right]^2 \quad (A.1)$$

Using \mathbf{W} denoting the hidden weights, and \mathbf{e} as the hidden weight changes, in each iteration, the optimal learning rate Z used in HWO is found by solving

$$\frac{dE(\mathbf{W} + Z \cdot \mathbf{e})}{dZ} = 0. \quad (A.2)$$

Then the hidden weights are updated as $\mathbf{W} \leftarrow \mathbf{W} + Z \cdot \mathbf{e}$.

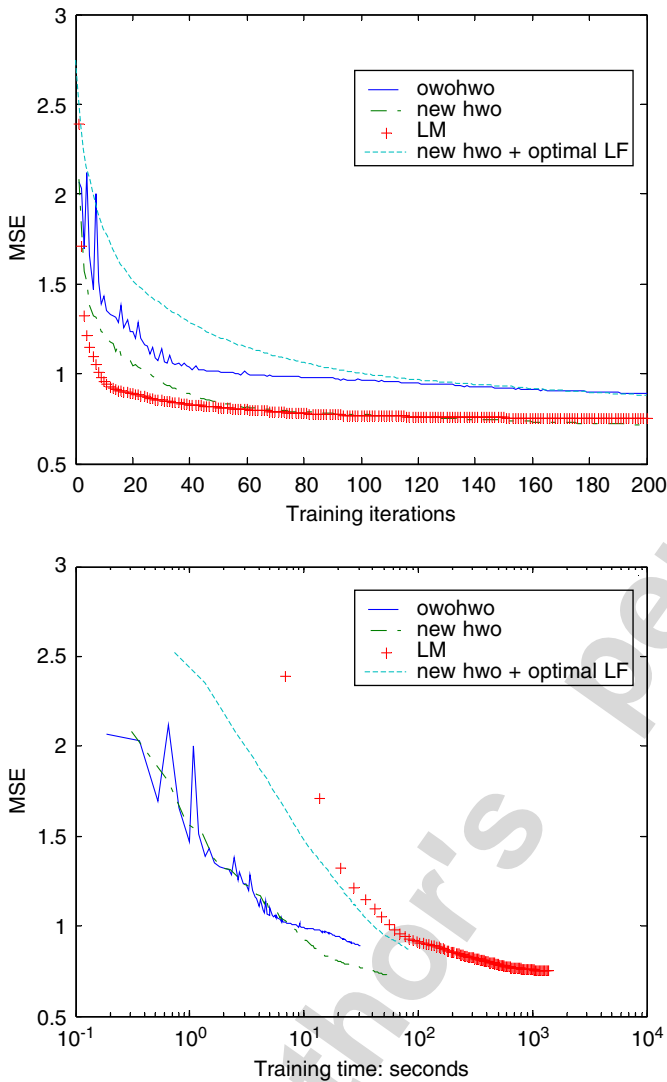


Fig. 6. Simulation results for example 3 Data: F17.dat, structure: 17-10-9.

Table 1
Training and testing errors for each training algorithm

		LM	OWO–HWO	New HWO	OWO + LBL	Layer-by-layer	Optimal LF
Twod	Training MSE	0.153103	0.140873	0.132289	0.353854	0.640023	0.191480
	Testing MSE	0.171514	0.149181	0.147620	0.391923	0.702386	0.210467
OH7	Training MSE	1.372671	1.392534	1.224596	2.3586	3.60798	1.464110
	Testing MSE	1.595475	1.562335	1.554806	2.39681	3.70564	1.610119
F17	Training MSE	0.749289	0.892191	0.716455	2.88507	3.15433	0.892393
	Testing MSE	0.821818	0.951841	0.753218	2.93159	3.16477	0.930940

As it is hard to solve (A.2) directly, we obtain Z using a Taylor series expansion of $E(\mathbf{W} + Z \cdot \mathbf{e})$ and obtaining the solution of equation (A.2). A third degree expansion of $E(\mathbf{W} + Z \cdot \mathbf{e})$ is

$$E(\mathbf{W} + Z \cdot \mathbf{e}) \cong A_0 + A_1 \cdot Z + A_2 \cdot Z^2 + A_3 \cdot Z^3 \quad (\text{A.3})$$

In Eq. (A.3), A_0 , A_1 , A_2 , and A_3 are the Taylor series coefficients. They are obtained as follows:

$$A_0 = E(\mathbf{W} + Z \cdot \mathbf{e})|_{Z=0}, \quad A_1 = \frac{\partial E(\mathbf{W} + Z \cdot \mathbf{e})}{\partial Z}|_{Z=0},$$

$$A_2 = \frac{1}{2!} \cdot \frac{\partial^2 E(\mathbf{W} + Z \cdot \mathbf{e})}{\partial Z^2}|_{Z=0}, \quad A_3 = \frac{1}{3!} \cdot \frac{\partial^3 E(\mathbf{W} + Z \cdot \mathbf{e})}{\partial Z^3}|_{Z=0}. \quad (\text{A.4})$$

Details for obtaining these coefficients are given in the following. Depending on whether the second degree or the third degree approximations of the error are used, the optimal learning rate Z_{OL} can be obtained from Eq. (A.2) either as

$$Z_{OL} = \frac{-A_1}{2 \cdot A_2}, \quad (\text{A.5})$$

or

$$Z_{OL} = \frac{-A_2 \pm \sqrt{A_2^2 - 3 \cdot A_1 \cdot A_3}}{3 \cdot A_3}. \quad (\text{A.6})$$

From Eq. (A.3) and (A.6), the second derivative of $E(Z)$ is expressed as

$$\frac{\partial^2 E}{\partial Z^2} = 2A_2 + 6A_3 \cdot Z = \pm 2\sqrt{A_2^2 - 2 \cdot A_1 \cdot A_3}. \quad (\text{A.7})$$

When $E(Z)$ is at the local minimum, $\partial^2 E(Z)/\partial Z^2$ should be greater than zero. So the resulting solution for Z_{OL} in Eq. (A.6) is

$$Z_{OL} = \frac{-A_2 + \sqrt{A_2^2 - 3 \cdot A_1 \cdot A_3}}{3 \cdot A_3}. \quad (\text{A.8})$$

Whenever the argument of the square root in Eq. (A.8) is negative, we use the solution from Eq. (A.5). Since the correctly obtained direction vector should point toward the trough of the error function, the optimal learning rate should be greater than zero when we obtain the direction vector correctly. So when the solution from either Eqs. (A.5) or (A.6) turns out to be negative, we use $-g$ as a new direction vector and obtain the optimal learning rate again from either Eqs. (A.5) or (A.8).

In order to find A_0 , A_1 , A_2 , and A_3 , we have to calculate the derivatives in (A.4). From the global MSE definition in (2.7), we have

$$\frac{\partial E}{\partial Z} = \frac{2}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M [t_{pk} - y_{pk}] \cdot \left(-\frac{\partial y_{pk}}{\partial Z} \right), \quad (\text{A.9})$$

$$\frac{\partial^2 E}{\partial Z^2} = \frac{2}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M \left[\left(\frac{\partial y_{pk}}{\partial Z} \right)^2 - [t_{pk} - y_{pk}] \cdot \frac{\partial^2 y_{pk}}{\partial Z^2} \right], \quad (\text{A.10})$$

$$\frac{\partial^3 E}{\partial Z^3} = \frac{2}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M \left[3 \cdot \left(\frac{\partial y_{pk}}{\partial Z} \right) \cdot \left(\frac{\partial^2 y_{pk}}{\partial Z^2} \right) - [t_{pk} - y_{pk}] \cdot \frac{\partial^3 y_{pk}}{\partial Z^3} \right]. \quad (\text{A.11})$$

And the output y_{pk} in terms of Z is

$$y_{pk} = \sum_{i=1}^{N+1} w_{oi}(k, i) \cdot x_{pi} + \sum_{j=1}^{N_h} w_{oh}(k, j) \cdot f(\text{net}_{pj} + Z \cdot \text{Net}_{pj}), \quad (\text{A.12})$$

where $\text{Net}_{pj} = \sum_{n=1}^{N+1} e(j, n) \cdot x_{pn}$.

So

$$\frac{\partial y_{pk}}{\partial Z} = \sum_{j=1}^{N_h} \{ w_{oh}(k, j) \cdot f \cdot (1 - f) \cdot \text{Net}_{pj} \}, \quad (\text{A.13})$$

where $f = f(\text{net}_{pj} + Z \cdot \text{NET}_{pj})$.

$$\frac{\partial^2 y_{pk}}{\partial Z^2} = \sum_{j=1}^{N_h} \{ w_{oh}(k, j) \cdot \text{Net}_{pj}^2 \cdot f \cdot (1 - f)(1 - 2f) \}, \quad (\text{A.14})$$

$$\frac{\partial^3 y_{pk}}{\partial Z^3} = \sum_{j=1}^{N_h} w_{oh}(k, j) \cdot \text{Net}_{pj}^3 \cdot f \cdot (1 - f) \cdot (1 - 6f + 6f^2). \quad (\text{A.15})$$

Substituting (A.13) (A.14) (A.15) into (A.9) (A.10) (A.11), A_0 , A_1 , A_2 , and A_3 can be calculated from (A.4). Then optimal learning rate can be found from (A.5) or (A.10).

References

- [1] S.A. Barton, A matrix method for optimizing a neural network, *Neural Computation* 3 (3) (1991) 450–459.
- [2] R. Battiti, First- and Second-order methods for learning: between steepest descent and Newton's method, *Neural computation* 4 (2) (1992) 141–166.
- [3] H.-H. Chen, M.T. Manry, Hema Chandrasekaran, A neural network training algorithm utilizing multiple set of linear equations, *Neurocomputing* 25 (1–3) (1999) 55–72.
- [4] Y. LeCun, Generalization and network design strategies, in: *Proceedings of Connectionism in perspective*, 1989.
- [5] M.S. Dawson, A.K. Fung, M.T. Manry, Surface parameter retrieval using fast learning neural networks, *Remote Sensing Reviews* 7 (1) (1993) 1–18.
- [6] R. Fletcher, *Practical Methods of Optimization*, Wiley, New York, 1987.
- [7] M.H. Fun, M.T. Hagan, Levenberg-Marquardt training for modular networks, *The 1996 IEEE International Conference on Neural Networks*, vol. 1, 1996, pp. 468–473.
- [8] C.M. Hadzler, R. Hasan, et al., Improved singular value decomposition by using neural networks, *IEEE International Conference on Neural Networks*, vol. 1, 1995 438–442.
- [9] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, *IEEE Transaction on Neural Networks* 5 (6) (1994) 989–993.
- [10] M.T. Hagan, H.B. Demuth, M.H. Beale, *Neural Network Design*, PWS publishing Company, 1996.
- [11] S. Haykin, *Neural networks: a comprehensive foundation*, Prentice-Hall, Englewood Cliffs, NJ, 1999.
- [12] T.H. Kim, M.T. Manry, F.J. Maldonado, New learning factor and testing methods for conjugate gradient training algorithm,

- IJCNN'03. International Joint Conference on Neural Networks, 2003, pp. 2011–2016.
- [13] H.-M. Lee, C.-M. Chen, T.-C. Huang, Learning efficiency improvement of back-propagation algorithm by error saturation prevention method, *Neurocomputing* 41 (2001) 125–143.
- [14] Y. Lee, S.-H. Oh, M.W. Kim, An analysis of premature saturation in back-propagation learning, *Neural Networks* 6 (1993) 719–728.
- [15] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the backpropagation algorithm using learning adaptation Methods, *Neural Computation* 11 (1999) 1769–1796.
- [16] M.T. Manry, H. Chandrasekaran, C.-H. Hsieh, *Signal Processing Using the Multiplayer Perceptron*, CRC Press, Boca Raton, 2001.
- [17] M.T. Manry, et al., Fast training of neural networks for remote sensing, *Remote Sensing Reviews* 9 (1994) 77–96.
- [18] T. Masters, *Neural, Novel & Hybrid Algorithms for Time Series Prediction*, Wiley, New York, 1995.
- [19] S.-H. Oh, Improving the error back-propagation algorithm with a modified error function, *IEEE Transactions on Neural Networks* 8 (3) (1997) 799–803.
- [20] S.-H. Oh, S.-Y. Lee, A new error function at hidden layers for fast training of multilayer perceptrons, *IEEE Transactions on Neural Networks* 10 (1999) 960–964.
- [21] S.-H. Oh, S.-Y. 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, 1999, pp. 1635–1638.
- [22] Y. Oh, K. Sarabandi, F.T. Ulaby, An empirical model and an inversion technique for radar scattering from bare soil surfaces, *IEEE Transactions on Geoscience and Remote Sensing* 30 (2) (1992) 370–381.
- [23] V. Ooyen, B. Nienhuis, Improving the convergence of the back-propagation algorithm, *Neural Networks* 5 (1992) 465–471.
- [24] W.H. Press, et al., *Numerical Recipes*, Cambridge University Press, New York, 1986.
- [25] K. Rohani, M.S. Chen, M.T. Manry, Neural subset design by direct polynomial mapping, *IEEE Transactions on Neural Networks* 3 (6) (1992) 1024–1026.
- [26] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representation by error propagation, *Parallel Distributed Processing* 1 (1986) 318–362.
- [27] R.S. Scalero, N. Tepedelenioglu, A fast new algorithm for training feedforward neural networks, *IEEE Transactions on signal processing*, vol. 40 (1) 1992 pp. 202–210.
- [28] G.-J. Wang, C.-C. Chen, A fast multilayer neural-network training algorithm based on the layer-by-layer optimizing procedures, *IEEE Transactions on Neural Networks* 7 (3) (1996) 768–775.
- [29] P.J. Werbos, *Beyond regression: new tools for prediction and analysis in the behavioral science*, Ph.D. Thesis, Harvard University, Cambridge, Mass, 1974.
- [30] J.Y.F. Yam, T.W.S. Chow, A weight initialization method for improving training speed in feedforward neural network, *Neurocomputing* 30 (2000) 219–232.



Changhua Yu received his B.S. in 1995 from Huazhong University of Science and Technology, Wuhan, China, and M. Sc. in 1998 from Shanghai Jiaotong University, Shanghai, China. He received his Ph.D. degree in Electrical Engineering from the University of Texas at Arlington in 2004. He joined the Neural Networks and Image Processing Lab in the EE department as a research assistant in 2001. His main research interests include neural networks,

image processing and pattern recognition. Currently, Dr. Yu works at FastVDO LLC in Columbia, Maryland.



Michael T. Manry was born in Houston, Texas in 1949. He received the B.S., M.S., and Ph.D. in Electrical Engineering in 1971, 1973, and 1976, respectively, from The University of Texas at Austin. After working there for two years as an Assistant Professor, he joined Schlumberger Well Services in Houston where he developed signal processing algorithms for magnetic resonance well logging and sonic well logging.

He joined the Department of Electrical Engineering at the University of Texas at Arlington in 1982, and has held the rank of Professor since 1993. In summer 1989, Dr. Manry developed neural networks for the Image Processing Laboratory of Texas Instruments in Dallas. His recent work, sponsored by the Advanced Technology Program of the state of Texas, E-Systems, Mobil Research, and NASA has involved the development of techniques for the analysis and fast design of neural networks for image processing, parameter estimation, and pattern classification. Dr. Manry has served as a consultant for the Office of Missile Electronic Warfare at White Sands Missile Range, MICOM (Missile Command) at Redstone Arsenal, NSF, Texas Instruments, Geophysics International, Halliburton Logging Services, Mobil Research and Verity Instruments. He is a Senior Member of the IEEE.



Jiang Li received his B.S. in Electrical Engineering from Shanghai Jiaotong University, China in 1992, his M.S. in Automation from Tsinghua University, China in 2000. He received his Ph.D degree in Electrical Engineering from the University of Texas at Arlington in 2004. His research interests focus on neural networks, wireless communication, pattern recognition and signal processing. Currently, Dr. Li works at the National Institute of Health in Bethesda Maryland.



Pramod Lakshmi Narasimha received his B.E. in Telecommunications Engineering from Bangalore University, India. He joined the Neural Networks and Image Processing Lab in the EE department, UTA as a research assistant in 2002. In 2003, he received his M.S. degree in Electrical Engineering at the University of Texas at Arlington. Currently, he is working on his Ph.D. His research interests focus on neural networks, pattern recognition, image and signal processing.